

**Proyecto 2 – This dataset represents a set of possible advertisements on Internet pages.**

**Taller integrado de ciencias de datos**

Yorch Sepúlveda

*Agosto de 2018*

1. **Construya la estructura de un proyecto de Ciencias de Datos y defina las tareas a realizar durante cada fase de un PCD.**

* **Identificar el problema:** En la actualidad, al momento navegar en las páginas web, se puede apreciar anuncios en estas. Ahora bien, existen funciones codifican la geometría de la imagen (si está disponible), así como las frases que aparecen en la URL, la URL de la imagen y el texto alternativo, el texto de anclaje y las palabras que se encuentran cerca del texto de anclaje. La tarea es predecir si una imagen es publicitaria ("publicitaria") o no ("nonad").
* **Fuentes de datos del instrumento:**

(a) Creador y donante: Nicholas Kushmerick <nick@ucd.ie>

(c) Generado: abril-julio de 1998

La información se obtuvo de N. Kushmerick (1999). Esta página hace referencia de "Aprender a eliminar publicidad en Internet", con el 3º Agentes autónomos de Int Conf. Disponible en [www.cs.ucd.ie/staff/nick/research/download/kushmerick-aa99.ps.gz](http://www.cs.ucd.ie/staff/nick/research/download/kushmerick-aa99.ps.gz). Este agente tiene una precisión mayor al 97% usando reglas c4.5 para predecir su una imagen es un anuncio.

El algoritmo C4.5 es una versión mejorada del ID3. Permite trabajar con valores continuos para los atributos, separando los posibles resultados en dos ramas en función de un umbral. Los arboles son menos frondosos debido a que cada hoja no cubre una clase en particular sino una distribución de clases.

Este algoritmo genera un árbol de decisión a partir de los datos mediante particiones realizadas recursivamente, según la estrategia de profundidad-primero (Depth-first). Antes de cada partición de datos, el algoritmo considera todas las pruebas posibles que pueden dividir el conjunto de datos y selecciona la prueba que resulta en la mayor ganancia de información o en la mayor proporción de ganancia de información. Para cada atributo discreto, se considera una prueba con n resultados, siendo n el numero de valores posibles que puede tomar el atributo. Para cada atributo continuo se realiza una prueba binaria sobre cada uno de los valores que toma el atributo en los datos.

* **Recopila datos:** Ver (Kushmerick, 1999) para detalles de los atributos; en formato "nombres":
* **altura: continua. | posiblemente falta**
* **ancho: continuo. | posiblemente falta**
* **aratio: continuo. | posiblemente falta**
* **local: 0,1.**

**| 457 características de términos url, cada una de las formas "url \* term1 + term2 ...";**

**| por ejemplo:**

**url \* imágenes + botones: 0,1.**

**...**

**| 495 características de términos origurl, en la misma forma; por ejemplo:**

**laberinto origurl \*: 0,1.**

**...**

**| 472 características de términos ancuros, en la misma forma; por ejemplo:**

**ancurl \* búsqueda + directa: 0,1.**

**...**

**| 111 características de términos alt, en la misma forma; por ejemplo:**

**alt \* tu: 0,1.**

**...**

**| 19 características de los términos de pie de foto**

**leyenda \* y: 0,1.**

**...**

* **Valores de atributo faltantes:** ¿cuántos por cada atributo?

Al 28% de las instancias les faltan algunos de los atributos continuos.

* **Preparar datos:** Esta etapa consiste en la preparación y limpieza de los datos extraídos desde las distintas fuentes de datos en una forma manejable, necesaria para las fases posteriores. En esta etapa se utilizan diversas estrategias para manejar datos faltantes o en blanco (N/A), datos inconsistentes o que están fuera de rango, obteniendo una estructura de datos adecuada para su posterior transformación.
  1. **Integrar**
  2. **Transformar:** para poder utilizar los datos provenientes del set de datos en los distintos métodos, se tuvo que utilizar la función **as.numeric** y para utilizar esta función se tuvo que hacer otra transformación antes de convertir a ***numeric*** los datos, la cual es **as.character**.
  3. **Limpiar**: primero, se analizan los datos en el estado que se encuentran en el documento, sin manipular los datos (en estado bruto). Esto se realiza debido a que los datos NA son representados por “?” y el problema es que puede haber distintas formas de estar escrito en el doc. lo cual no hace fácil reemplazar todos estos por NA.

Una vez realizado esto, se procede a eliminar las filas que contengan valores NA y así poder tener una tabla consistente.

Debido a la magnitud de datos que existen es imposible ver caso a caso las filas que contienen NA, pero como la cantidad de datos con filas consistentes es mayor a la de filas con NA es suficiente para proceder al análisis de los datos sin estas.

* 1. **Filtrar:** con la finalidad de realizar pruebas y pensando en la simplicidad de los datos y los sobresalientes, se opta por seleccionar los atributos, **height**, **width**, **aratio**, **local** y **ad/nonad(clase)**. Además, según la documentación, son los atributos más relevantes en comparación al resto. (Kushmerick, 1999)

**Nota:** La selección de los atributos mencionados anteriormente se debe a que existen muchos atributos que son irrelevantes, además de que la memoria de la computadora no soportaba tanta información.

1. **Caracterice la(s) herramienta(s) de preprocesado de datos que posea el lenguaje escogido por Usted, identificando ventajas y desventajas.**

**Resumen de los métodos que evaluar:**

* **KNN:** caret, class, kknn
* **SVM:** e1071, kernlab, caret
* **TREE:** c50, rpart
* **K-NN** es un algoritmo de **aprendizaje supervisado**, es decir, que a partir de un juego de datos inicial su objetivo será el de clasificar correctamente todas las instancias nuevas. El juego de datos típico de este tipo de algoritmos está formado por varios atributos descriptivos y un solo atributo objetivo (también llamado clase).

La idea es realmente sencilla: el algoritmo clasifica cada dato nuevo en el grupo que corresponda, según tenga **k** vecinos más cerca de un grupo o de otro. Es decir, calcula la distancia del elemento nuevo a cada uno de los existentes, y ordena dichas distancias de menor a mayor para ir seleccionando el grupo al que pertenecer. Este grupo será, por tanto, el de mayor frecuencia con menores distancias.

* **Support Vector Machine**  (SVM) es un modelo supervisado ​​de aprendizaje con algoritmos asociados que analizan los datos y reconocen patrones, que se utiliza para la clasificación y el análisis de regresión en [la Inteligencia de Negocios](https://www.webyempresas.com/que-es-la-inteligencia-de-negocios-business-intelligence/). El **SVM** básico toma un conjunto de datos de entrada y predice, para cada entrada dada, a cuál de las dos clases de salida pertenece, por lo que es un clasificador no-probabilístico lineal binario (solo escoge entre 2 opciones). Dado un conjunto de ejemplos de entrenamiento, cada uno marcado como perteneciente a una de dos categorías, un algoritmo de entrenamiento construye un modelo que asigna nuevos ejemplos en una categoría u otra. Este tipo de modelos es muy utilizado para el análisis de modelos en los cuales se tiene que clasificar un conjunto de datos a solo dos categorías, fraude o no-fraude, sí o no al crédito, etc. Un modelo de SVM es una representación de los ejemplos (base de datos con la cual se realizó la estimación) como puntos en el espacio, de modo que asignan los ejemplos de las categorías separadas que generalmente quedan divididas por un espacio definido, espacio que tiene que ser tan amplio como sea posible. Los nuevos datos de entrada serán clasificados en el mismo espacio y para predecir a que categoría pertenece.
* Los **árboles de regresión y clasificación** se basan en estratificar el espacio de los predictores en una serie de regiones simples que incluyen observaciones con valores similares para la variable respuesta en cuestión. Cuando la variable respuesta es numérica hablamos de árboles de regresión, mientras que las variables categóricas se analizan usando árboles de clasificación. En cualquier caso, su funcionamiento es relativamente similar: para hacer una predicción para una determinada observación, utilizaremos la media (o la moda) de las observaciones que se encuentran en la misma región del espacio multidimensional de predictores. Las reglas que se utilizan para dividir el espacio de predictores pueden ser representadas en forma de árbol; de ahí el nombre de estos métodos.

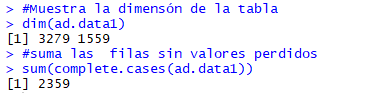
1. **Caracterice su correspondiente data set teniendo en cuenta: a. Número de instancias y de atributos, tipo de datos, valores perdidos, valores con ruido, outliers, otro que considere importante.** 
   1. **Numero de instancias y atributos: 3279 (2821 nonads, 458 anuncios)**



**Número de filas:** 3279

**Número de columnas:** 1559

* 1. **Tipos de datos:** Los tipos de datos fueron mencionados en el ítem 1 en la etapa de recopilación de datos.
  2. **Valores perdidos**



Este comando muestra todas las celdas que se encuentran vacías, pero es muy difícil de dimensionar la cantidad de datos perdidos ya que son muchos datos.



Este comando facilita la vida preguntando si existe alguna celda sin datos. Debió ser lo primero que se debería consultar al momento de analizar los datos.



Cuenta el total de celdas perdidas. Aunque aun así hay muchas perdidas de datos esas pueden ser filas completas o columnas completas. Aunque no da un indicio de que existe muchos datos perdidos y que ya debemos enfocarnos en un grupo de datos en donde se concentre la mayor cantidad de datos.

* 1. **Otros**
* **Porcentaje de bien clasificados según el método utilizado y la librería de donde se encontraba lo anteriormente mencionado**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **CLASS** | **KNN** | **CARET** |
| **KNN** | 92.65537 | **93.64407** | 90.53672 |

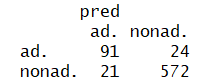
|  |  |  |  |
| --- | --- | --- | --- |
|  | **CARET** | **E1071** | **KERNLAB** |
| **SVM** | 91.52542 | **95.76271** | 91.10169 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **C50** | **RPART** | **TREE** |
| **TREE** | 94.0678 | **94.35028** | 91.24294 |

* **Matriz de confusión según el método utilizado y la librería de donde se encontraba lo mencionado anteriormente** 
  + **Knn-class**



* + **Knn – kknn**



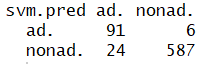
* + **Knn -carent**



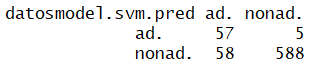
* + **Svm – carent**



* + **Svm – e1071**



* + **Svm – kernlab**



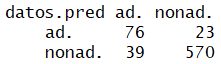
* + **Tree – c50**



* + **Tree – rpart**



* + **Tree – tree**



**Importante:** El script de R que contiene toda la configuración para la visualización de entrenamiento de los datos, matriz de confusión y porcentaje de bien clasificados, tiene como nombre ***scriptProyecto2.1.R*** y la URL del repositorio es: <https://github.com/YorchXD/Protecto2-TICD>

1. **Referencias**

Alexandros Karatzoglou [aut, c. A. (10 de Agosto de 2018). *CRAN.* Obtenido de https://cran.r-project.org/web/packages/kernlab/kernlab.pdf

Brian Ripley [aut, c. c. (30 de Agosto de 2015). *CRAN.* Obtenido de https://cran.r-project.org/web/packages/class/class.pdf

Brian Ripley [aut, c. c. (4 de Abril de 2018). *CRAN.* Obtenido de https://cran.r-project.org/web/packages/MASS/MASS.pdf

Calvo, D. (3 de Octubre de 2016). *Árboles de Clasificación en R*. Obtenido de http://www.diegocalvo.es/arboles-de-clasificacion-en-r/

David Meyer [aut, c. E.-C.-c.-C.-c. (28 de Julio de 2018). *CRAN.* Obtenido de https://cran.r-project.org/web/packages/e1071/e1071.pdf

Delaunay, F. (17 de 04 de 2015). *GitHub*. Obtenido de https://github.com/fdelaunay/tutorial-dplyr-es/blob/master/R/tutorial-dplyr.md

Klaus Schliep [aut, c. K. (26 de Marzo de 2016). *Weighted k-Nearest Neighbors.* Obtenido de https://cran.r-project.org/web/packages/kknn/kknn.pdf

Kushmerick, N. (1999). *Learning to remove Internet advertisements.* Dublin, Ireland: 3rd Int. Conf. on Autonomous Agents.

Ma, M. (27 de agosto de 2014). *Classification and Regression Trees (CART) with rpart and rpart.plot*. Obtenido de https://rstudio-pubs-static.s3.amazonaws.com/27179\_e64f0de316fc4f169d6ca300f18ee2aa.html

Marc Claesen, J. S. (2014). *Optunity*. Obtenido de https://optunity.readthedocs.io/en/latest/examples/r/svm.html

Max Kuhn. Contributions from Jed Wing, S. W. (27 de Mayo de 2018). *CRAN.* Obtenido de https://cran.r-project.org/web/packages/caret/caret.pdf

Meyer, D. (2018). *Support Vector Machines.* Austria.

Nitika. (3 de Septiembre de 2017). *kNN Classification demo*. Obtenido de http://rpubs.com/Nitika/kNN\_Iris

Ojeda, F. C. (3 de Marzo de 2016). *Github*. Obtenido de https://github.com/fcharte/CursoCienciaDatosR/blob/master/ejerciciosSVM.R

OrlandoM. (Agosto de 2016). *Aplicaciones Estadísticas en R*. Obtenido de http://orlandomoscote.blogspot.com/2016/08/arbol-de-decision-en-r.html

Pacheco, E. R. (s.f.). *RPubs*. Obtenido de https://rpubs.com/eropa1981/22869

Parra, F. (15 de Junio de 2017). *Estadística y Machine Learning con R*. Obtenido de https://rstudio-pubs-static.s3.amazonaws.com/293405\_4029f1f23f834b7195189d5504a436b2.html

Santiago, E. O. (31 de 10 de 2015). *blogger*. Obtenido de http://rstadistica.blogspot.cl/2015/10/VentajasDesventajasR.html