Universidad del País Vasco
Euskal Herriko Unibertsitatea

Máster Universitario en

Ingeniería **C**omputacional y **S**istemas **I**nteligentes

Konputazio Zientziak eta Adimen Artifiziala Saila –
Departamento de Ciencias de la Computación e Inteligencia Artificial

K I S A

I C S I

Tesis de Máster

Una aproximación integral desde la metodología del Descubrimiento del Conocimiento al problema de la predicción de supervivencia en el desastre del Titanic

Héctor Orlando Calderón Pezo

Tutor(a/es)

Dr. Roberto Santana Hermida
Departamento de Ciencia de la Computación e Inteligencia Artificial
Facultad de Informática

informatika fakultatea
facultad de informática

KZAA /CCIA

Septiembre 2013

Agradecimientos

Quiero agradecer a mi madre y mis tíos por su constante interés, apoyo moral y amor que entregan en la distancia durante este largo camino, y principalmente a Dios en quien me apoyo siempre.

Índice General

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

Hoy en día la información es una materia prima muy valiosa, para empresas, organizaciones así como simples usuario. Para todos ellos es importante obtener información óptima y oportuna. Es por esto que en la sociedad actual, la creación, distribución, uso, integración y manipulación de información es una actividad significativamente económica, política y cultural. Uno de los objetivos de la sociedad de la información es obtener una ventaja competitiva a nivel internacional, mediante el uso de tecnología de la información (TI) de una manera creativa y productiva.

La economía del conocimiento es su contrapartida económica, por la que se crea la riqueza a través de la explotación económica del conocimiento. Las personas que tienen los medios para participar en este tipo de sociedad a veces se llaman ciudadanos digitales. Esta es una de las muchas docenas de etiquetas que han sido identificadas para sugerir que los seres humanos están entrando en una nueva fase de la sociedad.

Ha habido mucha discusión sobre el impacto de la llamada "era de la información". La humanidad fue capaz de almacenar 295 exabytes (1.073.741.824 gigabytes) de información hasta 2007 tanto en medios analógicos como digitales, afirma un estudio publicado en Science Express

Es por ello que existen factores importantes que han influido en el aumento de la información., Entre estos factores están la acumulación rápida de datos, el desarrollo de sistemas gestores de base de datos más poderosos y el constante desarrollo tecnológico donde internet y las bases de datos dinámicas juegan un papel fundamental.

Frecuentemente se escuchan las palabras *Datos, Información y Conocimiento,* usados como si fueran la misma cosa, para definir el conocimiento primero debemos referirnos a

las dos fuentes del conocimiento que son los *datos*, y la *información*, los *datos* son los hechos del mundo, como por ejemplo las medidas de un individuo, su talla, el color de sus ojos, el color de su cabello. En sentido general los datos pueden ser considerados como una descripción del mundo. Se puede percibir esta información con nuestros sentidos, y entonces el cerebro puede procesarla. La *información* nos permite ampliar nuestro conocimiento más allá del alcance de nuestros sentidos. Podemos capturar datos en información y luego ponerlos a disposición de otras personas para que puedan acceder a ella en momentos diferentes. Por ejemplo una fotografía se puede hacer llegar a diferentes personas en diferentes momentos.

Por lo tanto se puede definir el *conocimiento* como una aplicación de los datos y la información, de manera que su intención es la de ser útil. El *conocimiento* es un proceso determinista.

En este trabajo para poder extraer dicho conocimiento se utiliza la Minería de Datos, una de las etapas del análisis del descubrimiento del conocimiento en bases de datos. Este es un campo dentro de las Ciencias de la Computación que se refiere al proceso que intenta descubrir patrones en grandes volúmenes de datos. En este campo se utilizan métodos de la Inteligencia Artificial tales como el aprendizaje automático, la estadística y también los sistemas de bases de datos [FPSSU96],[Dan02].

Para definir la Minería de Datos, también se utilizan otros términos con igual o similar significado, tales como Knowledge Mining from Databases (de «Minería de Conocimiento de Bases de Datos»), Knowledge Extraction (de «Extracción de Conocimiento»), Data Archeology (de «Arqueología de Datos») y Data Dredging (de «Excavación de Datos»).

En esta tesis utilizamos un conjunto de diversas herramientas de la minería de datos para realizar una aproximación exhaustiva a un problema de clasificación de notable dificultad. La tesis aborda el problema desde el estudio de las características de los datos, pasando por el análisis de cada una de las variables identificadas y su poder predictivo, hasta la propuesta y evaluación de diferentes clasificadores.

Para comprobar la validez de los enfoques propuestos en la tesis abordamos un problema de clasificación real. En esta Tesis de Maestría utilizamos una base de datos recopilada a partir de la información disponible del desastre del Titanic. El RMS Titanic se hundió el 15 de abril de 1912, en el Océano Atlántico Norte, en su viaje inaugural, viajando de South Hampton, Reino Unido a Nueva York, EE.UU. Era el barco más grande de mundo en el momento de su botadura. El hundimiento del Titanic causó la muerte de 1502 personas de las 2224 que iban a bordo, lo que lo convierte en uno de los mayores naufragios de la historia ocurridos en tiempos de paz. Consideramos un conjunto de datos sobre los registros de los pasajeros del Titanic en el momento del desastre, donde se dispone de 891 datos de entrenamiento y 418 casos de prueba.

El problema a resolver es predecir si un pasajero sobrevivirá o no a partir de los diferentes atributos categóricos y numéricos describiendo a las 1303 personas a bordo. En

la tesis desarrollamos todos los pasos necesarios que muestran como la metodología del Descubrimiento del Conocimiento puede dar solución a este problema.

1.1. Objetivos

El objetivo de este trabajo es estudiar el Desastre del Titanic, utilizando la metodología del Descubrimiento del Conocimiento (KDD). La tesis propone diferentes variantes de cómo aplicar técnicas de Minería de Datos y herramientas del Aprendizaje Automático para predecir de forma eficiente la sobrevivencia de los pasajeros. Con este fin se han adaptado diferentes algoritmos de pre-procesamiento de datos, selección de variables y clasificación, a las características particulares del problema tratado. Algunos de estos algoritmos han sido implementados o sus implementaciones han sido modificadas para el caso específico del problema del Titanic.

1.2. Estructura de la Tesis

Luego del presente capitulo introductorio, en el segundo capítulo, se hará una descripción de las diferentes etapas de la metodología del Descubrimiento del Conocimiento (Kwowledge Discovery in Databases), relacionándolos en algunos casos con el tema de este trabajo de tesis.

El tercer capítulo comprende las etapas primera y segunda de la metodología de descubrimiento del conocimiento, que son: la comprensión del dominio, luego la selección y la creación de un conjunto de datos en los que se realizará el descubrimiento.

El cuarto capítulo se refiere a la etapa del pre-procesamiento y transformación de datos (tercera y cuarta etapa del descubrimiento de conocimiento).

En el quinto capítulo se tocaran los aspectos algorítmicos dentro de la minería de datos y el aprendizaje automático (que son la quinta, sexta, séptima y octava etapas del descubrimiento del conocimiento). Estos aspectos comprenden la minería de datos propiamente, la evaluación e interpretación de los resultados. Se describirán y determinarán los métodos que permiten mejorar los resultados.

Finalmente en el sexto capítulo se presentan las conclusiones y recomendaciones sobre el trabajo y futuros estudios relacionados con la investigación.

Para dar un mejor entendimiento a este trabajo, se han incluido cinco apéndices:

- ✓ Apéndice A: Códigos en R para la selección de variables con el enfoque de subconjuntos.
- ✓ Apéndice B: Códigos para procesar algoritmos (RANDOM FOREST en R, Python) para el enfoque de rankeo de características.
- ✓ Apéndice C: Códigos para procesar algoritmos (diez algoritmos de minería de datos en R y Python), para el enfoque de selección de subconjuntos.

- ✓ Apéndice D: Códigos en R para la extracción de características de los datos.
- ✓ Apéndice E: Código para crear el paquete paqdatamining (en R que se usa para en los algoritmos de minería de datos).

Capítulo 2

Introducción al Descubrimiento del Conocimiento

En este capítulo se realizara un análisis general del proceso de extracción del conocimiento.

2.1. Introducción

El descubrimiento del conocimiento en bases de datos se puede definir como un proceso organizado de identificación de patrones válidos novedosos, útiles y comprensibles a partir de conjuntos de datos grandes y complejos [OMLR10]. Más formalmente se trata de la búsqueda de relaciones y patrones globales que existen en grandes bases de datos pero que permanecen ocultos ante los métodos convencionales.

2.2. Etapas en el proceso de KDD

El proceso de Descubrimiento del Conocimiento, es iterativo e interactivo y consiste de nueve pasos, haciéndose notar que es iterativo en cada paso, lo cual significa que se requiere moverse hacia atrás para ajustar pasos previos, siendo el usuario uno de los elementos más importantes a considerar, pues es él quien determina el dominio de la aplicación, y decide qué datos y cómo se utilizarán en el proceso.

Los pasos a considerar se muestran en la Figura 1, el proceso comienza con la determinación de las metas del KDD, y termina con la puesta en práctica del conocimiento descubierto. A continuación se describen los nueve pasos del KDD.

1. **Desarrollar un entendimiento del dominio de aplicación.-** Este es un paso preparatorio inicial en donde se prepara el escenario para la comprensión de lo que se debe hacer con las muchas decisiones (acerca de la transformación, los algoritmos, la representación, etc.). Los usuarios expertos que están a cargo de un proyecto de KDD necesitan entender y definir los objetivos del usuario final y el medio ambiente en el que el proceso de descubrimiento de conocimiento se llevará a cabo (incluyendo el conocimiento previo relevante). A medida que avanza en el proceso KDD, puede haber incluso una revisión y puesta a punto de este paso. Habiendo entendido los objetivos KDD, el pre-procesamiento de los datos se inicia, tal como se definen en los siguientes tres pasos del KDD (se debe notar que algunos de los métodos aquí son similares a los algoritmos de minería de datos, pero son usados en el contexto del pre-procesamiento).

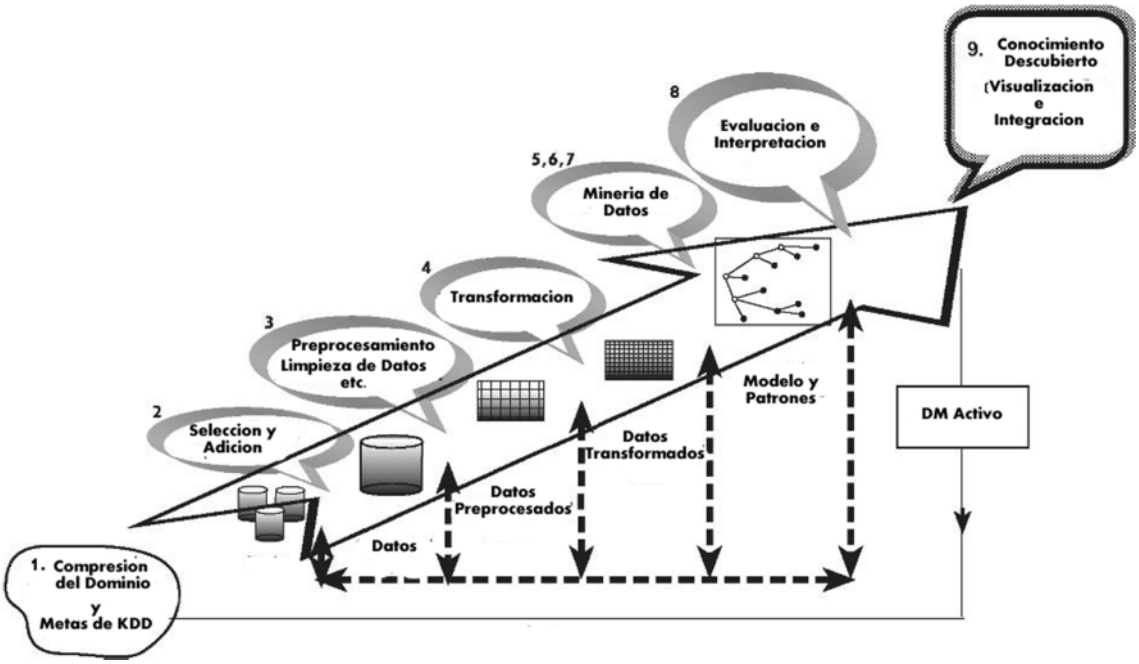


Figura 1.- El Proceso de Descubrimiento del Conocimiento en Bases de Datos.
(Tomado del libro "Data Mining and Knowledge Discovery Handbook" de Oded Maimon y Lior Rokach [MORL10])

2. **Selección y creación de un conjunto de datos en el que se realizará el descubrimiento.-** Una vez definidos los objetivos y metas, se deben determinar los datos que se utilizarán para el descubrimiento de conocimiento. Este paso incluye averiguar de qué datos se dispone, la obtención de datos adicionales necesarios, y luego la integración de todos los datos para el descubrimiento de conocimiento en un conjunto de datos, incluyendo los atributos que se tendrán en cuenta para el proceso. Este proceso es muy importante debido a que la minería de datos aprende y descubre a partir de los datos disponibles. Esta es la base de pruebas

para la construcción de los modelos. Si algunos atributos importantes se pierden, entonces todo el estudio puede fallar. Para el éxito del proceso, es bueno tener en cuenta el mayor número posible de atributos en esta etapa.

En este paso los datos significativos son seleccionados o creados. Buscando los atributos apropiados de entrada y la información de salida para representar la tarea. Es decir, lo primero que se tiene que tener en cuenta antes de comenzar con el proceso, es saber qué es lo que se quiere obtener y cuáles son los datos que nos permitirán realizar esta tarea.

3. **Pre-procesamiento y limpieza.-** En esta etapa, la fiabilidad de los datos se ve reforzada. Esto incluye limpieza de datos, tales como el manejo de los valores perdidos y la eliminación de ruido (valores incorrectos o inesperados) o valores atípicos. Para ello se dispone de varios métodos que luego se explicarán en este documento [MSTM11], este paso puede consumir la mayor parte (en términos de tiempo consumido) de un proceso KDD en ciertos proyectos. También puede involucrar métodos estadísticos complejos, o el uso específico de la minería de datos en este contexto. Por ejemplo, si se sospecha que un determinado atributo no es lo suficientemente confiable o tiene demasiados datos que faltan, entonces este atributo podría ser el objetivo de un algoritmo supervisado de minería de datos. Se desarrollará un modelo de predicción para este atributo, y los datos que faltan se podrán predecir. La extensión a la que se presta atención a este nivel depende de muchos factores. En cualquier caso, este proceso está formado por tres fases: definir y determinar los tipos de errores, buscar e identificar las instancias que contienen errores y corregir los errores descubiertos, usando cualquiera de los métodos de manejo de valores perdidos y la eliminación de ruido que serán expuestos posteriormente es esta tesis.

4. **Transformación de datos.-**En esta etapa, se persigue preparar y generar datos de mayor calidad para la minería de datos Los métodos aquí incluyen la reducción de la dimensión (como la selección y extracción de características, y el muestreo de registros), la transformación de atributos (tales como la discretización de atributos numéricos [FMRR05] [HLT99] [MORL10], y la transformación funcional). Este paso es a menudo crucial para el éxito de todo el proyecto KDD, pero por lo general es muy específico para el proyecto. La discretización es un procedimiento de tratamiento de datos que transforma los datos cuantitativos en cualitativos. Las transformaciones discretas de los datos mejoran la comprensión de las reglas descubiertas al transformar los datos de bajo nivel en datos de alto nivel y también reducen significativamente el tiempo de ejecución del algoritmo de inducción. Existen diversas taxonomías en la literatura para clasificar los métodos de discretización [MORL10], tales como los métodos de discretización Igual-

Ancho, Igual-frecuencia y frecuencia-fija entre otros. Una vez completados los cuatro pasos anteriores, los siguientes cuatro pasos están relacionados con la parte de minería de datos, donde la atención se centra en los aspectos algorítmicos empleados para cada proyecto.

5. **Selección de la tarea apropiada dé minería de datos**.-En esta etapa se decide sobre cuál tipo de minería de datos utilizar, por ejemplo, clasificación, regresión, o clustering. Esto depende en gran medida de los objetivos KDD, y también en los pasos anteriores. Hay dos objetivos principales en la minería de datos: predicción y descripción. La Predicción se refiere a menudo como minería de datos supervisada. Frecuentemente se hace referencia a la predicción como minería de datos supervisada, mientras que la minería de datos descriptiva incluye los aspectos no supervisados y de visualización de la minería de datos. La mayoría de las técnicas de minería de datos se basan en el aprendizaje inductivo, donde se construye un modelo de manera explícita o implícita, generalizando a partir de un número suficiente de ejemplos de entrenamiento. El supuesto básico del enfoque inductivo es que el modelo entrenado es aplicable a los casos futuros. La estrategia también tiene en cuenta el nivel de meta-aprendizaje para el conjunto particular de datos disponibles.

6. **Selección del algoritmo de minería de datos**.- Teniendo la estrategia, ahora se decide sobre las tácticas. Esta etapa incluye seleccionar el método específico que se utilizará para buscar patrones (incluyendo múltiples inductores). Por ejemplo, considerando la precisión frente a la comprensión, la primera es mejor con redes neuronales, mientras que el segundo es mejor con los árboles de decisión. Para cada estrategia de meta-aprendizaje hay varias posibilidades de cómo se puede lograr. El Meta-aprendizaje se centra en explicar qué causa que un algoritmo de minería de datos tenga éxito o no en un problema particular. Por lo tanto, este método trata de comprender las condiciones bajo las cuales un algoritmo de minería de datos es más apropiado. Cada algoritmo tiene parámetros y tácticas de aprendizaje (tal como el 10-fold cross-validation u otra división para el entrenamiento y la prueba).

7. **Empleo de algoritmos de minería de datos**.- Finalmente se llega a la implementación del algoritmo de minería de datos. En este paso, es posible emplear el algoritmo varias veces hasta que se obtenga un resultado satisfactorio, por ejemplo, mediante la regulación de los parámetros de control del algoritmo, como el número mínimo de instancias en una sola hoja de un árbol de decisión.

8. **Evaluación**.- En esta etapa, se evalúan e interpretan los patrones extraídos (reglas, fiabilidad, etc.), con respecto a los objetivos definidos en el primer paso. Aquí tenemos en cuenta los pasos de pre-procesamiento con respecto a su efecto sobre los resultados de los algoritmos de minería de datos (por ejemplo, adicionar características en el paso 4, y repetir desde allí). Este paso se centra en la comprensión y la utilidad del modelo inducido. En este paso el conocimiento descubierto también está documentado para su uso posterior. El último paso es el uso y la retroalimentación general respecto a los patrones y los resultados descubiertos obtenidos con la minería de datos.

9. **Usando el conocimiento descubierto**.-En esta etapa se incorpora el conocimiento a otro sistema para acciones futuras. El conocimiento se vuelve activo en el sentido de que es posible realizar cambios en el sistema y medir los efectos. En esta etapa el éxito de este paso determina la eficacia de todo el proceso de KDD. Hay muchos desafíos en esta etapa, como perder las "condiciones de laboratorio" en que hemos operado. Por ejemplo, el conocimiento fue descubierto a partir de una cierta instancia estática (por lo general una muestra) de los datos, pero ahora los datos se convierten en dinámicos. Las estructuras de datos pueden cambiar (ciertos atributos no están disponibles), y el dominio de datos puede ser modificado (tales como p. ej.: un atributo puede tener un valor que no fue asumido antes).

2.3. El Descubrimiento del Conocimiento y la Minería de Datos

Históricamente, al concepto de la búsqueda de patrones útiles de los datos se le ha dado una gran variedad de nombres, incluyendo la minería de datos, extracción de conocimiento, información sobre descubrimientos, recolección de información, la arqueología de datos, y el procesamiento de patrones de datos [UFP13].

El KDD se refiere al proceso en general del descubrimiento de conocimiento útil de los datos, y la minería de datos es un paso en el proceso KDD que consiste en aplicar análisis de datos y algoritmos de descubrimiento que producen una particular enumeración de patrones (o modelos) sobre los datos. Aunque los dos términos KDD y Minería de Datos son muy utilizados indistintamente, se refieren a dos conceptos relacionados pero ligeramente diferentes. KDD es el proceso general de extracción de conocimiento a partir de datos, mientras que la minería de datos es un paso dentro del proceso de KDD, que trata de identificar patrones en los datos. En otras palabras, la minería de datos es sólo la aplicación de un algoritmo específico basado en el objetivo general del proceso de KDD.

2.4. Minería de Datos

El componente de minería de datos del proceso KDD a menudo implica la aplicación repetitiva e iterativa de particulares métodos de minería de datos [UGP13]. Por ello las metas del descubrimiento del conocimiento son definidas por el uso intencionado del sistema. Se pueden distinguir dos tipos de objetivos: (1) la *verificación* y (2) el *descubrimiento*. Con la *verificación*, el sistema se limita a la verificación de la hipótesis del usuario. Con el *descubrimiento*, el sistema de forma autónoma encuentra nuevos patrones. De otra parte, se subdivide la meta del descubrimiento en la *predicción* (donde el sistema encuentra patrones para predecir el futuro comportamiento de algunas entidades), y la *descripción* (donde el sistema encuentra patrones para la presentación a un usuario en una forma humano-comprensible para un humano).

2.4.1. Taxonomía de los métodos de minería de Datos

Hay muchos métodos de minería de datos que se utilizan para diferentes propósitos y metas. La taxonomía es una ayuda para la comprensión de la variedad de métodos, su interrelación y agrupamiento. Es útil distinguir entre dos tipos principales de minería de datos: *orientado a la verificación* (el sistema verifica la hipótesis del usuario) y *orientado al descubrimiento* (el sistema encuentra nuevas reglas y patrones de forma autónoma). La Figura 2 presenta esta taxonomía.

En la Figura 2, se puede ver la taxonomía de la minería de datos ordenada jerárquicamente.

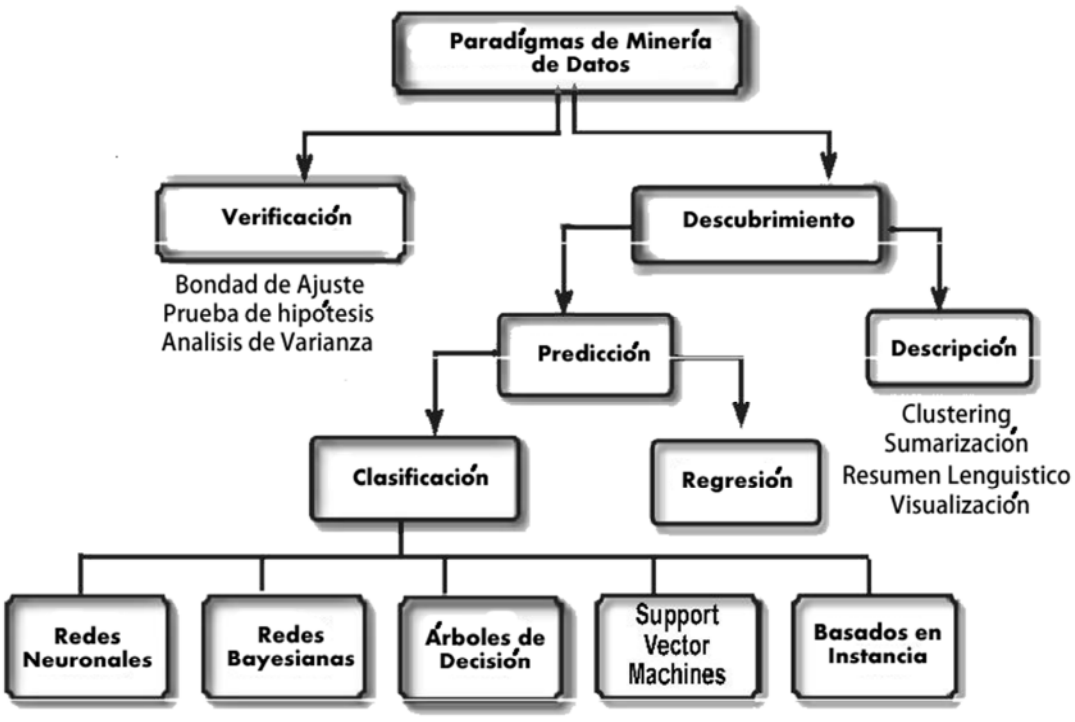


Figura 2.-Taxonomia de Minería de Datos

(Tomado del libro “Data Mining and Knowledge Discovery Handbook” de Oded Maimon y Lior Rokach [MORL10])

En este trabajo se utilizará el *aprendizaje supervisado*, más no el *aprendizaje no supervisado.* En el Aprendizaje Supervisado, se tiene un conjunto de datos (variables independientes y una variable independiente) que consta de atributos y etiquetas. La tarea es construir un estimador que sea capaz de predecir la etiqueta de un objeto dado a partir de un conjunto de características. Algunos ejemplos de aprendizaje supervisado son:

a) Dado una imagen multicolor de un objeto a través de un telescopio, determinar si ese objeto es una estrella, un quásar, o una galaxia.
b) Dada una fotografía de una persona, identificar a la persona en la foto.
c) Dada una lista de películas que una persona ha visto y en la calificación personal de la película, recomendar una lista de películas que les gustaría (denominados sistemas de recomendación: Un ejemplo famoso es Netflix, página de películas por internet).

Capítulo 3

Preparación del Escenario y la Base de Datos para la Minería de Datos

En este capítulo se presentan los primeros cuatro pasos del proceso de descubrimiento del conocimiento que son: la preparación del escenario, luego la selección y la creación de un conjunto de datos en la que se realizará el proceso de descubrimiento, seguidamente el pre-procesamiento y limpieza de datos, y por último se realizará la transformación de datos. En cada uno de estos casos se aplicarán los conceptos del proceso de descubrimiento del conocimiento, en la base de datos objeto de investigación.

3.1. Entendimiento del dominio de aplicación

Este es el paso preparatorio inicial en donde se prepara el escenario para la comprensión de lo que se debe hacer con las muchas decisiones (acerca de la transformación, los algoritmos, representación, etc.).

3.1.1. Identificación del Problema

El RMS Titanic (en inglés: *Royal Mail Steamship Titanic*, «Buque de vapor del Correo Real Titanic») fue un transatlántico británico, el mayor barco del mundo en el momento de su botadura, que se hundió en la madrugada del 14 al 15 de abril de 1912 durante su viaje inaugural desde Southampton a Nueva York. El hundimiento del Titanic causó la muerte de 1502 personas de las 2224 [ET13] [FW13] (aunque estas cifras difieren según

diferentes versiones), lo que lo convierte en uno de los mayores naufragios de la historia ocurridos en tiempos de paz.

Las cubiertas del Titanic en el momento del desastre (Figura 3) [FW13] eran las siguientes:

1. **Cubierta de botes**

   Los botes salvavidas, a 17,68 m por encima de la línea de flotación, estaban ubicados en 2 grupos, uno hacia la proa y otro hacia la popa. En la parte delantera se hallaban 12 botes (6 a cada lado), y hacia popa, se hallaban 8 botes (4 a cada lado), contando en total con veinte botes salvavidas de tres tipos diferentes:

   a. Botes 1 y 2: chinchorros de madera para emergencias, con capacidad para 40 personas.
   b. Botes del número 3 al 16: hechos de madera, con capacidad para 65 personas.
   c. Botes A, B, C y D: botes plegables marca Englehardt con capacidad para 47 personas, estos botes tenían los costados de tela.

2. **Cubierta A**

   Conocida también como la cubierta de paseo, este nivel, a 14,78 m por encima de la línea de flotación, albergaba los camarotes (hacia proa), y diferentes tipos de salas y cafés de uso únicamente para pasajeros de primera clase, pues la escalinata de segunda no tenía salida a ésta cubierta.

3. **Cubierta B**

   Esta cubierta, a 12,04 m por encima de la línea de flotación, fue diseñada principalmente para alojar a los pasajeros de primera clase.

4. **Cubierta C**

   Esta cubierta, a 9,30 m estaba dedicada principalmente a los alojamientos de los pasajeros de 1ª clase y los espacios de la tripulación. A popa, estaba el salón general de tercera a estribor y el de fumadores de 3ª a babor con las escaleras también de tercera clase. Entre el centro y la popa se ubicaban el salón general de segunda o librería con un paseo también de 2ª a ambos lados.

5. **Cubierta D**

   A 6,10 m por encima de la línea de flotación, en la parte delantera de esta cubierta se encontraban las estancias de los maquinistas y una sala común de tercera clase. Luego de ser separados por un mamparo seguían las habitaciones de primera clase.

6. **Cubierta E**

   Esta cubierta, a 3,35 m sobre la línea de flotación, alojaba sobre todo a las cabinas de la tripulación con su comedor correspondiente. A popa se hallaban los alojamientos de segunda con dos escaleras, una de ella con ascensor, una barbería, una tienda, una habitación donde los músicos guardaban sus instrumentos y a popa del todo los camarotes de 3ª clase con una escalera.

7. **Cubierta F**

   Esta cubierta, a 76 cm por encima de la línea de flotación, era ocupada por el centro por el salón-comedor de tercera clase, junto a su cocina y despensa, con unas escaleras para ingresar a éste desde la cubierta superior.

8. **Cubierta G**

   También conocida como cubierta inferior, ya que se situada - 1,68 m por debajo de la línea de flotación, estaba dividida en dos partes; la de proa y la de popa. Puesto que por el medio se hallaban los huecos que alojaban los grandes motores y las calderas que subían desde la sala de máquinas, dos cubiertas más abajo.

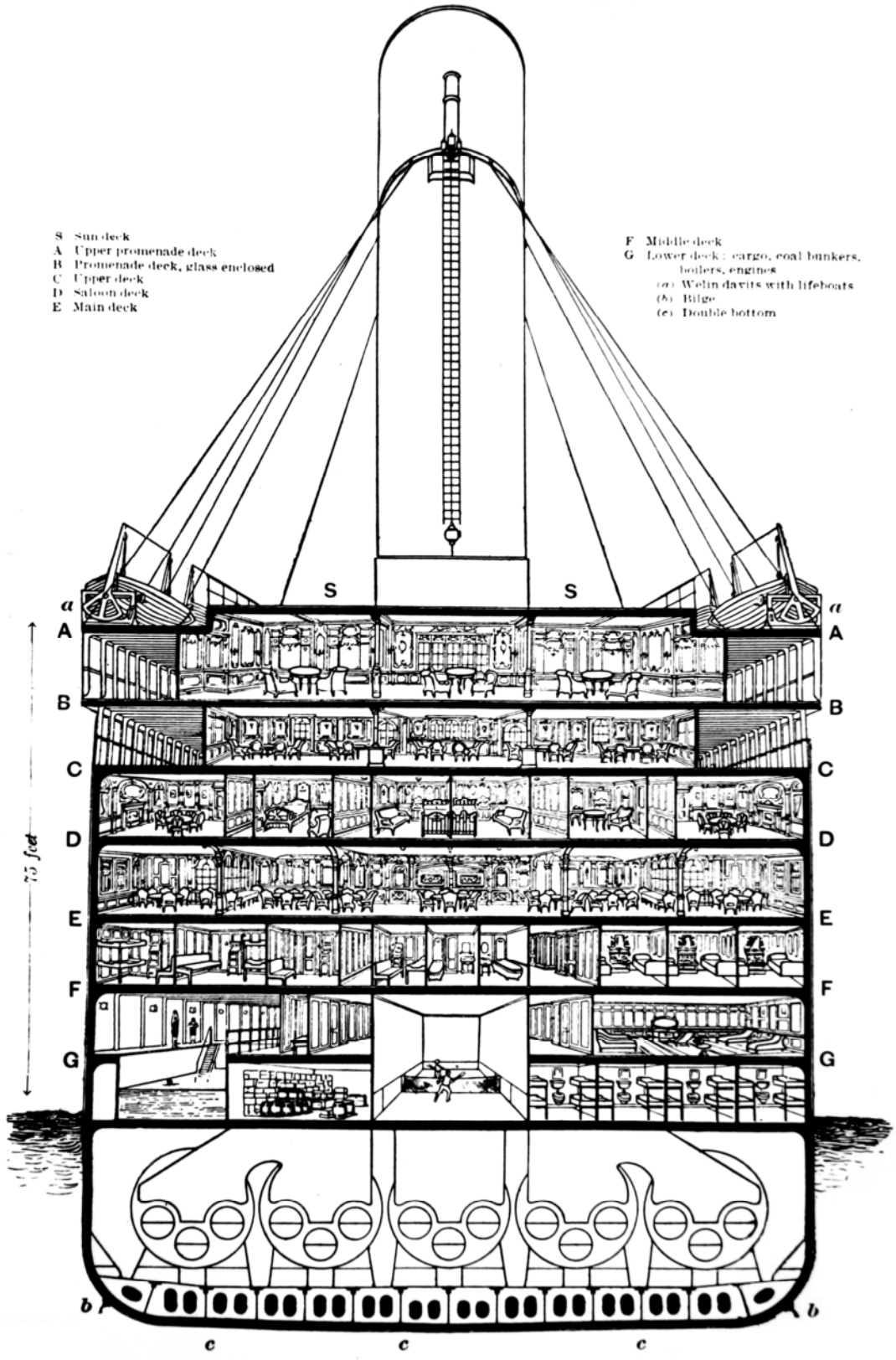


Figura 3.-Sección del centro del transatlántico en la que se pueden ver todas sus cubiertas.

(Obtenido de la enciclopedia Wikipedia [FW13])

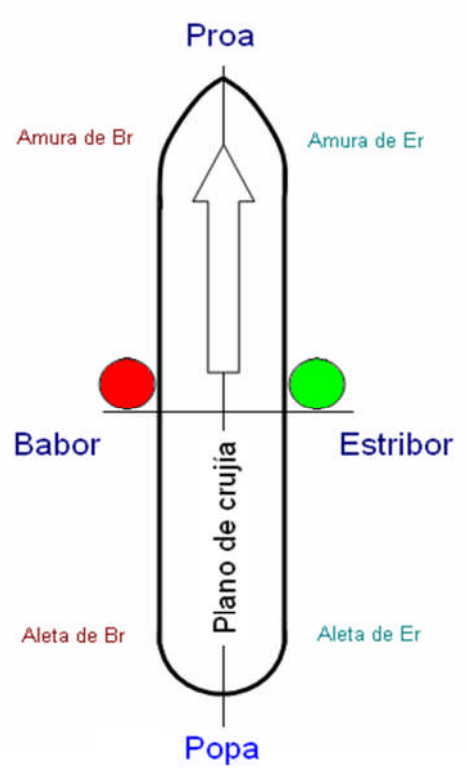


Figura 4.- Partes de un barco

(Obtenido de la enciclopedia Wikipedia [FW13])

3.2. Selección y creación de un conjunto de datos en el que se realizará descubrimiento del conocimiento

En el paso anterior se identificó el problema. En este paso se deben determinar los datos que se utilizarán para el descubrimiento de conocimiento. Si algunos atributos importantes se pierden, entonces todo el estudio puede fallar. Para el éxito del proceso, se deben tener en cuenta el mayor número posible de atributos en esta etapa.

Como pasos necesarios para la selección y la creación de un conjunto de datos en el que se realizará el descubrimiento realizamos las siguientes tareas:

- Creación de una bitácora electrónica para documentar los pasos realizados durante el proceso de descubrimiento del conocimiento (Capitulo 3).
- Pre procesamiento y limpieza de datos de datos con diferentes tipo de software introducidos en esta sección (Capitulo 4).
- Transformación de los datos.
- Procesamiento de diferentes algoritmos de minería de datos con los datos obtenidos del pre-procesamiento de datos (Capitulo 5).
- Evaluación de los resultados de los pasos anteriores.

3.2.1. Identificación de las fuentes de información

La información utilizada en este trabajo proviene de [Ka13] Kaggle Inc. La información obtenida consiste en dos archivos de formato csv (comma separated values). Además se obtuvo información sobre los pasajeros del Titanic, la lista de la tripulación y estadísticas de la Enciclopedia Titánica [ET13].

3.2.2. Descripción de la información disponible

Las variables del archivo (de los datos de entrenamiento) train.csv que contiene 891 registros (que en adelante se denominará **TRAIN),** se muestran en la siguiente tabla:

Tabla 1
Descripción de la variables del archivo train.csv o TRAIN

Nombre de la variable	Descripción de la variable
SURVIVED	ES LA CLASE O LA VARIABLE DEPENDIENTE QUE SE VA HA CLASIFICAR O PREDECIR, CON VALORES DE [0,1] QUE ES 1 SI SOBREVIVIÓ EL PASAJERO Y 0 SI EL PASAJERO NO SOBREVIVIÓ
PCLASS	CON VALORES {1,2,3} ES LA CLASE DE TICKET DONDE 1 ES PRIMERA CLASE, 2 ES SEGUNDA CLASE Y 3 ES TERCERA CLASE
NAME	ES EL NOMBRE DEL PASAJERO
SEX	ES EL SEXO DEL PASAJERO QUE PUEDE SER MALE O FEMALE
AGE	LA EDAD DEL PASAJERO
SIBSP	EL NÚMERO DE HERMANOS/CÓNYUGE QUE TENÍAN A BORDO CON ELLOS
PARCH	QUE ES EL NÚMERO DE PADRES / HIJOS QUE TENÍAN A BORDO CON ELLOS
TICKET	ES EL TIPO DE TICKET
FARE	ES EL MONTO PAGADO POR SUS TICKETS
CABIN	EL NÚMERO DE CABINA
EMBARKED	DONDE SE EMBARCARON (Q: QUEENSTOWN, C:CHERBOURGOR Y S: SOUTHAMPTON).

El otro archivo (de datos de prueba) se llama test.csv y contiene 418 registros. Este archivo, en adelante se denominara **TEST,** tiene las mismas variables que TRAIN.

Esta información será registrada en la bitácora electrónica que es presentada en la siguiente sección, y las subsiguientes operaciones que se lleven a cabo en estas variables como por ejemplo el paso de pre-procesamiento y transformación de datos.

3.2.3. Utilización de una bitácora electrónica o software para el descubrimiento del conocimiento KDD.

La bitácora se desarrolló utilizando la metodología RAD (Rapid Application Development) y es una aplicación Cliente/Servidor. Esta aplicación es una bitácora electrónica en la cual se van registrando los diferentes éxitos y fracasos de las distintas técnicas que se utilizan en la minería de datos, además de las diferentes observaciones o aspectos importantes que sean necesarios documentar. Todo ello se integra dentro de un sistema para poder tener un mejor control de los pasos que se están realizando dentro del descubrimiento del conocimiento.

Las figuras 5 y 6 muestran dos de las pantallas programadas para la aplicación: La pantalla de bienvenida y la pantalla de acceso.

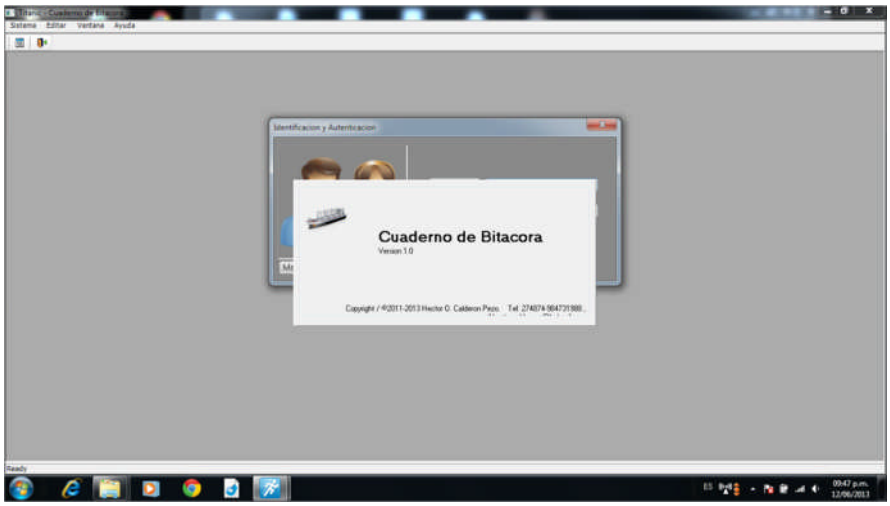


Figura 5.- Pantalla de bienvenida a la aplicación bitácora electrónica para la documentación de los diferentes pasos dentro del proceso de descubrimiento del conocimiento utilizando un DBMS.

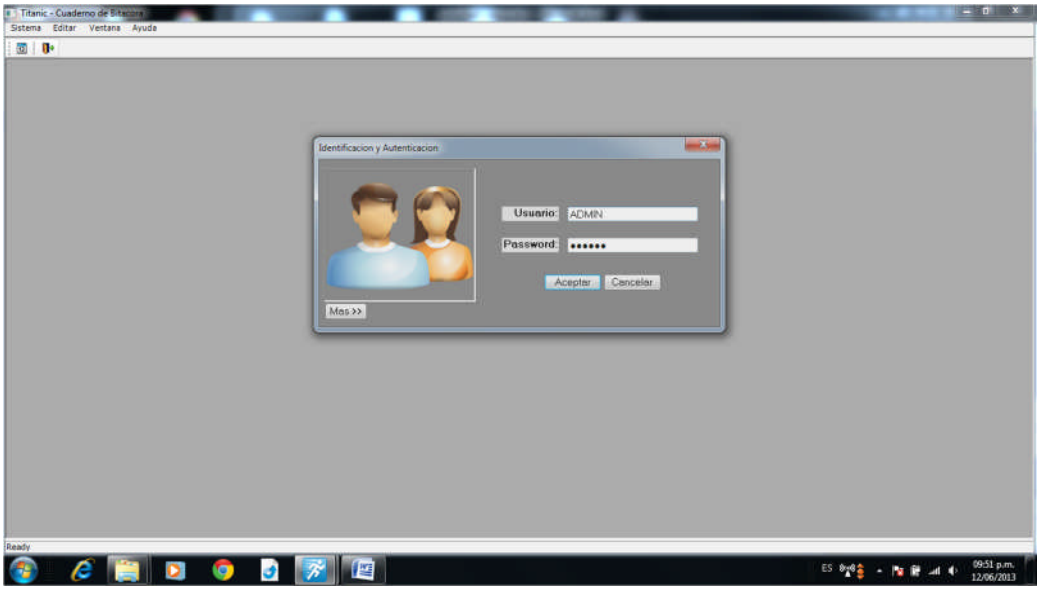


Figura 6.- Pantalla del acceso a la aplicación.

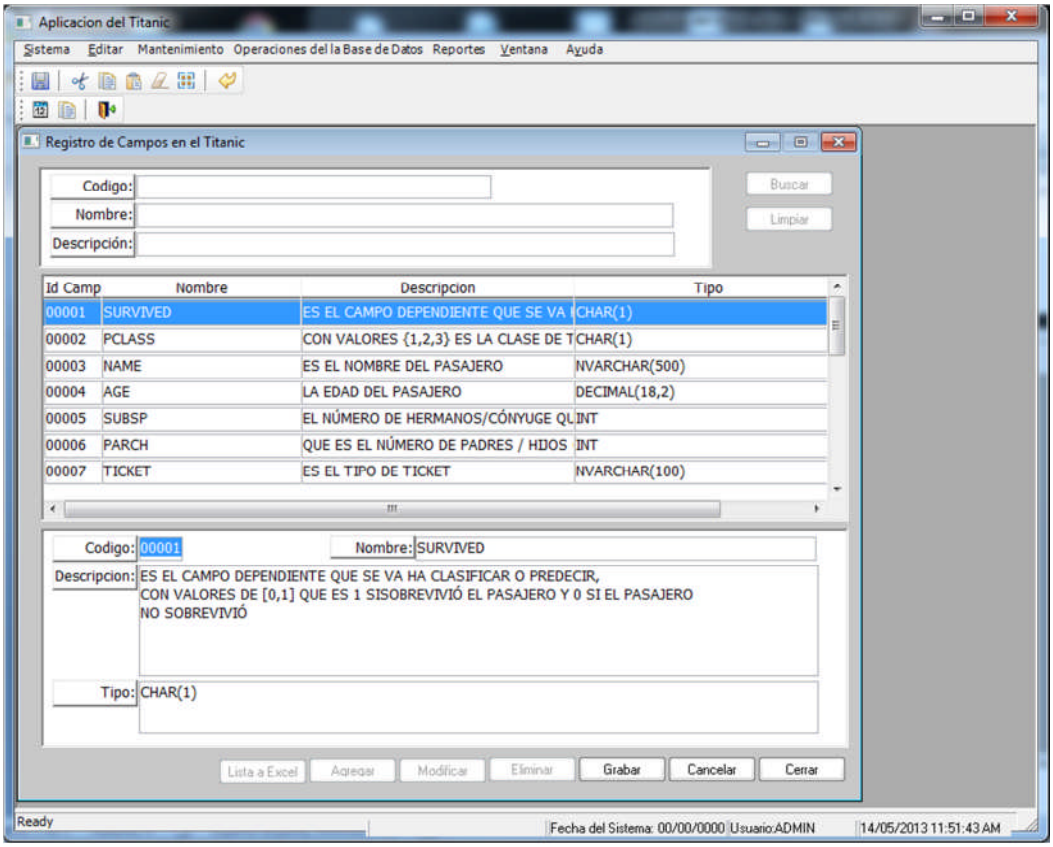


Figura 7.- Pantalla de registro de las variables utilizadas en los algoritmos de la minería de datos.

La ventana de la Figura 7 sirve para registrar las variables utilizadas en la minería de datos, algunas de ellas son producto de la transformación de variables. Esta información puede ser exportada a Excel para un mejor manejo de los datos.

La siguiente ventana sirve para registrar los éxitos y fracasos de las distintas técnicas del aprendizaje automático, además de los más importantes acontecimientos hallados a lo largo de los diferentes pasos del descubrimiento del conocimiento.

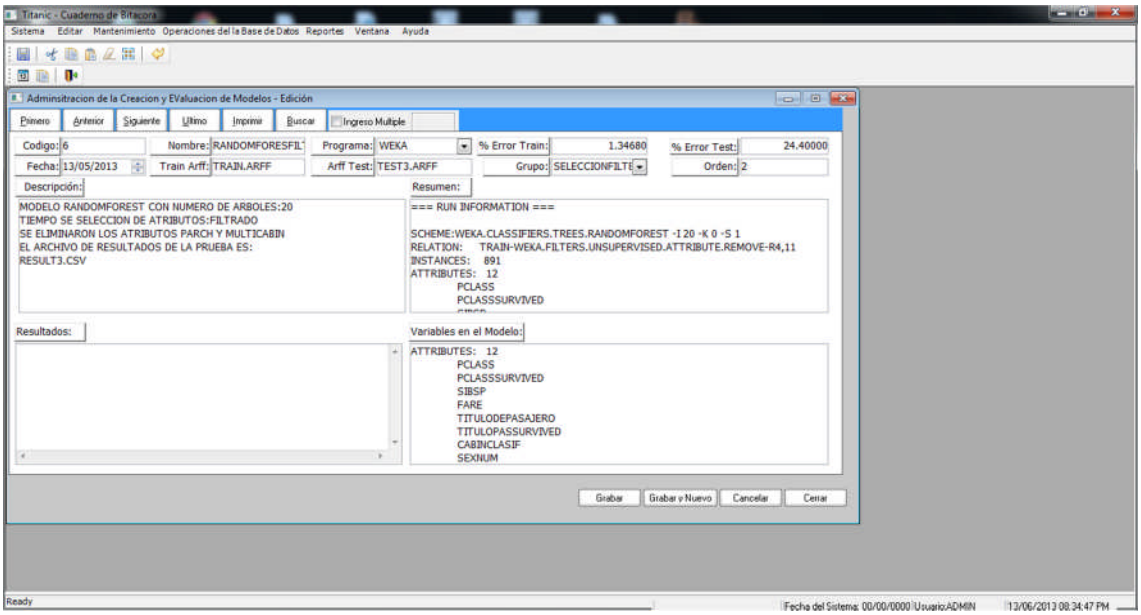


Figura 8.- Ventana del cuaderno de bitácora donde se registran los acontecimientos importantes en el proceso de descubrimiento del conocimiento.

3.2.4. Software de minería de datos utilizado para el proceso de Descubrimiento del Conocimiento

Para el análisis de los datos del problema se utilizará el siguiente software.

3.2.4.1 Software Estadístico R

R es un lenguaje y entorno de programación para análisis estadístico y gráfico. Se trata de un proyecto de software libre, resultado de la implementación GNU del premiado lenguaje S.

Dentro de R, se utilizará la librería Rattle para el proceso de descubrimiento del conocimiento. Rattle es una herramienta fácil de usar, rápida de implementar, y permite trabajar con rapidez a través del procesamiento de datos, modelado, y fases de evaluación de un proyecto de minería de datos.

3.2.4.2. Python

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis muy limpia y que favorezca un código legible. Usa tipado dinámico y es multiplataforma. El shell de Python, que es Ipython, añade funcionalidades como el resaltado de líneas y errores mediante colores, una sintaxis adicional para el shell, autocompletado mediante tabulador de variables, módulos y atributos; entre otras funcionalidades. Este shell es un componente del paquete Scipy.

En este trabajo se utilizara el Notebook IPython que es un entorno computacional interactivo basado en la web donde se puede combinar la ejecución de código, texto, matemáticas, gráficos y multimedia en un solo documento.

Estos cuadernos son archivos normales que se pueden compartir con otras personas, convertidos a otros formatos, como HTML o PDF, etc.

Para la minería de datos, junto con Python se utilizarán otras dependencias que son las siguientes:

a) NumPy.- NumPy es el paquete fundamental para la computación científica con Python.
b) Pandas.- pandas es una librería escrita en el lenguaje de programación Python para la manipulación de datos y el análisis.
c) SciKit-Learn.- Este módulo de aprendizaje automático ofrece herramientas versátiles para la minería de datos y análisis en cualquier campo de la ciencia y la ingeniería.
d) SciPy.- Es un software de código abierto para las matemáticas, la ciencia y la ingeniería para Python.
e) StatsModels.- Statsmodels es un módulo de Python que permite explorar datos, estimar los modelos estadísticos, y realizar pruebas estadísticas. Una extensa lista de estadísticas descriptivas, análisis estadísticos, funciones gráficas y estadísticas de resultados están disponibles para los diferentes tipos de datos y cada estimador.
f) Patsy.- Es un paquete de Python para la descripción de los modelos estadísticos y para la construcción de matrices de diseño.
g) Matplotlib.- matplotlib es una librería de trazado 2D python que produce figuras de calidad de publicación en una variedad de formatos impresos y entornos interactivos a través de plataformas.

3.2.4.3. WEKA

Es una plataforma de software para aprendizaje automático y minería de datos escrito en Java y desarrollado en la Universidad de Waikato. Weka es un software libre distribuido bajo licencia GNU-GPL.

Es una plataforma de software para aprendizaje automático y minería de datos escrita en Java y fue desarrollado en la Universidad de Waikato. Weka es un software libre distribuido bajo licencia GNU-GPL.

El paquete Weka contiene una colección de herramientas de visualización y algoritmos para análisis de datos y modelado predictivo, unidos a una interfaz gráfica de usuario para acceder fácilmente a sus funcionalidades.

Weka soporta varias tareas estándar de minería de datos, especialmente, pre procesamiento de datos, clustering, clasificación, regresión, visualización, y selección. Todas las técnicas de Weka se fundamentan en la asunción de que los datos están disponibles en un fichero plano (flat file) o una relación, en la que cada registro de datos está descrito por un número fijo de atributos (normalmente numéricos o nominales aunque también se soportan otros tipos).

Capítulo 4

Pre-procesamiento y transformación de los datos

Los métodos de limpieza de datos se utilizan para "limpiar" llenando los datos de los valores perdidos, suavizando datos ruidosos, identificando o eliminando los valores atípicos, y resolviendo las incoherencias. Los usuarios deben confiar en que los datos estén libres de inconsistencias, ya que si piensan que los datos están sucios, es poco probable que confíen en los resultados de minería de datos que se está aplicando.

En este capítulo describimos todo el proceso previo de análisis de los datos del problema, identificación de sus propiedades, caracterización de cada una de las variables y test estadísticos para comprobar las distribuciones de los datos entre los conjuntos de aprendizaje (TRAIN) y validación (TEST).

En la parte de transformación de los datos se utilizarán técnicas de reducción de dimensionalidad las cuales se dividen en selección de características, y transformación de características (también llamadas descubrimiento de características). En el primer caso se selecciona sólo un subconjunto de las características originales, mientras que el segundo enfoque se basa en la generación de características completamente nuevas a partir de las ya existentes.

4.1. Análisis descriptivo de datos

Previo a la limpieza de los datos se realizará el resumen descriptivo de los datos, pues este paso es importante para que el pre-procesamiento de datos tenga éxito. Aquí se utilizarán diferentes técnicas de resumen descriptivo de datos para poder identificar las propiedades típicas de los datos. Para el análisis descriptivo se utilizará el software R

Statistic para el resumen descriptivo de los datos y sus correspondientes gráficos.

Se utilizó el paquete Hmisc de R para las estadísticas descriptivas y su librería, así como el paquete gmodels con su librería para las tablas cruzadas. Para los histogramas se usó el paquete ggplot2, y también se usó el comando barplot de R para los gráficos de barras.

4.1.1. Análisis descriptivo de los datos de entrenamiento TRAIN

A continuación se presentan los análisis realizados a los datos del archivo train.csv, con todas las variables de la Tabla 1, el número de observaciones en total es de 891, con nueve variables.

4.1.1.1. Variable PCLASS

Tabla 2
Resumen descriptivo de la variable PCLASS

# de observaciones				Valores perdidos				Clases			
891				0				3			
Clase del pasajero											
Primera Clase				Segunda Clase				Tercera Clase			
# Pasaj.		% Pasaj.		# Pasaj.		% Pasaj.		# Pasaj.		% Pasaj.	
216		24%		184		21%		491		55%	
Sobrev.		No Sobrev.		Sobrev.		No Sobrev.		Sobrev.		No Sobrev.	
#	%	#	%	#	%	#	%	#	%	#	%
136	62.9	80	37.03	87	47.3	97	52.7	119	24.2	372	75.8

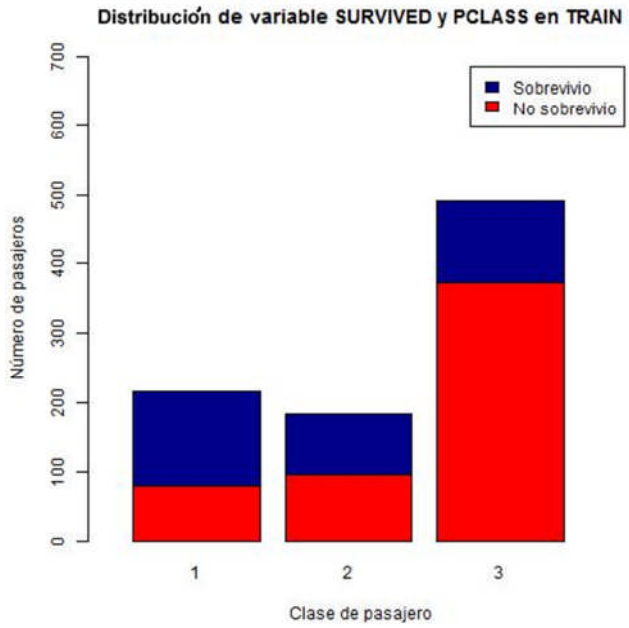
El gráfico de barras de la distribución de las variables SURVIVED con la variable PCLASS se muestra en la Figura 9.

En dicho gráfico se observa que mayormente sobreviven los pasajeros de primera clase con un 62.9%-

Los de segunda clase sobreviven con un 47.3% que se aproxima a los que no sobrevivieron que fue de un 52.7%.

En los de tercera clase mayormente no sobrevivieron pues solo sobrevivió un 24.2% y un 75.8% no sobrevivió.

Figura 9



4.1.1.2. Variable NAME

En este caso se crea una variable transformada reducida de nombre TITULOPASAJERO a partir de la variable NAME. La transformación se realiza extrayendo el título del pasajero como valor representativo de la variable NAME, con la metodología de "Extracción de Características".

Tabla 3

Descripción de la variable TITULOPASAJERO derivada de NAME

Nombre del rasgo	Regla de extracción de rasgo
TITULOPASAJERO	DERIVADA DE LA VARIABLE NAME, CON LOS SIGUIENTES VALORES CONSIDERANDO EL TITULO DEL PASAJERO: CUANDO EL TITULO ES 'MR.' LUEGO EL VALOR ES 1 CUANDO EL TITULO ES 'DON.' LUEGO EL VALOR ES 2 CUANDO EL TITULO ES 'SIR.' LUEGO EL VALOR ES 3 CUANDO EL TITULO ES 'JONKHEER.' LUEGO EL VALOR ES 4 CUANDO EL TITULO ES 'REV.' LUEGO EL VALOR ES 5 CUANDO EL TITULO ES 'DR.' LUEGO EL VALOR ES 6 CUANDO EL TITULO ES 'MAJOR.' LUEGO EL VALOR ES 7 CUANDO EL TITULO ES 'CAPT.' LUEGO EL VALOR ES 8 CUANDO EL TITULO ES 'COL.' LUEGO EL VALOR ES 9 CUANDO EL TITULO ES 'MRS' O ES 'DONA' O ES 'LADY' LUEGO EL VALOR ES 10 CUANDO EL TITULO ES 'MME.' LUEGO EL VALOR ES 11 CUANDO EL TITULO ES 'COUNTESS.' LUEGO EL VALOR ES 12 CUANDO EL TITULO ES 'MS.' LUEGO EL VALOR ES 13 CUANDO EL TITULO ES 'MISS.' LUEGO EL VALOR ES 14 CUANDO EL TITULO ES 'MLLE.' LUEGO EL VALOR ES 15 CUANDO EL TITULO ES 'MASTER.' LUEGO EL VALOR ES 16

Tabla 4

Resumen descriptivo de la variable TITULOPASAJERO

# de observaciones	Valores perdidos	Clases
891	0	16

Tabla 5

Tabla de frecuencias de la variable TITULOPASAJERO

Clases	1		2		3		4		5		6		7		8	
Frecuencia	517		1		1		1		6		7		2		1	
Sobrevivió	Si	No	Si	No	Si	No	Si	No	Si	No	Si	No	Si	No	Si	No
Frecuencia	81	436	0	1	1	0	0	1	0	6	3	4	1	1	0	1
%	58		0		0		0		1		1		0		0	

Clases	9		10		11		12		13		14		15		16	
Frecuencia	2		126		1		1		1		182		2		40	
Sobrevivió	Si	No	Si	No	Si	No	Si	No	Si	No	Si	No	Si	No	Si	No
Frecuencia	1	1	100	26	1	0	1	0	1	0	127	55	2	0	23	17
%	0		14		0		0		0		20		0		4	

En las Tabla 5, se observa que los pasajeros con título correspondiente al sexo femenino (valores 10, 11, 12, 13, 14 y 15) tienen tasas de sobrevivencia mayores a la de los varones (valores 1, 2, 3, 4, 5, 6, ,7, 8 7 9), y los correspondientes a los niños cuyo título o tratamiento es Master (el valor 16, que tienen edades menores a 13 años en los datos TRAIN y TEST). Su tasa de sobrevivencia es ligeramente mayor a la de no sobrevivencia (23 sobreviven y 17 no sobreviven). Su grafica está en la Figura 10.

4.1.1.3. Variable SEX

Tabla 6

Resumen descriptivo de la variable SEX

# de observaciones				Valores perdidos				Clases
891				0				2
Sexo								
Masculino				**Femenino**				
# Pasaj.		**% Pasaj.**		**# Pasaj.**		**% Pasaj.**		
577		65%		314		35%		
Sobrev.		**No Sobrev.**		**Sobrev.**		**No Sobrev.**		
#	**%**	**#**	**%**	**#**	**%**	**#**	**%**	
109	18.9	468	81.1	233	74.2	81	25.8	

Respecto a esta variable se observa que mayormente sobrevivieron las mujeres con un 74.2%, y un 25.8% no sobrevivió. En el caso del sexo masculino solo sobrevivió un 18.9% y un 81.1% no sobrevivió. El gráfico describiendo estos resultados se observa en la Figura 11.

Figura 10

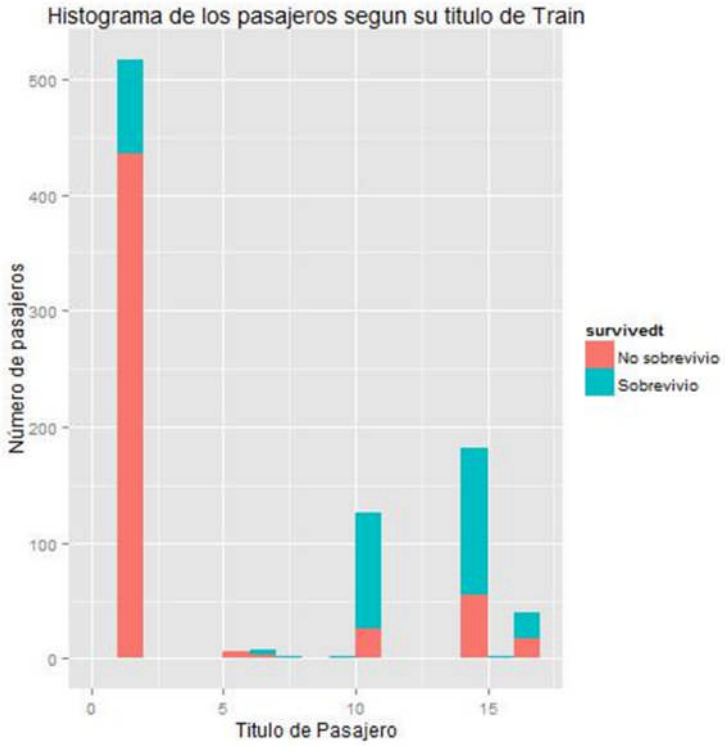
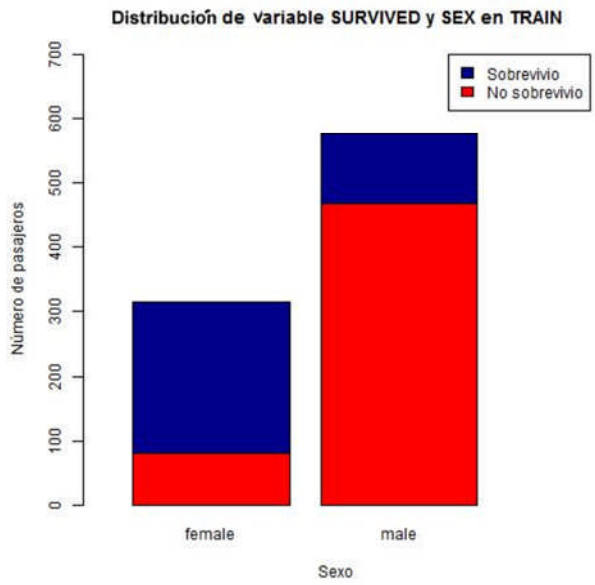


Figura 11



4.1.1.4. Variable AGE

Para este caso se usó el paquete de R "Hmisc" para las estadísticas. También el paquete "gmodels" para la tabla cruzada, y para el gráfico del histograma el paquete "ggplot2" con el comando "gplot".

Tabla 7

Resumen descriptivo de la variable AGE del sexo masculino

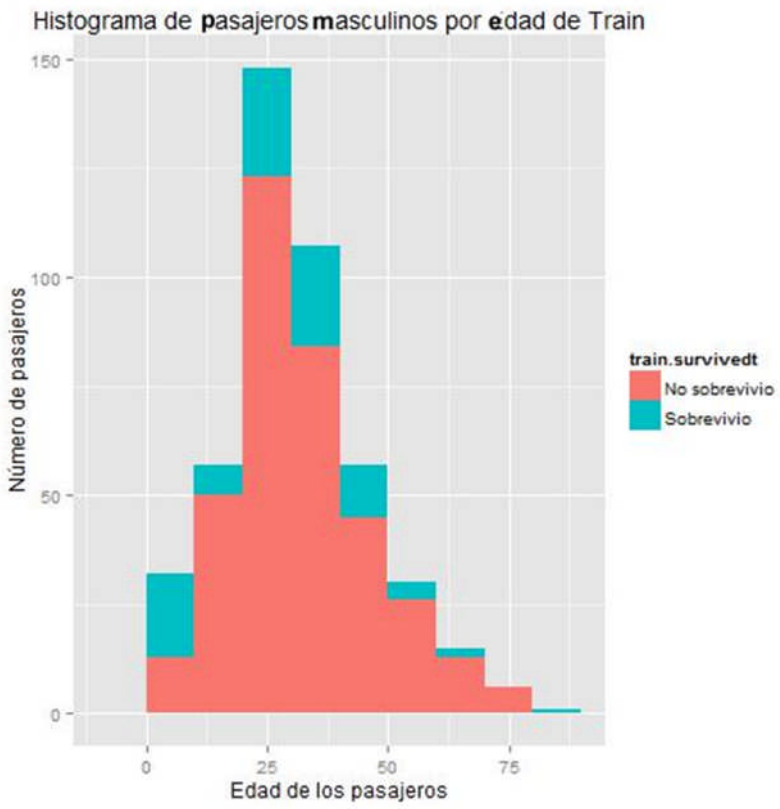
# de observaciones	Valores Perdidos	Media	Clases
453	124	30.73	9

Tabla 8

Tabla cruzada de las clases de la variable AGE vs SURVIVED de sexo masculino

Sobrev.	Clases según años									Total	
	[0-10>	[10-20>	[20-30>	[30-40>	[40-50>	[50-60>	[60-70>	[70-80>	[80>	#	%
No	13	50	123	84	45	26	13	6	0	360	79.5
Si	19	7	25	23	12	4	2	0	1	93	20.5
Frecuencia	32	57	148	107	57	30	15	6	1	453	100
%	7.1	12.6	32.7	23.6	12.6	6.6	3.3	1.3	0.02	100	

Figura 12



De las Tablas 7 y 8 y la Figura 12, se deduce que los pasajeros de sexo masculino mayormente no sobrevivieron en los diferentes rangos o clases de edades. Sin contar los valores perdidos de la variable AGE (que fueron 124), solo sobrevivió el 20.5%. Se puede observar que en el caso de los niños menores de 10 años, la cantidad de sobrevivientes (19) es mayor que la de no sobrevivientes (13). También en el caso de los mayores de 80 años sobrevivió una única persona.

Tabla 9

Resumen descriptivo de la variable AGE del sexo femenino

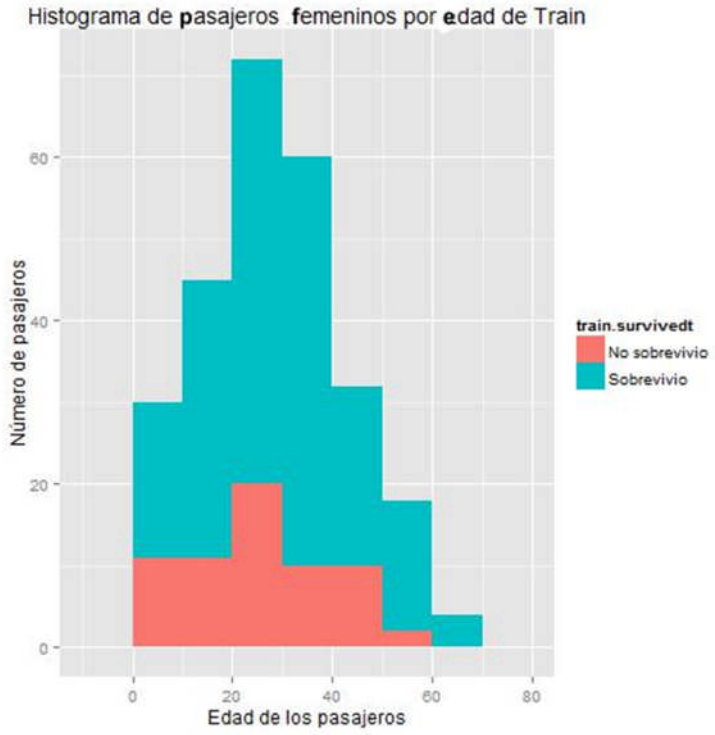
# de observaciones	Valores Perdidos	Media	Clases
261	53	27.92	7

Tabla 10

Tabla cruzada de las clases de la variable AGE vs SURVIVED de sexo femenino

Sobrev.	Clases según años							Total	
	[0-10>	[10-20>	[20-30>	[30-40>	[40-50>	[50-60>	[60-70>	#	%
No	11	11	20	10	10	2	0	64	24.52
Si	19	34	52	50	22	16	4	197	75.48
Frecuencia	30	45	72	60	32	18	4	261	100
%	11.5	17.2	27.6	23	12.3	6.9	1.5	1	

Figura 13



De las Tablas 9 y 10 y la Figura 13, se deduce que las pasajeras mayormente sobrevivieron en los diferentes rangos o clases de edades. Sin contar los valores perdidos de la variable AGE (que fueron 53) sobrevivió el 75.48%. Se puede observar que en el caso de las pasajeras mayores de 60 años todas sobrevivieron, y entre las edades de 50 y 60 años sobrevivieron 16 de 18, que es un 88.8%. En las demás edades se muestra que las tasas de sobrevivencia son mayores que las de los pasajeros masculinos en todos los casos.

4.1.1.5. Variable SIBSP (Número de hermanos/cónyuges)

Tabla 11

Resumen descriptivo de la variable SIBSP del número de hermanos/cónyuges a bordo

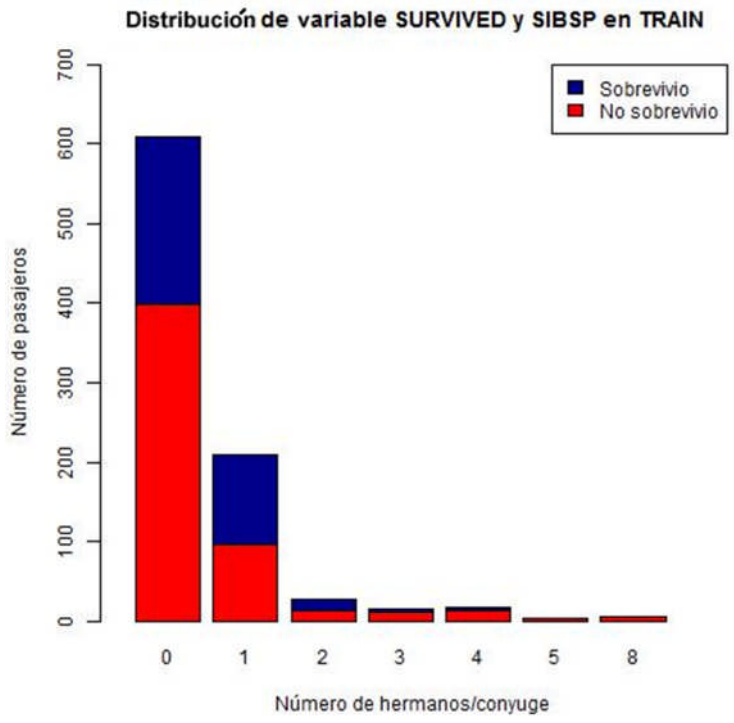
# de observaciones	Valores perdidos	Clases
891	0	7

Tabla 12

Tabla de frecuencias de la variable SIBSP

Clases	0		1		2		3		4		5		8	
Frecuencia	608		209		28		16		18		5		7	
Sobrevivió	Si	No	Si	No	Si	No	Si	No	Si	No	Si	No	Si	No
Frecuencia	210	398	112	97	13	15	4	12	3	15	0	5	0	7
%	68		23		3		2		2		1		1	

Figura 14



En las Tablas 11 y 12, y en la Figura 14, se observa que la mayoría de los pasajeros con hermanos/cónyuges mayores de 3 no sobrevivieron. Este hecho debería tomarse en consideración dentro del descubrimiento del conocimiento.

4.1.1.6. Variable PARCH(Número de padres/hijos)

Tabla 13

Resumen descriptivo de la variable PARCH del número de padres/hijos a bordo

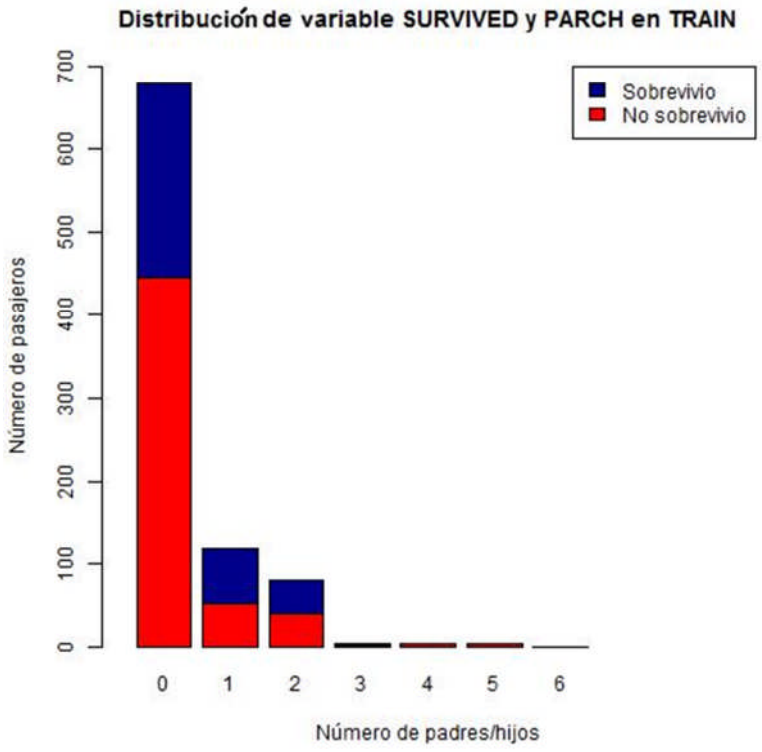
# de observaciones	Valores perdidos	Clases
891	0	7

Tabla 14

Tabla de frecuencias de la variable SIBSP

Clases	0		1		2		3		4		5		6	
Frecuencia	678		118		80		5		4		5		1	
Sobrevivió	Si	No	Si	No	Si	No	Si	No	Si	No	Si	No	Si	No
Frecuencia	233	445	65	53	40	40	3	2	0	4	1	4	0	1
%	73		13		9		1		0		1		0	

Figura 15



En las Tablas 13 y 14, y en la Figura 15, se observa que la mayoría de los pasajeros con padres/hijos mayores de 4 no sobrevivieron. Este hecho debería tomarse en consideración dentro del descubrimiento del conocimiento.

4.1.1.7. Variable TICKET (Ticket del pasajero)

En este caso se crea una variable transformada reducida de nombre TIPOTICKET a partir de la variable TICKET, de acuerdo a las reglas incluidas en la Tabla 15.

Tabla 15
Descripción de la variable TIPOTICKET derivada de TICKET

NOMBRE DE RASGO	CARACTERES CON LOS QUE EMPIEZA LOS VALORES DE LA VARIABLE TICKET DE LA TABLA 1	VALOR DE VARIABLE TIPOTICKET
TIPOTICKET	A./5 OR A.5 OR A/S OR A/5	1
	A/4 OR A4	2
	C	3
	C.A OR CA	4
	F.COR ,F.C.C.	5
	FA OR LINE OR p/PP OR AQ OR A OR 2. OR LP	6
	PC	7
	PP	8
	S.C.	9
	S.O.	10
	S.O.C.	11
	S.O.P.	12
	S.P.	13
	S.W.	14
	SC	15
	SCO	16
	SO/C	17
	SOTON	18
	STON	19
	SW/PP	20
	W./C.	21
	W.E.P.	22
	W/C	23
	WE/P	24
	1	25
	2	26
	3	27
	4	28
	5	29
	6	30
	7	31
	8	32
	9	33

Tabla 16

Resumen descriptivo de la variable TIPOTICKET

# de observaciones	Valores Perdidos	Clases
891	0	33

Tabla 17

Tabla de frecuencias de la variable TIPOTICKET

Clases	1		2		3		4		5		6		7	
Frecuencia	18		7		5		42		6		11		60	
Sobrevivió	Si	No	Si	No	Si	No	Si	No	Si	No	Si	No	Si	No
Frecuencia	2	16	0	7	2	3	14	28	4	2	2	9	39	21
%	2		0.8		0.6		4.7		0.7		1.2		6.7	

Clases	8		9		10		11		12		13		14	
Frecuencia	3		3		3		5		1		1		1	
Sobrevivió	Si	No	Si	No	Si	No	Si	No	Si	No	Si	No	Si	No
Frecuencia	2	1	1	2	0	3	0	5	0	1	0	1	1	0
%	0.3		0.3		0.3		0.6		0.1		0.1		0.1	

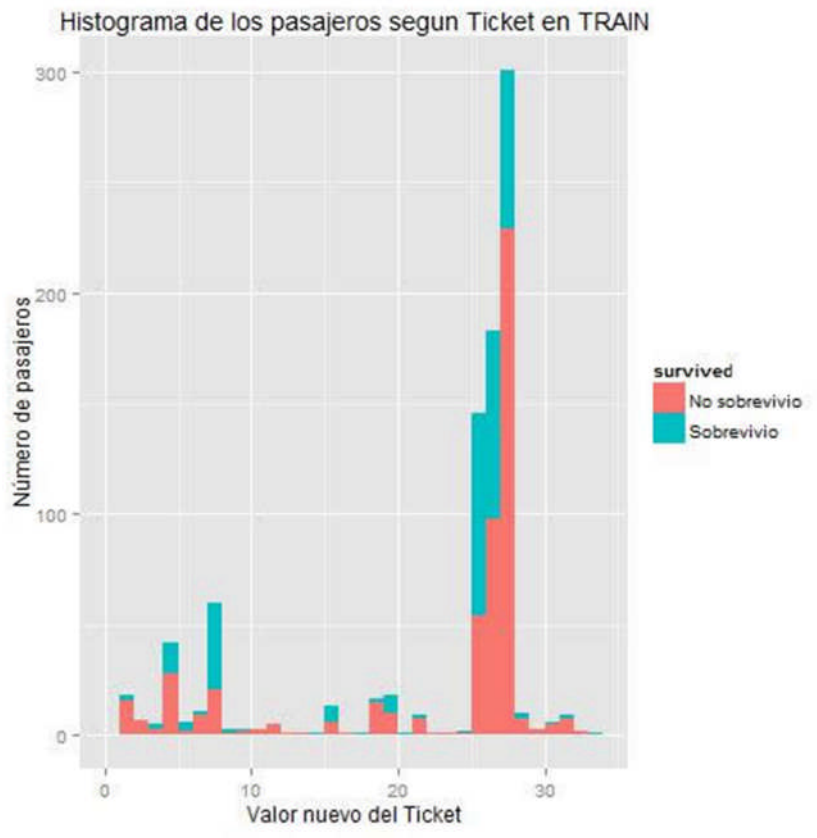
Clases	15		16		17		18		19		20		21	
Frecuencia	13		1		1		17		18		1		9	
Sobrevivió	Si	No	Si	No	Si	No	Si	No	Si	No	Si	No	Si	No
Frecuencia	7	6	0	1	1	0	2	15	8	19	1	0	1	8
%	1.5		0.1		0.1		1.9		2		0.1		1	

Clases	22		23		24		25		26		27		28	
Frecuencia	1		1		2		146		183		301		10	
Sobrevivió	Si	No	Si	No	Si	No	Si	No	Si	No	Si	No	Si	No
Frecuencia	0	1	0	1	1	1	92	54	85	98	72	229	2	8
%	0.1		0.1		0.2		16.4		20.5		33.8		1.1	

Clases	29		30		31		32		33	
Frecuencia	3		6		9		2		1	
Sobrevivió	Si	No	Si	No	Si	No	Si	No	Si	No
Frecuencia	0	3	1	5	1	8	0	2	1	0
%	0.3		0.7		1		0.2		0.1	

En las Tablas 16 y 17, y en la Figura 16, se observa que los pasajeros con tickets que empiezan con 4, 5, 6, 7, 8 y 9 (para “valores nuevos” de la variable Ticket mayores o iguales a 28), tienen más probabilidades de no sobrevivir. Este hecho debería tomarse en consideración dentro de descubrimiento del conocimiento.

Figura 16



4.1.1.8. Variable FARE(Monto pagado por los tickets)

Tabla 18

Resumen descriptivo de la variable FARE de monto pagado por los tickets

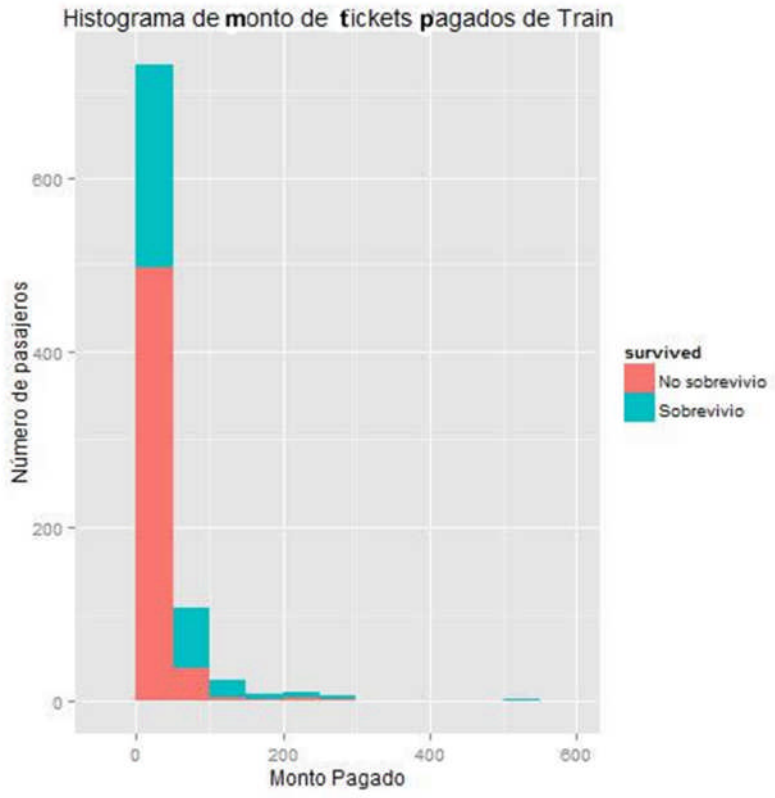
# obs.	Valores perdidos	Media	Clases
891	0	32.2	7

Tabla 19

Tabla cruzada de las clases de la variable FARE vs SURVIVED

Sobrev.	Clases según Monto Pagado							Total	
	[0-50>	[50-100>	[100-150>	[150-200>	[200-250>	[250-300>	[300-350>	#	%
No	497	38	5	3	4	2	0	549	61.62
Si	233	70	19	6	7	4	3	342	38.38
Frec.	730	108	24	9	11	6	3	891	100
%	82	12	3	1	1	1	0	100	

Figura 17



En las Tablas 18 y 19, y en la Figura 17, no se observa patrones que haya que resaltar en el descubrimiento del conocimiento. No se observa ningún patrón de interés para el proceso de descubrimiento del conocimiento.

4.1.1.9. Variable CABIN(El número de la cabina)

En este caso se hace el supuesto de que la variable CABIN corresponde a las cabinas ubicadas en cada cubierta y que la primera letra (del valor de la variable), se refiere a la cubierta donde está la cabina. Por ejemplo, la cabina "A1" estaría en la cubierta "A", la cabina "B1" estaría en la cubierta B y así sucesivamente, donde las cubiertas se muestran en la sección 3.1.1. Utilizando **R** se creó una nueva variable RANGECABIN, que contendría las cubiertas del barco según los valores de la variable CABIN. Los siguientes resultados se refieren a la variable RANGECABIN. A continuación se presenta la descripción de esta variable.

Tabla 20

Descripción de la variable RANGECABIN derivada de CABIN

Nombre del rasgo	Regla de extracción del rasgo
RANGECABIN	DERIVADA DE CABIN TIENE LOS VALORES: A, B, C, D, E, F, G SEGÚN EN QUE CUBIERTA ESTE LA CABINA. SI EL VALOR DE CABIN ES NULO O T, EL VALOR DE RANGECABIN ES T.

Tabla 21

Resumen descriptivo de la variable RANGECABIN de las cubiertas del Titanic

# de observaciones	Valores perdidos	Clases
891	0	8

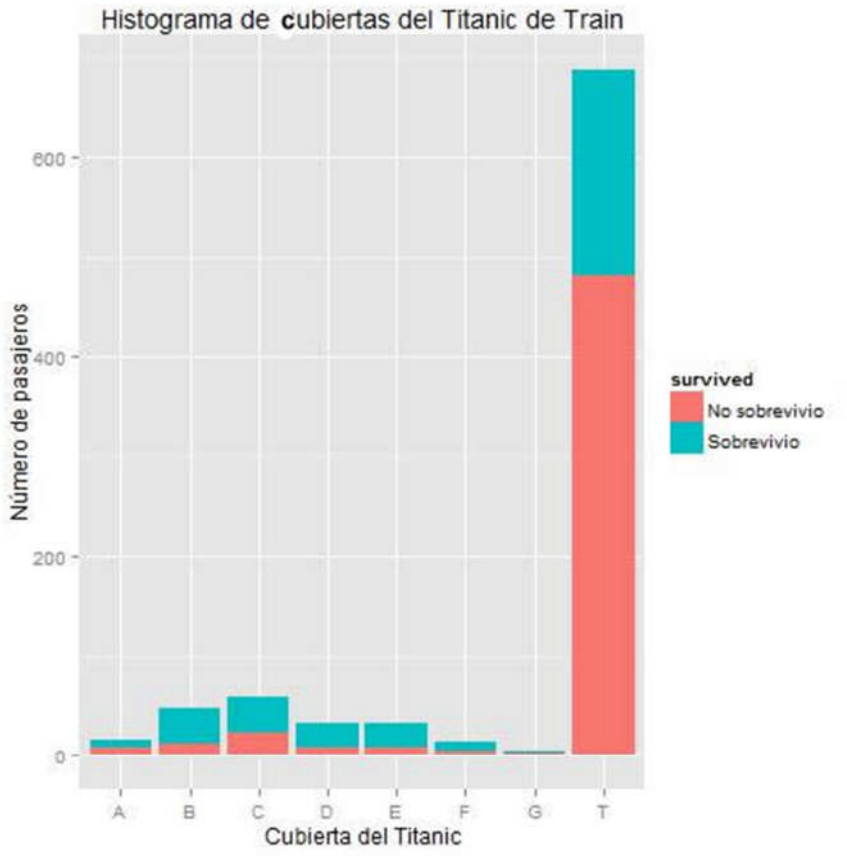
Tabla 22

Tabla de frecuencias de la variable RANGECABIN

Clases	A		B		C		D		E		F		G		T	
Frecuencia	15		47		59		33		32		13		4		688	
Sobrevivió	Si	No	Si	No	Si	No	Si	No	Si	No	Si	No	Si	No	Si	No
Frecuencia	7	8	35	12	35	24	25	8	24	8	8	5	2	2	206	482
%	2		5		7		4		4		1		0		77	

No se observa en las Tablas 21 y 22, y en la Figura 18, ningún patrón destacable en los datos a ser considerado en el proceso de descubrimiento de conocimiento.

Figura 18



4.1.1.10. Variable EMBARKED (En qué lugar se embarcaron los pasajeros)

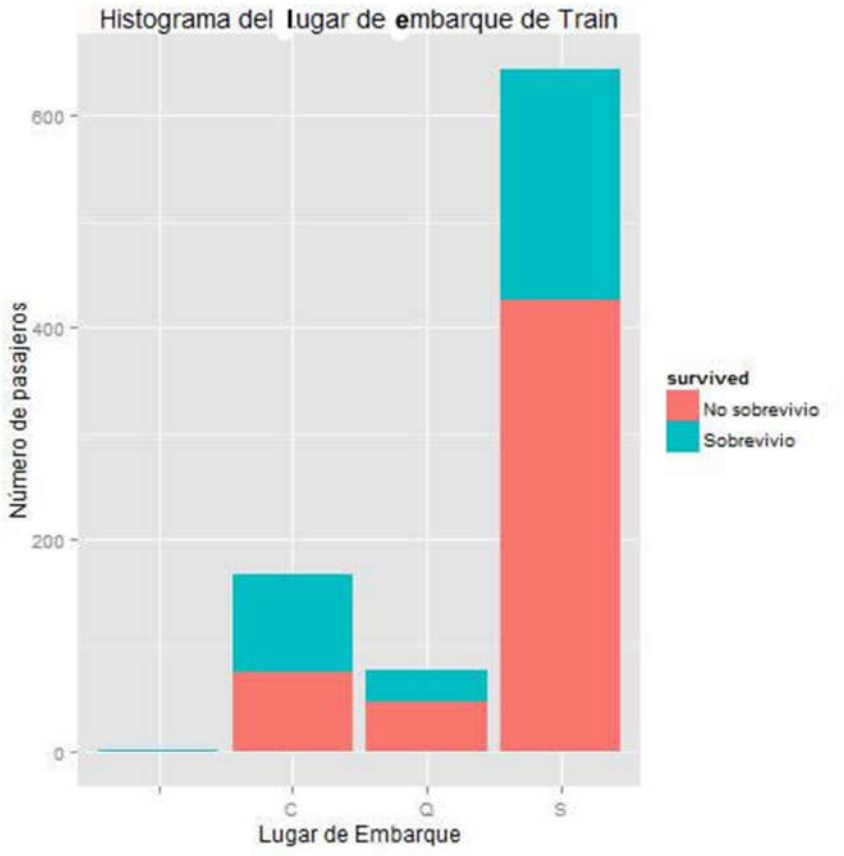
Tabla 23

Resumen descriptivo de la variable EMBARKED

# de observaciones				Valores Perdidos				Clases			
891				2				3			
Lugar de embarque											
Cherbourg C				Queenston Q				Southampton S			
# Pasaj.		% Pasaj.		# Pasaj.		% Pasaj.		# Pasaj.		% Pasaj.	
168		19%		77		9%		644		72%	
Sobrev.		No Sobrev.		Sobrev.		No Sobrev.		Sobrev.		No Sobrev.	
#	%	#	%	#	%	#	%	#	%	#	%
93	55.3	75	44.7	30	39	47	61	217	33.7	427	66.3

En las Tablas 21 y 22, y en la Figura 19, se observa que no hay ningún patrón particular destacable en los datos, respecto a los lugares de embarque de los pasajeros, que deba considerarse dentro del descubrimiento del conocimiento.

Figura 19



4.1.2. Análisis descriptivo de los datos de prueba TEST

A continuación se presentan los análisis realizados a los datos del archivo test.csv, donde la variable SURVIVED de la Tabla 1 no es considerada en los análisis. El número de observaciones en total es de 418, con ocho variables.

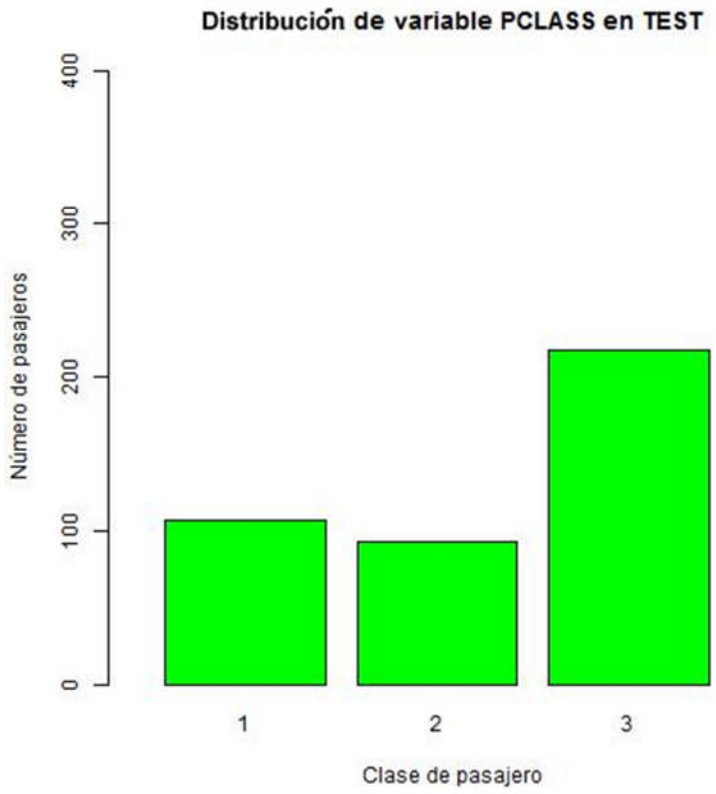
4.1.2.1. Variable PCLASS

Tabla 24
Resumen descriptivo de la variable PCLASS

# de observaciones		Valores Perdidos		Clases	
418		0		3	
Clase del pasajero					
Primera Clase		Segunda Clase		Tercera Clase	
# Pasaj.	% Pasaj.	# Pasaj.	% Pasaj.	# Pasaj.	% Pasaj.
107	26%	93	22%	281	52%

El gráfico de barras de distribución de la variable PCLASS se muestra en la Figura 20. Se puede apreciar la semejanza con la Figura 9.

Figura 20



4.1.2.2. Variable NAME

En este caso se crea una variable transformada reducida de nombre TITULOPASAJERO a partir de la variable NAME. El título del pasajero se utiliza como valor representativo de la variable NAME.

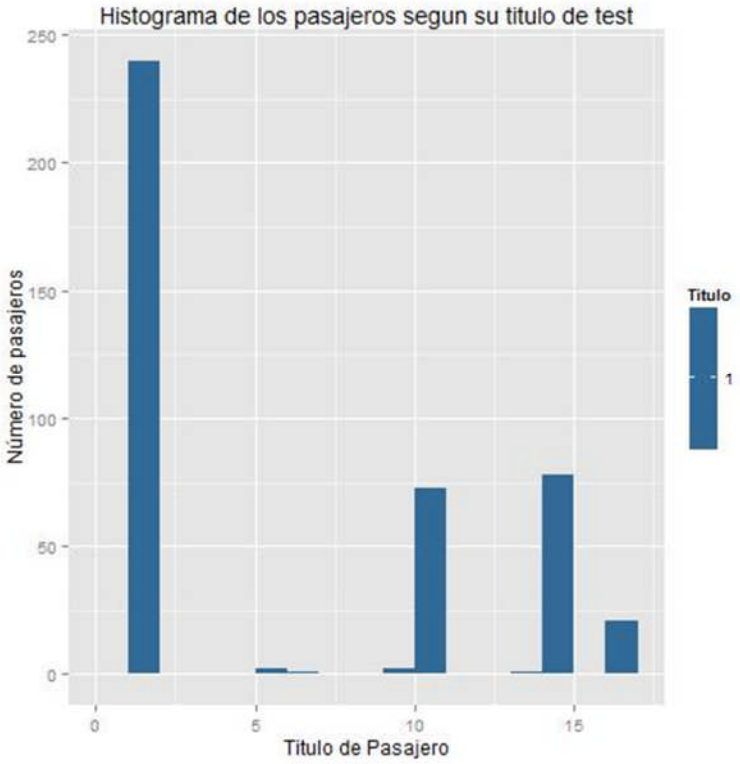
Tabla 25

Tabla de frecuencias de la variable TITULOPASAJERO

Clases	1	2	3	4	5	6	7	8
Frecuencia	240	0	0	0	2	1	0	0
%	57	0	0	0	0	0	0	0

Clases	9	10	11	12	13	14	15	16
Frecuencia	2	73	0	0	1	78	0	21
%	0	17	0	0	0	19	0	5

Figura 21



Se puede apreciar la semejanza con la Figura 10.

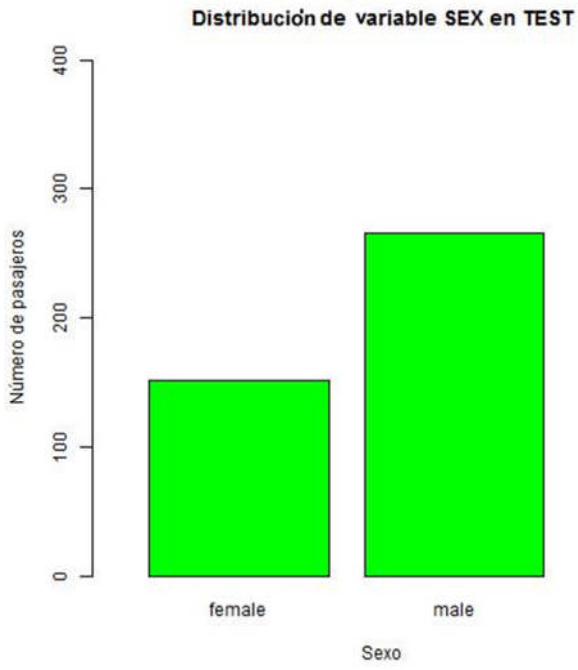
4.1.2.3. Variable SEX

Tabla 26
Resumen descriptivo de la variable SEX

# de observaciones		Valores perdidos		Clases
418		0		2
Sexo				
Masculino		Femenino		
# Pasaj.	% Pasaj.	# Pasaj.	% Pasaj.	
266	64%	152	36%	

Observando la Figura 22, se puede apreciar la semejanza con la Figura 11.

Figura 22



4.1.2.4. Variable AGE

Para este caso se usó el paquete de R “Hmisc” para el análisis estadístico. También el paquete “gmodels” para la tabla cruzada, y para el gráfico del histograma el paquete “ggplot2” con el comando “gplot”.

Tabla 27

Resumen descriptivo de la variable AGE del sexo masculino

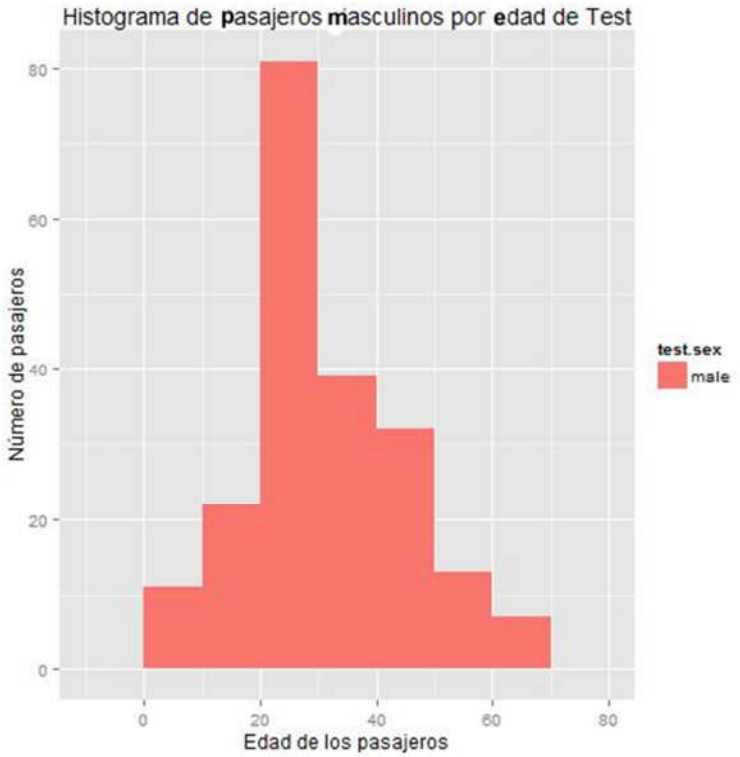
# de observaciones	Valores perdidos	Media	Clases
205	61	30.27	7

Tabla 28

Tabla cruzada de las clases de la variable AGE de sexo masculino

	Clases de edad							Total
	[0-10>	[10-20>	[20-30>	[30-40>	[40-50>	[50-60>	[60-70>	
Frecuencia	11	22	81	39	32	13	7	205
%	5	11	40	19	16	6	3	100

Figura 23



Para el sexo masculino, se puede apreciar la semejanza con la Figura 12.

Tabla 29

Resumen descriptivo de la variable AGE del sexo femenino

# de observaciones	Valores perdidos	Media	Clases
127	25	30.27	8

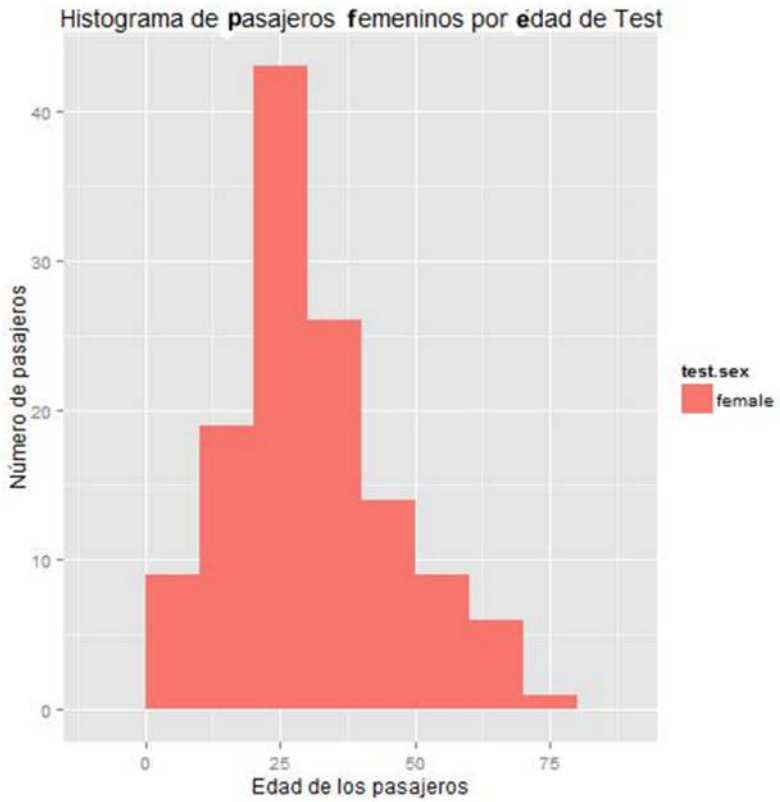
Tabla 30

Tabla cruzada de las clases de la variable AGE vs SURVIVED de sexo femenino

	Clase de Edad								Total
	[0-10>	[10-20>	[20-30>	[30-40>	[40-50>	[50-60>	[60-70>	[70-80>	
Frec.	9	19	43	26	14	9	6	1	127
%	7	15	34	20	11	7	5	1	100

Para el sexo femenino, observando la Figura 24 se puede apreciar la semejanza con la Figura 13.

Figura 24



4.1.2.5. Variable SIBSP (Número de hermanos/cónyuges)

Tabla 31

Resumen de la variable SIBSP del número de hermanos/cónyuges a bordo

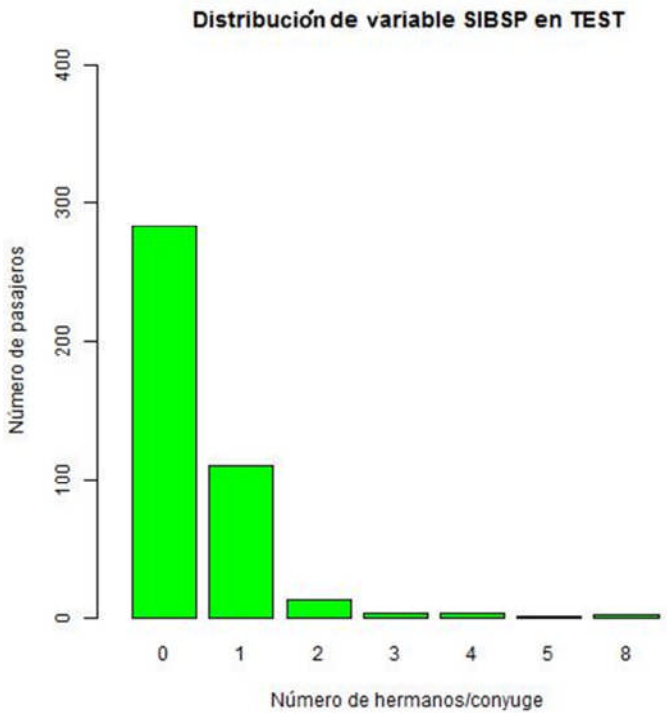
# de observaciones	Valores Perdidos	Clases
418	0	7

Tabla 32

Tabla de frecuencias de la variable SIBSP del sexo femenino

Clases	0	1	2	3	4	5	8
Frecuencia	283	110	14	4	4	1	2
%	68	26	3	1	1	0	0

Figura 25



Se puede apreciar la semejanza con la Figura 14.

4.1.2.6. Variable PARCH(Número de padres/hijos)

Tabla 33

Resumen descriptivo de la variable PARCH del número de padres/hijos a bordo

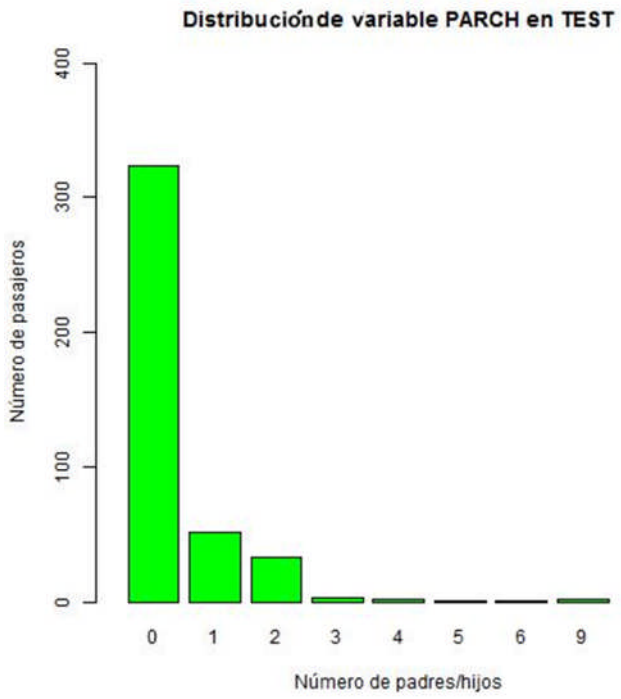
# de observaciones	Valores perdidos	Clases
418	0	8

Tabla 34

Tabla de frecuencias de la variable SIBSP

Clases	0	1	2	3	4	5	6	9
Frecuencia	324	52	33	3	2	1	1	2
%	78	12	8	1	0	0	0	0

Figura 26



Se puede apreciar la semejanza con la Figura 15.

4.1.2.7. Variable TICKET (Ticket del pasajero)

En este caso se crea una variable transformada reducida de nombre TIPOTICKET a partir de la variable TICKET, de acuerdo a la regla de la Tabla 15.

Tabla 35

Resumen descriptivo de la variable TIPOTICKET

# de observaciones	Valores perdidos	Clases
418	0	33

Tabla 36

Tabla de frecuencias de la variable TIPOTICKET

Clases	1	2	3	4	5	6	7
Frecuencia	5	3	3	27	6	11	32
%	1.2	0.7	0.7	6.5	0.7	1.4	7.7

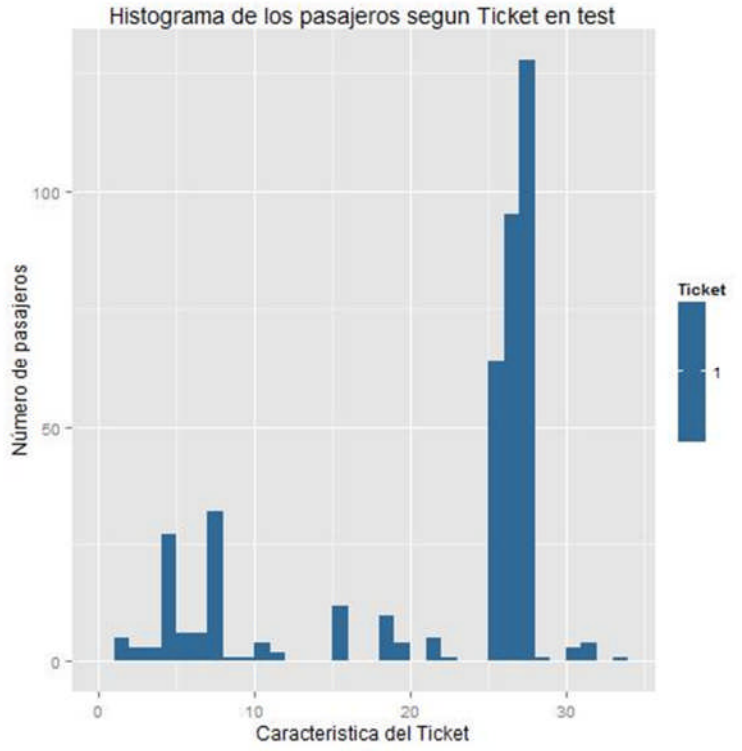
Clases	8	9	10	11	12	13	14
1	3	1	4	2	0	0	0
%	0.2	0.2	1	0.5	0	0	0

Clases	15	16	17	18	19	20	21
Frecuencia	12	0	0	10	4	0	5
%	2.9	0	0	2.4	1	0	1.2

Clases	22	23	24	25	26	27	28
Frecuencia	1	0	0	64	95	128	1
%	0.2	0	0	15.3	22.7	30.6	0.2

Clases	29	30	31	32	33
Frecuencia	0	3	4	0	1
%	0	0.7	1	0	0.1

Figura 27



Se puede apreciar la semejanza con la Figura 16.

4.1.2.8. Variable FARE(Monto pagado por los tickets)

Tabla 37

Resumen descriptivo de la variable FARE del monto pagado por los tickets

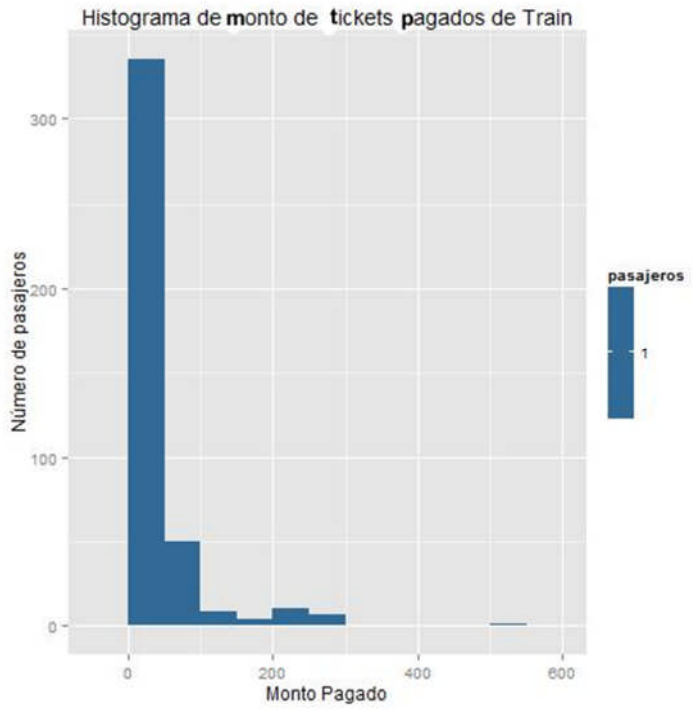
# obs.	Valores perdidos	Media	Clases
417	1	35.63	7

Tabla 38

Tabla cruzada de las clases de la variable FARE vs SURVIVED

	Clases según Monto Pagado							Total
	[0-50>	[50-100>	[100-150>	[150-200>	[200-250>	[250-300>	[500>	
Frec.	336	50	9	4	10	7	1	417
%	81	12	2	1	2	2	0	100

Figura 28



Se puede apreciar la semejanza con la Figura 17.

4.1.2.9. Variable CABIN(El Número de Cabina)

Al igual que en la sumarización de las variables de los datos TRAIN, en este análisis se creó la variable RANGECABIN para los datos TEST de la misma forma que en la sección 4.1.1.9, tal y como se muestra en la Tabla 20.

Tabla 39

Resumen descriptivo de la variable RANGECABIN de las cubiertas del Titanic

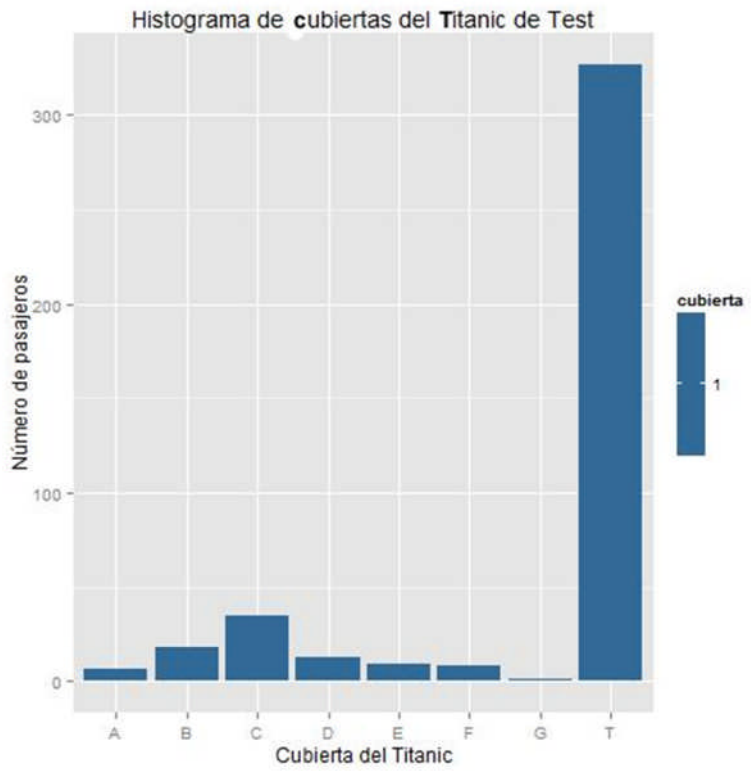
# de observaciones	Valores Perdidos	Clases
418	0	8

Tabla 40

Tabla de frecuencias de la variable RANGECABIN

Clases	A	B	C	D	E	F	G	T
Frecuencia	7	18	35	13	9	8	1	327
%	2	4	8	3	2	2	0	78

Figura 29



Se puede apreciar la semejanza con la Figura 18.

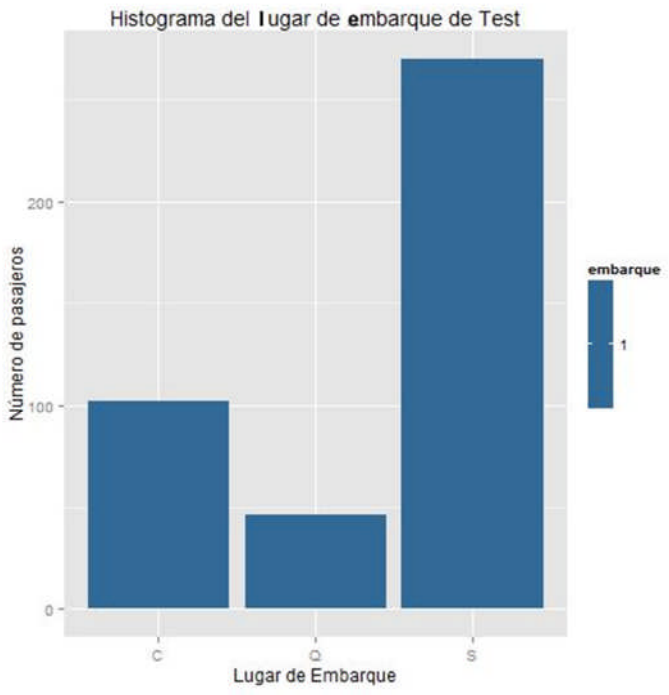
4.1.2.10. Variable EMBARKED (En qué lugar se embarcaron los pasajeros)

Tabla 41

Resumen descriptivo de la variable EMBARKED

# de observaciones		Valores Perdidos		Clases	
418		0		3	
Lugar de embarque					
Cherbourg C		Queenston Q		Southampton S	
# Pasaj.	% Pasaj.	# Pasaj.	% Pasaj.	# Pasaj.	% Pasaj.
102	24%	46	11%	270	65%

Figura 30



Se puede apreciar la semejanza con la Figura 19.

Una conclusión del exhaustivo análisis realizado, es que los conjuntos TRAIN y TEST tienen una distribución de datos muy similar. En la siguiente sección se comprobará la igualdad entre las distribuciones de las dos poblaciones utilizando análisis estadístico.

4.2. Análisis estadístico de los datos

Utilizaremos test estadísticos univariados y multivariados. El caso multivariado considera las correlaciones entre poblaciones por medio de la matriz de covarianza.

4.2.1. Análisis estadístico de una variable

Para el caso del análisis univariado se dividirá el conjunto de variables que son PCLASS, NAME, SEX, SIBSP, PARCH, TICKET, CABIN Y EMBARKED (que son valores numéricos enteros y que son frecuencias) en do grupos, aquí se utilizará el test estadístico univariado *T-Student* pareado para comparar las dos poblaciones que son TRAIN y TEST.

Para el segundo grupo, que son las variables continuas AGE para el género masculino, AGE pare el género femenino, y FARE (el costo de los tickets) se utilizará el test estadístico *t-test Welch de dos muestras* para variables continuas.

Las pruebas estadísticas en el primer grupo de variables se realizaron considerando los porcentajes de las frecuencias respecto al total.

Tabla 42
Test estadísticos t_student pareado

Nº Orden	Variable	Resultado
1	**PCLASS**	datos: trainpclass and testpclass t = 0, df = 2, p-value = 1 intervalo de confianza al 95%: -6.572411 6.572411 t=0 está dentro del intervalo de confianza Se acepta la hipótesis nula que la diferencia de medias de las dos poblaciones es 0.
2	**NAME**	datos: trainname and testname t = 0, df = 15, p-value = 1 intervalo de confianza al 95%: -0.514794 0.514794 t=0 está dentro del intervalo de confianza Se acepta la hipótesis nula que la diferencia de medias de las dos poblaciones es 0.
3	**SEX**	data: trainsex and testsex t = 0, df = 1, p-value = 1 intervalo de confianza al 95%: -12.7062 12.7062 t=0 está dentro del intervalo de confianza Se acepta la hipótesis nula que la diferencia de medias de las dos poblaciones es 0.
4	**SIBSP**	data: trainsibsp and testsibsp t = 0.2582, df = 6, p-value = 0.8049 intervalo de confianza al 95%: -1.210978 1.496693 t=0.2582 está dentro del intervalo de confianza Se acepta la hipótesis nula que la diferencia de medias de las dos poblaciones es 0.
5	PARCH	data: trainparch and testparch t = -0.3568, df = 7, p-value = 0.7318 intervalo de confianza al 95%: -1.907046 1.407046 t=-0.3568 está dentro del intervalo de confianza Se acepta la hipótesis nula que la diferencia de medias de las dos poblaciones es 0.
6	TICKET	data: traintipoticket and testtipoticket t = 0.1158, df = 32, p-value = 0.9086 intervalo de confianza al 95%: -0.3017380 0.3381016 t=0.1158 está dentro del intervalo de confianza Se acepta la hipótesis nula que la diferencia de medias de las dos poblaciones es 0.
7	CABIN	data: traincabin and testcabin t = 0.314, df = 7, p-value = 0.7627 intervalo de confianza al 95%: -0.8163526 1.0663526 t=0.314 está dentro del intervalo de confianza Se acepta la hipótesis nula que la diferencia de medias de las dos

		poblaciones es 0.
8	EMBARKED	data: trainembarked and testembarked t = 0, df = 2, p-value = 1 intervalo de confianza al 95%: -15.51344 15.51344 t=0 está dentro del intervalo de confianza Se acepta la hipótesis nula que la diferencia de medias de las dos poblaciones es 0.

Tabla 43
Test estadísticos t-test Welsh para dos muestras

Numero	Variable	Resultado
1	**AGE PARA POBLACION MASCULINA**	data: trainagemale$age and testagemale$age t = 0.3906, df = 428.962, p-value = 0.6963 intervalo de confianza al 95%: -1.829921 2.737747 t=0.3906 está dentro del intervalo de confianza Se acepta la hipótesis nula que la diferencia de medias de las dos poblaciones es 0.
2	**AGE PARA POBLACION FEMENINA**	data: trainagefemale$age and testagefemale$age t = -1.4512, df = 230.895, p-value = 0.1481 intervalo de confianza al 95%: -5.5562804 0.8429736 t=-1.4512 está dentro del intervalo de confianza Se acepta la hipótesis nula que la diferencia de medias de las dos poblaciones es 0.
3	**FARE**	data: train$fare and test$fare t = -1.0683, df = 733.63, p-value = 0.2858 intervalo de confianza al 95%: -9.713542 2.867580 t=-1.0683 está dentro del intervalo de confianza Se acepta la hipótesis nula que la diferencia de medias de las dos poblaciones es 0.

En las Tabla 42 y 43 las hipótesis que se están comprobando son:

- ✓ Hipótesis Nula:
  Ho: La diferencia verdadera en medias de la variable es igual a 0
- ✓ Hipótesis Alternativa:
  Ha: La diferencia verdadera en medias de la variable no es igual a 0.

A partir de las Tablas 42 y 43, se observa que se han aceptado las hipótesis nulas en todos los casos. Por lo tanto se concluye que con este análisis las dos poblaciones (TRAIN y TEST) son muestras de una misma población original, porque son iguales con un nivel de confianza del 95%.

4.2.2. Análisis estadístico multivariado

En la aproximación multivariada a la comparación de medias de poblaciones, se utilizarán los datos resultantes de la sección 4.3.1.1 (Tabla 46) que se verá más adelante,

correspondiente a la extracción de características donde a partir de los datos originales TRAIN y TEST se obtienen los datos transformados TRAINTRANSF y TESTTRANSF respectivamente.

Con este objetivo se obtendrá información de [CHYX99] (The R Project for Comparisons of Several Multivariate Means), de donde se utilizará el algoritmo para comparar las medias multivariadas de las dos poblaciones.

Luego dados los conjuntos de datos que son los datos de entrenamiento TRAIN de una población 1 con independientes respuestas: X_{11}, X_{12},…, X_{1n1} (n1=891) que se distribuyen aleatoriamente como $N_p(\mu_1, \Sigma)$, (donde μ_1 es el vector de medias de la población 1 y Σ es la matriz de covarianza de la población 1), y la población de prueba TEST que sería la población 2 con independientes respuestas: X_{21}, X_{22},…, X_{2n2} (n2=418) que se distribuye aleatoriamente como $N_p(\mu_1, \Sigma)$), (donde μ_2 es el vector de medias de la población 1 y Σ es la matriz de covarianza de la población 1). Asumiendo que ambas poblaciones tienen igual matriz de covarianza Σ.

Entonces la hipótesis nula seria la siguiente:

$$H_o: \mu_1 - \mu_2 = 0$$

El estadístico de prueba T^2 de Hotelling con parámetros (p, n1 + n2 -2) es como sigue:

$$T^2 = [\bar{X}_1 - \bar{X}_2 - (\mu_1 - \mu_2)]' \left[\left(\frac{1}{n1} + \frac{1}{n2}\right) S_{combinado}\right]^{-1} [\bar{X}_1 - \bar{X}_2 - (\mu_1 - \mu_2)]$$

$(donde\ \bar{X}_1 - \bar{X}_2$, es la diferencia entre medias muestrales de las dos poblaciones).

Y se distribuye como:

$$C^2 = \frac{(n1 + n2 - 2)p}{n1 + n2 - p - 1} F_{p,n1+n2-p-1}$$

La regla de decisión es que se acepta la hipótesis H_o, si $T^2 <= C^2$ en caso contrario se rechaza la hipótesis H_o, donde se tomará el valor F con un nivel de confianza de 95%.

Donde $S_{combinado}$ es la matriz de covarianza muestral combinada de las matrices de covarianza muestral S_1 y S_2 (se observa que ambas poblaciones son p variadas, p=12).

$$S_{combinado} = \frac{n1 - 1}{n1 + n2 - 1} S_1 + \frac{n2 - 1}{n1 + n2 - 1} S_2$$

(n1 – 1)S_1 es distribuida como $W_{n1-1}(\Sigma)$ y (n2 – 1)S_1 es distribuida como $W_{n2-1}(\Sigma)$, donde W es la distribución de Wishart.

A continuación se presentan los resultados de esta comparación.

Tabla 44

Test estadísticos de distribución normal multivariante para comparar las medias de dos poblaciones

Poblaciones	Resultado
TRAIN Y TEST	Vector de medias de la población uno 2.308642 5.782267 1.647587 3.198653 2.114478 1.13468 1.13468 11.79349 3.821549 3.619529 2.531987 3.272727 Vector de medias de las población dos 2.26555 5.849282 1.636364 3.138756 2.220096 1.086124 1.136364 11.43301 3.894737 3.62201 2.401914 3.244019 [1] Valor de T cuadrado [1,] 15.61233 [1] Valor de C cuadrado [1] 21.29484 Conclusion: Los vectores de medias poblaciones son iguales Intervalos simultaneos de confianza para la diferencia basados en T cuadrado EstimateLowerCIUpperCI Difer. Est.(D_{i1}) Limite Inf Limite Sup 1 0.043091736 -0.18614283 0.2723263 2 -0.067015181 -1.67580115 1.5417708 3 0.011223345 -0.11986171 0.1423084 4 0.059897218 -0.46742026 0.5872147 5 -0.105617579 -0.58230325 0.3710681 6 0.048555733 -0.07127058 0.1683820 7 -0.001683502 -0.12488583 0.1215188 8 0.360476106 -0.80412035 1.5250726 9 -0.073188021 -0.68253339 0.5361574 10 -0.002480950 -0.21086384 0.2059019 11 0.130072656 -0.09265830 0.3528036 12 0.028708134 -0.26141130 0.3188276

De la Tabla 44, se observa que el valor T^2 $es\ menor\ que$ C^2. Por los tanto se acepta la hipótesis de que la diferencia entre las medias poblacionales es 0, con un nivel de 95% de confianza.

Luego por esta prueba estadística multivariada. Se concluye que las dos poblaciones TRAIN y TEST están muestreadas de una misma población original.

4.3. Limpieza de datos

Generalmente los datos de los cuales se dispone en cualquier tipo de investigación en el mundo real están incompletos, con ruido y son inconsistentes. La limpieza de datos permite completar datos incompletos, suavizar el ruido, e identificar los valores atípicos.

En esta sección se explicará la forma en que distintos métodos de limpieza de datos fueron aplicados al problema tratado.

Los métodos que hemos utilizado en esta tesis para abordar el problema de predicción del Titanic son:

a) Imputación de datos incompletos (método EM)
b) Transformación de los datos (reducción de dimensionalidad)

4.3.1. Imputación de valores perdidos

Entre los métodos de imputación de datos incompletos o valores perdidos se encuentran los siguientes tres enfoques [WM06]:

1) Reglas de predicción de valores
2) Enfoque de la máxima verosimilitud.
3) Aproximación por mínimos cuadrados.

En este trabajo se utilizará el enfoque de la máxima verosimilitud, dentro del cual el método más popular es el de expectation-maximization (EM) para la imputación de valores perdidos.

4.3.1.1. Método de Expectation-Maximization (EM)

En este trabajo se utilizó el algoritmo EM regularizado, cuyo código hecho en MATLAB obtenido de [ST01], servirá para estimar los valores perdidos, tanto de los datos de entrenamiento TRAIN como de los datos de prueba TEST. En este caso los valores perdidos de TRAIN se encuentran en las variables AGE y EMBARKED, y los valores perdidos de TEST se encuentran en las variables AGE y FARE. En el siguiente cuadro se muestran las estimaciones realizadas utilizando el código en MATLAB donde para realizar los cálculos se realizaron uniendo las dos poblaciones en uno sola (TRAIN y TEST).

Tabla 45
Valores Perdidos con el Método EM

Datos	Variables	# Valores Perdidos	Valor estimado
TRAIN	AGE	177	29.8811
TRAIN	EMBARKED	2	2.49273APROX = 2
TEST	AGE	61	29.8811
TEST	FARE	1	33.2955

En el caso de EMBARKED, primeramente la variable se transformó a valores numéricos, donde 1 es Q, 2 es C y 3 es S (ver en Tabla 1). Los valores nulos o perdidos las variables EMBARKED, en TRAIN, y FARE, en TEST, se reemplazaron por sus respectivos valores estimados con el método EM.

4.4. Transformación de los datos

En muchas aplicaciones del mundo real, numerosas características se utilizan en un intento de garantizar una clasificación exacta. Por tanto se suele reducir la dimensionalidad de los datos para construir clasificadores. De esta forma se intenta evitar que el proceso de aprendizaje se vuelva computacionalmente y analíticamente complicado, resultando a menudo en la subida drástica de error de clasificación.

Las técnicas de reducción de dimensionalidad adoptadas se dividen en selección de características, y la transformación de características (además llamadas descubrimiento de características) [SKVK06] [PM05]. La diferencia fundamental entre la función de selección y transformación de características es que en el primer caso se selecciona sólo un subconjunto de las características originales, mientras que el segundo enfoque se basa en la generación de características completamente nuevas a partir de las ya existentes. La extracción de características es una técnica de reducción de la dimensionalidad que extrae un subconjunto de nuevas características de un conjunto original de características por medio de algún mapeo funcional, manteniendo la mayor cantidad posible de información en los datos.

4.4.1. Extracción de características o rasgos

Es uno de los procesos de reducción de la dimensionalidad de los datos, cuando los datos de entrada a un algoritmo son demasiado grandes para ser procesados y se sospecha son redundantes. En este caso los datos de entrada se transforman en una representación reducida del conjunto de características (también llamado vector de características). La transformación de los datos de entrada en el conjunto de características se llama extracción de características. Si las características extraídas se eligen cuidadosamente se espera que el conjunto de características vaya a extraer la información relevante de los datos de entrada con el fin de realizar la tarea deseada usando esta representación reducida en lugar de la entrada de tamaño completo.

En este trabajo se extraerá información importante contenida en algunas variables para deducir a partir de ellas nuevos rasgos o características, y también se podrá combinar varios rasgos en un nuevo rasgo.

4.4.1.1. Extracción de características o nuevas variables a partir de los datos

En este trabajo se extraerán nuevas características a partir de los datos originales, que fueron mostrados, en la Tabla 1 (del Capítulo 3). En la siguiente tabla se presentan los rasgos que permanecen tal y como fueron descritos en un principio, y los nuevos rasgos extraídos a partir de los ya existentes (el código en R correspondiente se encuentra en el Apéndice D en las tablas D1 y D2).

Tabla 46

Extracción de características de los datos de entrenamiento TRAIN y los datos de prueba TEST

Rasgo o característica	Transformación	
PCLASS	PERMANECE SIN CAMBIOS (TABLA 1)	
TITULOPASAJERO	TITULO DEL PASAJERO EN VARIABLE NAME	VALOR NUEVO
	MR	1
	DON	2
	SIR	3
	JONKHEER	4
	REV	5
	DR	6
	MAJOR	7
	CAPT	8
	COL	9
	MRS O DONA	10
	MME	11
	COUNTESS	12
	MS	13
	MISS	14
	MLLE	15
	MASTER	16
SEXNUEVO	SEXO DEL PASAJERO EN LA VARIABLE SEX	VALOR NUEVO
	FEMALE	1
	MALE	2
AGEMALE	INTERVALOS DE EDADES DE LA VARIABLE AGE SOLO SEXO MASCULINO	VALOR NUEVO DEL INTERVALO
	[-1,0]	1
	<0,15]	2
	<15,20]	3
	<20,30]	4
	<30,40]	5
	<40,50]	6
	<50,60]	7
	<60,100]	8
AGEFEMALE	INTERVALOS DE EDADES DE LA VARIABLE AGE SOLO SEXO FEMENINO	VALOR NUEVO DEL INTERVALO
	[-1,0]	1
	<0,15]	2
	<15,20]	3
	<20,30]	4
	<30,40]	5
	<40,50]	6

	<50,60]	7
	<60,100]	8
PARCHNUEVO	INTERVALOS DE NÚMERO DE PADRES/HIJOS DE LA VARIABLE PARCH (TABLA 1)	VALOR NUEVO DEL INTERVALO
	[0,1]	1
	<1,2]	2
	<2,3]	3
	<3,100]	4
SIBSPNUEVO	INTERVALOS DE NÚMERO DE HERMANOS /CONYUGES DE LA VARIABLE SIBSP (TABLA 1)	VALOR NUEVO DEL INTERVALO
	[0,1]	1
	<1,2]	2
	<2,100]	3
TIPOTICKET	PRIMER PASO: CREAR VARIABLE TEMPORAL T1	
	CARACTERES CON LOS QUE EMPIEZA LOS VALORES DE LA VARIABLE TICKET DE LA TABLA 1	VALOR NUEVO DE VARIABLE TEMPORAL T1
	A./5 OR A.5 OR A/S OR A/5	1
	A/4 OR A4	2
	C	3
	C.A OR CA	4
	F.COR ,F.C.C.	5
	FA OR LINE OR p/PP OR AQ OR A OR 2. OR LP	6
	PC	7
	PP	8
	S.C.	9
	S.O.	10
	S.O.C.	11
	S.O.P.	12
	S.P.	13
	S.W.	14
	SC	15
	SCO	16
	SO/C	17
	SOTON	18
	STON	19
	SW/PP	20
	W./C.	21
	W.E.P.	22
	W/C	23
	WE/P	24
	1	25
	2	26
	3	27
	4	28
	5	29
	6	30
	7	31
	8	32
	9	33
	PASO 2: HALLAR INTERVALOS PARA LA VARIABLE TEMPORAL T1	
	INTERVALOS PARA LA VARIABLE TEMPORAL T1	VALOR NUEVO DEL

		INTERVALO
	[0,2]	1
	<2,4]	2
	<4,5]	3
	<5,6]	4
	<6,8]	5
	<8,13]	6
	<13,14]	7
	<14,16]	8
	<16,17]	9
	<17,19]	10
	<19,20]	11
	<20,21]	12
	<21,24]	13
	<24,27]	14
	<27,33]	15
FARENUEVO	INTERVALOS PARA LA VARIABLE FARE (TABLA 1)	VALOR NUEVO DEL INTERVALO
	[0,7]	1
	<7,10]	2
	<10,20]	3
	<20,30]	4
	<30,40]	5
	<40,50]	6
	<50,75]	7
	<75,100]	8
	<100,VALOR MAXIMO DE TRAIN Y TEST=512.3292]	9
CABINNUEVO	PRIMER PASO:CREAR VARIABLE TEMPORAL T2	
	CARACTER CON EL QUE EMPIEZAN LOS VALORES DE LA VARIABLE CABIN DE LA TABLA 1 (QUE SERIAN LAS CUBIERTAS DEL BARCO)	VALOR NUEVO DE LA VARIABLE TEMPORAL T2
	A	1
	B	2
	C	3
	D	4
	E	5
	F	6
	G	7
	T O SI EL VALOR DE CABIN ES NULO	8
	SEGUNDO PASO: HALLAR INTERVALOS PARA LA VARIABLE TEMPORAL T2	
	INTERVALOS PARA LA VARIABLE TEMPORAL T2	VALOR NUEVO DEL INTERVALO
	[0,1]	1
	<1,3]	2
	<3,7]	3
	<7,8]	4
EMBARKEDNUEVO	VALOR DE LA VARIABLE EMBARKED (TABLA 1)	VALOR NUEVO
	C	1
	Q	2
	S	3
SOBREVFAMILIA	METODO: SE CONSIDERAN EL APELLIDO PATERNO DE LA VARIABLE NAME, LA CUBIERTA DEL BARCO QUE ES LA PRIMERA LETRA DE LA VARIABLE CABIN DONDE ESTA VARIABLE TIENE TAMBIEN VALORES NULOS Y LAS	

	VARIABLES SIBSP Y PARCH(ESTAS VARIABLE ESTAN EN LA TABLA 1)	
	REGLAS	VALOR NUEVO
	REGLA 1: SI EL PASAJERO ESTA SOLO ES DECIR SI SIBSP=0 Y PARCH=0	4
	REGLA 2: SI EL PASAJERO TIENE HERMANOS/CONYUGES O PADRES/HIJOS Y TIENE CABINAS CONTIGUAS EN LA MISMA CUBIERTA Y ESTAN EN PRIMERA CLASE EN OTRO CASO: (SI SIBSP>0 OR PARCH>0)AND PCLASS=1 AND (TIENEN CABINAS EN LA MISMA CUBIERTA)	1
	REGLA 3:SI EL PASAJERO TIENE HERMANOS/CONYUGES O PADRES/HIJOS Y TIENES CABINAS CONTIGUAS EN LA MISMA CUBIERTA Y ESTAN EN SEGUNDA CLASE EN OTRO CASO: (SI SIBSP>0 OR PARCH>0)AND PCLASS=2 AND (TIENEN CABINAS EN LA MISMA CUBIERTA)	2
	REGLA 4: SI EL PASAJERO TIENE HERMANOS/CONYUGES O PADRES/HIJOS Y TIENES CABINAS CONTIGUAS EN LA MISMA CUBIERTA Y ESTAN EN TERCERA CLASE EN OTRO CASO: (SI SIBSP>0 OR PARCH>0)AND PCLASS=3 AND (TIENEN CABINAS EN LA MISMA CUBIERTA)	3
	NOTA: LAS CUBIERTAS ESTAN DESCRITAS EN ESTA TABLA EN LA VARIABLE CABINNUEVO EN LA PARTE DEL PRIMER PASO	
SURVIVED	PERMANECE SIN CAMBIOS (TABLA 1)	

E

n la tabla 46, están todas las variables que serán utilizadas en este trabajo. Puesto que el descubrimiento del conocimiento es un proceso iterativo, podrán agregarse nuevos rasgos o características o tal vez modificarse las que se hallaron, las transformaciones hechas para los datos de entrenamiento y datos de prueba fueron enviadas a los archivos csv: TRAINTRANSF.CSV y TESTTRANSF.CSV respectivamente utilizando R, los códigos se encuentran en el Apéndice D (tablas D1 y D2).

4.4.2. Selección de características o variables

En el aprendizaje automático y la estadística, la selección de características, también conocida como la selección de variables, selección de atributos o selección de subconjuntos de variables, es el proceso de selección de un subconjunto de características relevantes para su uso en la construcción del modelo. El supuesto central cuando se utiliza una técnica de selección de características es que los datos contienen muchas características redundantes o irrelevantes. Características redundantes son las que no proporcionan más información que las características seleccionadas [LMSZ10].

Las técnicas de selección de características son un subconjunto de un campo más general de extracción de características. La extracción de características crea nuevas características de las funciones de los elementos originales, mientras que la función de selección devuelve un subconjunto de las características. Las técnicas de selección de características proporcionan tres ventajas principales en la construcción de modelos de predicción:

1) Mejoran la interpretabilidad del modelo.
2) Permiten tiempos más cortos de entrenamiento.
3) Generalización mejorada mediante la reducción del sobre-ajuste.

4.4.2.1. Enfoques de la selección de características o variables

Las técnicas para la selección de características se pueden dividir en dos enfoques

a) Rankeo de características.
b) Selección de subconjuntos.

En el primer enfoque, las características se rankean de acuerdo con algún criterio y luego se seleccionan las características cuyos valores están por encima de un umbral definido.

En el segundo enfoque, se busca a partir de un espacio de subconjuntos de características el subconjunto óptimo. Por otra parte, el enfoque de selección de subconjuntos se puede dividir en tres métodos:

1) Método de Selección de Filtro
2) Método de Selección de Envoltura o Wrapper
3) Método de selección embebido o incrustado.

En este trabajo se utilizarán básicamente dos enfoques que son el de Rankeo de características y el de selección de subconjuntos, dentro del cual se elegirá el enfoque Wrapper o Envoltura.

4.4.2.1.1. Enfoque de rankeo de características

En este enfoque se selecciona el subconjunto de características primero, y a continuación se utiliza este subconjunto para ejecutar un algoritmo de clasificación. Entre las medidas comunes se incluyen la información mutua, el coeficiente de correlación de Pearson y la distancia inter / intra clase. Para el método de selección de atributos por rankeo de características, se utilizará el enfoque de “ranking” [WI13], y se utilizarán las técnicas rankeo que se encuentran dentro del paquete FSELECTOR del software R, que son las siguientes:

a) Filtro Chi –Cuadrado.- El algoritmo encuentra pesos de atributos discretos basándose en una prueba de chi-cuadrado.

b) **Filtro de Correlación**.- El algoritmo encuentra pesos de los atributos continuos que se basan en su correlación con el atributo de clase continua. Se utilizan dos tipos de correlaciones que son:

I. Correlación de Pearson
II. Correlación de Spearman

c) **Filtro basado en Entropía**.- Los algoritmos encuentran pesos de atributos discretos que se basan en su correlación con el atributo de clase continua. Y se subdividen en dos que son:

i. Ganancia de información
ii. Ganancia de Ratio

d) **Filtro de Bosque Aleatorio (RandomForest).-** El algoritmo encuentra pesos de atributos utilizando el algoritmo Random Forest[WI13].

A continuación se presentan los resultados de la comparación entre las diferentes técnicas utilizando los datos de entrenamiento TRAINTRANSF (891 observaciones donde se hizo una partición aleatoria de 80/20 sin reemplazo es decir un 80% para los datos de entrenamiento y 20% para los datos de prueba).

a) Filtro Chi-Squared
b) Filtro Basado en Entropía - Ganancia de Información
c) filtro Basado en Entropía - Ganancia de Ratio
d) Filtro de Random Forest

Estos resultados se encuentran en la Tabla 47, donde se muestran las variables seleccionadas para el enfoque de rankeo de características, y su respectivo error de estimación (que corresponde a la partición de prueba).

Tabla 47
Resultados de la Selección de Atributos por el Método de Filtro

Nº Orden	Técnicas de Rankeo	Variables en el modelo	Algoritmo	Error
1	*Filtro Chi-squared*	SURVIVED, TITULOPASAJERO, AGEMALE, SEXNUEVO AGEFEMALE, PCLASS, SOBREVFAMILIA, CABINNUEVO, FARENUEVO, SIBSPNUEVO, PARCHNUEVO	RPART	0.1564246
			ADA	0.1452514
			RANDOM FOREST	0.1396648
			SVM	0.1564246
			LINEAL	0.1452514
		Variables no seleccionadas	NEURAL NETWORKS	0.1675978
	Filtro basado en Entropía - Ganancia de información	TIPOTICKET Y EMBARKEDNUEVO	GRADIENT BOOSTING	0.1564246
			BAGGING	0.1564246
	Filtro basado en		BAYES NAIVES	0.1675978

	Entropía - Ganancia de ratio		KNN	0.1508380
2	*Filtro de Random Forest*	SURVIVED, FARENUEVO, PCLASS, SIBSPNUEVO, TITULOPASAJERO, AGEMALE, CABINNUEVO, EMBARKEDNUEVO, SOBREVFAMILIA, AGEFEMALE, TIPOTICKET	RPART	0.1564246
			ADA	0.1508380
			RANDOM FOREST	0.1508380
			SVM	0.1508380
			LINEAL	0.1620112
		Variables no seleccionadas	NEURAL NETWORKS	0.1955307
		PARCHNUEVO Y SEXNUEVO	GRADIENT BOOSTING	0.1508380
			BAGGING	0.1620112
			BAYES NAIVES	0.1564246
			KNN	0.1843575

En la Tabla 47 se han elegido las variables: TITULOPASAJERO, AGEMALE, SEXNUEVO, AGEFEMALE, PCLASS, SOBREVFAMILI, CABINNUEVO, FARENUEVO, SIBSPNUEVO, PARCHNUEVO y se han excluido de esta selección a las variables TIPOTICKET Y EMBARKEDNUEVO, para los métodos a), b) y c).

se han elegido para el método de filtrado RANDOM FOREST (método d)), las variables FARENUEVO, PCLASS, SIBSPNUEVO, TITULOPASAJERO, AGEMALE, CABINNUEVO, EMBARKEDNUEVO, SOBREVFAMILIA, AGEFEMALE, TIPOTICKET, excluyéndose las variables PARCHNUEVO Y SEXNUEVO.

De esta selección, para los métodos a), b) y c), el mejor algoritmo es el RANDOM FOREST con un error de `0.1396648`, y para el Filtro siguiente d):

a) Filtro Random Forest

Los mejores algoritmos son el:

- ✓ ADA BOOST
- ✓ RANDOM FOREST
- ✓ SVM
- ✓ GRADIENT BOOSTING

Todos con un error de 0.1508380. Luego por ser un algoritmo que mayormente da buenos resultados, se elegirá al algoritmo RANDOM FOREST.

4.4.2.1.2. Enfoque de selección de subconjunto de variables por el método Wrapper

Este método de selección de características conduce una búsqueda para encontrar un buen subconjunto usando un algoritmo predictivo (de minería de datos) como parte de la función evaluadora de subconjuntos de características [KJ97].

Como este método entrena un nuevo modelo para cada subconjunto, esta metodología "*ofrece el subconjunto de características de mejor performance para un particular tipo de modelo*" [PS04].

Para realizar el método de selección de características Wrapper, se utilizará el paquete FSELECTOR de R contiene las siguientes técnicas de búsqueda:

a) Búsqueda BESTFIRST.- El algoritmo es similar al forward.search además del hecho de que se elige el mejor nodo de todos los ya evaluados, y se evalúa este. La selección del mejor nodo se repite un número máximo de veces, que es un parámetro del algoritmo en caso de no encontrar un mejor nodo.

b) Búsqueda GREEDY BACKWARD.- Este algoritmo implementa la búsqueda ávida (Greedy Search). Al principio, el algoritmo expande el nodo de inicio, evalúa sus hijos y elige la mejor opción que se convierte en un nuevo nodo de inicio. Este proceso va sólo en una dirección. Hacia adelante (esta técnica comienza a partir del conjunto completo de atributos).

c) **Búsqueda de ascenso de la colina (Hill Climbing)**.-El algoritmo comienza con un conjunto de atributos al azar. A continuación, se evalúa a todos sus vecinos y se opta por la mejor solución. Puede ser que sea susceptible de máximo local.

Los algoritmos de minería de datos que serán utilizados en este trabajo son:

1) Árbol de Decision
2) Gradient Boosting
3) Bagging
4) Ada Boost
5) Random Forest
6) Support Vector Machine SVM
7) Regresión Logística
8) Redes neuronales
9) Naïve Bayes
10) K- vecindad mas cercana

Las estimaciones que se describen en esta sección se llevarán a cabo con el software R y posteriormente en el CAPITULO 5 se utilizarán además los software Python (con el módulo Scikitlearn) y WEKA con las selecciones de variables hechas aquí.

A continuación se presentan los resultados de la estimación de las mejores selecciones de variables con el método de envoltura (Wrapper), para cada uno de los algoritmos presentados arriba que son 1), 2), 3), 4), 5), 6), 7), 8), 9) Y 10).

Los algoritmos son presentados en el APENDICE A, y corresponden a los algoritmos de minería de datos ya señalados anteriormente. En cada uno se tomó como el conjunto de entrenamiento TRAINTRANSF (891 observaciones) y se hizo una partición a esos datos

aleatoriamente de 80/20 sin reemplazo (es decir un 80% es para los datos de entrenamiento y 20% es para los datos de prueba). Luego los errores fueron estimados en la parte de los datos de prueba (el 20% de los datos).

Tabla 48

Mejores conjuntos de variables para los diferentes algoritmos de minería de datos obtenidos con el método de selección de subconjuntos

Nº Orden	Algoritmo	Técnica de Filtro	Variables en el modelo	Error
1	ÁRBOL DE DECISIÓN RPART	GREEDY SEARCH BACKWARD	SURVIVED, PCLASS, TITULOPASAJERO, SEXNUEVO, AGEMALE, AGEFEMALE, SIBSPNUEVO, PARCHNUEVO, TIPOTICKET, CABINNUEVO, SOBREVFAMILIA **Variables no seleccionadas** FARENUEVO Y EMBARKEDNUEVO	0.1508380
2	GRADIENT BOOSTING	BESTFIRST	SURVIVED, TITULOPASAJERO, SEXNUEVO, AGEMALE, AGEFEMALE, SIBSPNUEVO, EMBARKEDNUEVO, SOBREVFAMILIA **Variables no seleccionadas** PCLASS , PARCHNUEVO, TIPOTICKET, FARENUEVO Y CABINNUEVO	0.1508380
3	ADA BOOST (ADA)	GREEDY SEARCH BACKWARD	SURVIVED, PCLASS, TITULOPASAJERO, SEXNUEVO, AGEMALE, AGEFEMALE, SIBSPNUEVO, PARCHNUEVO, TIPOTICKET, CABINNUEVO, EMBARKEDNUEVO, SOBREVFAMILIA **Variables no seleccionadas** FARENUEVO	0.1396648
4	BAGGING	HILL CLIMBING	SURVIVED, TITULOPASAJERO, SEXNUEVO, SIBSPNUEVO, PARCHNUEVO, TIPOTICKET **Variables no seleccionadas** PCLASS, AGEMALE, AGEFEMALE, FARENUEVO, CABINNUEVO, EMBARKEDNUEVO Y SOBREVFAMILIA	0.1396648
5	RANDOM FOREST	HILL CLIMBING	SURVIVED, TITULOPASAJERO, SEXNUEVO, AGEMALE, SIBSPNUEVO, PARCHNUEVO, TIPOTICKET, CABINNUEVO, EMBARKEDNUEVO **Variables no seleccionadas** PCLASS, AGEFEMALE, FARENUEVO Y SOBREVFAMILIA	0.1284916
6	SUPPORT VECTOR MACHINE (SVM)	HILL CLIMBING	SURVIVED, TITULOPASAJERO, SEXNUEVO, AGEFEMALE , SIBSPNUEVO, PARCHNUEVO, TIPOTICKET , CABINNUEVO **Variables no seleccionadas** PCLASS, AGEMALE, FARENUEVO, EMBARKEDNUEVO Y SOBREVFAMILIA	0.1396648
7	REGRESIÓN LOGÍSTICA	BESTFIRST	SURVIVED , TITULOPASAJERO, SEXNUEVO, SIBSPNUEVO, PARCHNUEVO, TIPOTICKET, EMBARKEDNUEVO, SOBREVFAMILIA **Variables no seleccionadas** PCLASS, AGEMALE, AGEFEMALE, FARENUEVO YCABINNUEVO	0.1620112

8	REDES NEURONALES	GREEDY SEARCH BACKWARD	SURVIVED, PCLASS , TITULOPASAJERO, SEXNUEVO, AGEMALE , AGEFEMALE, SIBSPNUEVO, PARCHNUEVO, TIPOTICKET, CABINNUEVO, EMBARKEDNUEVO, SOBREVFAMILIA **Variables no seleccionadas** FARENUEVO	0.1620112
9	NAÏVE BAYES	GREEDY SEARCH BACKWARD	SURVIVED, TITULOPASAJERO, SEXNUEVO, AGEMALE, SIBSPNUEVO, PARCHNUEVO, EMBARKEDNUEVO, SOBREVFAMILIA **Variables no seleccionadas** PCLASS, AGEFEMALE, TIPOTICKET, FARENUEVO YCABINNUEVO	0.1508380
10	K.VECINDAD MAS CERCANA	BESTFIRST	SURVIVED, SEXNUEVO, EMBARKEDNUEVO, SOBREVFAMILIA **Variables no seleccionadas** PCLASS, TITULOPASAJERO, AGEMALE, AGEFEMALE, PARCHNUEVO, SIBSPNUEVO, TIPOTICKET, FARENUEVO Y CABINNUEVO	0.1731844

En la Tabla 58, se observa que la mejor estimación y selección corresponden al algoritmo RANDOM FOREST, y a la técnica de filtro HILL CLIMBING, con un error de 0.1284916, y donde la técnica HILL CLIMBING obtuvo bajos errores en el caso de los algoritmos BAGGING, RANDOMFOREST y SUPPORT VECTOR MACHINE (SVM). El algoritmo ADA BOOST con GREEDY SEARCH BACKWARD también obtuvo un error menor de 0.1396648.

La Tabla 58, servirá como base para realizar las estimaciones de los modelos en el Capítulo 5, con los softwares R y Python.

Capítulo 5

Aspectos algorítmicos de la minería de datos

En este capítulo se presentan los siguientes cuatro pasos del descubrimiento del conocimiento correspondientes a la minería de datos que son: la selección de la tarea apropiada de minería de datos, la selección del algoritmo de minería de datos, el empleo de algoritmos de minería de datos y la evaluación de los modelos de minería de datos.

5.1. Selección de la tarea apropiada de minería de datos

En esta etapa dentro del KDD, se decide sobre qué tipo de minería de datos o estrategia se va a utilizar. Basados en el Capítulo 2, en la sección 2.2, (etapas del descubrimiento del conocimiento) se ha decidido utilizar los diferentes métodos de aprendizaje supervisado para predecir esta variable SURVIVED (la variable dependiente).

Para realizar el aprendizaje de los algoritmos el conjunto de datos de entrenamiento, se particionará aleatoriamente sin reemplazo en un 80% para datos de entrenamiento y 20% para datos prueba.

5.2. Selección de los algoritmos de la minería de datos

Teniendo la estrategia, ahora se decidirá sobre las tácticas. En esta etapa se seleccionan los métodos específicos que se utilizarán para buscar patrones. Primeramente considerando el software que se va a utilizar (R, Python con el módulo ScitLearn y WEKA), así como la precisión, y la disponibilidad de algoritmos en estos software. Los algoritmos más apropiados [SS13] son:

1) Arboles de decisión
2) Métodos Ensemble
   a) GradientBoosting
   b) Bagging
   c) Ada Boost
   d) RandomForest
3) Support Vector Machine SVM
4) Regresión Logística
5) Redes Neuronales
6) Naïve Bayes
7) K-Vecindad mas Cercana

5.3. Empleo de los algoritmos de minería de datos

En esta sección se utilizarán los 10 algoritmos señalados en la sección 5.2. Se realizarán las estimaciones de los modelos utilizando los pasos previamente descritos en la sección 2.2 (etapas del descubrimiento del conocimiento).

5.3.1. Mejor selección de variables y algoritmos para el enfoque de rankeo de características o rasgos

En esta parte se realizará la estimación de modelos, considerando los resultados hallados en la sección 4.4.2.1.1 (Tabla 47).

5.3.1.1. Estimación con R, para las variables seleccionadas con el enfoque de rankeo de atributos

Aquí se utilizará el algoritmo Random Forest, de acuerdo a la Tabla 47 (con los métodos de rankeo de características) que producen los resultados de la Tabla 49, (el código está en R y se encuentra en el Apéndice B, Tablas B1 y B2):

Tabla 49

Resultados de las estimaciones con R del algoritmo Random Forest, para la selección de variables con el método rankeo de atributos

Caso	ALGORITMO RANDOM FOREST
1	**TECNICAS DE FILTRO**: *a) Filtro Chi-squared* *b) Filtro Basado en Entropía - Ganancia de información* *c) Filtro Basado en Entropía - Ganancia de ratio* **VARIABLES EN EL MODELO:** SURVIVED, TITULOPASAJERO, AGEMALE, SEXNUEVO, AGEFEMALE, PCLASS, SOBREVFAMILIA, CABINNUEVO FARENUEVO, SIBSPNUEVO, PARCHNUEVO **VARIABLES EXCLUIDAS:** TIPOTICKET Y EMBARKEDNUEVO **RESUMEN DEL ALGORITMO RANDOM FOREST CON R**

```
[1] "Error de los datos de prueba (20% de los datos)"
[1] 0.1396648
OOB estimado de tasa de error(para datos de entrenamiento (80%
de los datos):
    0.1896
[1] "Matriz de Confusión de los Datos de Prueba (20% de los datos)"
    Predicho
Real   0   1
   0 104   8
   1  15  52
[1] "Importancia de variables según orden (datos de entrenamiento 80% de
datos)"
                     0     1 MeanDecreaseAccuracy MeanDecreaseGini
titulopasajero 34.20 10.40                30.87            25.76
farenuevo      21.33 13.37                26.78            12.07
sibspnuevo     28.05 -0.65                26.70             6.37
pclass         17.04 19.75                26.56            12.21
agemale        16.43 10.20                18.01            23.24
agefemale      13.37  3.58                13.96            17.11
sexnuevo       11.43  0.08                11.32            12.11
cabinnuevo      7.79  4.36                10.21             8.20
sobrevfamilia   5.80  6.88                 9.48             6.50
parchnuevo      1.95 -4.67                -1.12             2.66

[1] "Área bajo la Curva ROC para los datos de entrenamiento"
[[1]]
[1] 0.8944539
```

2

TECNICAS DE FILTRO:

a) Filtro Random Forest

VARIABLES EN EL MODELO:

SURVIVED, FARENUEVO, PCLASS, SIBSPNUEVO, TITULOPASAJERO, AGEMALE, CABINNUEVO, EMBARKEDNUEVO, SOBREVFAMILIA, AGEFEMALE, TIPOTICKET

VARIABLES EXCLUIDAS:

PARCHNUEVO Y SEXNUEVO

RESUMEN DEL ALGORITMO RANDOM FOREST CON R

```
[1] "Error de los datos de prueba (20% de los datos)"
[1] 0.150838
OOB estimado de tasa de error(para datos de entrenamiento (80% de los datos):
    0.1882
[1] "Matriz de Confusión de los Datos de Prueba (20% de los datos)"
    Predicho
Real   0   1
   0 104   8
   1  15  52
[1] "Importancia de variables según orden (datos de entrenamiento 80% de datos)"
                    0     1 MeanDecreaseAccuracy MeanDecreaseGini
farenuevo       29.75 13.36                35.37            15.35
titulopasajero  29.89 13.73                28.93            28.67
pclass          22.16 17.65                28.89            13.10
sibspnuevo      24.59 -1.72                22.95             6.27
agemale         18.98 12.96                20.51            29.52
agefemale       16.09  3.65                16.11            22.11
embarkednuevo    4.59 15.34                14.86             5.60
cabinnuevo      10.51 -0.44                10.47             9.22
sobrevfamilia    7.70  4.68                10.19             8.03
tipoticket       5.20 -1.11                 3.78             8.33

[1] "Área bajo la Curva ROC para los datos de entrenamiento"
[[1]]
[1] 0.9289827
```

Según este cuadro para el primer caso, se produce en el 80% de los datos de entrenamiento un error de *0.1896*, y para el 20% de los datos de entrenamiento (de prueba) el error es de 0.1396648

El algoritmo Random Forest da como resultado también el orden de las variables, de dos formas, el MeanDecreaseAccuracy y el MeanDecreaseGini. De ellos, el primero da más importancia a la variable TITULOPASAJERO, luego viene FARENUEVO, y así sucesivamente, lo cual se explicará en la sección de evaluación de los modelos (sección 5.4).

Para el segundo caso, se produce en el 80% de los datos de entrenamiento un error de 0.1882, y para el 20% de los datos de entrenamiento (de prueba) el error es de 0.150838.

5.3.1.2. Estimación con Python, para las variables seleccionadas con el método de rankeo de atributos

Aquí se utilizará el algoritmo Random Forest, de acuerdo a la Tabla 47 (con los métodos de rankeo de características) que producen los resultados de la Tabla 50, (el código está en Python y se encuentra en el Apéndice B, Tablas B3 y B4):

Tabla 50
Resultados de las estimaciones con Python del algoritmo Random Forest, para la selección de variables con el método de rankeo de atributos

Caso	ALGORITMOS
1	**TECNICAS DE FILTRO**: *a) Filtro Chi-squared* *b) Filtro Basado en Entropía - Ganancia de información* *c) Filtro Basado en Entropía - Ganancia de ratio* **VARIABLES EN EL MODELO:** SURVIVED, TITULOPASAJERO, AGEMALE, SEXNUEVO, AGEFEMALE, PCLASS, SOBREVFAMILIA, CABINNUEVO FARENUEVO, SIBSPNUEVO, PARCHNUEVO **VARIABLES EXCLUIDAS:** TIPOTICKET Y EMBARKEDNUEVO **RESUMEN DEL ALGORITMO RANDOM FOREST CON PYTHON** ***"Error de los datos de prueba (20% de los datos de entrenamiento)"*** ***0.156424581006*** ***"Matriz de Confusión de los Datos de Prueba (20% de los datos de entrenamiento)"*** ***preds*** ***actual 0 1*** ***0 106 12*** ***1 16 45***
2	**TECNICAS DE FILTRO**: *b) Filtro Random Forest* **VARIABLES EN EL MODELO:** SURVIVED, FARENUEVO, PCLASS, SIBSPNUEVO, TITULOPASAJERO, AGEMALE, CABINNUEVO, EMBARKEDNUEVO, SOBREVFAMILIA, AGEFEMALE, TIPOTICKET **VARIABLES EXCLUIDAS:** PARCHNUEVO Y SEXNUEVO **RESUMEN DEL ALGORITMO RANDOM FOREST CON PYTHON**

	"Error de los datos de prueba (20% de los datos de entrenamiento)" ***0.150837988827*** ***"Matriz de Confusión de los Datos de Prueba (20% de los datos de entrenamiento)"*** ***preds*** ***actual 0 1*** ***0 105 13*** ***1 14 47***

Según el primer caso, el error producido en los datos de prueba (20% de los datos de entrenamiento) es de 0.156424581006.

Y según el segundo caso, el error producido en los datos de prueba (20% de los datos de entrenamiento) es de 0.150837988827**,** que se aproxima al error del primer caso.

5.3.1.3. Estimación con WEKA, para las variables seleccionadas con el método de rankeo de atributos

Aquí se utilizará el algoritmo Random Forest, de acuerdo a la Tabla 47 (con los métodos de rankeo de características) que producen los resultados de la Tabla 51.

Tabla 51
Resultados de las estimaciones con WEKA del algoritmo Random Forest, para la selección de variables con el método de filtro rankeado

Caso	ALGORITMOS
1	**TECNICAS DE FILTRO**: *a) Filtro Chi-squared* *b) Filtro Basado en Entropía - Ganancia de información* *c) Filtro Basado en Entropía - Ganancia de ratio* **VARIABLES EN EL MODELO:** SURVIVED, TITULOPASAJERO, AGEMALE, SEXNUEVO, AGEFEMALE, PCLASS, SOBREVFAMILIA, CABINNUEVO FARENUEVO, SIBSPNUEVO, PARCHNUEVO **VARIABLES EXCLUIDAS:** TIPOTICKET Y EMBARKEDNUEVO

RESUMEN DEL ALGORITMO RANDOM FOREST CON WEKA

=== Summary ===

Correctly Classified Instances	***139***	***78.0899 %***
Incorrectly Classified Instances	***39***	***21.9101 %***
Kappa statistic	***0.538***	
Mean absolute error	***0.2818***	
Root mean squared error	***0.4257***	
Relative absolute error	***59.9576 %***	
Root relative squared error	***88.3019 %***	
Total Number of Instances	***178***	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.796	0.246	0.849	0.796	0.822	0.787	0
	0.754	0.204	0.681	0.754	0.715	0.787	1
Weighted Avg.	0.781	0.231	0.788	0.781	0.783	0.787	

=== Confusion Matrix ===

a	b	<-- classified as
90	23	a = 0
16	49	b = 1

2

TECNICAS DE FILTRO:

c) *Filtro Random Forest*

VARIABLES EN EL MODELO:

SURVIVED, FARENUEVO, PCLASS, SIBSPNUEVO, TITULOPASAJERO, AGEMALE, CABINNUEVO, EMBARKEDNUEVO, SOBREVFAMILIA, AGEFEMALE, TIPOTICKET

VARIABLES EXCLUIDAS:

PARCHNUEVO Y SEXNUEVO

RESUMEN DEL ALGORITMO RANDOM FOREST CON WEKA

=== Summary ===

Correctly Classified Instances	***144***	***80.8989 %***
Incorrectly Classified Instances	***34***	***19.1011 %***
Kappa statistic	***0.588***	
Mean absolute error	***0.2479***	
Root mean squared error	***0.3846***	
Relative absolute error	***52.7422 %***	
Root relative squared error	***79.7927 %***	
Total Number of Instances	***178***	

=== Detailed Accuracy By Class ===

	TP Rate FP Rate Precision Recall F-Measure ROC Area Class ***0.85 0.262 0.85 0.85 0.85 0.846 0*** ***0.738 0.15 0.738 0.738 0.738 0.846 1*** ***Weighted Avg. 0.809 0.221 0.809 0.809 0.809 0.846*** ***=== Confusion Matrix ===*** ***a b <-- classified as*** ***96 17 | a = 0*** ***17 48 | b = 1***

Para el primer caso, en el *20%* de los datos de entrenamiento (de prueba) el error es de 0.219101.

En el segundo caso, en el *20%* de los datos de entrenamiento (de prueba) el error es de 0.191011.

5.3.2. Mejor selección del subconjunto de variables y algoritmos con el enfoque de selección de subconjunto Wrapper

En esta sección se realizará la estimación de modelos, considerando los resultados hallados en la sección 4.4.2.1.2. (Enfoque de selección de subconjunto Wrapper). Se utilizarán los resultados de la Tabla 48, y se ejecutarán los diez algoritmos de esta tabla hallándose el mejor conjunto de variables en cada caso. Para esto se utilizarán los software R y Python (con el módulo scikitlearn).

5.3.2.1. Estimación con R, con el enfoque de selección de subconjuntos

Aquí se utilizarán los algoritmos señalados en la sección 4.4.2.1.2 (Tabla 48); los códigos están en R y se encuentran en el Apéndice C, en las tablas C1, C2, C3, C4, C5, C6, C7, C8, C9 y C10.

Tabla 52

Resultados de las estimaciones con R, para el enfoque se selección de subconjuntos

ALGORITMOS
Algoritmo Árbol de Decisión: **METODO DE SELECCIÓN DE SUBCONJUNTO**: • GREEDY SEARCH BACKWARD

VARIABLES EN EL MODELO:
SURVIVED, PCLASS, TITULOPASAJERO, SEXNUEVO, AGEMALE, AGEFEMALE, SIBSPNUEVO, PARCHNUEVO, TIPOTICKET, CABINNUEVO, SOBREVFAMILIA

VARIABLES EXCLUIDAS:
FARENUEVO Y EMBARKEDNUEVO

RESUMEN DEL ALGORITMO ÁRBOL DE DECISIÓN RPART CON R

```
[1] "Error de los datos de prueba (20% de los datos)"
[1] 0.150838
[1] "Matriz de Confusión de los Datos de Prueba (20% de los datos)"
Predicho
Real   0   1
   0 102  10
   1  17  50

 [1] "Reglas de Decisión para el algoritmo RPART"
Rule number: 7 [as.factor(survived)=1 cover=145 (20%) prob=0.94]
titulopasajero>=8.5
pclass< 2.5

 Rule number: 27 [as.factor(survived)=1 cover=92 (13%) prob=0.64]
titulopasajero>=8.5
pclass>=2.5
sibspnuevo< 2.5
agefemale< 4.5

 Rule number: 26 [as.factor(survived)=0 cover=18 (3%) prob=0.22]
titulopasajero>=8.5
pclass>=2.5
sibspnuevo< 2.5
agefemale>=4.5

 Rule number: 2 [as.factor(survived)=0 cover=428 (60%) prob=0.17]
titulopasajero< 8.5

 Rule number: 12 [as.factor(survived)=0 cover=29 (4%) prob=0.14]
titulopasajero>=8.5
pclass>=2.5
sibspnuevo>=2.5

[1] "Área bajo la Curva ROC para los datos de entrenamiento"
[1] 0.8297316
```

Algoritmo Grandient Boosting:
METODO DE SELECCIÓN DE SUBCONJUNTO:

- BESTFIRST

VARIABLES EN EL MODELO:
SURVIVED, TITULOPASAJERO,SEXNUEVO, AGEMALE, AGEFEMALE, SIBSPNUEVO, EMBARKEDNUEVO, SOBREVFAMILIA

VARIABLES EXCLUIDAS:
PCLASS , PARCHNUEVO, TIPOTICKET, FARENUEVO Y CABINNUEVO

RESUMEN DEL ALGORITMO GRADIENT BOOSTING CON R

```
[1] "Error de los datos de prueba (20% de los datos)"
[1] 0.150838
```

```
[1] "Matriz de Confusión de los Datos de Prueba (20% de los datos)"
    Predicho
Real   0   1
   0 100  12
   1  15  52
[1] "Área bajo la Curva ROC para los datos de entrenamiento"
[1] 0.8182942
```

Algoritmo Ada Boost:

METODO DE SELECCIÓN DE SUBCONJUNTO:

- GREEDY SEARCH BACKWARD

VARIABLES EN EL MODELO:

SURVIVED, PCLASS, TITULOPASAJERO, SEXNUEVO, AGEMALE, AGEFEMALE, SIBSPNUEVO, PARCHNUEVO, TIPOTICKET, CABINNUEVO, EMBARKEDNUEVO, SOBREVFAMILIA

VARIABLES EXCLUIDAS:

FARENUEVO

RESUMEN DEL ALGORITMO ADA BOOST CON R

```
[1] "Error de los datos de prueba (20% de los datos)"
[1] 0.1396648
[1] "Matriz de Confusión de los Datos de Prueba (20% de los datos)"
    Predicho
Real   0   1
   0 104   8
   1  17  50
[1] "Área bajo la Curva ROC para los datos de entrenamiento"
 [1] 0.9054837
```

Algoritmo Bagging:

METODO DE SELECCIÓN DE SUBCONJUNTO:

- HILL CLIMBING

VARIABLES EN EL MODELO:

SURVIVED, TITULOPASAJERO, SEXNUEVO, SIBSPNUEVO, PARCHNUEVO, TIPOTICKET

VARIABLES EXCLUIDAS:

PCLASS, AGEMALE, AGEFEMALE, FARENUEVO, CABINNUEVO, EMBARKEDNUEVO Y SOBREVFAMILIA

RESUMEN DEL ALGORITMO BAGGING CON R

```
[1] "Error de los datos de prueba (20% de los datos)"
[1] 0.1396648
[1] "Matriz de Confusión de los Datos de Prueba (20% de los datos)"
    Predicho
Real   0   1
   0 104   8
   1  17  50
[1] "Área bajo la Curva ROC para los datos de entrenamiento"
[1] 0.8342334
```

Algoritmo Random Forest:

METODO DE SELECCIÓN DE SUBCONJUNTO:

- HILL CLIMBING

VARIABLES EN EL MODELO:
SURVIVED, TITULOPASAJERO, SEXNUEVO, AGEMALE, SIBSPNUEVO, PARCHNUEVO, TIPOTICKET, CABINNUEVO, EMBARKEDNUEVO

VARIABLES EXCLUIDAS:
PCLASS, AGEFEMALE, FARENUEVO Y SOBREVFAMILIA

RESUMEN DEL ALGORITMO RANDOM FOREST CON R

```
[1] "Error de los datos de prueba (20% de los datos)"
[1] 0.1284916
[1] "Matriz de Confusión de los Datos de Prueba (20% de los datos)"
    Predicho
Real   0   1
   0 102  10
   1  13  54
[1] "Importancia de variables según orden (datos de entrenamiento 80% de
datos)"
                  0     1 MeanDecreaseAccuracyMeanDecreaseGini
sibspnuevo     47.48  3.22                42.23            10.11
titulopasajero 31.80 15.80                34.24            35.69
cabinnuevo     21.28  6.76                22.20            14.35
agemale        13.52 11.64                17.75            25.70
sexnuevo       13.16  0.53                13.84            14.45
parchnuevo     13.95 -3.74                 9.16             5.21
embarkednuevo   6.06 -1.18                 4.32             4.29
tipoticket      6.18 -1.27                 4.06            10.40
[1] "Área bajo la Curva ROC para los datos de entrenamiento"
[[1]]
[1] 0.8813896
```

Algoritmo Support Vector Machine SVM:

METODO DE SELECCIÓN DE SUBCONJUNTO:

- HILL CLIMBING

VARIABLES EN EL MODELO:
SURVIVED, TITULOPASAJERO, SEXNUEVO, AGEFEMALE , SIBSPNUEVO, PARCHNUEVO, TIPOTICKET, CABINNUEVO

VARIABLES EXCLUIDAS:
PCLASS, AGEMALE, FARENUEVO, EMBARKEDNUEVO Y SOBREVFAMILIA

RESUMEN DEL ALGORITMO SUPPORT VECTOR MACHINE SVM CON R

```
[1] "Error de los datos de prueba (20% de los datos)"
[1] 0.1340782
[1] "Matriz de Confusión de los Datos de Prueba (20% de los datos)"
    Predicho
Real   0   1
   0 102  10
   1  14  53
[1] "Área bajo la Curva ROC para los datos de entrenamiento"
 [1] 0.8455294
```

Algoritmo Regresión Logística:

METODO DE SELECCIÓN DE SUBCONJUNTO:

- BEST FIRST

VARIABLES EN EL MODELO:
SURVIVED, TITULOPASAJERO, SEXNUEVO, SIBSPNUEVO, PARCHNUEVO, TIPOTICKET, EMBARKEDNUEVO, SOBREVFAMILIA

VARIABLES EXCLUIDAS:
PCLASS, AGEMALE, AGEFEMALE, FARENUEVO Y CABINNUEVO

RESUMEN DEL ALGORITMO REGRESIÓN LOGÍSTICA CON R

```
[1] "Error de los datos de prueba (20% de los datos)"
[1] 0.1620112
[1] "Matriz de Confusión de los Datos de Prueba (20% de los datos)"
    Predicho
Real   0   1
   0 101  11
   1  18  49
[1] "Área bajo la Curva ROC para los datos de entrenamiento"
 [1] 0.8315665
```

Algoritmo Redes neuronales:

METODO DE SELECCIÓN DE SUBCONJUNTO:

- GREEDY SEARCH BACKWARD

VARIABLES EN EL MODELO:
SURVIVED, PCLASS , TITULOPASAJERO, SEXNUEVO, AGEMALE, AGEFEMALE, SIBSPNUEVO, PARCHNUEVO, TIPOTICKET, CABINNUEVO, EMBARKEDNUEVO, SOBREVFAMILIA
VARIABLES EXCLUIDAS:
FARENUEVO

RESUMEN DEL ALGORITMO REDES NEURONALES NNET CON R

```
[1] "Error de los datos de prueba (20% de los datos)"
[1] 0.1620112
[1] "Matriz de Confusión de los Datos de Prueba (20% de los datos)"
    Predicho
Real  0  1
   0 96 16
   1 13 54
[1] "Área bajo la Curva ROC para los datos de entrenamiento"
 [1] 0.9354649
```

Algoritmo Naive Bayes:

METODO DE SELECCIÓN DE SUBCONJUNTO:

- GREEDY SEARCH BACKWARD

VARIABLES EN EL MODELO:
SURVIVED ~ TITULOPASAJERO + SEXNUEVO + AGEMALE + SIBSPNUEVO + PARCHNUEVO + EMBARKEDNUEVO + SOBREVFAMILIA

VARIABLES EXCLUIDAS:
PCLASS, AGEFEMALE, TIPOTICKET, FARENUEVO Y CABINNUEVO

RESUMEN DEL ALGORITMO BAYES NAIVES CON R

```
[1] "Error de los datos de prueba (20% de los datos)"
[1] 0.150838
[1] "Matriz de Confusión de los Datos de Prueba (20% de los datos)"
      Predicho
Real   0   1
   0 101  11
   1  16  51
```

Algoritmo k-vecindad más cercana:

METODO DE SELECCIÓN DE SUBCONJUNTO:

- BESTFIRST

VARIABLES EN EL MODELO:

SURVIVED, SEXNUEVO, EMBARKEDNUEVO, SOBREVFAMILIA

VARIABLES EXCLUIDAS:

PCLASS, TITULOPASAJERO, AGEMALE, AGEFEMALE, PARCHNUEVO, SIBSPNUEVO, TIPOTICKET, FARENUEVO Y CABINNUEVO

RESUMEN DEL ALGORITMO K-VECINDAD MAS CERCANA CON R

```
[1] 0.1731844
[1] "Matriz de Confusión de los Datos de Prueba (20% de los datos)"
    Predicho
Real   0   1
   0 103   9
   1  22  45
```

Un análisis de la Tabla 52 permite apreciar que el algoritmo que da mejor resultado es el RANDOM FOREST con un error de 0.1284. Los otros algoritmos que producen también buenos resultados, son el algoritmo ADA BOOST con 0.13966480, el algoritmo BAGGING con 0.1396648 y el algoritmo VECTOR SUPPORT VECTOR MACHINE SVM con 0.130782.

Se observa también que se ha añadido en esta tabla, como un resultado importante del algoritmo del Árbol de Decisión RPART, sus reglas de decisión.

5.3.2.2. Estimación con Python, con el enfoque se selección de subconjuntos

En este caso se utilizarán los algoritmos señalados en la sección 5.2, (Tabla 58), los códigos están en Python y se encuentra en el Apéndice C en las tablas C11, C12, C13, C14, C15, C16 y C17.

Tabla 53

Resultados de las estimaciones con Python, para el enfoque de selección de subconjuntos

ALGORITMOS
Algoritmo Árbol de Decisión: **METODO DE SELECCIÓN DE SUBCONJUNTO**: • GREEDY SEARCH BACKWARD **VARIABLES EN EL MODELO:** SURVIVED, PCLASS, TITULOPASAJERO, SEXNUEVO, AGEMALE, AGEFEMALE, SIBSPNUEVO, PARCHNUEVO, TIPOTICKET, CABINNUEVO, SOBREVFAMILIA **VARIABLES EXCLUIDAS:** FARENUEVO Y EMBARKEDNUEVO **RESUMEN DEL ALGORITMO ÁRBOL DE DECISIÓN RPART CON PYTHON** Matriz de Confusión de los Datos de Prueba (20% de los datos preds actual 0 1 0 104 14 1 17 44 Error de los datos de prueba (20% de los datos) 0.173184357542
Algoritmo Gradient Boosting: **METODO DE SELECCIÓN DE SUBCONJUNTO**: • BESTFIRST **VARIABLES EN EL MODELO:** SURVIVED, TITULOPASAJERO, SEXNUEVO, AGEMALE, AGEFEMALE, SIBSPNUEVO, EMBARKEDNUEVO, SOBREVFAMILIA **VARIABLES EXCLUIDAS:** PCLASS , PARCHNUEVO, TIPOTICKET, FARENUEVO Y CABINNUEVO **RESUMEN DEL ALGORITMO GRADIENT BOOSTING CON PYTHON** Matriz de Confusión de los Datos de Prueba (20% de los datos preds 0 1 actual 0 106 12 1 12 49 Error de los datos de prueba (20% de los datos) 0.134078212291
Algoritmo Random Forest: **METODO DE SELECCIÓN DE SUBCONJUNTO**: • HILL CLIMBING

VARIABLES EN EL MODELO:
SURVIVED, TITULOPASAJERO, SEXNUEVO, AGEMALE, SIBSPNUEVO, PARCHNUEVO, TIPOTICKET, CABINNUEVO, EMBARKEDNUEVO

VARIABLES EXCLUIDAS:
PCLASS, AGEFEMALE, FARENUEVO Y SOBREVFAMILIA

RESUMEN DEL ALGORITMO RANDOM FOREST CON PYTHON

```
Matriz de Confusión de los Datos de Prueba (20% de los datos
preds     0   1
actual
0       104  14
1        14  47
Error de los datos de prueba (20% de los datos)
0.156424581006
```

Algoritmo Support Vector Machine SVM:

METODO DE SELECCIÓN DE SUBCONJUNTO:

- HILL CLIMBING

VARIABLES EN EL MODELO:
SURVIVED , TITULOPASAJERO, SEXNUEVO, AGEFEMALE , SIBSPNUEVO, PARCHNUEVO, TIPOTICKET, CABINNUEVO

VARIABLES EXCLUIDAS:
PCLASS, AGEMALE, FARENUEVO, EMBARKEDNUEVO Y SOBREVFAMILIA

RESUMEN DEL ALGORITMO SUPPORT VECTOR MACHINE SVM CON PYTHON

```
Matriz de Confusión de los Datos de Prueba (20% de los datos
preds     0   1
actual
0       112  15
1        10  59
Error de los datos de prueba (20% de los datos)
0.127551020408
```

Algoritmo Regresión Logística:

METODO DE SELECCIÓN DE SUBCONJUNTO:

- BEST FIRST

VARIABLES EN EL MODELO:
SURVIVED , TITULOPASAJERO, SEXNUEVO, SIBSPNUEVO, PARCHNUEVO, TIPOTICKET, EMBARKEDNUEVO, SOBREVFAMILIA

VARIABLES EXCLUIDAS:
PCLASS, AGEMALE, AGEFEMALE, FARENUEVO Y CABINNUEVO

RESUMEN DEL ALGORITMO REGRESIÓN LOGÍSTICA CON PYTHON

```
Matriz de Confusión de los Datos de Prueba (20% de los datos
preds    0   1
actual
0       92   6
1       21  60
Error de los datos de prueba (20% de los datos)
0.150837988827
```

Algoritmo Naive Bayes:

METODO DE SELECCIÓN DE SUBCONJUNTO:

- GREEDY SEARCH BACKWARD

VARIABLES EN EL MODELO:

SURVIVED , TITULOPASAJERO, SEXNUEVO, AGEMALE, SIBSPNUEVO, PARCHNUEVO, EMBARKEDNUEVO, SOBREVFAMILIA

VARIABLES EXCLUIDAS:

PCLASS, AGEFEMALE, TIPOTICKET, FARENUEVO Y CABINNUEVO

RESUMEN DEL ALGORITMO NAIVE BAYES CON PYTHON

```
Matriz de Confusión de los Datos de Prueba (20% de los datos
preds    0   1
actual
0       88  10
1       18  63
Error de los datos de prueba (20% de los datos)
0.156424581006
```

Algoritmo k-vecindad mas cercana:

METODO DE SELECCIÓN DE SUBCONJUNTO:

- BESTFIRST

VARIABLES EN EL MODELO:

SURVIVED, SEXNUEVO, EMBARKEDNUEVO, SOBREVFAMILIA

VARIABLES EXCLUIDAS:

PCLASS, TITULOPASAJERO, AGEMALE, AGEFEMALE, PARCHNUEVO, SIBSPNUEVO, TIPOTICKET, FARENUEVO Y CABINNUEVO

RESUMEN DEL ALGORITMO K-VECINDAD MAS CERCANA CON PYTHON

```
Matriz de Confusión de los Datos de Prueba (20% de los datos)
preds     0   1
actual
0       104  14
1        21  40
Error de los datos de prueba (20% de los datos)
0.195530726257
```

De la Tabla 53 se resumen a continuación los errores producidos en el 20% de los datos de entrenamiento (891 observaciones).

- Error de Vector Support Machine SVM : 0.127551020408
- Error de Gradient Boosting : 0.134078212291
- Error de Raandom Forest : 0.156424581006
- Error de Regresión Logistica : 0.150837988827
- Error de Naive Bayes : 0.156424581006
- Error de Árbol de Decision : 0.173184357542
- Error de k-vecindad mas cercana : 0.195530726257

Como se puede observar el algoritmo que da menor error es el SUPPORT VECTOR MACHINE SVM, y luego el GRADIENT BOOSTING, en la siguiente sección se evaluarán estos resultados.

5.4. Evaluación de los modelos

En esta etapa, se van a evaluar e interpretar los resultados extraídos (reglas, fiabilidad, etc.). En este análisis se van a tener en cuenta los pasos de pre-procesamiento con respecto a su efecto sobre los resultados de los algoritmos de minería de datos. De acuerdo a las evaluaciones y resultados se añadirán o no otras características. Este paso se centra en la comprensión y el análisis de la utilidad del modelo inducido. En esta sección el conocimiento descubierto también será documentado para su uso posterior.

5.4.1. Tareas realizadas en la minería de datos previas a la evaluación

En esta sección, como una explicación previa a la evaluación, se debe señalar que se han realizado hasta el momento las tareas tales como el pre-procesamiento y limpieza de de los datos en el capítulo 4, siendo una de las labores realizadas, la imputación de valores perdidos en algunas variables de los conjuntos de datos.

Luego se realizó la verificación de los conjuntos de datos (TRAIN y TEST) en forma estadísticas comprobando si provenían de la misma población.

Luego se realizó la transformación de los datos, por medio de la extracción de características y la selección de características en donde la selección de variables, se realizó bajo dos enfoques que son:

a) Enfoque de rankeo de características
b) Enfoque se selección de subconjunto con el método Wrapper

Por último se eligieron diez algoritmos que iban a ser utilizados en este trabajo para realizar la investigación de minería de datos.

5.4.2. Comparación de los resultados obtenidos en la minería de datos

A continuación se presentan las comparaciones entre los algoritmos seleccionados en el capítulo anterior. La comparación se realiza utilizando los errores obtenidos por los algoritmos, tanto en el conjunto de entrenamiento (TRAIN) como en el conjunto final de prueba (TEST).

Tabla 54
Evaluación de los algoritmos de minería de datos

Nº Ord	Algoritmo	Software	Enfoque de selección de variables	Variables en el modelo	TRAIN Partición Entrenamiento		TRAIN Partición Test	Error en TEST (Kaggle)
					ROC	Error	Error	
1	RANDOM FOREST	R	RANKEO DE VARIABLES	SURVIVED, TITULOPASAJERO, AGEMALE, SEXNUEVO, AGEFEMALE, PCLASS , SOBREVFAMILIA, CABINNUEVO, FARENUEVO, SIBSPNUEVO, PARCHNUEVO	0.89	0.1896	0.139665	0.2488
2	RANDOM FOREST	R	RANKEO DE VARIABLES	SURVIVED, FARENUEVO, PCLASS,SIBSPNUEVO, TITULOPASAJERO, AGEMALE, CABINNUEVO, EMBARKEDNUEVO, SOBREVFAMILIA, AGEFEMALE, TIPOTICKET	0.92	0.1882	0.150838	0.2249
3	RANDOM FOREST	PYTHON	RANKEO DE VARIABLES	SURVIVED, TITULOPASAJERO, AGEMALE, SEXNUEVO, AGEFEMALE, PCLASS , SOBREVFAMILIA, CABINNUEVO, FARENUEVO, SIBSPNUEVO, PARCHNUEVO			0.156424	0.2345
4	RANDOM FOREST	PYTHON	RANKEO DE VARIABLES	SURVIVED, FARENUEVO, PCLASS,SIBSPNUEVO, TITULOPASAJERO, AGEMALE, CABINNUEVO, EMBARKEDNUEVO, SOBREVFAMILIA, AGEFEMALE, TIPOTICKET			0.150837	0.2488
5	RANDOM FOREST	WEKA	RANKEO DE VARIABLES	SURVIVED, TITULOPASAJERO, AGEMALE, SEXNUEVO, AGEFEMALE, PCLASS , SOBREVFAMILIA, CABINNUEVO, FARENUEVO, SIBSPNUEVO, PARCHNUEVO			0.219101	0.2441
6	RANDOM FOREST	WEKA	RANKEO DE VARIABLES	SURVIVED, FARENUEVO, PCLASS,SIBSPNUEVO, TITULOPASAJERO, AGEMALE, CABINNUEVO, EMBARKEDNUEVO, SOBREVFAMILIA, AGEFEMALE, TIPOTICKET			0.191011	0.2249

Continuación de la Tabla 54
Evaluación de los algoritmos de minería de datos

Nº Ord	Algoritmo	Software	Enfoque de selección de variables	Variables en el modelo	TRAIN Partición Entrenamiento		TRAIN Partición Test	Error en TEST (Kaggle)
					ROC	Error	Error	
7	ÁRBOL DE DECISIÓN	R	SELECCIÓN DE SUBCONJUNTO (WRAPPER)	SURVIVED,PCLASS,TITULOPASAJERO,SEXNUEVO, AGEMALE, AGEFEMALE, SIBSPNUEVO, PARCHNUEVO,TIPOTICKET, CABINNUEVO, SOBREVFAMILIA	0.83		0.150838	0.22488
8	GRADIENT BOOSTING	R	SELECCIÓN DE SUBCONJUNTO (WRAPPER)	SURVIVED,TITULOPASAJERO,SEXNUEVO, AGEMALE, AGEFEMALE, SIBSPNUEVO, EMBARKEDNUEVO,SOBREVFAMILIA	0.82		0.150838	0.21531
9	ADA BOOST (ADA)	R	SELECCIÓN DE SUBCONJUNTO (WRAPPER)	SURVIVED, PCLASS, TITULOPASAJERO, SEXNUEVO, AGEMALE, AGEFEMALE, SIBSPNUEVO, PARCHNUEVO,TIPOTICKET, CABINNUEVO, EMBARKEDNUEVO, SOBREVFAMILIA	0.90		0.139665	0.21053
10	BAGGING	R	SELECCIÓN DE SUBCONJUNTO (WRAPPER)	SURVIVED, TITULOPASAJERO,SEXNUEVO, SIBSPNUEVO, PARCHNUEVO, TIPOTICKET	0.83		0.139665	0.20096
11	RANDOM FOREST	R	SELECCIÓN DE SUBCONJUNTO (WRAPPER)	SURVIVED,TITULOPASAJERO,SEXNUEVO, AGEMALE, SIBSPNUEVO, PARCHNUEVO, TIPOTICKET, CABINNUEVO,EMBARKEDNUEVO	0.88		0.128492	0.21531
12	SUPPORT VECTOR MACHINE (SVM)	R	SELECCIÓN DE SUBCONJUNTO (WRAPPER)	SURVIVED, TITULOPASAJERO,SEXNUEVO, AGEFEMALE, SIBSPNUEVO, PARCHNUEVO, TIPOTICKET, CABINNUEVO	0.84		0.134078	0.2201
13	REGRESIÓN LOGÍSTICA	R	SELECCIÓN DE SUBCONJUNTO (WRAPPER)	SURVIVED, TITULOPASAJERO,SEXNUEVO, SIBSPNUEVO, PARCHNUEVO, TIPOTICKET, EMBARKEDNUEVO, SOBREVFAMILIA	0.83		0.162011	0.22967

Continuación de la Tabla 54
Evaluación de los algoritmos de minería de datos

Nº Ord	Algoritmo	Software	Enfoque de selección de variables	Variables en el modelo	TRAIN Partición Entrenamiento		TRAIN Partición Test	Error en TEST (Kaggle)
					ROC	Error	Error	
14	REDES NEURALES (NNET)	R	SELECCIÓN DE SUBCONJUNTO (WRAPPER)	SURVIVED, PCLASS, TITULOPASAJERO, SEXNUEVO, AGEMALE , AGEFEMALE, SIBSPNUEVO, PARCHNUEVO, TIPOTICKET, CABINNUEVO, EMBARKEDNUEVO, SOBREVFAMILIA	0.93		0.162011	0.23923
15	NAÏVE BAYES	R	SELECCIÓN DE SUBCONJUNTO (WRAPPER)	SURVIVED, TITULOPASAJERO, SEXNUEVO, AGEMALE, SIBSPNUEVO, PARCHNUEVO, EMBARKEDNUEVO, SOBREVFAMILIA			0.150838	0.21053
16	K.VECINDAD MAS CERCANA	R	SELECCIÓN DE SUBCONJUNTO (WRAPPER)	SURVIVED, SEXNUEVO, EMBARKEDNUEVO, SOBREVFAMILIA			0.173184	
17	ÁRBOL DE DECISIÓN	PYTHON	SELECCIÓN DE SUBCONJUNTO (WRAPPER)	SURVIVED,PCLASS,TITULOPASAJERO,SEXNUEVO, AGEMALE, AGEFEMALE, SIBSPNUEVO, PARCHNUEVO,TIPOTICKET, CABINNUEVO, SOBREVFAMILIA			0.173184	0.24402
18	GRADIENT BOOSTING	PYTHON	SELECCIÓN DE SUBCONJUNTO (WRAPPER)	SURVIVED,TITULOPASAJERO,SEXNUEVO, AGEMALE, AGEFEMALE, SIBSPNUEVO, EMBARKEDNUEVO,SOBREVFAMILIA			0.134078	0.21531
19	RANDOM FOREST	PYTHON	SELECCIÓN DE SUBCONJUNTO (WRAPPER)	SURVIVED,TITULOPASAJERO,SEXNUEVO, AGEMALE, SIBSPNUEVO, PARCHNUEVO, TIPOTICKET, CABINNUEVO,EMBARKEDNUEVO			0.156425	0.21531
20	SUPPORT VECTOR MACHINE (SVM)	PYTHON	SELECCIÓN DE SUBCONJUNTO (WRAPPER)	SURVIVED, TITULOPASAJERO,SEXNUEVO, AGEFEMALE, SIBSPNUEVO, PARCHNUEVO, TIPOTICKET, CABINNUEVO			0.127551	0.21531

Continuación de la Tabla 54
Evaluación de los algoritmos de minería de datos

Nº Ord	Algoritmo	Software	Enfoque de selección de variables	Variables en el modelo	TRAIN Partición Entrenamiento		TRAIN Partición Test	Error en TEST (Kaggle)
					ROC	Error	Error	
21	NAÏVE BAYES	PYTHON	SELECCIÓN DE SUBCONJUNTO (WRAPPER)	SURVIVED, TITULOPASAJERO, SEXNUEVO, AGEMALE, SIBSPNUEVO, PARCHNUEVO, EMBARKEDNUEVO, SOBREVFAMILIA			0.156425	0.21531
22	REGRESIÓN LOGÍSTICA	PYTHON	SELECCIÓN DE SUBCONJUNTO (WRAPPER)	SURVIVED, TITULOPASAJERO,SEXNUEVO, SIBSPNUEVO, PARCHNUEVO, TIPOTICKET, EMBARKEDNUEVO, SOBREVFAMILIA			0.150838	0.21531
23	K-VECINDAD MAS CERCANA	PYTHON	SELECCIÓN DE SUBCONJUNTO (WRAPPER)	SURVIVED, SEXNUEVO, EMBARKEDNUEVO, SOBREVFAMILIA			0.195530	0.22488

De la Tabla 54, se obtiene la Tabla 55 (extraída de la Tabla 54) para mostrar los mejores algoritmos de acuerdo al método de selección de variables y de acuerdo al software utilizado.

Tabla 55

Mejores algoritmos según los métodos de selección de atributos y software utilizados

Numero	Algoritmo	Metodo de selección de atributos	SOFTWARE	TRAIN Particion Test Error	Error en TEST (Kaggle)
1	RANDOM FOREST	RANKEO DE VARIABLES	R	0.150838	0.2249
2	RANDOM FOREST	RANKEO DE VARIABLES	PYTHON	0.156424	0.2345
3	RANDOM FOREST	RANKEO DE VARIABLES	WEKA	0.191011	0.2249
4	GRADIENT BOOSTING	SELECCIÓN DE SUBCONJUNTO (WRAPPER)	R	0.150838	0.21531
5	ADA BOOST (ADA)	SELECCIÓN DE SUBCONJUNTO (WRAPPER)	R	0.139665	0.21053
6	BAGGING	SELECCIÓN DE SUBCONJUNTO (WRAPPER)	R	0.139665	0.20096
7	RANDOM FOREST	SELECCIÓN DE SUBCONJUNTO (WRAPPER)	R	0.128492	0.21531
8	SUPPORT VECTOR MACHINE (SVM)	SELECCIÓN DE SUBCONJUNTO (WRAPPER)	R	0.134078	0.2201
9	NAÏVE BAYES	SELECCIÓN DE SUBCONJUNTO (WRAPPER)	R	0.150838	0.21053
10	GRADIENT BOOSTING	SELECCIÓN DE SUBCONJUNTO (WRAPPER)	PYTHON	0.134078	0.21531
11	RANDOM FOREST	SELECCIÓN DE SUBCONJUNTO (WRAPPER)	PYTHON	0.156425	0.21531
12	SUPPORT VECTOR MACHINE (SVM)	SELECCIÓN DE SUBCONJUNTO (WRAPPER)	PYTHON	0.127551	0.21531
13	NAÏVE BAYES	SELECCIÓN DE SUBCONJUNTO (WRAPPER)	PYTHON	0.156425	0.21531
14	REGRESIÓN LOGÍSTICA	SELECCIÓN DE SUBCONJUNTO (WRAPPER)	PYTHON	0.150838	0.21531

En la Tabla 55 se observa que el mejor algoritmo es el BAGGING con un error 0.20096, luego vendrían el ADABOOST y el NAIVE BAYES con un error de 0.21053 ambos, y después estarían el GRADIENT BOOSTING y el RANDOM FOREST con un error de 0.21531, todos implementados con R (los errores son en el conjunto de datos de prueba TEST).

En lo que respecta al software PYTHON se obtuvieron buenos resultados con los algoritmos GRADIENT BOOSTING, RANDOM FOREST, SUPPORT VECTOR MACHINE SVM, NAIVE BAYES Y REGRESION LOGISTICA, todos con el mismo error de 0.21531 en los datos de prueba TEST.

5.4.3. Evaluación del algoritmo árbol de decisión RPART de R

En esta parte se evalúa el algoritmo ÁRBOL DE DECISIÓN RPART que se obtuvo con R. Este algoritmo produjo un error en la partición test (de los datos de entrenamiento) de 0.150838, y en los datos de prueba TEST dio un error de 0.22488. Los resultados con este algoritmo aparecen en la Tabla 52. Además el resultado con R para el algoritmo RPART, incluye las reglas de decisión tal y como se describen a continuación.

[1] "Reglas de Decisión para el algoritmo RPART"
Rule number: 7 [as.factor(survived)=1 cover=145 (20%) prob=0.94]
titulopasajero>=8.5
pclass< 2.5

Rule number: 27 [as.factor(survived)=1 cover=92 (13%) prob=0.64]
titulopasajero>=8.5
pclass>=2.5
sibspnuevo< 2.5
agefemale< 4.5

Rule number: 26 [as.factor(survived)=0 cover=18 (3%) prob=0.22]
titulopasajero>=8.5
pclass>=2.5
sibspnuevo< 2.5
agefemale>=4.5

Rule number: 2 [as.factor(survived)=0 cover=428 (60%) prob=0.17]
titulopasajero< 8.5

Rule number: 12 [as.factor(survived)=0 cover=29 (4%) prob=0.14]
titulopasajero>=8.5
pclass>=2.5
sibspnuevo>=2.5

Estas reglas de decisión son reglas separadas, y están ordenadas en orden descendente de la probabilidad de ocurrencia del evento (prob).

La descripción de las reglas sería:

1. REGLA 7: El 94% de las veces cuando la variable TITULOPASAJERO es mayor o igual que 8.5 y la variable PCLASS es menor que 2.5, entonces el pasajero ***sobrevive***.

2. REGLA 27: El 64% de las veces cuando la variable TITULOPASAJERO es mayor o igual que 8.5 y la variable PCLASS es mayor o igual que 2.5, y la variables SIBSNUEVO es menor que 2.5 y AGEFEMALE es menor que 4.5, entonces el pasajero ***sobrevive***.

3. REGLA 26: El 22% de las veces cuando la variable TITULOPASAJERO es mayor o igual que 8.5 y la variable PCLASS es mayor o igual que 2.5 y la variables SIBSNUEVO es menor que 2.5 y AGEFEMALE es mayor o igual que 4.5, entonces el pasajero ***no sobrevive***.

4. REGLA 2: El 17% de las veces cuando la variable TITULOPASAJERO es menor que 8.5, entonces el pasajero ***no sobrevive***.

5. REGLA 26: El 12% de las veces cuando la variable TITULOPASAJERO es mayor o igual que 8.5 y la variable PCLASS es mayor o igual que 2.5, y la variable SIBSNUEVO es mayor o igual que 2.5, entonces el pasajero ***no sobrevive***.

El gráfico que esquematiza estas reglas de decisión, se muestra la figura 31.

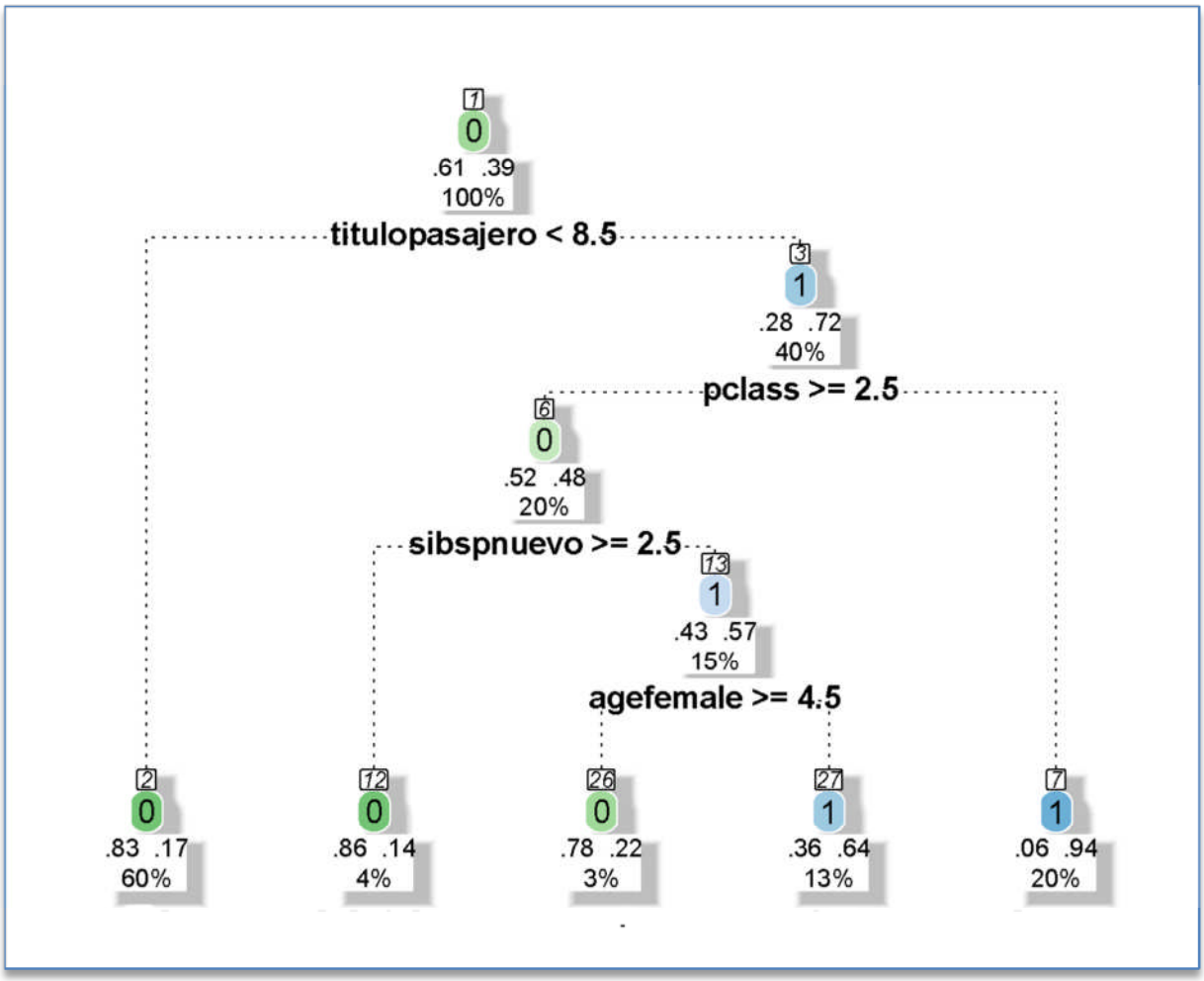
5.4.4. Importancia de las variables según orden con diferentes métodos

En esta parte se analiza la importancia de las variables de la Tabla 46, es decir todas las variables de este estudio. Para esto se considera lo hallado en la sección 4.3.2.1.1 del "Enfoque de Rankeo de Características", donde se identificaron los mejores subconjuntos de variables por el método de rankeo de características. Este orden de variables se tomará en cuenta aquí para determinar la importancia de las variables bajo este enfoque. Los siguientes son los filtros analizados:

a) Filtro Chi_squared
b) Filtro de ganancia de información
c) Filtro de ganancia de ratio
d) Filtro RandomForest

Las correspondientes ordenaciones de variables se encuentran en la Tabla 56 de la columna 2 a la columna 9.

Figura 31
Árbol de Decisión RPART



Dado que el algoritmo RANDOM FOREST es un algoritmo confiable, pues provee mejores resultados en comparación con otros algoritmos de minería de datos; y siendo uno de los resultados del algoritmo en R el orden de importancia de las variables; se tomará como base este algoritmo para hacer el ordenamiento de variables según su orden de importancia.

Teniendo en cuenta lo anterior se realizó una nueva estimación del modelo de predicción Random Forest con el conjunto de entrenamiento TRAIN, tomando en cuenta todas sus variables (Tabla 46); para así determinar la importancia de las variables bajo el enfoque del algoritmo Random Forest. La importancia se medirá con el índice MeanDecreaseAccuracy, que es un promedio escalado de la exactitud de la predicción [BL01] [WG11].

Tabla 56

Resultados del ordenamiento de variables, con el algoritmo RANDOM FOREST

ALGORITMOS

Algoritmo Random Forest:

METODO DE SELECCIÓN DE SUBCONJUNTO:

- HILL CLIMBING

VARIABLES EN EL MODELO:

SURVIVED, PCLASS, TITULOPASAJERO, SEXNUEVO, AGEMALE, AGEFEMALE, SIBSPNUEVO, PARCHNUEVO, TIPOTICKET, FARENUEVO, CABINNUEVO, EMBARKEDNUEVO, SOBREVFAMILIA

RESUMEN DEL ALGORTIMO RANDOM FOREST CON R

```
[1] "Error de los datos de prueba (20% de los datos)"
[1] 0.1452514
[1] "Matriz de Confusión de los Datos de Prueba (20% de los datos)"
    Predicho
Real   0   1
   0 102  10
   1  16  51

[1] "Importancia de variables según orden (datos de entrenamiento 80% de
datos)"
                    0     1 MeanDecreaseAccuracy MeanDecreaseGini
farenuevo       28.01  9.79                31.58            12.70
titulopasajero  30.48 12.84                29.34            28.72
pclass          21.03 16.07                27.85            12.36
sibspnuevo      23.80 -0.70                22.53             5.79
agemale         17.41  7.78                17.31            21.09
embarkednuevo    4.65 16.57                16.14             5.18
agefemale       11.34  5.70                13.02            16.03
sexnuevo        12.56  0.33                11.90            12.66
cabinnuevo      11.49  0.62                11.73             8.64
sobrevfamilia    7.04  4.23                 9.25             7.08
parchnuevo       6.55 -0.65                 5.46             3.03
tipoticket       6.28 -0.51                 4.86             7.04
[1] "Area bajo la Curva ROC para los datos de entrenamiento"
[[1]]
[1] 0.9269316
```

En la Tabla 57 se analizan de forma conjunta todos los métodos para hacer la comparación respectiva.

Tabla 57

Importancia de las variables según la ordenación obtenida por diferentes métodos

N° Ord	METODOS DE SELECCIÓN DE VARIABLE POR RANKEO								Algoritmo Random Forest (para ordenar las variables)	
	Filtro Chi-Squared		Filtro Ganancia de Información		Filtro Ganancia de Ratio		Filtro Random Forest			
	Variables	Importancia de Atributo	Variables	Importancia de Atributo	Variables	Importancia de Atributo	Variables	Importancia de Atributo	Variables	MeanDecrease Accuracy
1	titulopasajero	0.551679	titulopasajero	0.228547	titulopasajero	0.235938	farenuevo	54.0087	farenuevo	31.58
2	agemale	0.543399	agemale	0.220919	agemale	0.227903	pclass	53.14927	titulopasajero	29.34
3	sexnuevo	0.532569	sexnuevo	0.210179	sexnuevo	0.224501	sibspnuevo	49.28934	pclass	27.85
4	agefemale	0.532569	agefemale	0.210179	agefemale	0.224501	titulopasajero	45.18943	sibspnuevo	22.53
5	pclass	0.324503	pclass	0.07734	sobrevfamilia	0.098856	agemale	25.91698	agemale	17.31
6	sobrevfamilia	0.323446	sobrevfamilia	0.073277	cabinnuevo	0.091852	cabinnuevo	21.62795	embarkednuevo	16.14
7	cabinnuevo	0.317274	cabinnuevo	0.071113	pclass	0.077928	embarkednuevo	19.81208	agefemale	13.02
8	farenuevo	0.296127	farenuevo	0.067239	farenuevo	0.070336	sobrevfamilia	19.32737	sexnuevo	11.9
9	sibspnuevo	0	sibspnuevo	0	sibspnuevo	NaN	agefemale	18.66454	cabinnuevo	11.73
10	parchnuevo	0	parchnuevo	0	parchnuevo	NaN	tipoticket	16.44278	sobrevfamilia	9.25
11	tipoticket	0	tipoticket	0	tipoticket	NaN	sexnuevo	15.83762	parchnuevo	5.46
12	embarkednuevo	0	embarkednuevo	0	embarkednuevo	NaN	parchnuevo	-0.14978	tipoticket	4.86

A partir de la Tabla 57 se pueden distinguir las variables más importantes en base al algoritmo RANDOM FOREST [BL01]. El resumen de resultados extraídos se muestra a continuación en la tabla 58:

Tabla 58
Evaluación de la Importancia de las variables según orden

Variables más importantes según el RANDOM FOREST	Variables menos importantes según el RANDOM FOREST
titulopasajero	tipoticket
Farenuevo	parchnuevo
pclass	sobrevfamilia
sibsnuevo	cabinnuevo
agemale	sexnuevo

La selección de la tabla 58 se hizo tomando como base el resultado del algoritmo RANDOM FOREST. Seguidamente se muestran variables más importantes:

a) La variable "titulopasajero" se considera la más importante en todos los métodos.

b) La variable "farenuevo" es más importante en el algoritmo RANDOM FOREST por lo tanto se tomará en cuenta.

c) La variable "pclass" está entre las más o menos importantes en los primeros tres métodos de rankeo (Filtro Chi-Squared, Ganancia de Información, Ganancia de Ratio), y es la tercera más importante en el algoritmo RANDOM FOREST.

d) La variable "sibsnuevo" no es tan importante entre los tres primeros métodos pero sí es importante en el caso del algoritmo RANDOM FOREST.

e) La variable agemale es una variable que se ubica entre las cinco más importantes en todos los métodos.

En el caso de las variables menos importantes se observa lo siguiente:

a) La variable "tipoticket", no es muy importante en la mayoría de los métodos.

b) La variable "parchnuevo", es también una variable no muy importante en todos los métodos pero menos que la anterior.

c) La variable "sobrevfamilia" se ubica aproximadamente en la posición media entre los métodos de rankeo, y en el caso del algoritmo RANDOM FOREST se coloca como la antepenúltima.

d) La variable cabinnuevo tiene una importancia relativa en casi todos los métodos.

e) La variable "sexnuevo" está entre las primeras para los tres primeros métodos de rankeo, y en el algoritmo RANDOM FOREST está en el octavo puesto.

Las variables que se situarían en la posición media en todos los métodos, serían las variables "embarkednuevo" y "agefemale".

5.4.5. Evaluación de las curvas ROC

En esta sección se evaluarán las curvas ROC, que se obtuvieron con el software R, a partir de los datos de entrenamiento (80% del total de 891 observaciones). Se utilizan los algoritmos donde se observa que hay una mejor predicción:

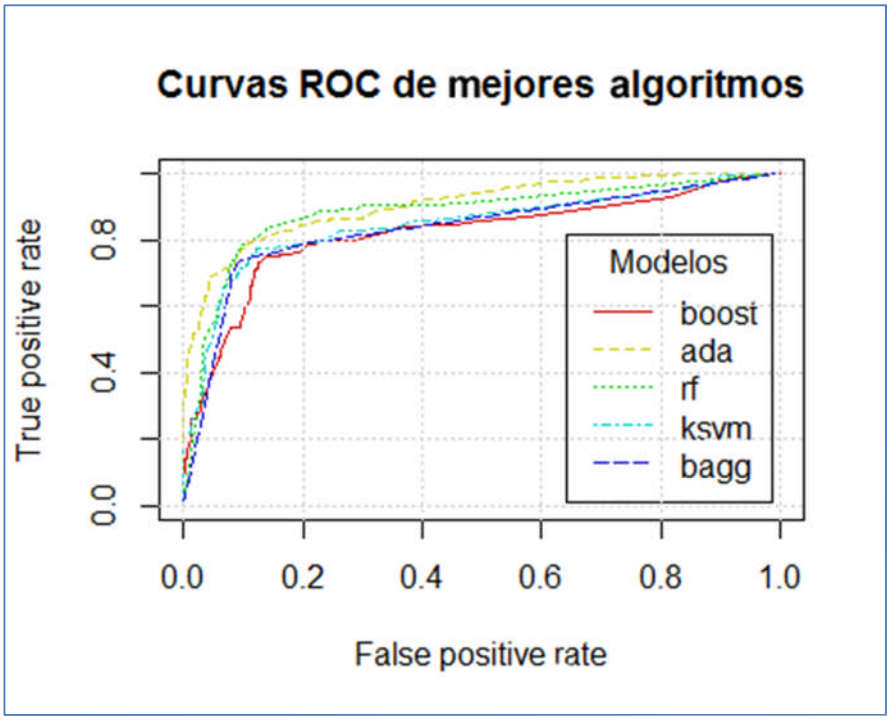
1) Gradient Boosting
2) Bagging
3) Ada Boost
4) Random Forest
5) Support Vector Machine SVM

La figura 32 muestra las curvas ROC, para estos algoritmos obtenidos utilizando R.

Tabla 59
Valores ROC de los algoritmos analizados

Algoritmo	Valor de ROC o AUC (Area bajo la curva)	Error en TEST (KAGGLE)
GRADIENT BOOSTING	0.82	0.21531
ADA BOOST	0.90	0.21053
BAGGING	0.83	0.20096
RANDOM FOREST	0.88	0.21531
SUPPORT VECTOR MACHINE	0.84	0.2201

Figura 32
Curvas ROC de los mejores modelos en R



A partir del análisis de la Figura 32 y de la Tabla 59, se observa que el algoritmo con mayor valor AUC es ADA BOOST (0.90), luego se encuentra el algoritmo RANDOM FOREST. Estos algoritmos obtuvieron menor error en los datos de prueba TEST (Kaggle). De lo anterior se concluye que estos algoritmos ajustan bien los datos de entrenamiento.

Se observa que el algoritmo BAGGING obtuvo un valor AUC bajo, pero obtuvo el menor error en los datos de prueba TEST (Kaggle). De lo anterior se concluye que en los datos de entrenamiento el algoritmo BAGGING no se comporta en forma eficiente, pero sí lo hace en los datos de prueba.

Capítulo 6

Conclusiones y trabajos futuros

6.1. Conclusiones

A lo largo de este trabajo el objetivo ha sido utilizar las técnicas de la Minería de Datos y las herramientas del aprendizaje automático para predecir en forma eficiente la probabilidad de supervivencia de los pasajeros del Titanic. Para ello se ha utilizado la metodología del descubrimiento del conocimiento (KDD), siguiendo todos sus pasos metodológicos, desde el paso 1 hasta el paso 8, los cuales fueron explicados en el capítulo 2.

Los pasos realizados en el proceso de descubrimiento del conocimiento KDD, han sido registrados en una bitácora electrónica o software creado con este propósito.

Con esta metodología se ha podido determinar como resultado los algoritmos que mejor se comportan para resolver el problema de predicción del Titanic.

Se ha logrado obtener una ubicación importante en el ranking de Kaggle (página web de competencias de minería de datos), en el tema del "Desastre del Titanic", donde la ubicación actual es la 239 con un 20.096% de error, entre un total de 6,634 equipos participantes.

Para obtener estos resultados se ha tenido que profundizar en cada uno de los pasos del descubrimiento del conocimiento, definiendo, modelando o implementando las técnicas adecuadas para la consecución de los objetivos planteados aquí. De forma detallada, la labor realizada en este trabajo de investigación es la siguiente:

- Se ha determinado la validez de la información:
  - Haciendo uso de la estadística descriptiva, y de test estadísticos, se ha podido comprobar que las poblaciones que son muestras que forman el conjunto de datos de entrenamiento y el conjunto de datos de prueba, provienen de una misma población que sería en este caso la población de los pasajeros del Titanic.
- Se ha transformado los datos en forma óptima
  - Se han obtenido a partir de los datos originales, mediante la extracción de características, nuevas variables que han demostrado un mejor comportamiento para predecir la variable respuesta (SURVIVED) en el conjunto de datos recopilado.
  - En el caso de la selección de características, se han hallado mejores subconjuntos de variables para los algoritmos de minería de datos. Donde la mejor metodología para la selección de variables fue la metodología de selección de subconjuntos, y dentro de la cual se eligió el método de filtro WRAPPER.
- Estimación de los modelos
  - Se han determinado los algoritmos de minería de datos que mejor predicen la probabilidad de supervivencia de los pasajeros del Titanic, tanto con el software R y con el software Python, y que son:

    Para el software R:

N°	ALGORITMO	ERROR EN LOS DATOS TEST
1	GRADIENT BOOSTING	0.21531
2	ADA BOOST (ADA)	0.21053
3	BAGGING	0.20096
4	RANDOM FOREST	0.21531
5	SUPPORT VECTOR MACHINE (SVM)	0.2201
6	NAÏVE BAYES	0.21053

En donde el mejor algoritmo es el BAGGING.

Para el software Python:

N°	ALGORITMO	ERROR EN LOS DATOS TEST
1	GRADIENT BOOSTING	0.21531
2	RANDOM FOREST	0.21531
3	SUPPORT VECTOR MACHINE (SVM)	0.21531
4	BAYES NAIVES	0.21531
5	LOGISTIC REGRESION	0.21531

Donde todos los algoritmos obtienen igual resultado.

- Importancia de la variables

  - Las más importantes en orden de mayor a menor importancia son:

N° Orden	Variables
1	titulopasajero
2	Farenuevo
3	pclass
4	sibsnuevo
5	agemale

  - Las menos importantes en orden de menor a mayor importancia son:

N° Orden	Variables
1	tipoticket
2	parchnuevo
3	sobrevfamilia
4	cabinnuevo
5	sexnuevo

6.2. Trabajos futuros

Finalizamos este trabajo señalando algunas líneas de investigación que han surgido del estudio realizado y que no han sido abordadas, debido a que sobrepasan los objetivos de este trabajo, o que han aparecido como consecuencia de los resultados. Se ha recogido una línea de investigación que es la siguiente:

- La obtención de una herramienta automatizada que genere de los mejores modelos de minería de datos aplicando la metodología del descubrimiento del conocimiento (KDD). Esta herramienta sería aplicable a cualquier tipo de investigación de minería de datos utilizando R, u otro software en particular.

Apéndice A

Códigos para la selección de variables con el enfoque de subconjuntos

Las siguientes tablas contienen el código en R, para estimar los modelos de la sección 4.4.2.1.2 (Enfoque de selección de subconjunto de variables por el método Wrapper), para seleccionara el mejor subconjunto de variables, donde estos códigos utilizan el paquete creado que se llama “paqdatamining”, que es mostrado en el Apendice E.

Se hacen las selecciones del mejor subconjunto de variables en R para los siguientes algoritmos:

a) ÁRBOL DE DECISIÓN
b) GRADIENT BOOSTING
c) BAGGING
d) ADA BOOST (ADA)
e) RANDOMFOREST (RF)
f) SUPPORT VECTOR MACHINE (SVM)
g) REGRESIÓN LOGÍSTICA
h) REDES NEURONALES
i) NAÏVE BAYES
j) K.VECINDAD MAS CERCANA

Tabla A1

Código en R para la selección de caracteristicas con el enfoque de selección de subconjunto por el método wrapper para el algoritmo RPART

Selección de variables del algoritmo RPART

```
library(rpart)
library(FSelector)
library(paqdatamining)

setwd("K:\\Datos\\datos tesis\\projecto_tesis")
# Cargando los datos de entrenamiento
basetrain <- read.csv("traintransf.csv",strip.white=TRUE,na.strings=c(""))

#creando data frame vacio de resultados
resultadosrpart <- data.frame(metodo= character(), formul=character(),error=numeric())

#METODO RPART  CON TECNICA DE BUSQUEDA BESTFIRST
print("procesando RPART bestfirst")
er=0.03
set.seed(46)
#numero minimo de variables
numvar=3
subset <- best.first.search(names(basetrain)[-13],evaluatorrpart,max.backtracks=10)
formula <- as.simple.formula(subset, 'as.factor(survived)')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
rpart=algorrpart(out$traindf,out$testdf,formula)
print(rpart$error)

# #METODO RPART  CON TECNICA DE BUSQUEDA FORWARD
print("procesando RPART forward")
er=0.03
set.seed(46)
#numero minimo de variables
numvar=3
subset <- forward.search(names(basetrain)[-13],evaluatorrpart)
formula <- as.simple.formula(subset, 'as.factor(survived)')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
rpart=algorrpart(out$traindf,out$testdf,formula)
print(rpart$error)

#METODO RPART  CON TECNICA DE BUSQUEDA BACKWARD
print("procesando RPART backward")
er=0.03
set.seed(46)
#numero minimo de variables
numvar=3
subset <- backward.search(names(basetrain)[-13],evaluatorrpart)
formula <- as.simple.formula(subset, 'as.factor(survived)')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
rpart=algorrpart(out$traindf,out$testdf,formula)
print(rpart$error)
```

```
# #METODO RPART  CON TECNICA DE BUSQUEDA HILL CLIMBING
# print("procesando RPART hill climbing")
er=0.03
set.seed(46)
#numero minimo de variables
numvar=3
subset <- hill.climbing.search(names(basetrain)[-13],evaluatorrpart)
formula <- as.simple.formula(subset, 'as.factor(survived)')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
rpart=algorrpart(out$traindf,out$testdf,formula)
print(rpart$error)
```

Tabla A2

Código en R para la selección de caracteristicas con el enfoque de selección de subconjunto por el método wrapper para el algoritmo GRADIENT BOOSTING

Selección de variables del algoritmo ADA BOOST

```
library(mboost)
library(FSelector)
library(paqdatamining)

setwd("K:\\Datos\\datos tesis\\projecto_tesis")
# Cargando los datos de entrenamiento
basetrain <- read.csv("traintransf.csv",strip.white=TRUE,na.strings=c(""))

#creando data frame vacio de resultados
resultadosboost <- data.frame(metodo= character(), formul=character(),error=numeric())

#METODO boost  CON TECNICA DE BUSQUEDA BESTFIRST
print("procesando boost bestfirst")
er=0.03
set.seed(46)
#numero minimo de variables
numvar=3
subset <- best.first.search(names(basetrain)[-13],evaluatorboost,max.backtracks=10)
formula <- as.simple.formula(subset, 'survived')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
boost=algorboosting(out$traindf,out$testdf,formula)
print(boost$error)

# #METODO boost  CON TECNICA DE BUSQUEDA FORWARD
print("procesando boost forward")
er=0.03
set.seed(46)
#numero minimo de variables
numvar=3
subset <- forward.search(names(basetrain)[-13],evaluatorboost)
formula <- as.simple.formula(subset, 'survived')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
```

```
boost=algorboosting(out$traindf,out$testdf,formula)
print(boost$error)

#METODO boost  CON TECNICA DE BUSQUEDA BACKWARD
print("procesando boost backward")
er=0.03
set.seed(46)
#numero minimo de variables
numvar=3
subset <- backward.search(names(basetrain)[-13],evaluatorboost)
formula <- as.simple.formula(subset, 'survived')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
boost=algorboosting(out$traindf,out$testdf,formula)
print(boost$error)

# #METODO boost  CON TECNICA DE BUSQUEDA HILL CLIMBING
print("procesando boost hill climbing")
er=0.03
set.seed(46)
#numero minimo de variables
numvar=3
subset <- hill.climbing.search(names(basetrain)[-13],evaluatorboost)
formula <- as.simple.formula(subset, 'survived')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
boost=algorboosting(out$traindf,out$testdf,formula)
print(boost$error)
```

Tabla A3

Código en R para la selección de caracteristicas con el enfoque de selección de subconjunto por el método wrapper para el algoritmo ADA BOOST

Selección de variables del algoritmo BAGGING

```
library(ada)
library(FSelector)
library(paqdatamining)

setwd("K:\\Datos\\datos tesis\\projecto_tesis")
# Cargando los datos de entrenamiento
basetrain <- read.csv("traintransf.csv",strip.white=TRUE,na.strings=c(""))

#creando data frame vacio de resultados
resultadosada <- data.frame(metodo= character(), formul=character(),error=numeric())

#METODO ada  CON TECNICA DE BUSQUEDA BESTFIRST
print("procesando ada bestfirst")
er=0.03
#numero minimo de variables
numvar=3
subset <- best.first.search(names(basetrain)[-13],evaluatorada,max.backtracks=10)
formula <- as.simple.formula(subset, 'as.factor(survived)')
```

```
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
ada=algorada(out$traindf,out$testdf,formula)
print(ada$error)

# #METODO ada  CON TECNICA DE BUSQUEDA FORWARD
print("procesando ada forward")
er=0.03
#numero minimo de variables
numvar=3
subset <- forward.search(names(basetrain)[-13],evaluatorada)
formula <- as.simple.formula(subset, 'as.factor(survived)')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
ada=algorada(out$traindf,out$testdf,formula)
print(ada$error)

#METODO ada  CON TECNICA DE BUSQUEDA BACKWARD
print("procesando ada backward")
er=0.03
#numero minimo de variables
numvar=3
subset <- backward.search(names(basetrain)[-13],evaluatorada)
formula <- as.simple.formula(subset, 'as.factor(survived)')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
ada=algorada(out$traindf,out$testdf,formula)
print(ada$error)

# #METODO ada  CON TECNICA DE BUSQUEDA HILL CLIMBING
print("procesando ada hill climbing")
er=0.03
#numero minimo de variables
numvar=3
subset <- hill.climbing.search(names(basetrain)[-13],evaluatorada)
formula <- as.simple.formula(subset, 'as.factor(survived)')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
ada=algorada(out$traindf,out$testdf,formula)
print(ada$error)
```

Tabla A4

Código en R para la selección de caracteristicas con el enfoque de selección de subconjunto por el método wrapper para el algoritmo BAGGING

Selección de variables del algoritmo BAGGING

```
library(ipred)
library(FSelector)
library(paqdatamining)

setwd("K:\\Datos\\datos tesis\\projecto_tesis")
# Cargando los datos de entrenamiento
basetrain <- read.csv("traintransf.csv",strip.white=TRUE,na.strings=c(""))
```

```
#creando data frame vacio de resultados
resultadosbagg <- data.frame(metodo= character(), formul=character(),error=numeric())

#METODO bagg  CON TECNICA DE BUSQUEDA BESTFIRST
print("procesando bagg bestfirst")
er=0.03
set.seed(46)
#numero minimo de variables
numvar=3
subset <- best.first.search(names(basetrain)[-13],evaluatorbagg,max.backtracks=10)
formula <- as.simple.formula(subset, 'as.factor(survived)')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
bagg=algorbagging(out$traindf,out$testdf,formula)
print(bagg$error)

# #METODO bagg  CON TECNICA DE BUSQUEDA FORWARD
print("procesando bagg forward")
er=0.03
set.seed(46)
#numero minimo de variables
numvar=3
subset <- forward.search(names(basetrain)[-13],evaluatorbagg)
formula <- as.simple.formula(subset, 'as.factor(survived)')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
bagg=algorbagging(out$traindf,out$testdf,formula)
print(bagg$error)

#METODO bagg  CON TECNICA DE BUSQUEDA BACKWARD
print("procesando bagg backward")
er=0.03
set.seed(46)
#numero minimo de variables
numvar=3
subset <- backward.search(names(basetrain)[-13],evaluatorbagg)
formula <- as.simple.formula(subset, 'as.factor(survived)')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
bagg=algorbagging(out$traindf,out$testdf,formula)
print(bagg$error)
# #METODO bagg  CON TECNICA DE BUSQUEDA HILL CLIMBING
print("procesando bagg hill climbing")
er=0.03
set.seed(46)
#numero minimo de variables
numvar=3
subset <- hill.climbing.search(names(basetrain)[-13],evaluatorbagg)
formula <- as.simple.formula(subset, 'as.factor(survived)')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
bagg=algorbagging(out$traindf,out$testdf,formula)
print(bagg$error)
```

Tabla A5

Código en R para la selección de caracteristicas con el enfoque de selección de subconjunto por el método wrapper para el algoritmo RANDOM FOREST

Selección de variables del algoritmo RANDOMFOREST

```
library(randomForest)
library(FSelector)
library(paqdatamining)

setwd("K:\\Datos\\datos tesis\\projecto_tesis")
# Cargando los datos de entrenamiento
basetrain <- read.csv("traintransf.csv",strip.white=TRUE,na.strings=c(""))

#creando data frame vacio de resultados
resultadosrf <- data.frame(metodo= character(), formul=character(),error=numeric())

#METODO rf  CON TECNICA DE BUSQUEDA BESTFIRST
print("procesando rf bestfirst")
er=0.03
set.seed(46)
#numero minimo de variables
numvar=3
subset <- best.first.search(names(basetrain)[-13],evaluatorrf,max.backtracks=10)
formula <- as.simple.formula(subset, 'as.factor(survived)')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
rf=algorrf(out$traindf,out$testdf,formula)
print(rf$error)

# #METODO rf  CON TECNICA DE BUSQUEDA FORWARD
print("procesando rf forward")
er=0.03
set.seed(46)
#numero minimo de variables
numvar=3
subset <- forward.search(names(basetrain)[-13],evaluatorrf)
formula <- as.simple.formula(subset, 'as.factor(survived)')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
rf=algorrf(out$traindf,out$testdf,formula)
print(rf$error)

#METODO rf  CON TECNICA DE BUSQUEDA BACKWARD
print("procesando rf backward")
er=0.03
set.seed(46)
#numero minimo de variables
numvar=3
subset <- backward.search(names(basetrain)[-13],evaluatorrf)
formula <- as.simple.formula(subset, 'as.factor(survived)')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
rf=algorrf(out$traindf,out$testdf,formula)
print(rf$error)
```

```
# #METODO rf  CON TECNICA DE BUSQUEDA HILL CLIMBING
print("procesando rf hill climbing")
er=0.03
set.seed(46)
#numero minimo de variables
numvar=3
subset <- hill.climbing.search(names(basetrain)[-13],evaluatorrf)
formula <- as.simple.formula(subset, 'as.factor(survived)')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
rf=algorrf(out$traindf,out$testdf,formula)
print(rf$error)
```

Tabla A6

Código en R para la selección de caracteristicas con el enfoque de selección de subconjunto por el método wrapper para el algoritmo SUPPORT VECTOR MACHINES SVM

Selección de variables del algoritmo SUPPORT VECTOR MACHINES SVM

```
library(kernlab)
library(FSelector)
library(paqdatamining)

setwd("K:\\Datos\\datos tesis\\projecto_tesis")
# Cargando los datos de entrenamiento
basetrain <- read.csv("traintransf.csv",strip.white=TRUE,na.strings=c(""))

#creando data frame vacio de resultados
resultadossvm <- data.frame(metodo= character(), formul=character(),error=numeric())

#METODO svm  CON TECNICA DE BUSQUEDA BESTFIRST
print("procesando svm bestfirst")
er=0.03
set.seed(46)
#numero minimo de variables
numvar=3
subset <- best.first.search(names(basetrain)[-13],evaluatorsvm,max.backtracks=10)
formula <- as.simple.formula(subset, 'as.factor(survived)')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
svm=algorsvm(out$traindf,out$testdf,formula)
print(svm$error)

# #METODO svm  CON TECNICA DE BUSQUEDA FORWARD
print("procesando svm forward")
er=0.03
set.seed(46)
#numero minimo de variables
numvar=3
subset <- forward.search(names(basetrain)[-13],evaluatorsvm)
formula <- as.simple.formula(subset, 'as.factor(survived)')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
```

```
svm=algorsvm(out$traindf,out$testdf,formula)
print(svm$error)

#METODO svm  CON TECNICA DE BUSQUEDA BACKWARD
print("procesando svm backward")
er=0.03
set.seed(46)
#numero minimo de variables
numvar=3
subset <- backward.search(names(basetrain)[-13],evaluatorsvm)
formula <- as.simple.formula(subset, 'as.factor(survived)')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
svm=algorsvm(out$traindf,out$testdf,formula)
print(svm$error)

# #METODO svm  CON TECNICA DE BUSQUEDA HILL CLIMBING
print("procesando svm hill climbing")
er=0.03
set.seed(46)
#numero minimo de variables
numvar=3
subset <- hill.climbing.search(names(basetrain)[-13],evaluatorsvm)
formula <- as.simple.formula(subset, 'as.factor(survived)')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
svm=algorsvm(out$traindf,out$testdf,formula)
print(svm$error)
```

Tabla A7

Código en R para la selección de caracteristicas con el enfoque de selección de subconjunto por el método wrapper para el algoritmo REGRESION LOGISTICA

Selección de variables del algoritmo REGRESION LOGISTICA

```
library(FSelector)
library(paqdatamining)

setwd("K:\\Datos\\datos tesis\\projecto_tesis")
# Cargando los datos de entrenamiento
basetrain <- read.csv("traintransf.csv",strip.white=TRUE,na.strings=c(""))

#creando data frame vacio de resultados
resultadosreg <- data.frame(metodo= character(), formul=character(),error=numeric())

#METODO reg  CON TECNICA DE BUSQUEDA BESTFIRST
print("procesando reg bestfirst")
er=0.03
set.seed(46)
#numero minimo de variables
numvar=3
subset <- best.first.search(names(basetrain)[-13],evaluatorreg,max.backtracks=10)
formula <- as.simple.formula(subset, 'as.factor(survived)')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
```

```
reg=algorreg(out$traindf,out$testdf,formula)
print(reg$error)

# #METODO reg  CON TECNICA DE BUSQUEDA FORWARD
print("procesando reg forward")
er=0.03
set.seed(46)
#numero minimo de variables
numvar=3
subset <- forward.search(names(basetrain)[-13],evaluatorreg)
formula <- as.simple.formula(subset, 'as.factor(survived)')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
reg=algorreg(out$traindf,out$testdf,formula)
print(reg$error)

#METODO reg  CON TECNICA DE BUSQUEDA BACKWARD
print("procesando reg backward")
er=0.03
set.seed(46)
#numero minimo de variables
numvar=3
subset <- backward.search(names(basetrain)[-13],evaluatorreg)
formula <- as.simple.formula(subset, 'as.factor(survived)')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
reg=algorreg(out$traindf,out$testdf,formula)
print(reg$error)

# #METODO reg  CON TECNICA DE BUSQUEDA HILL CLIMBING
print("procesando reg hill climbing")
er=0.03
set.seed(46)
#numero minimo de variables
numvar=3
subset <- hill.climbing.search(names(basetrain)[-13],evaluatorreg)
formula <- as.simple.formula(subset, 'as.factor(survived)')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
reg=algorreg(out$traindf,out$testdf,formula)
print(reg$error)
```

Tabla A8

Código en R para la selección de caracteristicas con el enfoque de selección de subconjunto por el método wrapper para el algoritmo RED NEURONAL NNET

Selección de variables del algoritmo NNET

```
library(nnet)
library(FSelector)
library(paqdatamining)

setwd("K:\\Datos\\datos tesis\\projecto_tesis")
# Cargando los datos de entrenamiento
basetrain <- read.csv("traintransf.csv",strip.white=TRUE,na.strings=c(""))
```

```
#creando data frame vacio de resultados
resultadosnnet <- data.frame(metodo= character(), formul=character(),error=numeric())

#METODO nnet  CON TECNICA DE BUSQUEDA BESTFIRST
print("procesando nnet bestfirst")
er=0.03
set.seed(46)
#numero minimo de variables
numvar=3
subset <- best.first.search(names(basetrain)[-13],evaluatornnet,max.backtracks=10)
formula <- as.simple.formula(subset, 'as.factor(survived)')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
nnet=algornnet(out$traindf,out$testdf,formula)
print(nnet$error)

# #METODO nnet  CON TECNICA DE BUSQUEDA FORWARD
print("procesando nnet forward")
er=0.03
set.seed(46)
#numero minimo de variables
numvar=3
subset <- forward.search(names(basetrain)[-13],evaluatornnet)
formula <- as.simple.formula(subset, 'as.factor(survived)')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
nnet=algornnet(out$traindf,out$testdf,formula)
print(nnet$error)

#METODO nnet  CON TECNICA DE BUSQUEDA BACKWARD
print("procesando nnet backward")
er=0.03
set.seed(46)
#numero minimo de variables
numvar=3
subset <- backward.search(names(basetrain)[-13],evaluatornnet)
formula <- as.simple.formula(subset, 'as.factor(survived)')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
nnet=algornnet(out$traindf,out$testdf,formula)
print(nnet$error)

# #METODO nnet  CON TECNICA DE BUSQUEDA HILL CLIMBING
print("procesando nnet hill climbing")
er=0.03
set.seed(46)
#numero minimo de variables
numvar=3
subset <- hill.climbing.search(names(basetrain)[-13],evaluatornnet)
formula <- as.simple.formula(subset, 'as.factor(survived)')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
nnet=algornnet(out$traindf,out$testdf,formula)
print(nnet$error)
```

Tabla A9

Código en R para la selección de caracteristicas con el enfoque de selección de subconjunto por el método wrapper para el algoritmo BAYES NAIVES

Selección de variables del algoritmo NNET

```
library(e1071)
library(FSelector)
library(paqdatamining)

setwd("K:\\Datos\\datos tesis\\projecto_tesis")
# Cargando los datos de entrenamiento
basetrain <- read.csv("traintransf.csv",strip.white=TRUE,na.strings=c(""))

#creando data frame vacio de resultados
resultadosnb <- data.frame(metodo= character(), formul=character(),error=numeric())

#METODO nb  CON TECNICA DE BUSQUEDA BESTFIRST
print("procesando nb bestfirst")
er=0.03
set.seed(46)
#numero minimo de variables
numvar=3
subset <- best.first.search(names(basetrain)[-13],evaluatornb,max.backtracks=10)
formula <- as.simple.formula(subset, 'as.factor(survived)')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
nb=algornaivebayes(out$traindf,out$testdf,formula)
print(nb$error)

# #METODO nb  CON TECNICA DE BUSQUEDA FORWARD
print("procesando nb forward")
er=0.03
set.seed(46)
#numero minimo de variables
numvar=3
subset <- forward.search(names(basetrain)[-13],evaluatornb)
formula <- as.simple.formula(subset, 'as.factor(survived)')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
nb=algornaivebayes(out$traindf,out$testdf,formula)
print(nb$error)

#METODO nb  CON TECNICA DE BUSQUEDA BACKWARD
print("procesando nb backward")
er=0.03
set.seed(46)
#numero minimo de variables
numvar=3
subset <- backward.search(names(basetrain)[-13],evaluatornb)
formula <- as.simple.formula(subset, 'as.factor(survived)')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
nb=algornaivebayes(out$traindf,out$testdf,formula)
print(nb$error)
```

```
# #METODO nb  CON TECNICA DE BUSQUEDA HILL CLIMBING
print("procesando nb hill climbing")
er=0.03
set.seed(46)
#numero minimo de variables
numvar=3
subset <- hill.climbing.search(names(basetrain)[-13],evaluatornb)
formula <- as.simple.formula(subset, 'as.factor(survived)')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
nb=algornaivebayes(out$traindf,out$testdf,formula)
print(nb$error)
```

Tabla A10

Código en R para la selección de caracteristicas con el enfoque de selección de subconjunto por el método wrapper para el algoritmo K-NEAREST NEIGHTBOUR

Selección de variables del algoritmo KNN

```
library(DMwR)
library(FSelector)
library(paqdatamining)

setwd("K:\\Datos\\datos tesis\\projecto_tesis")
# Cargando los datos de entrenamiento
basetrain <- read.csv("traintransf.csv",strip.white=TRUE,na.strings=c(""))

#creando data frame vacio de resultados
resultadosknn <- data.frame(metodo= character(), formul=character(),error=numeric())

#METODO knn  CON TECNICA DE BUSQUEDA BESTFIRST
print("procesando knn bestfirst")
er=0.03
set.seed(46)
#numero minimo de variables
numvar=3
subset <- best.first.search(names(basetrain)[-13],evaluatorknn,max.backtracks=10)
formula <- as.simple.formula(subset, 'survived')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
knn=algorknn(out$traindf,out$testdf,formula)
print(knn$error)

# #METODO knn  CON TECNICA DE BUSQUEDA FORWARD
print("procesando knn forward")
er=0.03
set.seed(46)
#numero minimo de variables
numvar=3
subset <- forward.search(names(basetrain)[-13],evaluatorknn)
formula <- as.simple.formula(subset, 'survived')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
knn=algorknn(out$traindf,out$testdf,formula)
print(knn$error)
```

```
#METODO knn  CON TECNICA DE BUSQUEDA BACKWARD
print("procesando knn backward")
er=0.03
set.seed(46)
#numero minimo de variables
numvar=3
subset <- backward.search(names(basetrain)[-13],evaluatorknn)
formula <- as.simple.formula(subset, 'survived')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
knn=algorknn(out$traindf,out$testdf,formula)
print(knn$error)

# #METODO knn  CON TECNICA DE BUSQUEDA HILL CLIMBING
print("procesando knn hill climbing")
er=0.03
set.seed(46)
#numero minimo de variables
numvar=3
subset <- hill.climbing.search(names(basetrain)[-13],evaluatorknn)
formula <- as.simple.formula(subset, 'survived')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
knn=algorknn(out$traindf,out$testdf,formula)
print(knn$error)
```

Apéndice B

Códigos para procesar algoritmos para el enfoque de rankeo de características

Las siguientes tablas contienen el código en R y Python, para estimar el modelo Random Forest (pues fue el único mejor algortimo seleccionado para este enfoque), de la seccion 5.3.1.1 (Estimación con R, para las variables seleccionadas con el enfoque de rankeo de atributos), con las variables seleccionadas por el enfoque se rankeo de caracterisitcas (sección 4.4.2.1.1 – Enfoque de rankeo de caracterisiticas), donde los códigos en R utilizan el paquete creado “paqdatamining”, del Apendice E.

Tabla B1

Código en R para modelo RANDOMFOREST con las técnicas Chi-squared, Filtro Basado en Entropía - Ganancia de información y Filtro Basado en Entropía – Ganancia de Ratio del enfoque de rankeo de variables

Variables en el Modelo (Formula)
SURVIVED, TITULOPASAJERO, AGEMALE, SEXNUEVO AGEFEMALE, PCLASS, SOBREVFAMILIA, CABINNUEVO, FARENUEVO, SIBSPNUEVO, PARCHNUEVO
Variables excluidas
TIPOTICKET Y EMBARKEDNUEVO
Modelo de RANDOMFOREST

```
#Modelo RANDOMFOREST
library(randomForest)
library(paqdatamining)

setwd("K:\\Datos\\datos tesis\\projecto_tesis")
# Cargando los datos de entrenamiento
basetrain <- read.csv("traintransf.csv",strip.white=TRUE,na.strings=c(""))
testData <- read.csv("testtransf.csv",strip.white=TRUE)

#Modelo RANDOMFOREST METODO DE SELECCION DE FILTRO (TODOS LOS METODOS MENOS RANDOM FOREST)

#CALCULANDO EL MODELO Y EL ERROR
subset=c('titulopasajero','agemale','sexnuevo','agefemale',
        'pclass','sobrevfamilia','cabinnuevo','farenuevo','sibspnuevo','parchnuevo')
formula <- as.simple.formula(subset, 'as.factor(survived)')
out=inicio(basetrain,subset)
rf=algorrf(out$traindf,out$testdf,formula)
print(rf$error)
rf1=rf$alg
# Generar la matriz de confusion.
matconf=table(out$testdf$survived, pr,dnn=c("Real", "Predicho"))
print("Matriz de Confusion de los Datos de Prueba")
print(matconf)

# Ejecutar el modelo constra los datos reales test y escribir los resultados
fileName <- "rf_solucion_r_filtro_srf.csv"
prediction <- predict(rf1, newdata=testData)
write.table(prediction,fileName,row.names = FALSE, col.names=FALSE,quote = FALSE)
```

Las entradas a este codigo son el archivo de datos de entrenamiento TRAINTRANSF.CSV, y el archivo de datos de prueba TESTTRANSF.CSV y la salida es el archivo de predicciones “rf_solucion_r_filtro_srf.csv”.

Tabla B2

Código en R para modelo RANDOMFOREST con variables seleccionadas con la técnica Random Forest del enfoque de rankeo de variables

Variables en el Modelo (Formula)
SURVIVED, FARENUEVO, PCLASS, SIBSPNUEVO, TITULOPASAJERO, AGEMALE, CABINNUEVO, EMBARKEDNUEVO, SOBREVFAMILIA, AGEFEMALE, TIPOTICKET
Variables excluidas
PARCHNUEVO Y SEXNUEVO
Modelo de RANDOMFOREST

```
#Modelo RANDOMFOREST
library(randomForest)
library(paqdatamining)

setwd("K:\\Datos\\datos tesis\\projecto_tesis")
# Cargando los datos de entrenamiento
basetrain <- read.csv("traintransf.csv",strip.white=TRUE,na.strings=c(""))
testData <- read.csv("testtransf.csv",strip.white=TRUE)

#Modelo RANDOMFOREST METODO DE SELECCION DE FILTRO DE RANKEO (METODO RANDOM FOREST)
#CALCULANDO EL MODELO Y EL ERROR
subset=c('farenuevo','pclass','sibspnuevo','titulopasajero','agemale',
      'cabinnuevo','embarkednuevo','sobrevfamilia','agefemale','tipoticket')

formula <- as.simple.formula(subset, 'as.factor(survived)')
out=inicio(basetrain,subset)
rf=algorrf(out$traindf,out$testdf,formula)
print(rf$error)
rf2=rf$alg
# Generar la matriz de confusion.
matconf=table(out$testdf$survived, pr,dnn=c("Real", "Predicho"))
print("Matriz de Confusion de los Datos de Prueba")
print(matconf)

# Ejecutar el modelo constra los datos reales test y escribir los resultados
fileName <- "rf_solucion_r_filtro_rf.csv"
prediction <- predict(rf2, newdata=testData)
write.table(prediction,fileName,row.names = FALSE, col.names=FALSE,quote = FALSE)
```

Las entradas a este código son el archivo de entrenamiento TRAINTRANSF.CSV, y el archivo de prueba TESTTRANSF.CSV el cual produce la salida que es el archivo de predicciones “rf_solucion_r_filtro_rf.csv”, se utiliza el paquete creado para este trabajo “paqdatamining”

Tabla B3

Código en Python para el modelo RANDOMFOREST con las tecnicas Chi-squared, Filtro Basado en Entropía - Ganancia de información, Filtro Basado en Entropía – Ganancia de Ratio del enfoque de rankeo de variables

Variables en el Modelo (Formula)
SURVIVED, TITULOPASAJERO, AGEMALE, SEXNUEVO AGEFEMALE, PCLASS, SOBREVFAMILIA, CABINNUEVO, FARENUEVO, SIBSPNUEVO, PARCHNUEVO
Variables excluidas
TIPOTICKET Y EMBARKEDNUEVO
Modelo de RANDOMFOREST

```
import pandas as pd
import numpy as np
import statsmodels.api as sm

from statsmodels.nonparametric import KDE, smoothers_lowess
from pandas import Series,DataFrame
from patsy import dmatrices
from sklearn import datasets, svm
from __future__ import division

import sys
sys.path.append('K:\Datos\datos tesis\projecto_tesis\python')
import csv_io as csv_io
import os
os.chdir('K:\Datos\datos tesis\projecto_tesis\python')

from sklearn.ensemble import RandomForestClassifier
from sklearn.utils import shuffle
from sklearn.metrics import roc_curve, auc
dftr = pd.read_csv("traintransf.csv")
dfte = pd.read_csv("testtransf.csv")
random_state = np.random.RandomState(1)
#survived ~  titulopasajero + sexnuevo + agemale + sibspnuevo + parchnuevo + tipoticket + cabinnuevo + sobrevfamilia
values=['titulopasajero','agemale','sexnuevo','agefemale',
        'pclass','sobrevfamilia','cabinnuevo','farenuevo','sibspnuevo','parchnuevo','survived']
dftrain=dftr[values]
n_samples, n_features = dftrain.shape
dftrain.head()
#df=dftit.loc[:,'pclass':'survived']
#dftrain['is_train'] = np.random.uniform(0, 1, len(dftr)) <= .80
lenf=len(values)-1
features = dftrain.columns[:lenf]
X=dftrain[features]
y, _ = pd.factorize(dftrain['survived'])
#test=test[values]
# shuffle and split training and test sets
X, y = shuffle(X, y, random_state=random_state)
ochoporc = int(n_samples * .8)
X_train, X_test = X[:ochoporc], X[ochoporc:]
y_train, y_test = y[:ochoporc], y[ochoporc:]

clf = RandomForestClassifier(n_estimators=18, max_depth=None,min_samples_split=1, random_state=0,n_jobs=2)
#clf.fit(train[features], y)
clf.fit(X_train, y_train)
```

```
preds=clf.predict(X_test)
cross=pd.crosstab(y_test, preds, rownames=['actual'], colnames=['preds'])
print(cross)
error=(cross[0][1]+cross[1][0])/len(preds)
print(error)

dftest=dfte[values]
predicted=clf.predict(dftest[features])
print(predicted)
dfpred=DataFrame(predicted)
dfpred.to_csv("rf_solucion_py_filtro_srf.csv ", sep=',', na_rep='',header=False,index=False)
```

Las entradas a este código son el archivo de entrenamiento TRAINTRANSF.CSV, y el archivo de prueba TESTTRANSF.CSV el cual produce la salida que es el archivo de predicciones "rf_solucion_py_filtro_srf.csv

Tabla B4

Código en Python para modelo RANDOMFOREST con variables seleccionadas con la técnica Random Forest del enfoque de rankeo de variables

Variables en el Modelo (Formula)
SURVIVED, FARENUEVO, PCLASS, SIBSPNUEVO, TITULOPASAJERO, AGEMALE, CABINNUEVO, EMBARKEDNUEVO, SOBREVFAMILIA, AGEFEMALE, TIPOTICKET
Variables excluidas
PARCHNUEVO Y SEXNUEVO
Modelo de RANDOMFOREST

```
import pandas as pd
import numpy as np
import statsmodels.api as sm

from statsmodels.nonparametric import KDE, smoothers_lowess
from pandas import Series,DataFrame
from patsy import dmatrices
from sklearn import datasets, svm
from __future__ import division

import sys
sys.path.append('K:\Datos\datos tesis\projecto_tesis\python')
import csv_io as csv_io
import os
os.chdir('K:\Datos\datos tesis\projecto_tesis\python')

from sklearn.ensemble import RandomForestClassifier
from sklearn.utils import shuffle
from sklearn.metrics import roc_curve, auc
dftr = pd.read_csv("traintransf.csv")
dfte = pd.read_csv("testtransf.csv")
random_state = np.random.RandomState(1)

values=['farenuevo','pclass','sibspnuevo','titulopasajero','agemale','cabinnuevo','embarkednuevo',
    'sobrevfamilia','agefemale','tipoticket','survived']

dftrain=dftr[values]
n_samples, n_features = dftrain.shape
```

```
dftrain.head()
#df=dftit.loc[:,'pclass':'survived']
#dftrain['is_train'] = np.random.uniform(0, 1, len(dftr)) <= .80
lenf=len(values)-1
features = dftrain.columns[:lenf]
X=dftrain[features]
y, _ = pd.factorize(dftrain['survived'])
#test=test[values]
# shuffle and split training and test sets
X, y = shuffle(X, y, random_state=random_state)
ochoporc = int(n_samples * .8)
X_train, X_test = X[:ochoporc], X[ochoporc:]
y_train, y_test = y[:ochoporc], y[ochoporc:]

clf = RandomForestClassifier(n_estimators=18, max_depth=None,min_samples_split=1, random_state=0,n_jobs=2)
#clf.fit(train[features], y)
clf.fit(X_train, y_train)

preds=clf.predict(X_test)
cross=pd.crosstab(y_test, preds, rownames=['actual'], colnames=['preds'])
print(cross)
error=(cross[0][1]+cross[1][0])/len(preds)
print(error)

dftest=dfte[values]
predicted=clf.predict(dftest[features])
print(predicted)
dfpred=DataFrame(predicted)
dfpred.to_csv("rf_solucion_py_filtro_rf.csv ", sep=',', na_rep='',header=False,index=False)
```

Las entradas a este código son el archivo de entrenamiento TRAINTRANSF.CSV, y el archivo de prueba TESTTRANSF.CSV el cual produce la salida que es el archivo de predicciones “rf_solucion_py_filtro_rf.csv

Apéndice C

Códigos para procesar algoritmos, para el enfoque de selección de subconjuntos

Las siguientes tablas contienen el código en R y Python, para estimar los algoritmos de la sección 5.2 (Selección de los algoritmos de la minería de datos), que fueron seleccionados, para el enfoque de selección de subconjuntos de caracterisitcas de la sección 4.4.2.1.2 (Enfoque de selección de subconjunto de variables por el método wrapper), donde los códigos en R utilizan el paquete creado que se llama "paqdatamining", que es mostrado en el Apendice E.

Se hacen las selecciones de mejor subconjunto de variables en R para los siguientes algoritmos:

a) ÁRBOL DE DECISIÓN
b) GRADIENT BOOSTING
c) BAGGING
d) ADA BOOST (ADA)
e) RANDOMFOREST (RF)
f) SUPPORT VECTOR MACHINE (SVM)
g) REGRESIÓN LOGÍSTICA
h) REDES NEURONALES
i) NAÏVE BAYES
j) K.VECINDAD MAS CERCANA

Tabla C1

Código en R para modelo RPART con la selección de subconjunto de variables por el método wrapper

Variables en el Modelo (Formula)
SURVIVED, PCLASS, TITULOPASAJERO, SEXNUEVO, AGEMALE, AGEFEMALE, SIBSPNUEVO, PARCHNUEVO, TIPOTICKET, CABINNUEVO, SOBREVFAMILIA
Variables excluidas
FARENUEVO Y EMBARKEDNUEVO
CODIGO

```
library(rpart)
library(rpart.plot)
library(rattle)
library(FSelector)
library(paqdatamining)

setwd("K:\\Datos\\datos tesis\\projecto_tesis")
# Cargando los datos de entrenamiento
basetrain <- read.csv("traintransf.csv",strip.white=TRUE,na.strings=c(""))
testData <- read.csv("testtransf.csv",strip.white=TRUE)

#METODO RPART  CON TECNICA DE BUSQUEDA BESTFIRST
print("procesando RPART")

subset=c('pclass','titulopasajero','sexnuevo','agemale','agefemale',
        'sibspnuevo','parchnuevo','tipoticket','cabinnuevo','sobrevfamilia')

formula <- as.simple.formula(subset, 'as.factor(survived)')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
rpart=algorrpart(out$traindf,out$testdf,formula)
print("Error del los datos de prueba (20% de los datos)")
print(rpart$error)

# Ejecutar el modelo constra los datos reales test y escribir los resultados
fileName <- "rpart_solucion_r_wrapper.csv"
prediction <- predict(rpart$alg, newdata=testData,type="class")
write.table(prediction,fileName,row.names = FALSE, col.names=FALSE,quote = FALSE)


# Generar la matriz de confusion.
matconf=table(out$testdf$survived, rpart$pr,dnn=c("Real", "Predicho"))
print("Matriz de Confusion de los Datos de Prueba (20% de los datos)")
print(matconf)

#dibujar el arbol de decision
fancyRpartPlot(rpart$alg, main="Árbol de decisión",uniform=T, branch=1, yesno=F, border.col=0,under=T)
#reglas de decision
print("Reglas de Decision para el algoritmo RPART")
asRules(rpart$alg)
model=rpart$alg

#curva ROC para RPART
require(ROCR, quietly=TRUE)

# Generar una curva ROC para el modelo model
prrocmodel <- predict(model, newdata=out$traindf, type="prob")[,2]
```

```
missmodel <- na.omit(out$traindf$survived)
listmodel <- attr(missmodel, "na.action")
prmodel <- prediction(prrocmodel, missmodel)
pemodel=performance(prmodel, "auc")
print("Area bajo la Curva ROC para los datos de entrenamiento")
print(pemodel@y.values)
```

Las entradas a este código son el archivo de datos de entrenamiento TRAINTRANSF.CSV, y el archivo de datos de prueba TESTTRANSF.CSV el cual produce la salida que es el archivo de predicciones "rpart_solucion_r_wrapper.csv".

Tabla C2

Código en R para modelo GRADIENT BOOSTING con la selección de subconjunto de variables por el método wrapper

Variables en el Modelo (Formula)
SURVIVED, TITULOPASAJERO, SEXNUEVO, AGEMALE, AGEFEMALE, SIBSPNUEVO, EMBARKEDNUEVO, SOBREVFAMILIA
Variables excluidas
PCLASS , PARCHNUEVO, TIPOTICKET, FARENUEVO Y CABINNUEVO
CODIGO

```
library(mboost)
library(FSelector)
library(paqdatamining)

setwd("K:\\Datos\\datos tesis\\projecto_tesis")
# Cargando los datos de entrenamiento
basetrain <- read.csv("traintransf.csv",strip.white=TRUE,na.strings=c(""))
testData <- read.csv("testtransf.csv",strip.white=TRUE)

#METODO boost
print("procesando boost")

subset=c( 'titulopasajero','sexnuevo','agemale','agefemale','sibspnuevo',
        'embarkednuevo','sobrevfamilia')

formula <- as.simple.formula(subset, 'survived')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
boost=algorboosting(out$traindf,out$testdf,formula)
print("Error del los datos de prueba (20% de los datos)")
print(boost$error)

# Ejecutar el modelo constra los datos reales test y escribir los resultados
fileName <- "boost_solucion_r_wrapper.csv"
prediction = as.vector(ifelse(predict(boost$alg, type = "response", newdata = out$testdf) >
                0.5, "1", "0"))
write.table(prediction,fileName,row.names = FALSE, col.names=FALSE,quote = FALSE)

# Generar la matriz de confusion.
matconf=table(out$testdf$survived, boost$pr,dnn=c("Real", "Predicho"))
print("Matriz de Confusion de los Datos de Prueba (20% de los datos)")
print(matconf)
```

```
model=boost$alg

#curva ROC para RPART
require(ROCR, quietly=TRUE)

# Generar una curva ROC para el modelo GRADIENT BOOSTING
prrocmodel <- predict(model, type="response", newdata=out$traindf)
missmodel <- na.omit(out$traindf$survived)
listmodel <- attr(missmodel, "na.action")
prmodel <- prediction(prrocmodel, missmodel)
pemodel=performance(prmodel, "auc")
print("Area bajo la Curva ROC para los datos de entrenamiento")
print(pemodel@y.values)
```

Las entradas a este código son el archivo de datos de entrenamiento TRAINTRANSF.CSV, y el archivo de datos de prueba TESTTRANSF.CSV el cual produce la salida que es el archivo de predicciones “boost_solucion_r_wrapper.csv”.

Tabla C3

Código en R para modelo ADA BOOST con la selección de subconjunto de variables por el método wrapper

Variables en el Modelo (Formula)
SURVIVED, PCLASS, TITULOPASAJERO, SEXNUEVO, AGEMALE, AGEFEMALE, SIBSPNUEVO, PARCHNUEVO, TIPOTICKET, CABINNUEVO, EMBARKEDNUEVO, SOBREVFAMILIA
Variables excluidas
FARENUEVO
CODIGO

```
library(ada)
library(FSelector)
library(paqdatamining)

setwd("K:\\Datos\\datos tesis\\projecto_tesis")
# Cargando los datos de entrenamiento
basetrain <- read.csv("traintransf.csv",strip.white=TRUE,na.strings=c(""))
testData <- read.csv("testtransf.csv",strip.white=TRUE)

#METODO ada  CON TECNICA DE BUSQUEDA BESTFIRST
print("procesando ada")

subset=c( 'pclass','titulopasajero','sexnuevo','agemale','agefemale',
        'sibspnuevo','parchnuevo','tipoticket','cabinnuevo','embarkednuevo','sobrevfamilia')

formula <- as.simple.formula(subset, 'as.factor(survived)')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
ada=algorada(out$traindf,out$testdf,formula)
print("Error del los datos de prueba (20% de los datos)")
print(ada$error)
```

```
# Ejecutar el modelo constra los datos reales test y escribir los resultados
fileName <- "ada_solucion_r_wrapper.csv"
prediction <- predict(ada$alg, newdata=testData)
write.table(prediction,fileName,row.names = FALSE, col.names=FALSE,quote = FALSE)

model=ada$alg

# Generar la matriz de confusion.
matconf=table(out$testdf$survived, ada$pr,dnn=c("Real", "Predicho"))
print("Matriz de Confusion de los Datos de Prueba (20% de los datos)")
print(matconf)

#curva ROC para RPART
require(ROCR, quietly=TRUE)

# Generar una curva ROC para el modelo GRADIENT BOOSTING
prrocmodel <- predict(model, newdata=out$traindf, type="prob")[,2]
# prrocmodel <- predict(model, type="response", newdata=out$traindf)
missmodel <- na.omit(out$traindf$survived)
listmodel <- attr(missmodel, "na.action")
prmodel <- prediction(prrocmodel, missmodel)
pemodel=performance(prmodel, "auc")
print("Area bajo la Curva ROC para los datos de entrenamiento")
print(pemodel@y.values)
```

Las entradas a este código son el archivo de datos de entrenamiento TRAINTRANSF.CSV, y el archivo de datos de prueba TESTTRANSF.CSV el cual produce la salida que es el archivo de predicciones “ada_solucion_r_wrapper.csv”.

Tabla C4

Código en R para modelo BAGGING con la selección de subconjunto de variables por el método wrapper

Variables en el Modelo (Formula)
SURVIVED, TITULOPASAJERO, SEXNUEVO, SIBSPNUEVO, PARCHNUEVO, TIPOTICKET
Variables excluidas
PCLASS, AGEMALE, AGEFEMALE, FARENUEVO, CABINNUEVO, EMBARKEDNUEVO Y SOBREVFAMILIA
CODIGO

```
library(ipred)
library(FSelector)
library(paqdatamining)

setwd("K:\\Datos\\datos tesis\\projecto_tesis")
# Cargando los datos de entrenamiento
basetrain <- read.csv("traintransf.csv",strip.white=TRUE,na.strings=c(""))
testData <- read.csv("testtransf.csv",strip.white=TRUE)


#METODO bagg
print("procesando bagg")

subset=c( 'titulopasajero','sexnuevo','sibspnuevo','parchnuevo','tipoticket')
```

```
formula <- as.simple.formula(subset, 'as.factor(survived)')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
bagg=algorbagging(out$traindf,out$testdf,formula)
print("Error del los datos de prueba (20% de los datos)")
print(bagg$error)

# Ejecutar el modelo constra los datos reales test y escribir los resultados
fileName <- "bagg_solucion_r_wrapper.csv"
prediction <- predict(bagg$alg, newdata=testData)
write.table(prediction,fileName,row.names = FALSE, col.names=FALSE,quote = FALSE)

model=bagg$alg

# Generar la matriz de confusion.
matconf=table(out$testdf$survived, bagg$pr,dnn=c("Real", "Predicho"))
print("Matriz de Confusion de los Datos de Prueba (20% de los datos)")
print(matconf)

#curva ROC para RPART
require(ROCR, quietly=TRUE)

# Generar una curva ROC para el modelo GRADIENT BOOSTING
prrocmodel <- predict(model, newdata=out$traindf, type="prob")[,2]
# prrocmodel <- predict(model, type="response", newdata=out$traindf)
missmodel <- na.omit(out$traindf$survived)
listmodel <- attr(missmodel, "na.action")
prmodel <- prediction(prrocmodel, missmodel)
pemodel=performance(prmodel, "auc")
print("Area bajo la Curva ROC para los datos de entrenamiento")
print(pemodel@y.values)
```

Las entradas a este código son el archivo de datos de entrenamiento TRAINTRANSF.CSV, y el archivo de datos de prueba TESTTRANSF.CSV el cual produce la salida que es el archivo de predicciones “bagg_solucion_r_wrapper.csv”.

Tabla C5

Código en R para modelo RANDOM FOREST con la selección de subconjunto de variables por el método wrapper

Variables en el Modelo (Formula)
SURVIVED, TITULOPASAJERO, SEXNUEVO, AGEMALE, SIBSPNUEVO, PARCHNUEVO, TIPOTICKET, CABINNUEVO, EMBARKEDNUEVO
Variables excluidas
PCLASS, AGEFEMALE, FARENUEVO Y SOBREVFAMILIA
CODIGO

```
library(randomForest)
library(FSelector)
library(paqdatamining)

setwd("K:\\Datos\\datos tesis\\projecto_tesis")
# Cargando los datos de entrenamiento
basetrain <- read.csv("traintransf.csv",strip.white=TRUE,na.strings=c(""))
testData <- read.csv("testtransf.csv",strip.white=TRUE)
```

```
#METODO rf
print("procesando rf")

subset=c( 'titulopasajero','sexnuevo','agemale','sibspnuevo','parchnuevo','tipoticket',
        'cabinnuevo','embarkednuevo')

formula <- as.simple.formula(subset, 'as.factor(survived)')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
rf=algorrf(out$traindf,out$testdf,formula)
print("Error del los datos de prueba (20% de los datos)")
print(rf$error)

# Ejecutar el modelo constra los datos reales test y escribir los resultados
fileName <- "rf_solucion_r_wrapper.csv"
prediction <- predict(rf$alg, newdata=testData)
write.table(prediction,fileName,row.names = FALSE, col.names=FALSE,quote = FALSE)

# Generar la matriz de confusion.
matconf=table(out$testdf$survived, rf$pr,dnn=c("Real", "Predicho"))
print("Matriz de Confusion de los Datos de Prueba (20% de los datos)")
print(matconf)

# El paquete `pROC' implementa varias funciones AUC.
require(pROC, quietly=TRUE)
model=rf$alg

# Calculando el area bajo la Curva (AUC).
roc(model$y,model$votes)
# Calculando el intervalo de confianza de AUC.
auc(model$y,model$votes)
# Enumerarando la importancia de las variables.
rn <- round(importance(model), 2)
ord=rn[order(rn[,3], decreasing=TRUE),]
print("Importancia de variables segun orden (datos de entrenamiento 80% de datos)")
print(ord)

#curva ROC para RPART
require(ROCR, quietly=TRUE)

# Generar una curva ROC para el modelo GRADIENT BOOSTING
prrocmodel <- predict(model, newdata=out$traindf, type="prob")[,2]
# prrocmodel <- predict(model, type="response", newdata=out$traindf)
missmodel <- na.omit(out$traindf$survived)
listmodel <- attr(missmodel, "na.action")
prmodel <- prediction(prrocmodel, missmodel)
pemodel=performance(prmodel, "auc")
print("Area bajo la Curva ROC para los datos de entrenamiento")
print(pemodel@y.values)
```

Las entradas a este código son el archivo de datos de entrenamiento TRAINTRANSF.CSV, y el archivo de datos de prueba TESTTRANSF.CSV el cual produce la salida que es el archivo de predicciones “rf_solucion_r_wrapper.csv”.

Tabla C6

Código en R para modelo SUPPORT VECTOR MACHINE SVM con la selección de subconjunto de variables por el método wrapper

Variables en el Modelo (Formula)
SURVIVED, TITULOPASAJERO, SEXNUEVO, AGEFEMALE , SIBSPNUEVO, PARCHNUEVO, TIPOTICKET , CABINNUEVO
Variables excluidas
PCLASS, AGEMALE, FARENUEVO, EMBARKEDNUEVO Y SOBREVFAMILIA
CODIGO

```
library(kernlab)
library(FSelector)
library(paqdatamining)

setwd("K:\\Datos\\datos tesis\\projecto_tesis")
# Cargando los datos de entrenamiento
basetrain <- read.csv("traintransf.csv",strip.white=TRUE,na.strings=c(""))
testData <- read.csv("testtransf.csv",strip.white=TRUE)


#METODO svm
print("procesando svm")
subset=c('titulopasajero','sexnuevo','agefemale','sibspnuevo','parchnuevo','tipoticket','cabinnuevo')

formula <- as.simple.formula(subset, 'as.factor(survived)')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
svm=algorsvm(out$traindf,out$testdf,formula)
print("Error del los datos de prueba (20% de los datos)")
print(svm$error)

# Ejecutar el modelo constra los datos reales test y escribir los resultados
fileName <- "svm_solucion_r_wrapper.csv"
prediction <- predict(svm$alg, newdata=testData)
write.table(prediction,fileName,row.names = FALSE, col.names=FALSE,quote = FALSE)

# Generar la matriz de confusion.
matconf=table(out$testdf$survived, svm$pr,dnn=c("Real", "Predicho"))
print("Matriz de Confusion de los Datos de Prueba (20% de los datos)")
print(matconf)

#curva ROC para RPART
require(ROCR, quietly=TRUE)
model=svm$alg

# Generar una curva ROC para el modelo GRADIENT BOOSTING
prrocmodel <- predict(model, newdata=out$traindf, type="prob")[,2]
# prrocmodel <- predict(model, type="response", newdata=out$traindf)
missmodel <- na.omit(out$traindf$survived)
listmodel <- attr(missmodel, "na.action")
prmodel <- prediction(prrocmodel, missmodel)
pemodel=performance(prmodel, "auc")
print("Area bajo la Curva ROC para los datos de entrenamiento")
print(pemodel@y.values)
```

Las entradas a este código son el archivo de datos de entrenamiento TRAINTRANSF.CSV, y el archivo de datos de prueba TESTTRANSF.CSV el cual produce la salida que es el archivo de predicciones “svm_solucion_r_wrapper.csv”.

Tabla C7

Código en R para modelo REGRESION LOGISTICA con la selección de subconjunto de variables por el método wrapper

Variables en el Modelo (Formula)
SURVIVED , TITULOPASAJERO, SEXNUEVO, SIBSPNUEVO, PARCHNUEVO, TIPOTICKET, EMBARKEDNUEVO, SOBREVFAMILIA
Variables excluidas
PCLASS, AGEMALE, AGEFEMALE, FARENUEVO Y CABINNUEVO
CODIGO

```
library(FSelector)
library(paqdatamining)

setwd("K:\\Datos\\datos tesis\\projecto_tesis")
# Cargando los datos de entrenamiento
basetrain <- read.csv("traintransf.csv",strip.white=TRUE,na.strings=c(""))
testData <- read.csv("testtransf.csv",strip.white=TRUE)


#METODO reg
print("procesando reg")
survived ~  titulopasajero + sexnuevo + sibspnuevo + parchnuevo + tipoticket +
  embarkednuevo + sobrevfamilia
subset=c( 'titulopasajero','sexnuevo','sibspnuevo','parchnuevo','tipoticket',
          'embarkednuevo','sobrevfamilia')

formula <- as.simple.formula(subset, 'as.factor(survived)')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
reg=algorreg(out$traindf,out$testdf,formula)
print("Error del los datos de prueba (20% de los datos)")
print(reg$error)

# Ejecutar el modelo constra los datos reales test y escribir los resultados
fileName <- "reg_solucion_r_wrapper.csv"
prediction <- as.vector(ifelse(predict(reg$alg, type = "response", newdata = out$testdf) >
                    0.5, "1", "0"))
write.table(prediction,fileName,row.names = FALSE, col.names=FALSE,quote = FALSE)

# Generar la matriz de confusion.
matconf=table(out$testdf$survived, reg$pr,dnn=c("Real", "Predicho"))
print("Matriz de Confusion de los Datos de Prueba (20% de los datos)")
print(matconf)

model=reg$alg
#curva ROC para RPART
require(ROCR, quietly=TRUE)

# Generar una curva ROC para el modelo GRADIENT BOOSTING
prrocmodel <- predict(model, type="response", newdata=out$traindf)
missmodel <- na.omit(out$traindf$survived)
```

```
listmodel <- attr(missmodel, "na.action")
prmodel <- prediction(prrocmodel, missmodel)
pemodel=performance(prmodel, "auc")
print("Area bajo la Curva ROC para los datos de entrenamiento")
print(pemodel@y.values)
```

Las entradas a este código son el archivo de datos de entrenamiento TRAINTRANSF.CSV, y el archivo de datos de prueba TESTTRANSF.CSV el cual produce la salida que es el archivo de predicciones "reg_solucion_r_wrapper.csv".

Tabla C8

Código en R para modelo REDES NEURONALES NNET con la selección de subconjunto de variables por el método wrapper

Variables en el Modelo (Formula)
SURVIVED, PCLASS , TITULOPASAJERO, SEXNUEVO, AGEMALE , AGEFEMALE, SIBSPNUEVO, PARCHNUEVO, TIPOTICKET, CABINNUEVO, EMBARKEDNUEVO, SOBREVFAMILIA
Variables excluidas
FARENUEVO
CODIGO

```
library(nnet)
library(FSelector)
library(paqdatamining)

setwd("K:\\Datos\\datos tesis\\projecto_tesis")
# Cargando los datos de entrenamiento
basetrain <- read.csv("traintransf.csv",strip.white=TRUE,na.strings=c(""))
testData <- read.csv("testtransf.csv",strip.white=TRUE)


#METODO nnet
print("procesando nnet")
subset=c('pclass','titulopasajero','sexnuevo','agemale','agefemale','sibspnuevo','parchnuevo',
        'tipoticket','cabinnuevo','embarkednuevo','sobrevfamilia')

formula <- as.simple.formula(subset, 'as.factor(survived)')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
nnet=algornnet(out$traindf,out$testdf,formula)
print("Error del los datos de prueba (20% de los datos)")
print(nnet$error)

# Ejecutar el modelo constra los datos reales test y escribir los resultados
fileName <- "nnet_solucion_r_wrapper.csv"
prediction <- predict(nnet$alg, newdata=testData,type="class")
write.table(prediction,fileName,row.names = FALSE, col.names=FALSE,quote = FALSE)

model=nnet$alg

# Generar la matriz de confusion.
matconf=table(out$testdf$survived, nnet$pr,dnn=c("Real", "Predicho"))
print("Matriz de Confusion de los Datos de Prueba (20% de los datos)")
print(matconf)

#curva ROC para RPART
```

```
require(ROCR, quietly=TRUE)

# Generar una curva ROC para el modelo GRADIENT BOOSTING
prrocmodel <- predict(model, newdata=out$traindf)
missmodel <- na.omit(out$traindf$survived)
listmodel <- attr(missmodel, "na.action")
prmodel <- prediction(prrocmodel, missmodel)
pemodel=performance(prmodel, "auc")
print("Area bajo la Curva ROC para los datos de entrenamiento")
print(pemodel@y.values)
```

Las entradas a este código son el archivo de datos de entrenamiento TRAINTRANSF.CSV, y el archivo de datos de prueba TESTTRANSF.CSV el cual produce la salida que es el archivo de predicciones "nnet_solucion_r_wrapper.csv".

Tabla C9

Código en R para modelo NAIVE BAYES con la selección de subconjunto de variables por el método wrapper

Variables en el Modelo (Formula)
SURVIVED, TITULOPASAJERO, SEXNUEVO, AGEMALE, SIBSPNUEVO, PARCHNUEVO, EMBARKEDNUEVO, SOBREVFAMILIA
Variables excluidas
PCLASS, AGEFEMALE, TIPOTICKET, FARENUEVO Y CABINNUEVO
CODIGO

```
library(e1071)
library(paqdatamining)

setwd("K:\\Datos\\datos tesis\\projecto_tesis")
# Cargando los datos de entrenamiento
basetrain <- read.csv("traintransf.csv",strip.white=TRUE,na.strings=c(""))
testData <- read.csv("testtransf.csv",strip.white=TRUE)


#METODO nb
print("procesando nb")
subset=c('titulopasajero','sexnuevo','agemale','sibspnuevo','parchnuevo','embarkednuevo',
        'sobrevfamilia')

formula <- as.simple.formula(subset, 'as.factor(survived)')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
nb=algornaivebayes(out$traindf,out$testdf,formula)
print("Error del los datos de prueba (20% de los datos)")
print(nb$error)

# Ejecutar el modelo constra los datos reales test y escribir los resultados
fileName <- "nb_solucion_r_wrapper.csv"
prediction <- predict(nb$alg, newdata=testData)
write.table(prediction,fileName,row.names = FALSE, col.names=FALSE,quote = FALSE)

# Generar la matriz de confusion.
matconf=table(out$testdf$survived, nb$pr,dnn=c("Real", "Predicho"))
print("Matriz de Confusion de los Datos de Prueba (20% de los datos)")
print(matconf)
```

Las entradas a este código son el archivo de datos de entrenamiento TRAINTRANSF.CSV, y el archivo de datos de prueba TESTTRANSF.CSV el cual produce la salida que es el archivo de predicciones “nb_solucion_r_wrapper.csv”.

Tabla C10
Código en R para modelo K- NEAREST NEIGHTBOUR KNN con la selección de subconjunto de variables por el método wrapper

Variables en el Modelo (Formula)
SURVIVED, SEXNUEVO, EMBARKEDNUEVO, SOBREVFAMILIA
Variables excluidas
PCLASS, TITULOPASAJERO, AGEMALE, AGEFEMALE, PARCHNUEVO, SIBSPNUEVO, TIPOTICKET, FARENUEVO Y CABINNUEVO
CODIGO

```
library(DMwR)
library(FSelector)
library(paqdatamining)

setwd("K:\\Datos\\datos tesis\\projecto_tesis")
# Cargando los datos de entrenamiento
basetrain <- read.csv("traintransf.csv",strip.white=TRUE,na.strings=c(""))
testData <- read.csv("testtransf.csv",strip.white=TRUE)

#METODO knn
print("procesando knn")
subset=c('sexnuevo','embarkednuevo','sobrevfamilia')

formula <- as.simple.formula(subset, 'survived')
print(formula)
#CALCULANDO EL MODELO Y EL ERROR
out=inicio(basetrain,subset)
knn=algorknn(out$traindf,out$testdf,formula)
print(knn$error)

# Ejecutar el modelo constra los datos reales test y escribir los resultados
fileName <- "knn_solucion_r_wrapper.csv"
write.table(knn$pr,fileName,row.names = FALSE, col.names=FALSE,quote = FALSE)

# Generar la matriz de confusion.
matconf=table(out$testdf$survived, knn$pr,dnn=c("Real", "Predicho"))
print("Matriz de Confusion de los Datos de Prueba (20% de los datos)")
print(matconf)
```

Las entradas a este código son el archivo de datos de entrenamiento TRAINTRANSF.CSV, y el archivo de datos de prueba TESTTRANSF.CSV el cual produce la salida que es el archivo de predicciones “knn_solucion_r_wrapper.csv”.

Tabla C11

Código en Python para el algoritmo de Árbol de Decision con la selección de subconjunto de variables por el método wrapper

Variables en el Modelo (Formula)
SURVIVED, PCLASS, TITULOPASAJERO, SEXNUEVO, AGEMALE, AGEFEMALE, SIBSPNUEVO, PARCHNUEVO, TIPOTICKET, CABINNUEVO, SOBREVFAMILIA
Variables excluidas
FARENUEVO Y EMBARKEDNUEVO
CODIGO

```
import pandas as pd
import numpy as np
import statsmodels.api as sm

from statsmodels.nonparametric import KDE, smoothers_lowess
from pandas import Series,DataFrame
from patsy import dmatrices
from sklearn import datasets, svm
from __future__ import division

import sys
sys.path.append('K:\Datos\datos tesis\projecto_tesis\python')
import csv_io as csv_io
import os
os.chdir('K:\Datos\datos tesis\projecto_tesis\python')

from sklearn import tree
from sklearn.utils import shuffle
from sklearn.metrics import roc_curve, auc
dftr = pd.read_csv("traintransf.csv")
dfte = pd.read_csv("testtransf.csv")
random_state = np.random.RandomState(1)

values=['titulopasajero','sexnuevo','agemale','agefemale','tipoticket','farenuevo',
'cabinnuevo','embarkednuevo','sobrevfamilia','survived']
dftrain=dftr[values]
n_samples, n_features = dftrain.shape
dftrain.head()
#df=dftit.loc[:,'pclass':'survived']
#dftrain['is_train'] = np.random.uniform(0, 1, len(dftr)) <= .80
lenf=len(values)-1
features = dftrain.columns[:lenf]
X=dftrain[features]
y, _ = pd.factorize(dftrain['survived'])
#test=test[values]
# shuffle and split training and test sets
X, y = shuffle(X, y, random_state=random_state)
ochoporc = int(n_samples * .8)
X_train, X_test = X[:ochoporc], X[ochoporc:]
y_train, y_test = y[:ochoporc], y[ochoporc:]

clf = tree.DecisionTreeClassifier()
clf.fit(X_train, y_train)

print("Matriz de Confusion de los Datos de Prueba (20% de los datos")
preds=clf.predict(X_test)
cross=pd.crosstab(y_test, preds, rownames=['actual'], colnames=['preds'])
print(cross)
```

```
print("Error del los datos de prueba (20% de los datos)")
error=(cross[0][1]+cross[1][0])/len(preds)

print(error)

dftest=dfte[values]
predicted=clf.predict(dftest[features])
print(predicted)
dfpred=DataFrame(predicted)
dfpred.to_csv("tree_solution_py_wrapper.csv", sep=',', na_rep='',header=False,index=False)
```

Las entradas a este código son el archivo de datos de entrenamiento TRAINTRANSF.CSV, y el archivo de datos de prueba TESTTRANSF.CSV el cual produce la salida que es el archivo de predicciones "rpart_solucion_py_wrapper.csv".

Tabla C12

Código en Python para el algoritmo Gradient Boosting con la selección de subconjunto de variables por el método wrapper

Variables en el Modelo (Formula)
SURVIVED, TITULOPASAJERO, SEXNUEVO, AGEMALE, AGEFEMALE, SIBSPNUEVO, EMBARKEDNUEVO, SOBREVFAMILIA
Variables excluidas
PCLASS , PARCHNUEVO, TIPOTICKET, FARENUEVO Y CABINNUEVO
CODIGO

```
import pandas as pd
import numpy as np
import statsmodels.api as sm

from statsmodels.nonparametric import KDE, smoothers_lowess
from pandas import Series,DataFrame
from patsy import dmatrices
from sklearn import datasets, svm
from __future__ import division

import sys
sys.path.append('K:\Datos\datos tesis\projecto_tesis\python')
import csv_io as csv_io
import os
os.chdir('K:\Datos\datos tesis\projecto_tesis\python')
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.utils import shuffle
from sklearn.metrics import roc_curve, auc
dftr = pd.read_csv("traintransf.csv")
dfte = pd.read_csv("testtransf.csv")
random_state = np.random.RandomState(1)

values=['titulopasajero','sexnuevo','agemale','agefemale','sibspnuevo','embarkednuevo','sobrevfamilia','survived']
dftrain=dftr[values]
n_samples, n_features = dftrain.shape
dftrain.head()
lenf=len(values)-1
features = dftrain.columns[:lenf]
X=dftrain[features]
```

```
y, _ = pd.factorize(dftrain['survived'])
# shuffle and split training and test sets
X, y = shuffle(X, y, random_state=random_state)
ochoporc = int(n_samples * .8)
X_train, X_test = X[:ochoporc], X[ochoporc:]
y_train, y_test = y[:ochoporc], y[ochoporc:]

clf = GradientBoostingClassifier(n_estimators=100, learning_rate=1.0,
...     max_depth=1, random_state=0).fit(X_train, y_train)
clf.fit(X, y)

print("Matriz de Confusion de los Datos de Prueba (20% de los datos")
preds=clf.predict(X_test)
cross=pd.crosstab(y_test, preds, rownames=['actual'], colnames=['preds'])
print(cross)
print("Error del los datos de prueba (20% de los datos)")
error=(cross[0][1]+cross[1][0])/len(preds)

dftest=dfte[values]
predicted=clf.predict(dftest[features])
print(predicted)
dfpred=DataFrame(predicted)
dfpred.to_csv("boost_solution_py_wrapper.csv", sep=',', na_rep='',header=False,index=False)
```

Las entradas a este código son el archivo de datos de entrenamiento TRAINTRANSF.CSV, y el archivo de datos de prueba TESTTRANSF.CSV el cual produce la salida que es el archivo de predicciones “boost_solucion_py_wrapper.csv”.

Tabla C13

Código en Python para el algoritmo Random Forest con la selección de subconjunto de variables por el método wrapper

Variables en el Modelo (Formula)
SURVIVED, TITULOPASAJERO, SEXNUEVO, AGEMALE, SIBSPNUEVO, PARCHNUEVO, TIPOTICKET, CABINNUEVO, EMBARKEDNUEVO
Variables excluidas
PCLASS, AGEFEMALE, FARENUEVO Y SOBREVFAMILIA
CODIGO

```
import pandas as pd
import numpy as np
import statsmodels.api as sm

from statsmodels.nonparametric import KDE, smoothers_lowess
from pandas import Series,DataFrame
from patsy import dmatrices
from sklearn import datasets, svm
from __future__ import division

import sys
sys.path.append('K:\Datos\datos tesis\projecto_tesis\python')
import csv_io as csv_io
import os
os.chdir('K:\Datos\datos tesis\projecto_tesis\python')

from sklearn.ensemble import RandomForestClassifier
from sklearn.utils import shuffle
```

```
from sklearn.metrics import roc_curve, auc
dftr = pd.read_csv("traintransf.csv")
dfte = pd.read_csv("testtransf.csv")
random_state = np.random.RandomState(1)
values=['titulopasajero','sexnuevo','agemale','sibspnuevo','parchnuevo','tipoticket',
        'cabinnuevo','embarkednuevo','survived']
dftrain=dftr[values]
n_samples, n_features = dftrain.shape
dftrain.head()

lenf=len(values)-1
features = dftrain.columns[:lenf]
X=dftrain[features]
y, _ = pd.factorize(dftrain['survived'])

# shuffle and split training and test sets
X, y = shuffle(X, y, random_state=random_state)
ochoporc = int(n_samples * .8)
X_train, X_test = X[:ochoporc], X[ochoporc:]
y_train, y_test = y[:ochoporc], y[ochoporc:]

clf = RandomForestClassifier(n_estimators=10, max_depth=None,
...     min_samples_split=1, random_state=0)
clf.fit(X_train, y_train)

print("Matriz de Confusion de los Datos de Prueba (20% de los datos")
preds=clf.predict(X_test)
cross=pd.crosstab(y_test, preds, rownames=['actual'], colnames=['preds'])
print(cross)
print("Error del los datos de prueba (20% de los datos)")
error=(cross[0][1]+cross[1][0])/len(preds)

dftest=dfte[values]
predicted=clf.predict(dftest[features])
print(predicted)
dfpred=DataFrame(predicted)
dfpred.to_csv("rf_solution_py_wrapper.csv", sep=',', na_rep='',header=False,index=False)
```

Las entradas a este código son el archivo de datos de entrenamiento TRAINTRANSF.CSV, y el archivo de datos de prueba TESTTRANSF.CSV el cual produce la salida que es el archivo de predicciones “rf_solucion_py_wrapper.csv”.

Tabla C14

Código en Python para el algoritmo Support Vector Machine SVM con la selección de subconjunto de variables por el método wrapper

Variables en el Modelo (Formula)
SURVIVED, TITULOPASAJERO, SEXNUEVO, AGEFEMALE , SIBSPNUEVO, PARCHNUEVO, TIPOTICKET , CABINNUEVO
Variables excluidas
PCLASS, AGEMALE, FARENUEVO, EMBARKEDNUEVO Y SOBREVFAMILIA
CODIGO
import pandas as pd import numpy as np import statsmodels.api as sm

```
from statsmodels.nonparametric import KDE, smoothers_lowess
from pandas import Series,DataFrame
from patsy import dmatrices
from sklearn import datasets, svm
from __future__ import division

import sys
sys.path.append('K:\Datos\datos tesis\projecto_tesis\python')
import csv_io as csv_io
import os
os.chdir('K:\Datos\datos tesis\projecto_tesis\python')

from sklearn import svm
dftr = pd.read_csv("traintransf.csv")
dfte = pd.read_csv("testtransf.csv")
dftrain = pd.DataFrame()
#El mejor modelo encontrado que da el menor error con R es el siguiente:
values=['titulopasajero','sexnuevo','agefemale','sibspnuevo','parchnuevo','tipoticket','cabinnuevo','survived']
dftrain=dftr[values]
#df=dftit.loc[:,'pclass':'survived']

dftrain['is_train'] = np.random.uniform(0, 1, len(dftr)) <= .80
dftrain.head()
train, test = dftrain[dftrain['is_train']==True], dftrain[dftrain['is_train']==False]
train=train[values]
test=test[values]
lenf=len(values)-1
features = dftrain.columns[:lenf]
y, _ = pd.factorize(train['survived'])

# train the model

clf = svm.LinearSVC()
clf.fit(train[features], y)

print("Matriz de Confusion de los Datos de Prueba (20% de los datos")
preds=clf.predict(X_test)
cross=pd.crosstab(y_test, preds, rownames=['actual'], colnames=['preds'])
print(cross)
print("Error del los datos de prueba (20% de los datos)")
error=(cross[0][1]+cross[1][0])/len(preds)

dfp=dfte[values]
predicted=clf.predict(dfp[features])
print(predicted)
dfpred=DataFrame(predicted)
dfpred.to_csv("svm_solution_py_wrapper.csv", sep=',', na_rep='',header=False,index=False)
```

Las entradas a este código son el archivo de datos de entrenamiento TRAINTRANSF.CSV, y el archivo de datos de prueba TESTTRANSF.CSV el cual produce la salida que es el archivo de predicciones “svm_solucion_py_wrapper.csv”.

Tabla C15

Código en Python para el algoritmo Regresión Logistica con la selección de subconjunto de variables por el método wrapper

Variables en el Modelo (Formula)
SURVIVED , TITULOPASAJERO, SEXNUEVO, SIBSPNUEVO, PARCHNUEVO, TIPOTICKET, EMBARKEDNUEVO, SOBREVFAMILIA
Variables excluidas
PCLASS, AGEMALE, AGEFEMALE, FARENUEVO Y CABINNUEVO
CODIGO

```
import pandas as pd
import numpy as np
import statsmodels.api as sm

from statsmodels.nonparametric import KDE, smoothers_lowess
from pandas import Series,DataFrame
from patsy import dmatrices
from sklearn import datasets, svm
from __future__ import division

import sys
sys.path.append('K:\Datos\datos tesis\projecto_tesis\python')
import csv_io as csv_io
import os
os.chdir('K:\Datos\datos tesis\projecto_tesis\python')

from sklearn.linear_model import LogisticRegression
from sklearn import linear_model
from sklearn.svm import LinearSVC
from sklearn.utils import shuffle
from sklearn.metrics import roc_curve, auc
dftr = pd.read_csv("traintransf.csv")
dfte = pd.read_csv("testtransf.csv")
random_state = np.random.RandomState(0)
values=[ 'titulopasajero','sexnuevo','sibspnuevo','parchnuevo','tipoticket',
        'embarkednuevo','sobrevfamilia','survived']
dftrain=dftr[values]
n_samples, n_features = dftrain.shape
dftrain.head()

lenf=len(values)-1
features = dftrain.columns[:lenf]
X=dftrain[features]
y, _ = pd.factorize(dftrain['survived'])

# shuffle and split training and test sets
X, y = shuffle(X, y, random_state=random_state)
ochoporc = int(n_samples * .8)
X_train, X_test = X[:ochoporc], X[ochoporc:]
y_train, y_test = y[:ochoporc], y[ochoporc:]


clf = LogisticRegression(penalty='l2', tol=0.00001)
clf.fit(X_train, y_train)

print("Matriz de Confusion de los Datos de Prueba (20% de los datos")
preds=clf.predict(X_test)
cross=pd.crosstab(y_test, preds, rownames=['actual'], colnames=['preds'])
```

```
print(cross)
print("Error del los datos de prueba (20% de los datos)")
error=(cross[0][1]+cross[1][0])/len(preds)

dftest=dfte[values]
predicted=clf.predict(dftest[features])
print(predicted)
dfpred=DataFrame(predicted)
dfpred.to_csv("log_solution_py_wapper.csv", sep=',', na_rep='',header=False,index=False)
```

Las entradas a este código son el archivo de datos de entrenamiento TRAINTRANSF.CSV, y el archivo de datos de prueba TESTTRANSF.CSV el cual produce la salida que es el archivo de predicciones "reg_solucion_py_wrapper.csv".

Tabla C16

Código en Python para el algoritmo Naive Bayes con la selección de subconjunto de variables por el método wrapper

Variables en el Modelo (Formula)
SURVIVED, TITULOPASAJERO, SEXNUEVO, AGEMALE, SIBSPNUEVO, PARCHNUEVO, EMBARKEDNUEVO, SOBREVFAMILIA
Variables excluidas
PCLASS, AGEFEMALE, TIPOTICKET, FARENUEVO Y CABINNUEVO
CODIGO

```
import pandas as pd
import numpy as np
import statsmodels.api as sm

from statsmodels.nonparametric import KDE, smoothers_lowess
from pandas import Series,DataFrame
from patsy import dmatrices
from sklearn import datasets, svm
from __future__ import division

import sys
sys.path.append('K:\Datos\datos tesis\projecto_tesis\python')
import csv_io as csv_io
import os
os.chdir('K:\Datos\datos tesis\projecto_tesis\python')

from sklearn.naive_bayes import GaussianNB
from sklearn.utils import shuffle
from sklearn.metrics import roc_curve, auc
dftr = pd.read_csv("traintransf.csv")
dfte = pd.read_csv("testtransf.csv")
random_state = np.random.RandomState(0)
lues=['titulopasajero','sexnuevo','agemale','sibspnuevo','parchnuevo','embarkednuevo','sobrevfamilia','survived']
dftrain=dftr[values]
n_samples, n_features = dftrain.shape
dftrain.head()

lenf=len(values)-1
features = dftrain.columns[:lenf]
X=dftrain[features]
y, _ = pd.factorize(dftrain['survived'])
```

```
# shuffle and split training and test sets
X, y = shuffle(X, y, random_state=random_state)
ochoporc = int(n_samples * .8)
X_train, X_test = X[:ochoporc], X[ochoporc:]
y_train, y_test = y[:ochoporc], y[ochoporc:]

clf = GaussianNB()
clf.fit(X_train, y_train)

print("Matriz de Confusion de los Datos de Prueba (20% de los datos")
preds=clf.predict(X_test)
cross=pd.crosstab(y_test, preds, rownames=['actual'], colnames=['preds'])
print(cross)
print("Error del los datos de prueba (20% de los datos)")
error=(cross[0][1]+cross[1][0])/len(preds)

dftest=dfte[values]
predicted=clf.predict(dftest[features])
print(predicted)
dfpred=DataFrame(predicted)
dfpred.to_csv("log_solution_py_wapper.csv", sep=',', na_rep='',header=False,index=False)

dfpred.to_csv("bn_solution_py_wrapper.csv", sep=',', na_rep='',header=False,index=False)
```

Las entradas a este código son el archivo de datos de entrenamiento TRAINTRANSF.CSV, y el archivo de datos de prueba TESTTRANSF.CSV el cual produce la salida que es el archivo de predicciones “nb_solucion_py_wrapper.csv”.

Tabla C17

Código en Python para el algoritmo k-Nearest Neightbour KNN con la selección de subconjunto de variables por el método wrapper

Variables en el Modelo (Formula)
SURVIVED, SEXNUEVO, EMBARKEDNUEVO, SOBREVFAMILIA
Variables excluidas
PCLASS, TITULOPASAJERO, AGEMALE, AGEFEMALE, PARCHNUEVO, SIBSPNUEVO, TIPOTICKET, FARENUEVO Y CABINNUEVO
CODIGO

```
import pandas as pd
import numpy as np
import statsmodels.api as sm

from statsmodels.nonparametric import KDE, smoothers_lowess
from pandas import Series,DataFrame
from patsy import dmatrices
from sklearn import datasets, svm
from __future__ import division

import sys
sys.path.append('K:\Datos\datos tesis\projecto_tesis\python')
import csv_io as csv_io
import os
os.chdir('K:\Datos\datos tesis\projecto_tesis\python')

from sklearn.neighbors import KNeighborsClassifier
from sklearn.utils import shuffle
```

```
from sklearn.metrics import roc_curve, auc
dftr = pd.read_csv("traintransf.csv")
dfte = pd.read_csv("testtransf.csv")
random_state = np.random.RandomState(1)

values=['titulopasajero','embarkednuevo','sobrevfamilia','survived']
dftrain=dftr[values]
n_samples, n_features = dftrain.shape
dftrain.head()
lenf=len(values)-1
features = dftrain.columns[:lenf]
X=dftrain[features]
y, _ = pd.factorize(dftrain['survived'])
# shuffle and split training and test sets
X, y = shuffle(X, y, random_state=random_state)
ochoporc = int(n_samples * .8)
X_train, X_test = X[:ochoporc], X[ochoporc:]
y_train, y_test = y[:ochoporc], y[ochoporc:]

clf = KNeighborsClassifier(n_neighbors=8)
clf.fit(X, y)

print("Matriz de Confusion de los Datos de Prueba (20% de los datos")
preds=clf.predict(X_test)
cross=pd.crosstab(y_test, preds, rownames=['actual'], colnames=['preds'])
print(cross)
print("Error del los datos de prueba (20% de los datos)")
error=(cross[0][1]+cross[1][0])/len(preds)

dftest=dfte[values]
predicted=clf.predict(dftest[features])
print(predicted)
dfpred=DataFrame(predicted)
dfpred.to_csv("knn_solution_py_wrapper.csv", sep=',', na_rep='',header=False,index=False)
```

Las entradas a este código son el archivo de datos de entrenamiento TRAINTRANSF.CSV, y el archivo de datos de prueba TESTTRANSF.CSV el cual produce la salida que es el archivo de predicciones “knn_solucion_py_wrapper.csv”.

Apéndice D

Códigos para la estracción de características de los datos

Las siguientes tablas contienen el código en R, de las transformaciones a los datos originales TRAIN y TEST, (sección 4.4.1.1 Extracción de características o nuevas variables a partir de los datos). Los datos contenidos en los archivos train.csv se transforman en el archivo traintransf.csv y los datos contenidos en el archivo test.csv se transforman en el archivo testtransf.csv.

Tabla D1

Código en R para generar el archivo TRAINTRANSF.CSV a partir de TRAIN.CSV

CODIGO

```
setwd("K:\\Datos\\datos tesis\\projecto_tesis")

# Cargando los datos de entrenamiento y realizando algunas categorias

readtrain <- read.csv("train.csv",strip.white=TRUE,na.strings=c(""))
readtest <- read.csv("test.csv",strip.white=TRUE,na.strings=c(""))

#convirtiendo la variable embarked si su valor es Q=2,S=3,C=1
readtrain$embarked[ is.na(readtrain$embarked) ] <- 'C'
readtrain$embarked <- as.integer(readtrain$embarked)
readtrain$embarked[ readtrain$embarked=="Q"] <- 2
readtrain$embarked[ readtrain$embarked=="S"]<- 3
readtrain$embarked[ readtrain$embarked=="C"]<- 1


# Reemplazando valores perdidos en TRAIN con los valores encontrados con el metodo EM
readtrain$age[ is.na(readtrain$age) ] <- 29.8811
readtest$age[ is.na(readtest$age) ] <- 29.8811
readtest$fare[ is.na(readtest$fare) ] <- 33.2955
#readtrain$cabin <- as.character(readtrain$cabin)
#readtest$cabin[ is.na(readtest$cabin) ] <- "T"

#dividiendo la variable age en agemale y agefemale
#para male
readtrain["agemale"]=NA
agemalev <- rep(NA,length(readtrain$age))
readtrain$agemale <- agemalev
readtrain$agemale[readtrain$sex=="male"] <- readtrain$age[readtrain$sex=="male"]
readtrain$agemale[readtrain$sex=="female"] <- 0

#para female
readtrain["agefemale"]=NA
agefemalev <- rep(NA,length(readtrain$age))
readtrain$agefemale <- agefemalev
readtrain$agefemale[readtrain$sex=="male"] <- 0
readtrain$agefemale[readtrain$sex=="female"] <- readtrain$age[readtrain$sex=="female"]

#transformado la variable name a la variable titulopasajero

readtrain["titulopasajero"]=NA
titulop <- rep(NA,length(readtrain$name))

for (i in 1:length(readtrain$name)) {
  if( any(grep('MR.',toupper(readtrain$name[i]), fixed=TRUE)>=1))
    titulop[i]=1
  if (any(grep('DON.',toupper(readtrain$name[i]), fixed=TRUE)>=1))
    titulop[i]=2
  if (any(grep('SIR.',toupper(readtrain$name[i]), fixed=TRUE)>=1))
    titulop[i]=3
  if (any(grep('JONKHEER.',toupper(readtrain$name[i]), fixed=TRUE)>=1))
```

```
   titulop[i]=4
  if (any(grep('REV.',toupper(readtrain$name[i]), fixed=TRUE)>=1))
   titulop[i]=5
  if (any(grep('DR.',toupper(readtrain$name[i]), fixed=TRUE)>=1))
   titulop[i]=6
  if (any(grep('MAJOR.',toupper(readtrain$name[i]), fixed=TRUE)>=1))
   titulop[i]=7
  if (any(grep('CAPT.',toupper(readtrain$name[i]), fixed=TRUE)>=1))
   titulop[i]=8
  if (any(grep('COL.',toupper(readtrain$name[i]), fixed=TRUE)>=1))
   titulop[i]=9
  if (any(grep('MRS.',toupper(readtrain$name[i]), fixed=TRUE)>=1) ||
any(grep('DONA.',toupper(readtrain$name[i]), fixed=TRUE)>=1 ) ||
any(grep('LADY.',toupper(readtrain$name[i]), fixed=TRUE)>=1 ))
   titulop[i]=10
  if (any(grep('MME.',toupper(readtrain$name[i]), fixed=TRUE)>=1))
   titulop[i]=11
  if (any(grep('COUNTESS.',toupper(readtrain$name[i]), fixed=TRUE)>=1))
   titulop[i]=12
  if (any(grep('MS.',toupper(readtrain$name[i]), fixed=TRUE)>=1))
   titulop[i]=13
  if (any(grep('MISS.',toupper(readtrain$name[i]), fixed=TRUE)>=1))
   titulop[i]=14
  if (any(grep('MLLE.',toupper(readtrain$name[i]), fixed=TRUE)>=1))
   titulop[i]=15
  if (any(grep('MASTER.',toupper(readtrain$name[i]), fixed=TRUE)>=1))
   titulop[i]=16
}
readtrain$titulopasajero = titulop

#transformado la variable ticket a la variable tipotcket

readtrain["tipoticket"]=NA
tipotick <- rep(NA,length(readtrain$ticket))
for (i in 1:length(readtrain$ticket)) {
  if( any(regexpr('A./5.',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1) ||
      any(regexpr('A.5.',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1) ||
      any(regexpr('A/S',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1) ||
      any(regexpr('A/5',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1) )
   tipotick[i]=1
  if( any(regexpr('A/4',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1) ||
      any(regexpr('A4',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1))
   tipotick[i]=2
  if( any(regexpr('C',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1))
   tipotick[i]=3
  if( any(regexpr('C.A.',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1) ||
      any(regexpr('CA',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1))
   tipotick[i]=4
  if( any(regexpr('F.C.',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1) ||
      any(regexpr('F.C.C.',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1))
   tipotick[i]=5
  if( any(regexpr('FA',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1) ||
      any(regexpr('LINE',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1) ||
      any(regexpr('P/PP',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1) ||
```

```
      any(regexpr('AQ',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1) ||
      any(regexpr('A.',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1) ||
      any(regexpr('2.',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1) ||
      any(regexpr('LP',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=6
if( any(regexpr('PC',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=7
if( any(regexpr('PP',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=8
if( any(regexpr('S.C.',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=9
if( any(regexpr('S.O.',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=10
if( any(regexpr('S.O.C.',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=11
if( any(regexpr('S.O.P.',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=12
if( any(regexpr('S.P.',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=13
if( any(regexpr('S.W.',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=14
if( any(regexpr('SC',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=15
if( any(regexpr('SCO',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=16
if( any(regexpr('SO/C',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=17
if( any(regexpr('SOTON',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=18
if( any(regexpr('STON',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=19
if( any(regexpr('SW/PP',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=20
if( any(regexpr('W./C.',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=21
if( any(regexpr('W.E.P.',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=22
if( any(regexpr('W/C',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=23
if( any(regexpr('WE/P',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=24
if( any(regexpr('1',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=25
if( any(regexpr('2',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=26
if( any(regexpr('3',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=27
if( any(regexpr('4',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=28
if( any(regexpr('5',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=29
if( any(regexpr('6',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=30
if( any(regexpr('7',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1))
```

```
   tipotick[i]=31
  if( any(regexpr('8',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1))
   tipotick[i]=32
  if( any(regexpr('9',toupper(readtrain$ticket[i]), fixed=TRUE)[1]==1))
   tipotick[i]=33
}
readtrain$tipoticket = tipotick

#transformado la variable cabin a la variable rangecabin

readtrain["rangecabin"]=NA
rangecabint <- rep(NA,length(readtrain$fare))
rangecabint[ substr(readtrain$cabin,1,1) == "A"] <- 1
rangecabint[ substr(readtrain$cabin,1,1) == "B"] <- 2
rangecabint[ substr(readtrain$cabin,1,1) == "C"] <- 3
rangecabint[ substr(readtrain$cabin,1,1) == "D"] <- 4
rangecabint[ substr(readtrain$cabin,1,1) == "E"] <- 5
rangecabint[ substr(readtrain$cabin,1,1) == "F"] <- 6
rangecabint[ substr(readtrain$cabin,1,1) == "G"] <- 7
rangecabint[ substr(readtrain$cabin,1,1) == "T"] <- 8
rangecabint[ substr(readtrain$cabin,1,1) != "A" &
             substr(readtrain$cabin,1,1) != "B" &
             substr(readtrain$cabin,1,1) != "C" &
             substr(readtrain$cabin,1,1) != "D" &
             substr(readtrain$cabin,1,1) != "E" &
             substr(readtrain$cabin,1,1) != "F" &
             substr(readtrain$cabin,1,1) != "G" &
             substr(readtrain$cabin,1,1) != "T"]  <- 8
readtrain$rangecabin = rangecabint
readtrain$rangecabin[ is.na(readtrain$rangecabin) ] <- 8

#agregando nuevo rasgo deducidos apartir de los otros rasgos denominado
#sobrevfamilia


#uniendo los dos dataframes train y test en un dataframe  ordenado por
#la variable name
readtrainem=data.frame(readtrain[2:11])
readtotal=rbind(readtrainem,readtest)

readtotal["sobrevfamilia"]=NA
sobrevfam <- rep(NA,length(readtotal$name))
readtotal$sobrevfamilia=sobrevfam


readtotal["index"]=NA
ind <- rep(NA,length(readtotal$name))
for (i in 1:length(readtotal$name))
{
  ind[i]=i
}
readtotal$index=ind

sortreadtotal <- readtotal[order(readtotal$name) , ]
```

```
sortreadtotal$cabin <- as.character(sortreadtotal$cabin)
sortreadtotal$cabin[ is.na(sortreadtotal$cabin) ] <- "9"
sortreadtotal$sobrevfamilia <- as.integer(sortreadtotal$sobrevfamilia)
#creando el dataframe de apellidos unicos

apell1=substr(sortreadtotal$name[1],1,regexpr(',',sortreadtotal$name[1])[1] - 1)
pclass1=sortreadtotal$pclass[1]
cabin1=substr(sortreadtotal$cabin[1],1,1)

if(any(sortreadtotal$sibsp[1]==0) & any(sortreadtotal$parch[1]==0)) {
 sortreadtotal$sobrevfamilia[1]=4
} else {
 if (any(sortreadtotal$pclass[1])==1 ) {
      sortreadtotal$sobrevfamilia[1]=1
 }  else if (any(sortreadtotal$pclass[1])==2) {
      sortreadtotal$sobrevfamilia[1]=2
 }  else if (any(sortreadtotal$pclass[1])==3) {
      sortreadtotal$sobrevfamilia[1]=3
 }
}
pclassvar=sortreadtotal$sobrevfamilia[1]
for (i in 2:length(sortreadtotal$name)) {
 apell2=substr(sortreadtotal$name[i],1,regexpr(',',sortreadtotal$name[i])[1] - 1)
 pclass2=sortreadtotal$pclass[i]
 cabin2=substr(sortreadtotal$cabin[i],1,1)
 if ( any(toupper(apell1)!=toupper(apell2)) || any(pclass1!=pclass2)
    || any(cabin1!=cabin2 )) {
  if (any(sortreadtotal$sibsp[i]==0) & any(sortreadtotal$parch[i]==0)) {
   sortreadtotal$sobrevfamilia[i]=4
  } else {
   if (any(sortreadtotal$pclass[i]==1)) {
    sortreadtotal$sobrevfamilia[i]<-1
   }  else if (any(sortreadtotal$pclass[i]==2)) {
    sortreadtotal$sobrevfamilia[i]<-2
   }  else if (any(sortreadtotal$pclass[i]==3)) {
    sortreadtotal$sobrevfamilia[i]<-3
   }
  }
  apell1=apell2
  pclass1=pclass2
  cabin1=cabin2
  pclassvar=sortreadtotal$sobrevfamilia[i]
 } else {
  if (any(sortreadtotal$sibsp[i]==0) & any(sortreadtotal$parch[i]==0)) {
   sortreadtotal$sobrevfamilia[i]=4
  } else if (any(pclassvar!=4)) {
  sortreadtotal$sobrevfamilia[i]=pclassvar
  } else {
   if (any(sortreadtotal$pclass[i]==1)) {
    sortreadtotal$sobrevfamilia[i]<-1
   }  else if (any(sortreadtotal$pclass[i]==2)) {
    #print('c2')
    sortreadtotal$sobrevfamilia[i]<-2
   }  else if (any(sortreadtotal$pclass[i]==3)) {
```

```
      #print('c3')
      sortreadtotal$sobrevfamilia[i]<-3
    }
    pclassvar=sortreadtotal$sobrevfamilia[i]
  }
 }
}

#extraendo la parte de datos de entrenamiento del dataframe sortreadtotal
sortreadtotaltemp <- sortreadtotal[order(sortreadtotal$index) , ]
readtrainn=subset(sortreadtotaltemp,sortreadtotaltemp$index<892  )


#   realizando los cortes para crear los intervalos
maxFare <- max(max(readtrain$fare),max(readtest$fare))
loscortes <- c(0,7,10,20,30,40,50,75,100,maxFare)
cutFare <- cut(readtrainn$fare,breaks=loscortes,include.lowest=T)
loscortes <- c(-1,0,15,20,30,40,50,60,100)
cutAgemale <- cut(readtrain$agemale,breaks=loscortes,include.lowest=T)
loscortes <- c(-1,0,15,20,30,40,50,60,100)
cutAgefemale <- cut(readtrain$agefemale,breaks=loscortes,include.lowest=T)
loscortes <- c(0,1,2,3,100)
cutParch <- cut(readtrain$parch,breaks=loscortes,include.lowest=T)
loscortes <- c(0,1,2,100)
cutSibsp <- cut(readtrain$sibsp,breaks=loscortes,include.lowest=T)
loscortes <- c(0,2,4,5,6,8,13,14,16,17,19,20,21,24,27,33)
cuttipoticket <- cut(readtrain$tipoticket,breaks=loscortes,include.lowest=T)
loscortes <- c(0,1,3,7,8)
cutrangecabin <- cut(readtrain$rangecabin,breaks=loscortes,include.lowest=T)


# Poninedo lo datos en un dataframe sin todas las categorias que se esta ignorando

baseData <- data.frame(cbind(readtrain$pclass,
                     readtrain$titulopasajero,
                     readtrain$sex,
                     cutAgemale,
                     cutAgefemale,
                     cutSibsp,
                     cutParch,
                     cuttipoticket,
                     cutFare,
                     cutrangecabin,
                     readtrain$embarked,
                     readtrainn$sobrevfamilia,
                     readtrain$survived)
                  )

# Porniendo los nombres de las variables
names(baseData)[1] <- "pclass"
names(baseData)[2] <- "titulopasajero"
names(baseData)[3] <- "sexnuevo"
names(baseData)[4] <- "agemale"
names(baseData)[5] <- "agefemale"
```

```
names(baseData)[6] <- "sibspnuevo"
names(baseData)[7] <- "parchnuevo"
names(baseData)[8] <- "tipoticket"
names(baseData)[9] <- "farenuevo"
names(baseData)[10] <- "cabinnuevo"
names(baseData)[11] <- "embarkednuevo"
names(baseData)[12] <- "sobrevfamilia"
names(baseData)[13] <- "survived"

basetrain=baseData

#escribiendo el dataframe en un archivo csv
write.table(baseData,file="traintransf.csv",sep=",",row.names=F)
```

Tabla D2

Código en R para generar el archivo TESTTRANSF.CSV a partir de TEST.CSV

CODIGO

```
#   configurando el directorio de trabajo
setwd("K:\\Datos\\datos tesis\\projecto_tesis")

# Cargando los datos de prueba y haciendo algunas categorias

readtest <- read.csv("test.csv",strip.white=TRUE)

#convirtiendo la variable embarked si su valor es Q=2,S=3,C=1
readtest$embarked[ is.na(readtest$embarked) ] <- 'C'
readtest$embarked <- as.integer(readtest$embarked)
readtest$embarked[ readtest$embarked=="Q"] <- 2
readtest$embarked[ readtest$embarked=="S"]<- 3
readtest$embarked[ readtest$embarked=="C"]<- 1

# Reemplazando valores perdidos en TEST con los valores encontrados con el metodo EM

readtest$age[ is.na(readtest$age) ] <- 29.8811
readtest$fare[ is.na(readtest$fare) ] <- 33.2955

#dividiendo la variable age en agemale y agefemale
#para male
readtest["agemale"]=NA
agemalev <- rep(NA,length(readtest$age))
readtest$agemale <- agemalev
readtest$agemale[readtest$sex=="male"] <- readtest$age[readtest$sex=="male"]
readtest$agemale[readtest$sex=="female"] <- 0

#para female
readtest["agefemale"]=NA
agefemalev <- rep(NA,length(readtest$age))
readtest$agefemale <- agefemalev
readtest$agefemale[readtest$sex=="male"] <- 0
readtest$agefemale[readtest$sex=="female"] <- readtest$age[readtest$sex=="female"]
```

```
#transformado la variable name a la variable titulopasajero

readtest["titulopasajero"]=NA
titulop <- rep(NA,length(readtest$name))

for (i in 1:length(readtest$name)) {
  if( any(grep('MR.',toupper(readtest$name[i]), fixed=TRUE)>=1))
    titulop[i]=1
  if (any(grep('DON.',toupper(readtest$name[i]), fixed=TRUE)>=1))
    titulop[i]=2
  if (any(grep('SIR.',toupper(readtest$name[i]), fixed=TRUE)>=1))
    titulop[i]=3
  if (any(grep('JONKHEER.',toupper(readtest$name[i]), fixed=TRUE)>=1))
    titulop[i]=4
  if (any(grep('REV.',toupper(readtest$name[i]), fixed=TRUE)>=1))
    titulop[i]=5
  if (any(grep('DR.',toupper(readtest$name[i]), fixed=TRUE)>=1))
    titulop[i]=6
  if (any(grep('MAJOR.',toupper(readtest$name[i]), fixed=TRUE)>=1))
    titulop[i]=7
  if (any(grep('CAPT.',toupper(readtest$name[i]), fixed=TRUE)>=1))
    titulop[i]=8
  if (any(grep('COL.',toupper(readtest$name[i]), fixed=TRUE)>=1))
    titulop[i]=9
  if (any(grep('MRS.',toupper(readtest$name[i]), fixed=TRUE)>=1) ||
any(grep('DONA.',toupper(readtest$name[i]), fixed=TRUE)>=1 ) ||
any(grep('LADY.',toupper(readtest$name[i]), fixed=TRUE)>=1 ))
    titulop[i]=10
  if (any(grep('MME.',toupper(readtest$name[i]), fixed=TRUE)>=1))
    titulop[i]=11
  if (any(grep('COUNTESS.',toupper(readtest$name[i]), fixed=TRUE)>=1))
    titulop[i]=12
  if (any(grep('MS.',toupper(readtest$name[i]), fixed=TRUE)>=1))
    titulop[i]=13
  if (any(grep('MISS.',toupper(readtest$name[i]), fixed=TRUE)>=1))
    titulop[i]=14
  if (any(grep('MLLE.',toupper(readtest$name[i]), fixed=TRUE)>=1))
    titulop[i]=15
  if (any(grep('MASTER.',toupper(readtest$name[i]), fixed=TRUE)>=1))
    titulop[i]=16
}
readtest$titulopasajero = titulop

#transformado la variable ticket a la variable tipotcket

readtest["tipoticket"]=NA
tipotick <- rep(NA,length(readtest$ticket))
for (i in 1:length(readtest$ticket)) {
  if( any(regexpr('A./5.',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1) ||
      any(regexpr('A.5.',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1) ||
      any(regexpr('A/S',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1) ||
      any(regexpr('A/5',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1) )
    tipotick[i]=1
  if( any(regexpr('A/4',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1) ||
```

```
     any(regexpr('A4',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=2
 if( any(regexpr('C',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=3
 if( any(regexpr('C.A.',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1) ||
     any(regexpr('CA',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=4
 if( any(regexpr('F.C.',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1) ||
     any(regexpr('F.C.C.',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=5
 if( any(regexpr('FA',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1) ||
     any(regexpr('LINE',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1) ||
     any(regexpr('P/PP',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1) ||
     any(regexpr('AQ',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1) ||
     any(regexpr('A.',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1) ||
     any(regexpr('2.',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1) ||
     any(regexpr('LP',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=6
 if( any(regexpr('PC',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=7
 if( any(regexpr('PP',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=8
 if( any(regexpr('S.C.',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=9
 if( any(regexpr('S.O.',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=10
 if( any(regexpr('S.O.C.',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=11
 if( any(regexpr('S.O.P.',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=12
 if( any(regexpr('S.P.',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=13
 if( any(regexpr('S.W.',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=14
 if( any(regexpr('SC',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=15
 if( any(regexpr('SCO',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=16
 if( any(regexpr('SO/C',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=17
 if( any(regexpr('SOTON',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=18
 if( any(regexpr('STON',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=19
 if( any(regexpr('SW/PP',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=20
 if( any(regexpr('W./C.',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=21
 if( any(regexpr('W.E.P.',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=22
 if( any(regexpr('W/C',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=23
 if( any(regexpr('WE/P',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1))
  tipotick[i]=24
```

```
  if( any(regexpr('1',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1))
    tipotick[i]=25
  if( any(regexpr('2',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1))
    tipotick[i]=26
  if( any(regexpr('3',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1))
    tipotick[i]=27
  if( any(regexpr('4',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1))
    tipotick[i]=28
  if( any(regexpr('5',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1))
    tipotick[i]=29
  if( any(regexpr('6',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1))
    tipotick[i]=30
  if( any(regexpr('7',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1))
    tipotick[i]=31
  if( any(regexpr('8',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1))
    tipotick[i]=32
  if( any(regexpr('9',toupper(readtest$ticket[i]), fixed=TRUE)[1]==1))
    tipotick[i]=33
}
readtest$tipoticket = tipotick

#transformado la variable cabin a la variable rangecabin

readtest["rangecabin"]=NA
rangecabint <- rep(NA,length(readtest$fare))
rangecabint[ substr(readtest$cabin,1,1) == "A"] <- 1
rangecabint[ substr(readtest$cabin,1,1) == "B"] <- 2
rangecabint[ substr(readtest$cabin,1,1) == "C"] <- 3
rangecabint[ substr(readtest$cabin,1,1) == "D"] <- 4
rangecabint[ substr(readtest$cabin,1,1) == "E"] <- 5
rangecabint[ substr(readtest$cabin,1,1) == "F"] <- 6
rangecabint[ substr(readtest$cabin,1,1) == "G"] <- 7
rangecabint[ substr(readtest$cabin,1,1) == "T"] <- 8
rangecabint[ substr(readtest$cabin,1,1) != "A" &
             substr(readtest$cabin,1,1) != "B" &
             substr(readtest$cabin,1,1) != "C" &
             substr(readtest$cabin,1,1) != "D" &
             substr(readtest$cabin,1,1) != "E" &
             substr(readtest$cabin,1,1) != "F" &
             substr(readtest$cabin,1,1) != "G" &
             substr(readtest$cabin,1,1) != "T"]  <- 8
readtest$rangecabin = rangecabint
readtest$rangecabin[ is.na(readtest$rangecabin) ] <- 8


#agregando nuevo rasgo deducidos apartir de los otros rasgos denominado
#sobrevfamilia


#uniendo los dos dataframes train y test en un dataframe  ordenado por
#la variable name
readtrainem=data.frame(readtrain[2:11])
readtestem=data.frame(readtest[1:10])
readtotal=rbind(readtrainem,readtestem)
```

```
#creando la variable sobrevfamilia
readtotal["sobrevfamilia"]=NA
sobrevfam <- rep(NA,length(readtotal$name))
readtotal$sobrevfamilia=sobrevfam


readtotal["index"]=NA
ind <- rep(NA,length(readtotal$name))
for (i in 1:length(readtotal$name))
{
  ind[i]=i
}
readtotal$index=ind

sortreadtotal <- readtotal[order(readtotal$name) , ]
sortreadtotal$cabin <- as.character(sortreadtotal$cabin)
sortreadtotal$cabin[ is.na(sortreadtotal$cabin) ] <- "9"
sortreadtotal$sobrevfamilia <- as.integer(sortreadtotal$sobrevfamilia)
#creando el dataframe de apellidos unicos

apell1=substr(sortreadtotal$name[1],1,regexpr(',',sortreadtotal$name[1])[1] - 1)
pclass1=sortreadtotal$pclass[1]
cabin1=substr(sortreadtotal$cabin[1],1,1)

if(any(sortreadtotal$sibsp[1]==0) & any(sortreadtotal$parch[1]==0)) {
  sortreadtotal$sobrevfamilia[1]=4
} else {
  if (any(sortreadtotal$pclass[1])==1 ) {
    sortreadtotal$sobrevfamilia[1]=1
  }  else if (any(sortreadtotal$pclass[1])==2) {
    sortreadtotal$sobrevfamilia[1]=2
  }  else if (any(sortreadtotal$pclass[1])==3) {
    sortreadtotal$sobrevfamilia[1]=3
  }
}
pclassvar=sortreadtotal$sobrevfamilia[1]
for (i in 2:length(sortreadtotal$name)) {
  apell2=substr(sortreadtotal$name[i],1,regexpr(',',sortreadtotal$name[i])[1] - 1)
  pclass2=sortreadtotal$pclass[i]
  cabin2=substr(sortreadtotal$cabin[i],1,1)
  if ( any(toupper(apell1)!=toupper(apell2)) || any(pclass1!=pclass2)
       || any(cabin1!=cabin2 )) {
    if (any(sortreadtotal$sibsp[i]==0) & any(sortreadtotal$parch[i]==0)) {
      sortreadtotal$sobrevfamilia[i]=4
    } else {
      if (any(sortreadtotal$pclass[i]==1)) {
        sortreadtotal$sobrevfamilia[i]<-1
      }  else if (any(sortreadtotal$pclass[i]==2)) {
        sortreadtotal$sobrevfamilia[i]<-2
      }  else if (any(sortreadtotal$pclass[i]==3)) {
        sortreadtotal$sobrevfamilia[i]<-3
      }
    }
```

```
   apell1=apell2
   pclass1=pclass2
   cabin1=cabin2
   pclassvar=sortreadtotal$sobrevfamilia[i]
  } else {
   if (any(sortreadtotal$sibsp[i]==0) & any(sortreadtotal$parch[i]==0)) {
    sortreadtotal$sobrevfamilia[i]=4
   } else if (any(pclassvar!=4)) {
    sortreadtotal$sobrevfamilia[i]=pclassvar
   } else {
    if (any(sortreadtotal$pclass[i]==1)) {
     sortreadtotal$sobrevfamilia[i]<-1
    }  else if (any(sortreadtotal$pclass[i]==2)) {
     #print('c2')
     sortreadtotal$sobrevfamilia[i]<-2
    }  else if (any(sortreadtotal$pclass[i]==3)) {
     #print('c3')
     sortreadtotal$sobrevfamilia[i]<-3
    }
    pclassvar=sortreadtotal$sobrevfamilia[i]
   }
  }
}

#extraendo la parte de datos de prueba del dataframe sortreadtotal
sortreadtotaltemp <- sortreadtotal[order(sortreadtotal$index) , ]
readtestn=subset(sortreadtotaltemp,sortreadtotaltemp$index>891  )

#   realizando los cortes para crear los intervalos
maxFare <- max(max(readtest$fare),max(readtest$fare))
loscortes <- c(0,7,10,20,30,40,50,75,100,maxFare)
cutFare <- cut(readtest$fare,breaks=loscortes,include.lowest=T)
loscortes <- c(-1,0,15,20,30,40,50,60,100)
cutAgemale <- cut(readtest$agemale,breaks=loscortes,include.lowest=T)
loscortes <- c(-1,0,15,20,30,40,50,60,100)
cutAgefemale <- cut(readtest$agefemale,breaks=loscortes,include.lowest=T)
loscortes <- c(0,1,2,3,100)
cutParch <- cut(readtest$parch,breaks=loscortes,include.lowest=T)
loscortes <- c(0,1,2,100)
cutSibsp <- cut(readtest$sibsp,breaks=loscortes,include.lowest=T)
loscortes <- c(0,2,4,5,6,8,13,14,16,17,19,20,21,24,27,33)
cuttipoticket <- cut(readtest$tipoticket,breaks=loscortes,include.lowest=T)
loscortes <- c(0,1,3,7,8)
cutrangecabin <- cut(readtest$rangecabin,breaks=loscortes,include.lowest=T)


# Poninedo lo datos en un dataframe sin todas las categorias que se esta ignorando

testData <- data.frame(cbind(readtest$pclass,
                             readtest$titulopasajero,
                             readtest$sex,
                             cutAgemale,
                             cutAgefemale,
                             cutSibsp,
```

```
                    cutParch,
                    cuttipoticket,
                    cutFare,
                    cutrangecabin,
                    readtest$embarked,
                    readtestn$sobrevfamilia,
                    0)
)

# Porniendo los nombres de las variables
names(testData)[1] <- "pclass"
names(testData)[2] <- "titulopasajero"
names(testData)[3] <- "sexnuevo"
names(testData)[4] <- "agemale"
names(testData)[5] <- "agefemale"
names(testData)[6] <- "sibspnuevo"
names(testData)[7] <- "parchnuevo"
names(testData)[8] <- "tipoticket"
names(testData)[9] <- "farenuevo"
names(testData)[10] <- "cabinnuevo"
names(testData)[11] <- "embarkednuevo"
names(testData)[12] <- "sobrevfamilia"
names(testData)[13] <- "survived"

#escribiendo el dataframe en un archivo csv
write.table(testData,file="testtransf.csv",sep=",",row.names=F)
```

Apéndice E

Código para crear el paquete paqdatamining

El siguiente código es para crear el paquete paqdatamining que sera utilizado para estimar los modelos de mineria de datos que se utilizan en este trabajo, y que son los siguientes:

k) ÁRBOL DE DECISIÓN
l) GRADIENT BOOSTING
m) BAGGING
n) ADA BOOST (ADA)
o) RANDOMFOREST (RF)
p) SUPPORT VECTOR MACHINE (SVM)
q) REGRESIÓN LOGÍSTICA
r) REDES NEURONALES
s) NAÏVE BAYES
t) K.VECINDAD MAS CERCANA

Será utilizado en el Capitulo 4 (Pre-procesamiento y transformación de datos), y en el Capitulo 5 (Aspectos Algoritmicos de la Mineria de Datos).

Tabla E1

Código en R para la creacion del paquete "paqdatamining"

Creacion del paquete paqdatamining

```
require(stats)
inicio=function(basetrain,subset){
set.seed(46)
nobs <- nrow(basetrain) # 891 observations
sample <- trainin <- sample(nrow(basetrain), 0.8*nobs) # 712 observations
testin <- sample(setdiff(seq_len(nrow(basetrain)), trainin)) # 179 observations
target  <- "survived"
traindf <- basetrain [trainin, c(subset, target)]
testdf <- basetrain [testin, c(subset, target)]
lista=list(traindf=traindf,testdf=testdf)
return(lista)
}

#funcion para el algortimo RPART
algorrpart=function(traindf,testdf,formula){
tree <- rpart(formula,
          data=traindf,
          method="class",
          parms=list(split="information"),
          control=rpart.control(usesurrogate=0,
                        maxsurrogate=0))
pr=predict(tree, newdata=testdf,type="class")
error.rate = sum(testdf$survived != pr ) / nrow(testdf)
lista=list(alg=tree,pr=pr,error=error.rate)
return(lista)
}

#funcion para el algortimo ADABOOST
algorada=function(traindf,testdf,formula){
modelada <- ada(formula,
            data=traindf,
            control=rpart.control(maxdepth=30,
                        cp=0.010000,
                        minsplit=20,
                        xval=10),
            iter=50)
pr=predict(modelada, testdf)
error.rate = sum(testdf$survived != pr ) / nrow(testdf)
lista=list(alg=modelada,pr=pr,error=error.rate)
return(lista)
}

#funcion para el algortimo RANDOMFOREST
algorrf=function(traindf,testdf,formula){
rf <- randomForest(formula,
              data=traindf,
              ntree=500,
              mtry=3,
              importance=TRUE,
              na.action=na.roughfix,
              replace=FALSE)
pr=predict(rf, testdf)
error.rate = sum(testdf$survived != pr ) / nrow(testdf)
lista=list(alg=rf,pr=pr,error=error.rate)
return(lista)
```

```
}

#funcion para el algortimo SUPPORT VECTOR MACHINE SVM
algorsvm=function(traindf,testdf,formula){
ksvm <- ksvm(formula,
          data=traindf,
          kernel="rbfdot",
          prob.model=TRUE)

pr=predict(ksvm, testdf)
error.rate = sum(testdf$survived != pr ) / nrow(testdf)
lista=list(alg=ksvm,pr=pr,error=error.rate)
return(lista)
}

#funcion para el algortimo REGRESION LOGISTICA
algorreg=function(traindf,testdf,formula){
glm <- glm(formula,
        data=traindf,
        family=binomial(link="logit"))
pr <- as.vector(ifelse(predict(glm, type="response", newdata=testdf) > 0.5, "1", "0"))
error.rate = sum(testdf$survived != pr ) / nrow(testdf)
lista=list(alg=glm,pr=pr,error=error.rate)
return(lista)
}

#funcion para el algortimo REDES NURONALES NNET
algornnet=function(traindf,testdf,formula){
nnet <- nnet(formula,
          data=traindf,
          size=8, skip=TRUE, MaxNWts=10000, trace=FALSE, maxit=100)
pr <- predict(nnet, newdata=testdf, type="class")
error.rate = sum(testdf$survived != pr ) / nrow(testdf)
lista=list(alg=nnet,pr=pr,error=error.rate)
return(lista)
}

#funcion para el algortimo NAIVE BAYES
algornaivebayes=function(traindf,testdf,formula){
nb<-naiveBayes(formula,data=traindf,threshold=0.003)
prednb=predict(nb, testdf)
error.rate = sum(testdf$survived != prednb ) / nrow(testdf)
lista=list(alg=nb,pr=pr,error=error.rate)
return(lista)
}


#funcion para el algortimo K NERAREST NEIGHBORHOOD
algorknn=function(traindf,testdf,formula){
 prknn5 <- kNN(formula,traindf,testdf,norm=TRUE,k=5)
 error.rate = sum(testdf$survived != prknn5 ) / nrow(testdf)
 lista=list(alg=prknn5,pr=pr,error=error.rate)
 return(lista)
}

#funcion para el algortimo GRADIENT BOOSTING
algorboosting=function(traindf,testdf,formula){

boost= glmboost(formula, data = traindf)
pr=as.vector(ifelse(predict(boost, type="response", newdata=testdf) > 0.5, "1", "0"))
```

```
error.rate = sum(testdf$survived != pr ) / nrow(testdf)
lista=list(alg=boost,pr=pr,error=error.rate)
return(lista)
}


#funcion para el algortimo BAGGINGRPART
algorbagging=function(traindf,testdf,formula){
bagg= bagging(formula, data = traindf)
pr=predict(bagg, newdata=testdf)
error.rate = sum(testdf$survived != pr ) / nrow(testdf)
lista=list(alg=bagg,pr=pr,error=error.rate)
return(lista)
}


evaluatorrpart <- function(subset) {
 #k-fold cross validation
 k <- 5
 #numvar=6
 assign("numvar", numvar, envir = .GlobalEnv)
 splits <- runif(nrow(basetrain))
   results = sapply(1:k, function(i) {
   test.idx <- (splits >= (i - 1) / k) & (splits < i / k)
   train.idx <- !test.idx
   test <- basetrain[test.idx, , drop=FALSE]
   train <- basetrain[train.idx, , drop=FALSE]
   formula<-as.simple.formula(subset, 'as.factor(survived)')
    if ( any(!is.na(subset[numvar]))) {
     out=algorrpart(train,test,formula)
     return(1 - out$error)
   }
  })
 if ( any(!is.na(subset[numvar]))) {
   #print(results)
   #print(mean(results))
   return(mean(results))
 } else {
   rd <- runif(1, 0.001,0.002)
   er<-er + rd
   assign("er", er + rd, envir = .GlobalEnv)
   return(er)
 }
}

evaluatorada <- function(subset) {
 #k-fold cross validation
 k <- 5
 #numvar=6
 assign("numvar", numvar, envir = .GlobalEnv)
 splits <- runif(nrow(basetrain))
  results = sapply(1:k, function(i) {
   test.idx <- (splits >= (i - 1) / k) & (splits < i / k)
   train.idx <- !test.idx
   test <- basetrain[test.idx, , drop=FALSE]
   train <- basetrain[train.idx, , drop=FALSE]
   formula<-as.simple.formula(subset, 'as.factor(survived)')
    if ( any(!is.na(subset[numvar]))) {
     out=algorada(train,test,formula)
     return(1 - out$error)
```

```
    }
  })
  if ( any(!is.na(subset[numvar]))) {
     return(mean(results))
  } else {
    rd <- runif(1, 0.001,0.002)
    er<-er + rd
    assign("er", er + rd, envir = .GlobalEnv)
    return(er)
  }
}

evaluatorrf <- function(subset) {
   #k-fold cross validation
  k <- 5
  #numvar=6
  assign("numvar", numvar, envir = .GlobalEnv)
  splits <- runif(nrow(basetrain))
   results = sapply(1:k, function(i) {
    test.idx <- (splits >= (i - 1) / k) & (splits < i / k)
    train.idx <- !test.idx
    test <- basetrain[test.idx, , drop=FALSE]
    train <- basetrain[train.idx, , drop=FALSE]
    formula<-as.simple.formula(subset, 'as.factor(survived)')
     if ( any(!is.na(subset[numvar]))) {
      out=algorrf(train,test,formula)
      return(1 - out$error)
    }
  })
 if ( any(!is.na(subset[numvar]))) {
    #print(results)
    #print(mean(results))
    return(mean(results))
  } else {
    rd <- runif(1, 0.001,0.002)
    er<-er + rd
    assign("er", er + rd, envir = .GlobalEnv)
    return(er)
  }
}

evaluatorsvm <- function(subset) {
  #k-fold cross validation
  k <- 5
  #numvar=6
  assign("numvar", numvar, envir = .GlobalEnv)
  splits <- runif(nrow(basetrain))
   results = sapply(1:k, function(i) {
    test.idx <- (splits >= (i - 1) / k) & (splits < i / k)
    train.idx <- !test.idx
    test <- basetrain[test.idx, , drop=FALSE]
    train <- basetrain[train.idx, , drop=FALSE]
    formula<-as.simple.formula(subset, 'as.factor(survived)')
     if ( any(!is.na(subset[numvar]))) {
      #print(formula)
      out=algorsvm(train,test,formula)
      return(1 - out$error)
    }
   })
  if ( any(!is.na(subset[numvar]))) {
```

```
    #print(results)
    #print(mean(results))
    return(mean(results))
  } else {
    rd <- runif(1, 0.001,0.002)
    er<-er + rd
    assign("er", er + rd, envir = .GlobalEnv)
    #print(er)
    return(er)
  }
}

evaluatorreg <- function(subset) {
 #k-fold cross validation
  k <- 5
  #numvar=6
  assign("numvar", numvar, envir = .GlobalEnv)
  splits <- runif(nrow(basetrain))
   results = sapply(1:k, function(i) {
    test.idx <- (splits >= (i - 1) / k) & (splits < i / k)
    train.idx <- !test.idx
    test <- basetrain[test.idx, , drop=FALSE]
    train <- basetrain[train.idx, , drop=FALSE]
    formula<-as.simple.formula(subset, 'as.factor(survived)')
     if ( any(!is.na(subset[numvar]))) {
      #print(formula)
      out=algorreg(train,test,formula)
      return(1 - out$error)
    }
   })
  if ( any(!is.na(subset[numvar]))) {
    #print(results)
    #print(mean(results))
    return(mean(results))
  } else {
    rd <- runif(1, 0.001,0.002)
    er<-er + rd
    assign("er", er + rd, envir = .GlobalEnv)
    #print(er)
    return(er)
  }
}

evaluatornnet <- function(subset) {
  #k-fold cross validation
  k <- 5
  #numvar=6
  assign("numvar", numvar, envir = .GlobalEnv)
  splits <- runif(nrow(basetrain))
  results = sapply(1:k, function(i) {
    test.idx <- (splits >= (i - 1) / k) & (splits < i / k)
    train.idx <- !test.idx
    test <- basetrain[test.idx, , drop=FALSE]
    train <- basetrain[train.idx, , drop=FALSE]
    formula<-as.simple.formula(subset, 'as.factor(survived)')
    if ( any(!is.na(subset[numvar]))) {
      out=algornnet(train,test,formula)
      return(1 - out$error)
    }
  })
```

```
  if ( any(!is.na(subset[numvar]))) {
    #print(results)
    #print(mean(results))
    return(mean(results))
  } else {
    rd <- runif(1, 0.001,0.002)
    er<-er + rd
    assign("er", er + rd, envir = .GlobalEnv)
    #print(er)
    return(er)
  }
}

evaluatornb <- function(subset) {
  #k-fold cross validation
  k <- 5
  assign("numvar", numvar, envir = .GlobalEnv)
  splits <- runif(nrow(basetrain))
  results = sapply(1:k, function(i) {
    test.idx <- (splits >= (i - 1) / k) & (splits < i / k)
    train.idx <- !test.idx
    test <- basetrain[test.idx, , drop=FALSE]
    train <- basetrain[train.idx, , drop=FALSE]
    formula<-as.simple.formula(subset, 'as.factor(survived)')
    if ( any(!is.na(subset[numvar]))) {
      out=algornaivebayes(train,test,formula)
      return(1 - out$error)
    }
  })
  if ( any(!is.na(subset[numvar]))) {
    #print(results)
    #print(mean(results))
    return(mean(results))
  } else {
    rd <- runif(1, 0.001,0.002)
    er<-er + rd
    assign("er", er + rd, envir = .GlobalEnv)
    #print(er)
    return(er)
  }
}

evaluatorknn <- function(subset) {

 #k-fold cross validation
  k <- 5
  assign("numvar", numvar, envir = .GlobalEnv)
  splits <- runif(nrow(basetrain))
   results = sapply(1:k, function(i) {
    test.idx <- (splits >= (i - 1) / k) & (splits < i / k)
    train.idx <- !test.idx
    test <- basetrain[test.idx, , drop=FALSE]
    train <- basetrain[train.idx, , drop=FALSE]
    formula<-as.simple.formula(subset, 'survived')
     if ( any(!is.na(subset[numvar]))) {
      out=algorknn(train,test,formula)
#       prknn5 <- kNN(formula,train,test,norm=TRUE,k=5)
#       error.rate = sum(test$survived != prknn5 ) / nrow(test)
      return(1 - out$error)
    }
```

```
  })
  if ( any(!is.na(subset[numvar]))) {

    return(mean(results))
  } else {
    rd <- runif(1, 0.001,0.002)
    er<-er + rd
    assign("er", er + rd, envir = .GlobalEnv)
    return(er)
  }
}

evaluatorboost <- function(subset) {
  #k-fold cross validation
  k <- 5
  assign("numvar", numvar, envir = .GlobalEnv)
  splits <- runif(nrow(basetrain))
  results = sapply(1:k, function(i) {
    test.idx <- (splits >= (i - 1) / k) & (splits < i / k)
    train.idx <- !test.idx
    test <- basetrain[test.idx, , drop=FALSE]
    train <- basetrain[train.idx, , drop=FALSE]
    formula<-as.simple.formula(subset, 'survived')
    if ( any(!is.na(subset[numvar]))) {
      out=algorboosting(train,test,formula)
      return(1 - out$error)
    }
  })
  if ( any(!is.na(subset[numvar]))) {
    #print(results)
    #print(mean(results))
    return(mean(results))
  } else {
    rd <- runif(1, 0.001,0.002)
    er<-er + rd
    assign("er", er + rd, envir = .GlobalEnv)
    #print(er)
    return(er)
  }
}

evaluatorbagg <- function(subset) {
  #k-fold cross validation
  k <- 5
  assign("numvar", numvar, envir = .GlobalEnv)
  splits <- runif(nrow(basetrain))
  results = sapply(1:k, function(i) {
    test.idx <- (splits >= (i - 1) / k) & (splits < i / k)
    train.idx <- !test.idx
    test <- basetrain[test.idx, , drop=FALSE]
    train <- basetrain[train.idx, , drop=FALSE]
    formula<-as.simple.formula(subset, 'as.factor(survived)')

    if ( any(!is.na(subset[numvar]))) {
      out=algorbagging(train,test,formula)
      return(1 - out$error)
    }
  })
  if ( any(!is.na(subset[numvar]))) {
    #print(results)
```

```
    #print(mean(results))
    return(mean(results))
  } else {
    rd <- runif(1, 0.001,0.002)
    er<-er + rd
    assign("er", er + rd, envir = .GlobalEnv)
    return(er)
  }
}

package.skeleton(list = c("inicio","evaluatorrpart","evaluatorada","evaluatorrf",
                          "evaluatorsvm","evaluatorreg","evaluatornnet",
                          "evaluatornb","evaluatorknn","evaluatorboost",
                          "evaluatorbagg",
                          "algorrpart","algorada","algorrf",
                         "algorsvm","algorreg","algornnet","algornaivebayes","algorknn",
                        "algorboosting","algorbagging"), name = "paqdatamining")
```

Bibliografía

[Abi97] Abiteboul S. (1997) Querying semi-structured data. En Proceedings of the Inter national Conference on Databases Theory (ICDT) páginas 1-18.

[BA01] Sunita Beniwal, Jitender Arora.(2012) Classification and Feature Selection Techniques in Data Mining. Department of Information Technology, Maharishi Markandeshwar University, Mullanayvol 1.

[BL01] Breiman L.(2001) Random Forest. Statistics Department University of California, Berkely Machine Learning 45, paginas 5-32.

[BFR+99] Burt M., Fowlkes C., Roden J., Stechert A. y Mukhtar S. (April 1999) Diamondeye:A distributed architecture for image data mining. En SPIE DMKD, Orlando páginas 71-80.

[CDH+02] Chen Y., Dong G., Han J., Wah B. y Wang J. (2002) Multidimensional regressionanalysis of time-series data streams. En: 2002 International Conference on VeryLarge Data Bases páginas 323-334.

[CHYX99] Chu-yu C, Hang D, Yi S, Xiangmin Z(December 2009) The R project for Comparisons of SeveralMultivariate Means.

[CMS99] Cooley R., Mobasher B. y Srivastava J. (1999) Data preparation for mining world wide web browsing patterns. Knowledge and information Systems1(1): 5-32.

[Dan02] Dandretta G. (Junio, 2002) Web mining: Implementando técnicas de data mining en un servidor web. Technical Report, Universidad de Belgrano, Buenos Aires.

[DHS00] Duda R., Hart P. y Stork D. (2000) Pattern classification. John Wiley & Sons,Nueva York, Estados Unidos.

[DKS95] Dougherty J., Kohavi R. y Sahami M. (1995) Supervised and unsupervised discretization of continuos features. Proc. of the 12th International Conference Machine Learning páginas 194-202.

[DL01] Dzeroski S. y Lavrac N. (2001) Relational data mining: Inductive logic programming for knowledge discovery in databases. Springer-Verlag.

[DOF03] DeCaceres M., Oliva F. y Font X. (2003) Ginkgo, un programa de análisis multivariante orientado a la clasificación basada en distancias. 27 Congreso Nacional de Estadstica e Investigaci on Operativa, Lleida páginas 1-9.

[ET13] Encyclopedia Titanica (2013), Titanic facts and history: Titanic passenger and crew biography. [Fecha de consulta: 11 Junio 2013]. Disponible en:

http://www.encyclopedia-titanica.org/

[Fa00] Fukuda M. y Ńanri I. (2000) Mining from literary texts: Pattern discovery andsimilarity computation. Progress in Discovery Science 2000 páginas 518-531.

[FMRR05] Fabrice Muhlenbach y Ricco Rakotomalala (2005) Discretization for Continuous Attributes, Encyclopedia of Data Warehousing and Mining, **397-402.**

[FPSM92] Frawley W., Piatetsky-Shapiro G. y Matheus C. (1992) Knowledge discovery in databases: An overview. SAAAI/MIT Presspáginas 57-70.

[FPSSU96] Fayyad U., Piatetsky-Shapiro G., Smyth P. y Uthurusamy P. (1996) Advances in knowledge discovery and data mining. AAAI/MIT Press.

[FW13] Fundación Wikimedia, Inc., (2013), RMS Titanic. [Fecha de consulta: 11 Junio 2013]. Disponible en:http://es.wikipedia.org/wiki/RMS_Titanic.

[HCC93] Han J., Cai Y. y Cercone N. (1993) Data-driven discovery of quantitative rulesin relational databases. En IEEE Transactions on Knowledge and Data Eng. 5:29-40.

[HH03] Hilderman R. y Hamilton H. (2003) Measuring the interestingness of discoveredknowledge: A principal approach. Intelligent Data Analysis 7(4): 347-382.

[HLT99] Hussain F., Liu H. y Tan C. (1999) Discretization : an enabling technique. Technical Report TRC6/99, The National University of Singaporepáginas 1022-1027.

[HPS04] Huang X., Peng F. y Schuurmans D. (2004) Dynamic web log session identification with statistical language models. En: Journal of American society for information science and technology 55(13): 1.290-1.303.

[JHMK06] Jiawei Han, Micheline Kamber (2006) Data Mining: Concepts and Techniques, Second Edition. The Morgan Kaufmann Series in DataManagement Systems, Morgan Kaufmann Publishers.

[Ka13] Kaggle Inc, Titanic: Machine Learning from Disaster [Fecha de consulta: 11 Junio 2013]. Disponible en:http://www.kaggle.com/c/titanic-gettingStarted**.**

[Kan] Kandel A.Fuzzy techniques in pattern recognition. New York: John Wiley & Sonspágina 356.

[Kar01] Kargupta H. (2001) Career: Ubiquitous distributed knowledge discovery from heterogeneous data. NSF Information and Data Management (IDM) Workshop pági-nas 18-23.

[Kar03] Kargupta H. (2003) Vehicle data stream mining (vedas proyect). NSF Information and Data Management (IDM) Workshop páginas 37-46.

[KLR+98] Kennedy R., Lee Y., Roy B., Reed C. y Lippman R. (1998) Solving data mining problems through pattern recognition. Prentice Hall, Upper Saddler River, NewJersey.

[KW96] Kwork C. y Weld D. (1996) Planning to gather information. En Proc. 14th National Conference on AI 9: 32-39.

[LHML99] Liu B., Hsu W., Mun L. y Lee H. (1999) Finding interesting patterns using userexpectations. IEEE Transactions on Knowledge and Data Engeniering 11(6):817-832.

LMSZ10] Liu H.Motoda Hiroshi. Setiono R. y Zhao Zheng**(2010)** Feature Selection: An Ever Evolving Frontier in Data Mining.Journal of Machine Learning Research - Proceedings Track.

[McG05] McGarry K. (2005) A survey of interestingness measures for knowledge discovery. The Knowledge Engineering Review20(1): 39-61.

[McQ67] McQueen J. (1967) Some methods for classification and analisys of multivariate observations. En Proceedings of the Fifth Berkeley Symposium on Mathematcal Statistic and Probability páginas 281- 287.

[Mit97] Mitchell T. (New York, 1997) Machine learning. Book McGraw-Hill.

[Mol02] Molina L. (2002) Data mining: torturando a los datos hasta que confiesen http://www.uoc.edu/web/esp/art/uoc/molina1102/molina1102.html .

[MORL10] Maimon O,Rokach L (2010) Data Mining and Knowledge Discovery Handbook. Springer New York Dordrecht Heidelberg London, paginas 1-5.

[MPM02] Mitra S., Pal S. y Mitra P. (January 2002) Data mining in soft computing framework: a survey. IEEE Transactions on Neural Networks 13(1): 3-14.

[MSTM11] Michael R. Smith and Tony Martinez (2011) Improving Classification Accuracy by Identifying and RemovingInstances that Should Be Misclassified paginas 1-8.

[PM05] Pechenizkiy M. (2005) The impact of feature extraction on the performance of a classifier: kNN, Naïve Bayes and C4.5AI'05 Proceedings of the 18th Canadian Society conference on Advances in Artificial Intelligence: Pages 268-279.

[PS04] Piramuthu S, (2004) Evaluating feature selection methods for learning in data mining applications. 156(2): 483-494**.**

[Pyl99] Pyle D. (1999) Data preparation for data mining. Book Morgan Kaufmann, SanFrancisco California.

[Sah01] Sahar S. (2001) Interestingness preprocessing. En Proceedings of the 2001, IEEEInternational Conference on Data Mining, San Jose, CA páginas 489-496.

[Sah02] Sahar S. (Japan, 2002) On incorporating subjective interestingness into the miningprocess. En Proceedings of the 2002, IEEE International Conference on DataMining, Maebashi City páginas 681-684.

[SKVK06] Srinivasa K G *, Venugopal K R 1 and L M Patnaik (2006) Feature Extraction using Fuzzy C - Means Clustering for Data Mining Systems Data Mining IJCSNS International Journal of Computer Science and Network SecurityVolumen(6).

[Ser03] Serrano J. (2003) Knowledge fusion in relational databases: Aggregation andsummarization measures. Proceedings of the 1st ICEIS Doctoral Consortium (DCEIS-2003) in conjunction with ICEIS 2003, Angers, France páginas 1-4.

[SS13] Sayad S. (2013), An introduction to Data Mining [Fecha de consulta: 09 de setiembre 2013]. Disponible en:http://saedsayad.com/data_mining_map.htm.

[ST01] Shneider T (2001), Analysis of Incomplete Climate Data: Estimation of Mean Values and Covariance Matrices and Imputation of Missing Values, Atmospheric and Oceanic Sciences Program, Princeton University, Princeton, New Jersey, paginas 853-871, Disponible en:,http://www.clidyn.ethz.ch/imputation/index.html.

[Swa94] Swanson D. (1994) Assessing a gap in the biomedical literature: Magnesium deficiency and neurologic disease. En : Neuroscinse Research Communications (15):1-9.

[Tan99] Tan A. (1999) Text mining: Promises and challenges. En Pacic Asia Conference on Knowledge Discovery and Data Mining PAKDD'99 workshop on knowledge Discovery from Advanced Databases páginas 63-70.

[TRJ01] Trivellore E. Raghunathan, James M. Lepkowski, John Van Hoewykand Peter Solenberger (2001) A Mutilvariate Technique for Multiply Imputing Missing Values Using a Sequence of Regression Models Survey. Statistic Canada. 27(1) 85-95.

[UGP13] Usama Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth. (2013) From Data Mining to Knowledge Discovery in Databases Association for the Advancement of Artificial Intelligence (www.aaai.org) 17(3).

[VDS00] Vila M. A., Delgado M. y Sanchez D. (2000) Acquisition of fuzzy associationrules from decimal. data. En: Barro S, Marn R, editors. Fuzzy logic in medicine,Physical-Verlag.

[WI13] Wikibooks (2013), Data Mining Algorithms In R/Dimensionality Reduction/Feature Selection [Fecha de consulta: 09 de setiembre 2013]. Disponible en: http://en.wikibooks.org/wiki/Data_Mining_Algorithms_In_R/Dimensionality_Reduction/Feature_Selection.

[WG11] Williams G (2011) Data Mining With Rattle and R, Springer Science-Business Media, pp 245-368.

[WM06] Wasito, I. y Mirkin, B. (2006) Nearest neighbor approach in the data lest-squares data imputation algorithms. Information Sciences169(1-2): 1-25.

[XKPS02] Xu F., Kurz D., Piskorski J. y Shmeier S. (2002) Term extraction and mining ofterm relations from unrestricted texts in the financial domain. En Proceedings ofBIS 2002.

[ZMFA07] Z. Marzuki, F. Ahmad. (2007)Data Mining Discretization Methods and Performances. Faculty of Information Technology, Universiti Utara Malaysia, 06010 UUM Sintok, Kedah, Malaysia.