

Desarrollo y análisis comparativo de una librería en R para clasificación supervisada

Jorge Antolinos

1 de noviembre de 2019

Índice general

1. Introducción	2
1.1. Resumen	2
1.2. Abstract	2
1.3. Palabras Clave	2
1.4. Objetivos del trabajo	2
1.5. Enfoque y método seguido	3
1.6. Descripción de los capítulos	3
2. Planificación del proyecto	5
3. Estado del arte	8
3.1. R	8
3.2. Historia de R	9
3.3. Factores de utilización R	10
3.4. Competencia de R	11
3.5. Software libre Vs Software propietario	13
3.6. Introducción a Data Science	16
3.7. R en el mundo Data Science	20
3.8. Casos de éxito R	21
3.9. Eventos sobre R	22
4. Clasificación Supervisada	24
4.1. Definición de Clasificación Supervisada	24
4.2. Comparación entre Clasificación Supervisada y No Supervisada	24
4.3. Técnicas de Clasificación Supervisada	25
5. Análisis De Paquetes	26
5.1. Random Forest	26
5.2. Decision Tree	40
5.3. Support Vector Machines	62
6. Conclusiones	82

Capítulo 1

Introducción

1.1. Resumen

En este proyecto vamos a analizar las principales librerías de clasificación supervisada desarrolladas por la comunidad de desarrolladores R.

Para ello realizaremos una búsqueda y análisis de librerías en el portal R-CRAN. El primer paso consistirá en distinguir las librerías de clasificación supervisada del resto de librerías y para cada una se seleccionarán los paquetes más importantes que nos ayudarán a resolver esa tipología de problema.

El método de análisis consiste en la descripción detallada del manual de uso, la descarga y el desarrollo del mismo en el entorno de desarrollo R-studio para comprobar los resultados solicitados del problema utilizado.

1.2. Abstract

In this project we are going to analyse the main supervised classification libraries carried out by the R. developers community.

To that end we will conduct a library search and test in R-CRAN portal. The first step will be to distinguish between the supervised libraries and the rest of them and for every classification made we will choose the most important suites that will help us solve that problem.

The test method lies in the detailed description of user guide, its download and progress in R-Studios developing environment in order to check the requested results from the used problem.

1.3. Palabras Clave

Data Mining, Data Science, Clasificación supervisada, R.

1.4. Objetivos del trabajo

Muchos de nosotros escuchamos en la televisión, radio, internet o a nuestros amigos hablar sobre el concepto del análisis de datos. Nos cuentan como las empresas consiguen pasar de tener datos a información y una vez que tienen esa

información la convierten en conocimiento para mejorar sus procesos y modelos de negocios. KPMG durante el 2015 ya indicaba que en los últimos 2 años se han generado más datos que en toda la historia de la humanidad [1]

Para diferenciarnos de la competencia en un mundo cada vez más competitivo tenemos tres opciones:

- Ser los más baratos.
- Ofrecer valor al cliente.
- Tener un producto que la competencia no haya sido aún capaz de replicar.

R es una herramienta de software libre que va a ser la herramienta de análisis de datos que vamos utilizar durante este trabajo de Fin de Grado. La vamos a enfocar en resolver problemas de Clasificación supervisada. Con ella vamos a comprobar que utilizando la tecnología:

Podemos reducir costes Con R tenemos algoritmos para realizar por ejemplo: Predicciones de stock, con estas predicciones podemos reducir los gastos de almacenaje y mejorar en la rotación.

Podemos ofrecer un Valor diferencial al cliente Con R podemos realizar clasificaciones según variables que tengamos de nuestros clientes, por lo que podemos ofrecer un valor diferencial mediante una comunicación única con cada cliente.

Para ello explicaremos los algoritmos utilizados para cada caso y mostraremos ejemplos de implementación en R. Todas las implementaciones que realizaremos durante este proyecto de fin de grado serán sobre Bases de Datos accesibles por todo el mundo.

1.5. Enfoque y método seguido

El método que vamos a utilizar para llegar al objetivo marcado es el siguiente: Análisis de los paquetes R-Cran y similares. Recopilando todos aquellos que son de Clasificación supervisada, identificando las funcionalidades que contiene cada uno y para poder así conseguir un documento en el que los desarrolladores puedan de una manera clara, rápida y concisa encontrar la función que en ese momento necesiten. El documento va a ser realizado desde un punto de vista científico. Este se basa en un conjunto de métodos y técnicas que organizan la información adquirida mediante la experiencia o la introspección.

1.6. Descripción de los capítulos

En este primer capítulo se realizará una introducción en el que se explicará el objetivo que perseguimos con este Trabajo de Fin de Grado. Una vez explicado las funciones que va a tener este trabajo de Fin de grado, pasaremos a la planificación del proyecto en el que se detallarán los tiempos que ha llevado el desarrollo de cada uno de los capítulos que cuenta este Proyecto. Una vez contrastado los tiempos del proyecto nos vamos a introducir en el mundo R, que va a ser el lenguaje que vamos a utilizar para desarrollar este proyecto. En este capítulo contaremos todo sobre R (Para qué sirve, ventajas, desventajas, y mucho más...). También introduciremos otros lenguajes que son competencia de

R como puede ser SAS o IBM SPSS. Durante las hojas de este capítulo también pondremos casos prácticos de empresas que han utilizado R, indicando lo que han conseguido. De nada sirve una herramienta y muchos datos si no consigues explotar los mismos y conseguir ventajas competitivas para tu empresa. Por último durante este tercer capítulo hablaremos sobre los eventos que tiene R durante el 2016. Este es un gran indicador ya que si cuenta con muchos eventos significa que a las personas les importa este lenguaje y lo ven como un lenguaje con mucho uso y potencialidad. Durante el cuarto capítulo nos introduciremos en la clasificación supervisada que es el tipo de clasificación que vamos a llevar a cabo durante nuestro proyecto. Con ella explicaremos para que sirve y que sentido de negocio puede tener en el mundo empresarial. También contaremos como tiene unos rasgos generales diferentes a otros tipos de clasificación como puede ser la clasificación no supervisada. En el quinto capítulo ya nos introduciremos en la parte más importante de nuestro proyecto de fin de grado, en el explicaremos los distintos algoritmos que tiene R. Los clasificaremos para que un desarrollador pueda encontrar de una manera clara y sencilla los distintos algoritmos que tienes en Clasificación supervisada. Este es un punto bastante importante ya que si haces búsquedas en la web, hay pocos foros o páginas que puedas encontrar esta información en español. Todavía la comunidad de España está varios pasos por detrás de comunidades como puede ser la americana. Por último y para terminar el Proyecto de Fin de Grado mostraremos las conclusiones que hemos sacado durante los meses que hemos desarrollado el proyecto y mostraremos una bibliografía con todos los libros y enlaces de interés.

Capítulo 2

Planificación del proyecto

El proyecto va a contar con cuatro bloques claramente diferenciados.

1. Introducción al trabajo de Fin de Grado.

	Modo de	Nombre de tarea	Duración	Comienzo	Fin	Predecesora
1		Introducción	11 días	vie 18/03/16	vie 01/04/16	
2		Objetivos del trabajo	5 días	vie 18/03/16	jue 24/03/16	
3		Enfoque y método seguido	3 días	vie 25/03/16	mar 29/03/16	2
4		Sumario - Descripción de los capítulos	3 días	mié 30/03/16	vie 01/04/16	3

En el se contará cuál es el objetivo que se tiene con este trabajo, qué es lo que vamos a aprender, la forma en la que vamos a explicar los diversos contenidos y un breve resumen de los capítulos que va a encontrar la persona que lea este proyecto.

Este es uno de los bloques que contará con menos extensión debido a su claro carácter introductorio al mismo. Debido a este motivo como podemos comprobar en la planificación del proyecto, este bloque tendrá una duración de 11 respecto al total de días de finalización de proyecto.

2. R y Clasificación supervisada

	Modo de	Nombre de tarea	Duración	Comienzo	Fin	Predecesora
5		Capítulos	49 días	lun 04/04/16	jue 09/06/16	4
6		1 Estado del arte	14 días	lun 04/04/16	jue 21/04/16	4
7		1.1 ¿Qué es R?	1 día	lun 04/04/16	lun 04/04/16	4
8		1.2 ¿Por qué utilizar R?	1 día	mar 05/04/16	mar 05/04/16	7
9		1.3 Introducción al Data Science	3 días	mié 06/04/16	vie 08/04/16	8
10		1.4 Comparación - ¿Con quién compete R?	5 días	lun 11/04/16	vie 15/04/16	9
11		1.5 Software libre VS. Software propietario	2 días	lun 18/04/16	mar 19/04/16	10
12		1.6 R en el mundo del Data Science	2 días	mié 20/04/16	jue 21/04/16	11
13		2 Clasificación Supervisada	10 días	vie 22/04/16	jue 05/05/16	12
14		2.1 ¿Qué es la clasificación supervisada?	2 días	vie 22/04/16	lun 25/04/16	12
15		2.2 Comparación entre clasificación supervisada y no supervisada	1 día	mar 26/04/16	mar 26/04/16	14
16		2.3 Técnicas de clasificación supervisada	7 días	mié 27/04/16	jue 05/05/16	15

Como podemos comprobar en el título del Proyecto de Fin de Grado, durante las páginas de este proyecto vamos a hablar de R y de Clasificación supervisada. Antes de entrar profundamente, hemos creído conveniente hacer una pequeña introducción a R, la historia de R, los competidores de R (que son muchos), tanto de software libre como de software propietario y de los eventos que tiene R durante el año 2016 en todo el mundo. Este último punto es muy importante ya que puedes conocer si un software es bueno o malo con indicadores como el número de eventos que hay. En el caso de que existan muchos eventos consiste en un gran indicador ya que nos está indicando que los desarrolladores quieren conocer R, saber novedades, etc.

Una vez que dejamos de hablar de R, hacemos una pequeña introducción a Data Science y explicamos en el punto que se encuentra R.

Por último hablamos de la clasificación supervisada, ya que es el tipo de clasificación que hemos elegido para realizar el Trabajo de Fin de Grado. Con ella explicamos las diferencias con la no supervisada.

Este es un punto que cuenta con muchos temas muy importantes para poder adentrarte un poco más en el análisis de datos, por lo que al contrario del punto anterior, este es un apartado con un peso muy grande dentro del proyecto y en el que se han invertido muchos días para poder desarrollarlo, como podemos comprobar en la planificación desarrollada con Microsoft project.

3. Identificación de paquetes de clasificación supervisada

17		3 Identificación de paquetes de clasificación en R-CRAN Packages y similares	25 días	vie 06/05/16	jue 09/06/16	16
----	---	--	---------	--------------	--------------	----


Investigación, búsqueda y recopilación exhaustiva de los distintos paquetes de clasificación supervisada que se encuentran en la red para poder así facilitar el trabajo de los desarrolladores que necesitan utilizar métodos de clasificación que se encuentran en estos paquetes.

Este es el núcleo del Trabajo de Fin de Grado. Es un apartado que lleva mucho tiempo desarrollarlo debido a que primero tienes que buscar tipología de acciones como puede ser Random forest, una vez encontrado tienes que buscar algoritmos que puedan resolver los problemas de esta tipología y cuando los encuentras los debes explicar de una manera clara y sencilla y descargarlo e instalarlo en el ordenador para poder probar que realmente cumple su función como es necesario.

En algunos de estos algoritmos hemos también mostrado su salida de forma gráfica para poder comprobar de una manera más sencilla el resultado.

En cuanto a los días invertidos durante este apartado, han sido la mayoría ya que es la parte más importante y más costosa de desarrollar.

4. Conclusiones y bibliografía

18		4 Conclusiones	3 días	vie 10/06/16	mar 14/06/16	17
19		5 Bibliografía	1 día	mié 15/06/16	mié 15/06/16	18

En este punto finalizaremos el proyecto de Fin de Grado.

Durante las conclusiones sintetizaremos los logros conseguidos con la investigación de este proyecto, es un punto muy importante porque produce una visión global del contenido del proyecto en un número reducido de líneas.

En el apartado de bibliografía indicaremos todas las fuentes consultadas para la realización del mismo

Este es el punto que menos tiempo ha llevado ya que durante el desarrollo de las anteriores partes se ha ido completando poco a poco estos puntos. Por lo que los días invertidos han servido de perfeccionamiento.

Capítulo 3

Estado del arte

3.1. R

R es un lenguaje de programación interpretado, de distribución libre, bajo Licencia GNU, y se mantiene en un ambiente para el cómputo estadístico y gráfico. Este software funciona en los grandes sistemas operativos Linux, Windows, Mac OSX. El término ambiente pretende caracterizarlo como un sistema totalmente planificado y coherente, en lugar de una acumulación gradual de herramientas muy específicas y poco flexibles, como suele ser con otro software de análisis de datos. El hecho que R sea un lenguaje y un sistema, es porque forma parte de la filosofía de creación.¹ John Chambers la siguiente perspectiva (Chambers and Hastie [1991]), cito: Buscamos que los usuarios puedan iniciar en un entorno interactivo, en el que no se vean, conscientemente, a ellos mismos como programadores. Conforme sus necesidades sean más claras y su complejidad se incrementa, deberían gradualmente poder profundizar en la programación, es cuando los aspectos del lenguaje y el sistema se vuelven más importantes. Por esta razón, en lugar de pensar de R como un sistema estadístico, es preferible verlo como un ambiente en el que se aplican técnicas estadísticas. [2]

El paquete R está desarrollado y mantenido por algunos de los más prestigiosos estadísticos actuales, además de caracteriza por tener una descarga e instalación sencilla y rápida.

Muchas multinacionales lo utilizan, por ejemplo: Google lo utiliza para explorar datos y construir modelos y Facebook lo usa para mostrar los enlaces entre sus usuarios.

Una de sus grandes fortalezas que tiene R es que puede ser ampliado mediante paquetes que extienden sus funcionalidades. Actualmente hay más de 2860 paquetes publicados con licencias libres y disponibles en un repositorio general (CRAN)[3].

Los paquetes publicados en CRAN están organizados en lo que denominan Task Views, abarcando aplicaciones tales como Empirical Finance, Computational Econometrics, Social Sciences, Analysis of Ecological and Environmental Data, Official Statistics & Survey Methodology, Chemometrics and Computational Physics, Natural Language Processing, Time Series Analysis, y también simulaciones de sistemas fotovoltaicos).

¹Desde la codificación del lenguaje S, lenguaje progenitor de R.

R es un lenguaje de programación y, por tanto, la interacción con él es a base de código. Sin embargo, es posible diseñar interfaces gráficas para facilitar la interacción a usuarios que así lo prefieran.

Existen herramientas IDE adaptadas a R, pero dos destacan por encima de la multitud: Emacs Speaks Statistics (un módulo para Emacs) y el recientemente aparecido RStudio.

Otras características de R son:

- Es posible utilizar código C y C++ a través de la librería RCpp, código Python a través de la librería RPython y código java a través de rJava.
- Existen un buen número de paquetes en CRAN destinados a facilitar el parallel computing con R. En particular, es posible utilizar GPUs con R.[5]

3.2. Historia de R

El lenguaje R fue desarrollado inicialmente por Robert Gentleman y Ross Ihaka del Departamento de Estadística de la Universidad de Auckland de Nueva Zelanda en 1993. La intención inicial era utilizarlo como base académica con la que utilizarlo en el curso de Introducción a la Estadística de la Universidad de Nueva Zelanda. Por ello decidieron adoptar la sintaxis del lenguaje S desarrollado por Bell Laboratories. Por ello, la sintaxis es similar al lenguaje S, pero la semántica, que aparentemente es parecida a la de S, en realidad es sensiblemente diferente, sobre todo en los detalles un poco más profundos de la programación. A modo de broma Ross y Robert, comienzan a llamar R al lenguaje que implementaron, por las iniciales de sus nombres, y desde entonces así se le conoce en la muy extendida comunidad amante de dicho lenguaje. Debido a que R es una evolución de S. S es un lenguaje que fue desarrollado por John Chambers y colaboradores en Laboratorios Bell (AT&T), actualmente Lucent Technologies, en 1976. Este lenguaje, originalmente fue codificado e implementado como unas bibliotecas de FORTRAN. Por razones de eficiencia, en 1988 S fue reescrito en lenguaje C, dando origen al sistema estadístico S, Versión 3. Con la finalidad de impulsar comercialmente a S, Bell Laboratories dio a StatSci (ahora Insightful Corporation) en 1993, una licencia exclusiva para desarrollar y vender el lenguaje S. En 1998, S ganó el premio de la Association for Computing Machinery a los Sistemas de Software, y se liberó la versión 4, la cual es prácticamente la versión actual. El éxito de S fue tal que, en 2004 Insightful decide comprar el lenguaje a Lucent (Bell Laboratories) por la suma de 2 millones de dólares, convirtiéndose hasta la fecha en el dueño. Desde entonces, Insightful vende su implementación del lenguaje S bajo el nombre de S-PLUS, donde le añade un ambiente gráfico amigable. En el año 2008, TIBCO compra Insightful por 25 millones de dólares y se continúa vendiendo S-PLUS, sin modificaciones. R, que define su sintaxis a partir de esa versión de S, no ha sufrido en lo fundamental ningún cambio dramático desde 1998. Regresemos ahora al lenguaje que nos ocupa: R. Luego de la creación de R (en 1992), se da un primer anuncio al público del software R en 1993. En el año de 1995 Martin Mächler, de la Escuela Politécnica Federal de Zúrich, convence a Ross y Robert a usar la Licencia GNU para hacer de R un software libre. Como consecuencia, a partir de 1997, R forma parte del proyecto GNU. Con el propósito de crear algún tipo de soporte para el lenguaje, en 1996 se crea una lista pública de correos; sin embargo debido al gran éxito

de R, los creadores fueron rebasados por la continua llegada de correos. Por esta razón, se vieron en la necesidad de crear, en 1997, dos listas de correos, a saber: R-help y R-devel, que son las que actualmente funcionan para responder las diversas dudas que los usuarios proponen en muy diversos asuntos relativos al lenguaje. Además se consolida el grupo núcleo de R, donde se involucran personas asociadas con S-PLUS, con la finalidad de administrar el código fuente de R. Fue hasta febrero de 29 del 2000, que se considera al software completo y lo suficientemente estable, para liberar la versión 1.0. [4]

Su desarrollo actual es responsabilidad del R Development Core Team. Para saber más al respecto y en el entorno del programa, puede teclearse `contributors()`; en la lista desplegada aparecen los nombres de los autores iniciales y los actuales pertenecientes al R Development Core Team (Equipo Central de Desarrolladores R).

3.3. Factores de utilización R

Algunas de las ventajas de utilizar R son las siguientes [6]:

1. Lenguaje robusto: R es un lenguaje con una curva de aprendizaje compleja, pero muy robusto y efectivo para el manejo de datos estadísticos. Es un lenguaje orientado a objetos, muy similar a las sintaxis C y C++. Para los desarrolladores especializados en estos lenguajes, puede ser sencillo. Además, R es un lenguaje de programación que está en constante evolución y del que se dispone de una amplia documentación.
2. Facilidad en la preparación de los datos: Cuando un programador maneja volúmenes de datos muy elevados, gran parte del tiempo se dedica a preparar la información para la visualización de la que se pueden extraer conclusiones. Con R esa preparación es relativamente sencilla, en gran medida porque automatiza muchos procesos mediante la programación de scripts.
3. R funciona con cualquier tipo de archivo: R es muy flexible, puede trabajar con datos procedentes de todo tipo de archivos: un .txt, un .csv, un JSON o un EXCEL.
4. Gestión de un gran volumen de datos: R es un lenguaje que permite la implementación de paquetes adicionales que le dan una capacidad de gestión de datos enorme. En proyectos de gran volumen, la escalabilidad es un elemento clave.
5. Es de código abierto y gratuito:
Si eres programador y quieres empezar con R, esto es un detalle importante. No hay limitaciones. El código está en cualquier repositorio de plataformas de desarrollo colaborativo como GitHub o foros de dudas para desarrolladores como Stackoverflow. Hay librerías y paquetes adicionales para impulsar proyectos, cuyo código se puede modificar al gusto para implementar nuevas funcionalidades. Y gratis.
6. Visualizaciones: Si hay algo que define a R es su capacidad para visualizar información compleja de una forma sencilla. Cualquier desarrollador

dispone de librerías especializadas en visualización de datos para hacer gráficos.

3.4. Competencia de R

R cuenta con una competencia muy dura de otros fabricantes. Para comparar funcionalidades vamos a realizar un análisis de los principales puntos entre Sas, R y Python. [8]

En este análisis ofrecido por Kunal Jain (profesional del mundo del Data Science y reconocido a nivel mundial) influirán las siguientes variables:

- Disponibilidad / Coste Sas es un software comercial. Queda fuera del alcance de la mayoría de los profesionales ya que es un software caro. Sin embargo cuenta con la mayor cuota de mercado en las organizaciones privadas. Por lo que a menos que seas una organización privada que invierte fuertemente en SAS, es difícil acceder a él.

R & Python, por el otro lado son gratuitas y pueden ser descargadas por todo el mundo, por lo que las puntuaciones son las siguientes:

Sas - 2 R - 5 Python - 5

- Facilidad de aprendizaje

SAS es fácil de aprender y ofrece la opción *FOCUS* (PROC SQL) para las personas que ya conocen SQL. Incluso en caso contrario, tiene una buena interfaz GUI estable en su repositorio. En términos de recursos, hay tutoriales disponibles en los sitios web de varias universidades y SAS tiene una amplia documentación. Hay certificaciones de los institutos de formación de SAS, la única desventaja es que tienen un precio muy elevado.

R tiene la curva de aprendizaje más pronunciada entre los 3 lenguajes de la comparación. Se requiere que usted pueda aprender y entender la codificación. R es un lenguaje de programación de bajo nivel y por lo tanto los procedimientos simples pueden tener códigos más largos.

Python es conocido por su simplicidad en el mundo de la programación. Esto sigue siendo cierto para el análisis de datos también. No hay interfaces GUI generalizadas pero se tiene previsto que esto sea más común en el medio plazo.

Las puntuaciones son las siguientes:

SAS - 4.5 R - 2.5 Python - 3.5

- Capacidades de manejo de datos Esto solía ser una ventaja para SAS hasta hace poco tiempo. R calcula en la memoria (RAM) y, por tanto, los cálculos fueron limitados por la cantidad de RAM en las máquinas de 32 bits. Esto ya no es un problema. Los tres lenguajes tienen buenas capacidades y opciones para cálculos paralelos de tratamiento de datos. Por lo que no hay grandes diferencias. Las puntuaciones son las siguientes:

SAS 3 R 4.5 Python 4

- Capacidades gráficas Sas tiene capacidades gráficas pero no es su core. Las funcionalidades son complejas y requiere que conozcas su paquete SAS Gráfico.

R tiene las capacidades gráficas más avanzadas entre los tres. Existen numerosos paquetes que le proporcionan capacidades gráficas avanzadas.

Las capacidades de Python se encuentran en punto intermedio, con opciones para utilizar las bibliotecas nativas (matplotlib) o bibliotecas derivados (que permite llamar a funciones R).

Las puntuaciones son las siguientes:

SAS 3 R 4.5 Python 4

- Avances en la herramienta Los tres ecosistemas tienen todas las funciones básicas y más usadas disponibles.

Debido a su naturaleza de código abierto R & Python cuentan con las últimas funcionalidades de una forma más rápida (R aún más que Python). Sas por el lado contrario, tarda más en actualizar. Hay que remarcar que SAS cuando lanza actualizaciones son en ambiente controlado y probadas, mientras que las de R & Python por su contribución académica y abierta puede que contengan errores.

Las puntuaciones son las siguientes:

SAS 4 R 4.5 Python 4

- Escenario de trabajo A nivel mundial, SAS sigue siendo el líder del mercado en puestos de trabajo corporativos disponibles. La mayoría de las grandes organizaciones siguen trabajando en el SAS. R & Python, por otra parte, son mejores opciones para la creación de empresas y empresas en busca de la eficiencia de costes. Además, los puestos de trabajos disponibles según diversos estudios de R & Python aumentarán en los próximos años.

Las puntuaciones son las siguientes:

SAS 4.5 R 3.5 Python 2.5

- Servicio al cliente y comunidad R tiene la comunidad en línea más grande, pero no hay soporte de servicio al cliente. Así que si tienen problemas, lo tendrán que solucionar por su cuenta.

SAS, por otra parte tiene dedicado un servicio al cliente junto con una comunidad. Por lo tanto, si usted tiene problemas en la instalación o cualquier otro desafío técnico, puede contactar con ellos.

Las puntuaciones son las siguientes:

SAS 4 R 3.5 Python 3

Conclusiones

Claramente no hay un ganador por goleada en estos momentos. Y es muy prematuro de hacer apuestas de lo que ocurrirá en el futuro, ya que estamos en un sector muy cambiante.

Las recomendaciones son que si usted se encuentra en la industria del análisis, sería bueno que el lenguaje que se usase fuese SAS, ya que es fácil de aprender y tiene una alta cuota de mercado. En el caso de ser una start-up / freelance R & Python son dos grandes opciones.

Parameter	SAS	R	Python
Availability / Cost	2	5	5
Ease of learning	4.5	2.5	3.5
Data handling capabilities	4	4	4
Graphical capabilities	3	4.5	4
Advancements in tool	4	4.5	4
Job scenario	4.5	3.5	2.5
Customer service support and Community	4	3.5	3

3.5. Software libre Vs Software propietario

Para realizar las ventajas y desventajas que tiene el software libre Vs el propietario, vamos a realizar una comparativa del software libre R contra los grandes del software propietario como son SAS y SPSS.

Para ello iremos punto por punto indicando las características de cada uno de ellos. Historia

- Historia

Los tres nacieron en el siglo XX.

En concreto SAS empezó a distribuirse masivamente en 1971. Es un lenguaje creado por Jim Goodnight y Jim Barr. Nació debido a que necesitaban un software que permitiera analizar una gran cantidad de datos para la agricultura. El instituto SAS fue creado en 1976 y en 2013 ya contaba con 13.733 empleados. También en 2013, SAS invirtió un 25 % de su beneficio en R&D.

En cuanto a R, este lenguaje fue lanzado en 1995. Es un lenguaje desarrollado por Ross Ihaka y Robert Gentleman. Este lenguaje es una implementación del lenguaje de programación S creado en los laboratorios Bell. El desarrollo y evolución de R es controlado por el grupo Rcore y la fundación R. El código fuente del que cuenta R está escrito principalmente en C, Fortran y R.

Por último tenemos el software de pago SPSS. Este lenguaje fue lanzado en el año 1968. Los creadores fueron Norman H. Nie, Dale H. Bent y Hadlai Tex Hull. Este lenguaje fue adquirido por IBM en 2009 por 1.2 billones de dolares.

- Propósito y usabilidad Sas desde de los años 1970 acumula mucho prestigio. Se caracteriza por tener un código de alta calidad, que se puede utilizar para muchos propósitos. Cuenta con una posición de liderazgo. Sas cuenta con unas capacidades enormes de manejo de datos. Realiza sus actualizaciones en un entorno desarrollado, que producen que todas se controlen y se reduzcan el número de errores. El lado malo es que es una solución de muy alto precio.

Por el contrario R, es una solución muy utilizada en la investigación y Universidades desde hace mucho tiempo. Actualmente, también cuenta con una fuerte aceptación en el mundo empresarial. Características que definen a R son las grandes capacidades gráficas con las que cuenta, gracias a paquetes como ggplot2, googleVis y rCharts. Debido a que es de código abierto, R cuenta con una gran comunidad. Las últimas técnicas son desarrolladas muy rápido.

Por último contamos con la solución de IBM. Es la llamada herramienta para los "noestadísticos" que cuenta con una interfaz y menús muy fácil de usar. Spss cuenta con aplicaciones en muchos campos, pero se caracteriza por ser muy fuerte en las Ciencias Sociales.

- Empresas que utilizan estas herramientas Multinacionales en sus sectores utilizan estas herramientas.

En el caso de SAS: HP, Scotiabank, AIG o Staples confían en la herramienta.

Con R: Grandes compañías como Facebook, The New York Times, FDA o Google la utilizan para realizar análisis internos.

En el caso de SPSS: ABN-AMRO, Canon, BT o CreditSuisse.

- Facilidad de aprendizaje

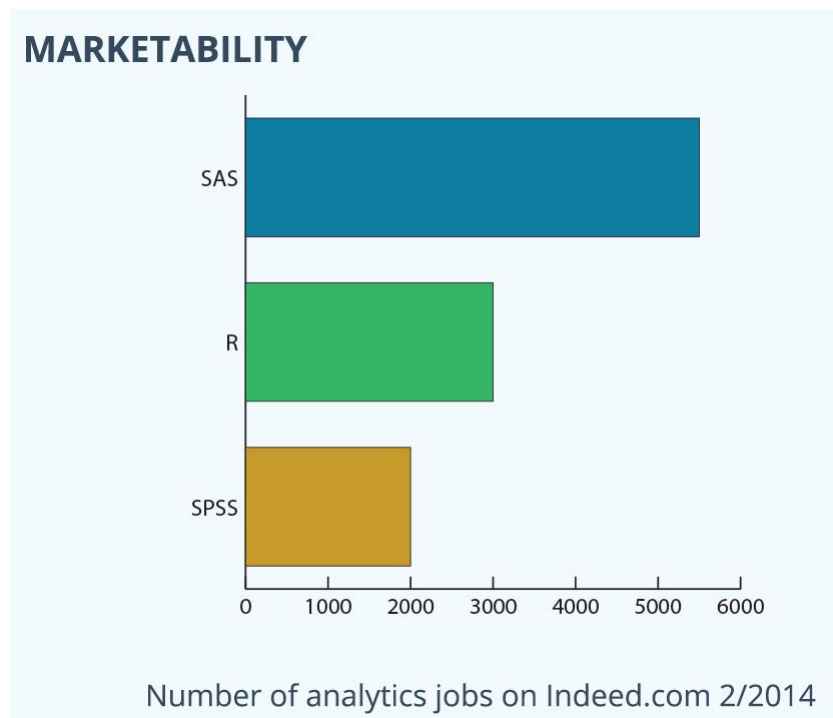
SAS: Aunque no es tan fácil como aprender a utilizar un Sistema Operativo de Microsoft, tener un entendimiento básico de la herramienta no debería llevar mucho tiempo. Sin embargo, ser un gran profesional de SAS te costará bastante tiempo ya que tendrás muchos módulos de especialización. Hay muchos tutoriales oficiales disponibles, también cuentan con certificaciones oficiales.

R: Cuenta con una reputación de ser un lenguaje muy lento de aprender. Los usuarios de R tienen que aprender como analizar los datos interactivamente. Para muchos análisis de datos, las personas se tienen que someter a cambios de mentalidad y ver los datos con preparación para poder llegar a entenderlos. La gran comunidad que tiene R a sus espaldas propicia que la curva de aprendizaje sea menor, ya que cuenta con tutoriales introductorios de muy alta calidad.

SPSS: El lenguaje de IBM es de lejos el lenguaje más fácil de aprender de los tres. Una de las grandes ventajas que tiene SPSS son sus similitudes con excel. Lo que provoca que mucho de los conceptos sean familiares.

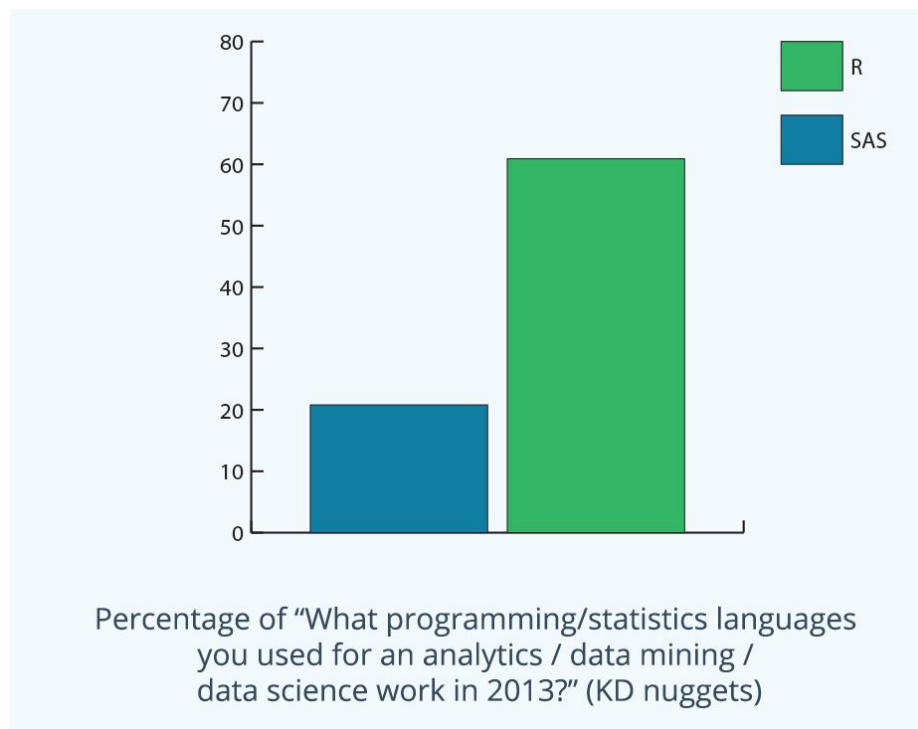
- Funcionamiento en el mercado

Como podemos comprobar en el gráfico el mayor número de usuarios que se requieren en el portal Indeed son de SAS. Seguido por R y SPSS de cerca.



- Popularidad Según Stackoverflow: El número de post contestados en Stack Overflow de R es más de 7 veces el número de post de SAS.

También contamos con el estudio de KD Nuggets, en el que preguntaban a los usuarios cuál era su lenguaje de programación estadístico preferido para la minería de datos o ciencia de datos en el trabajo. Este estudio es de 2013. [9]



Los resultados de este estudio desprendían que los usuarios de KdNugets preferían R.

Conclusiones de Software Libre vs Software propietario

- Si el presupuesto con el que cuentas es limitado, debes optar a herramientas de software libre como R. Son muy potentes y podrás realizar la mayoría de las cosas que harías con herramientas de pago
- En general, las herramientas de Software Libre cuentan con una comunidad muy amplia de usuarios que la utilizan. Podrás encontrar en muchos blogs y foros famosos información sobre los algoritmos más usados.
- En cuanto a la resolución de incidencias y soporte, la mayoría de las herramientas de software libre no cuentan con él aunque si puedes optar a él pagando cuotas anuales como es el caso de RStudio que cuenta con una licencia comercial.[10]
- Si deseas que todo funcione a la perfección y no haya fallos de diseño, debes optar por opciones de software de pago. Son altamente analizadas y no suelen salir al mercado con fallos.

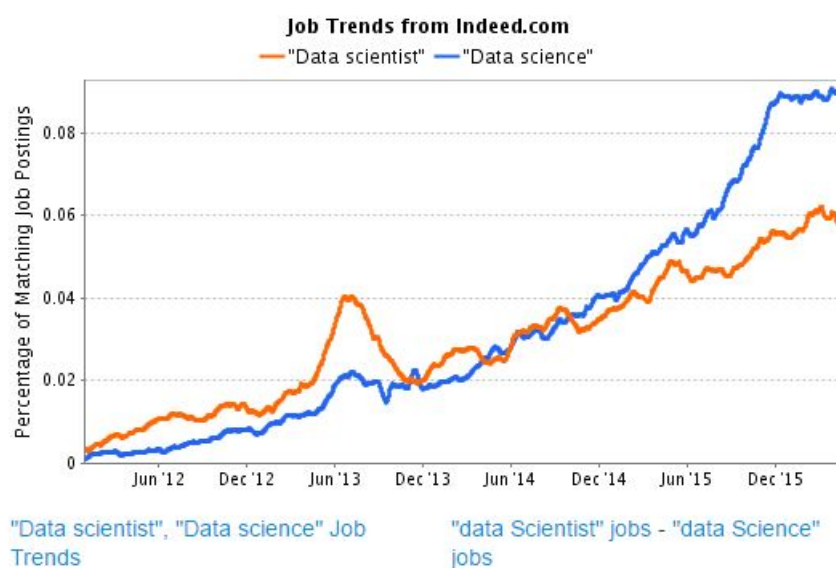
3.6. Introducción a Data Science

La persona que es mejor en estadística que cualquier ingeniero de software y mejor en ingeniería de software que cualquier estadístico. Así definió la profesión de Data Scientist Josh Wills, Director de Data Science de Cloudera en una

conferencia en 2012, titulada La vida de un Científico de Datos. [13] Hoy en día multinacionales que todos conocemos como pueden ser Google, IBM, Facebook, HP, Oracle, Amazon o LinkedIn se mueven día a día en el mundo de los Big Data para obtener ventajas competitivas. La clave de todo ese proceso es la Ciencia de Datos: la mejora de algoritmos que permitan ahorrar costes, perfeccionar sistemas de recomendación o búsqueda, modernizar los procesos industriales, controlar los niveles de riesgo Y cómo no, transformar el modelo de negocio de cualquier empresa.

El Científico de Datos debe tener conocimientos en ciencia aplicada, con una larga experiencia en su industria y con formación en materia científica (aprendizaje supervisado, no supervisado). Esto le permite llegar a soluciones creativas y con criterio. El Data Scientist va mucho ms allá del Power Point al que estamos acostumbrados en el mundo de la innovación: su responsabilidad empieza diseando un prototipo con las tecnologías que mejor se adapten al problema en cuestión (Hadoop, MongoDB, Spark, Python, R) y acaba en la supervisión de su puesta en marcha en producción, afirma Sergio Álvarez Telena, Responsable de Global Strategies & Data Science en BBVA, Global Markets.

Debido a esta mezcla de experiencia y conocimientos tan especial, encontrar profesionales que respondan a los desafíos del mercado es complicado. Tanto es así, que el sector se refiere a ellos como unicornios. Sin embargo, la demanda de información y formación profesional en este campo no deja de aumentar. Una simple búsqueda en Indeed, la plataforma de ofertas de empleo, muestra cmo el interés por esta disciplina ha progresado mucho desde 2011.



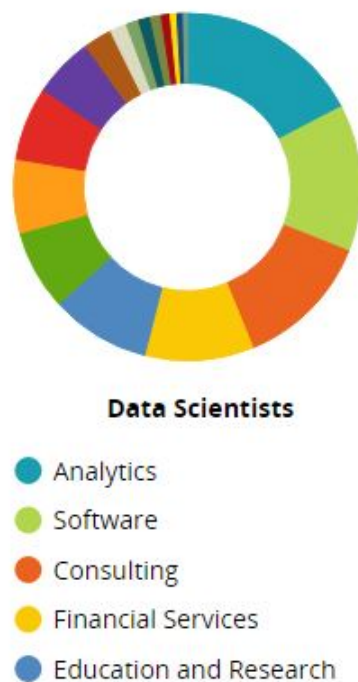
Crecimientos en el sector del Big Data

Lo cierto es que el sector de Big Data crece por encima de un 50 % anual. En 2011 lo hizo un 59 % y en 2012, un 58 %. Las previsiones son que el sector alcance los 38.000 millones de dólares en ventas en 2015, los 45.000 millones en 2016 y rompa todos los registros al situarse en los 50.000 millones en 2017. [14]

EEUU, Reino Unido e Israel, los tres puntos más calientes en innovación internacional, son y serán los países que apuesten con más fuerza por estos

perfiles y financien equipos y proyectos de este tipo. En España es más difícil, exceptuando aquellas empresas que se dedican a dar estos servicios. El miedo a lo desconocido y la comodidad son dos de las principales causas por las que este país no evoluciona al ritmo de otras economías desarrolladas, pero todo llegará aquí lo que sobra es talento, afirma Álvarez.

En Estados Unidos, por poner un ejemplo, los sectores que utilizan la Ciencia de Datos son la analítica, el desarrollo de software, la consultoría, la energía, los servicios financieros, la educación y la investigación, la publicidad y los medios, las infraestructuras o el reclutamiento de nuevo talento. Aquí tienes un gráfico con el reparto porcentual de la demanda en esta disciplina al otro lado del Atlántico. [11]



Pese a esa demanda, hay escasez de titulaciones educativas. 'Las mejores alternativas son comenzar con una formación base, como Estadística o Ingeniería Informática y complementarla con estudios de postgrado para obtener un espectro de habilidades más amplio', asegura Guerrero. Para Álvarez, 'el mejor sitio del mundo para aprender todo esto es el UK PhD Centre for Financial Computing & Analytics, un centro muy elitista en términos de talento al que solo se accede por beca del gobierno británico'.

Las herramientas de trabajo del Científico de Datos

La mejor virtud de un profesional que quiera dedicarse a la Ciencia de Datos es su creatividad y capacidad para alcanzar soluciones óptimas a los problemas. Desarrollar librerías y herramientas que acaben en mejoras de negocio. Y para ello es indispensable dominar distintos lenguajes de programación. R, C++, Python, Matlab, Pascal 'Mi primera opción suele ser R, dispone de un enorme

conjunto de paquetes generales y especializados que cubren la mayor parte de las necesidades para el análisis de información', dice Guerrero.

El lenguaje C++ suele utilizarse para mitigar las deficiencias de R, una gestión eficaz de los objetos en memoria o de la velocidad en las operaciones tipo bucle. En el caso de desarrollo para proyectos de deep learning, los desarrolladores en Python pueden encontrar un campo profesional muy interesante. Además, existen otras opciones como Pascal, una solución rápida para estructuras de datos complejas.

Casos de éxito

El entorno de los Big Data y la Ciencia de Datos se ha convertido en un círculo cerrado, en el que se evitan filtraciones que den ventajas a los competidores. En muchas ocasiones, las empresas alcanzan soluciones exitosas que no publicitan por miedo a ser imitados. Empresas como Google, Microsoft, Facebook, Twitter o entidades financieras de medio planeta invierten miles de millones para liderar sus respectivos mercados.

Hoy en día, más de la mitad de las operaciones en bolsa se ejecutan a partir de algoritmos que no necesitan de la intervención de una persona. 'Comencé dentro de un departamento de renta variable en Europa y ahora mismo tengo un equipo de Data Scientists con el que controlamos todo la innovación del trading y el comercio electrónico a nivel global para todos los activos del BBVA', afirma Álvarez. 'Muchos describen esta apuesta por la innovación como un notable éxito en gestión del cambio. De hecho, mi unidad ha sido la única iniciativa presentada por Global Markets a unos premios internos de excelencia. Mi objetivo ahora es difundir el conocimiento de Data Science en otras reas para que entre todos podamos exprimir al máximo esta nueva disciplina', asegura.

Redes sociales como Facebook analizan la interacción de los usuarios dentro de su plataforma para determinar la imagen de marca o producto de las empresas. Ese análisis es vital para el modelo de negocio de muchas empresas, por lo que su explotación con fines comerciales es una fuente de financiación enorme.

Microsoft, por ejemplo, utilizó la Ciencia de Datos para mejorar Kinect, el sistema que permite jugar a los videojuegos con el cuerpo. La empresa de Redmond recurrió a la comunidad científica para mejorar su sistema de reconocimiento de gestos corporales. El español Alfonso Nieto, que en la actualidad trabaja en la Universidad de Boston, ganó en dos ocasiones el reto lanzado por la empresa americana. En esa línea, los Data Scientists han desarrollado aplicaciones de reconocimiento del lenguaje de signos o la realización de ejercicios de rehabilitación en el campo de la medicina.

Hacia dónde van los Big Data

El futuro de la Ciencia de Datos se debate en lo que podríamos llamar la Trinidad de los Big Data: volumen, diferenciación y velocidad. Hasta hace muy poco, cualquier solución en el análisis de los datos tenía como una de sus prioridades la gestión de un gran volumen de datos. En la actualidad, ofrecen muchos ms problemas la heterogeneidad de esos datos (texto, imagen, vídeo, conversaciones en foros, redes sociales y aplicaciones móviles) y la velocidad con la que esos datos se generan.

Por tanto, 'debemos estar preparados para trabajar con bases de datos noSQL (not only SQL) y almacenamientos distribuidos', afirma Guerrero. Además, las empresas cada vez más reclaman soluciones en tiempo real. 'Hoy se están desarrollando técnicas denominadas online incremental learning para resolver este tipo de situaciones en mercados como la publicidad online, el trading au-

tomático o la detección de intrusiones en redes en materia de seguridad’, dice este profesional.

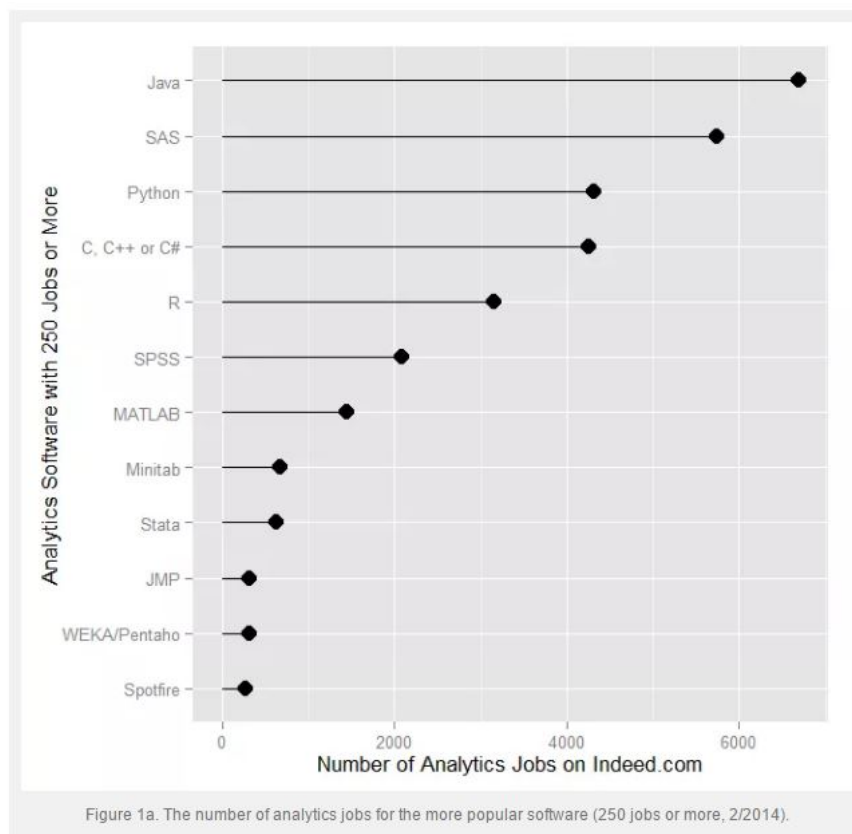
Otros de los avances en los últimos años es el aumento de las soluciones paralelizables. Las empresas apuestan cada vez más por OpenMP para optimizar los procesadores multihilo; Hadoop y MapReduce para los clústeres multinodo; y, de una forma más reciente, la irrupción de Spark, que permite ‘una gestión de los datos en memoria cien veces más rápida que Hadoop para la computación de algoritmos’, asegura Guerrero.

El futuro profesional es de los Científicos de Datos y las empresas que no se suban a la gran ola de los Big Data perderán la gran batalla del conocimiento.

Fuente: BBVA [7]

3.7. R en el mundo Data Science

R dentro del mundo Data Science es uno de los lenguajes más usados, famosos y utilizados. Para medir el éxito del lenguaje dentro del sector, qué mejor si las empresas están buscando profesionales de R. Para ello podemos el siguiente ranking de empleos más buscados en el portal de empleo Indeed. R está entre los primeros en la clasificación. [12]

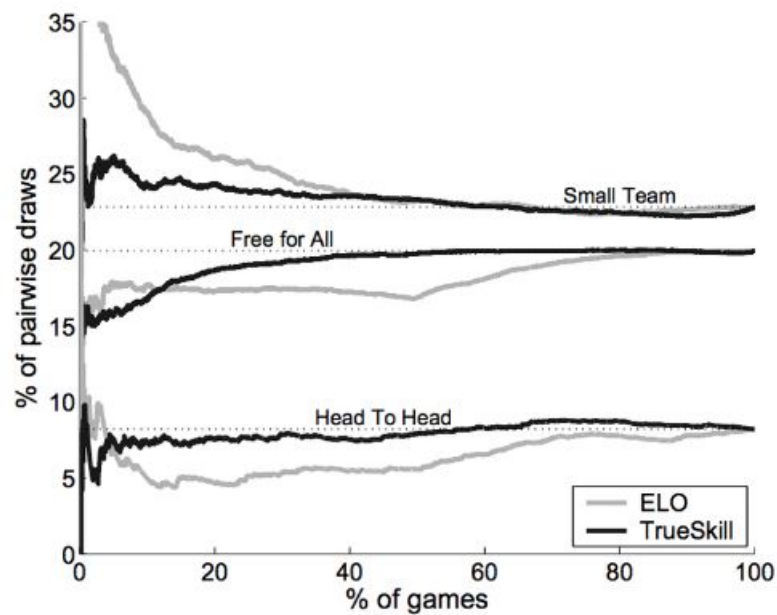


Existen muchas empresas que realizan proyectos de Data Science con R, aquí mostramos algunos de estos proyectos:

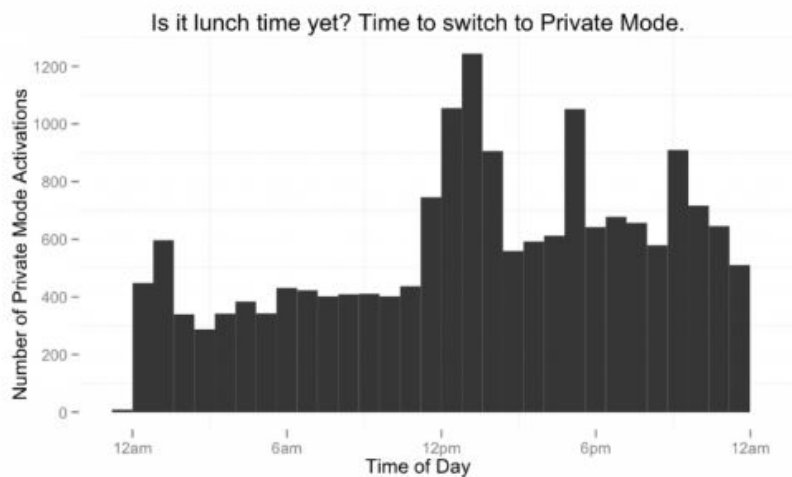
- Bank of America: Mientras que los clásicos analistas financieros estudian minuciosamente a través de Excel. Bank of America utiliza R para la renovación financiera. Su presidente Niall O'Connor indicó que ellos empezaron a utilizarlo como herramienta de visualizaciones. [15]
- Chicago: Gracias a un sistema automatizado que utiliza el análisis textual en tiempo real implementado con R, la ciudad tiene una poderosa herramienta para la identificación de las necesidades de sus ciudadanos, desde los tweets.
- Facebook: Los científicos de Facebook utilizan R con paquetes como ggplot2, dplyr, plyr para explorar los datos a través de visualizaciones personalizadas.
- Twitter: Twitter creó el paquete de R BreakoutDetection para supervisar la experiencia del usuario en su red.

3.8. Casos de éxito R

- Microsoft usa R para el emparejamiento de rival en XBOX: Si alguna vez has jugado a juegos multijugador en línea como Halo en la consola Xbox, sabrás que la consola de Microsoft empareja a los compañeros de equipo y oponentes. Esta es una de las funciones más importantes ya que es fundamental para tener una buena experiencia de juego. Microsoft debe emparejarte con un oponente que no esté muy por encima de tu nivel de habilidad ya que no es sólo desmoralizador para ti; también es un desperdicio del tiempo para el jugador por encima de tu nivel (por lo general no se avanza en el juego a menos que ganes a un jugador que es de nivel más alto que el tuyo). El problema es aún más complejo para los juegos de equipo: no hay nada peor que estar en un equipo con un grupo de novatos y jugar contra un grupo de veteranos en el otro lado. Para mejorar estos puntos Microsoft pone en marcha su sistema "TrueSkill" que es una generalización del sistema bayesiano del sistema ELO usado en partidas de ajedrez. Este sistema de Microsoft utiliza C # y F # para el análisis de datos y R para las visualizaciones. Un sistema de concordancia ideal sería generar el mayor número de empates (atados), ya que los jugadores de la misma habilidad son más propensos a atar. De acuerdo con el gráfico mostrado a continuación, el sistema TrueSkill funciona mejor para "Free for All" "Head-to-Head" (uno-a-uno) de las partidas. [17] [19]



- Es la hora de comer. Navegación privada de Mozilla: En el blog de Mozilla métricas, Mozillan Hamilton Ulmer utiliza R y ggplot2 para saber qué hace los usuarios con navegación privada (o al menos, los usuarios de Firefox que se ofrecieron para compartir sus datos de uso). Mediante gráficos conoció que su uso principal resulta ser la hora del almuerzo, con este tipo de navegación conseguían navegar lejos de la mirada de los jefes. [20]



3.9. Eventos sobre R

El lenguaje de programación estadístico R cuenta con muchos eventos alrededor del mundo. Alguno de los eventos más importantes son los siguientes:

- useR international R User conference: Este evento tiene lugar en la Universidad de Stanford. El programa cuenta con dos partes, una de ellas son presentaciones de invitados por parte del staff de R y una segunda parte son presentaciones por parte de la comunidad de R. Las presentaciones cubren un amplio rango de temas, que van desde problemas de computación ubícuos a problemas de estadística general.
- R in Insurance: Es un evento ofrecido en la escuela de negocio Cass. Este evento es una conferencia de un día que se centra en las aplicaciones de los seguros y la ciencia actuarial que utilizan R. Los temas cubiertos incluyen la Reserva y la fijación de precios, el modelado de la pérdida, el uso de R en un entorno de producción, y mucho más. Todos los temas son discutidos en el contexto de la utilización de R como herramienta principal para la gestión de riesgos de seguros, análisis y modelado.
- JSM (the Joint Statistical Meetings): Es la mayor reunión de estadísticos celebrada en América del Norte. Cuenta con la asistencia de más de 6.000 personas, las actividades incluyen presentaciones orales, sesiones de paneles, presentaciones de carteles, cursos de educación continua, una sala de exposiciones, actividades sociales y oportunidades de networking.
- Teradata PARTNERS: Es el lugar donde comerciales, empleados de marketing y de tecnología líderes en su sector se reúnen para aprender y compartir las últimas estrategias y tácticas para optimizar el uso de los datos de su empresa. Se desarrolla en Atlanta.
- EARL: Es una conferencia para los usuarios y desarrolladores del lenguaje de programación R de código abierto. El objetivo principal de la conferencia es el uso comercial de R a través de ejemplos prácticos de una variedad de sectores de la industria, con el objetivo de compartir conocimientos y aplicaciones. Este evento se lleva a cabo en Boston.

Capítulo 4

Clasificación Supervisada

4.1. Definición de Clasificación Supervisada

Con este tipo de clasificación no vamos a ciegas, es decir para la tarea de clasificar un objeto dentro de una categoría o clase contamos con modelos ya clasificados (objetos agrupados que tienen características comunes). [16] Ej: Sabemos que son pétalos y sépalos de flores y queremos clasificarlo según una variable como puede ser la anchura de las mismas. Esta es una de las diferencias que tiene la clasificación supervisada con la clasificación no supervisada, que veremos en detalle en el siguiente punto. Podemos diferenciar dos fases dentro de la clasificación:

- Primera fase: Contamos con un conjunto de entrenamiento o de aprendizaje (para el diseño del clasificador) y otro llamado de test o de validación (para clasificación), estos nos servirán para construir un modelo o regla general para la clasificación.
- Segunda fase: Proceso en sí de clasificar los objetos o muestras de las que se desconoce la clase a las que pertenecen.

Ejemplos de clasificación supervisada son: Predicciones de pérdidas dentro de nuestra empresa, reconocimiento de caracteres escritos a mano, clasificaciones según las variables indicadas. Entre las técnicas dentro del grupo de clasificación supervisada se encuentran los algoritmos como los de clasificación por vecindad.

4.2. Comparación entre Clasificación Supervisada y No Supervisada

Con la clasificación no supervisada no contamos con conocimiento a priori, por lo que tendremos un área de entrenamiento disponible para la tarea de clasificación. A la clasificación no supervisada se llama también clustering. [18]

En este tipo de clasificación contamos con objetos o muestras que tienen un conjunto de características, de las que no sabemos a qué clase o categoría pertenece, entonces la finalidad es el descubrimiento de grupos de objetos cuyas características afines nos permitan separar las diferentes clases.

4.3. Técnicas de Clasificación Supervisada

El algoritmo de los k vecinos más cercanos:

un sistema de clasificación supervisado basado en criterios de vecindad. El algoritmo del vecino más cercano explora todo el conocimiento almacenado en el conjunto de entrenamiento para determinar cuál será la clase a la que pertenece una nueva muestra, pero únicamente tiene en cuenta el vecino más próximo a ella, por lo que es lógico pensar que es posible que no se esté aprovechando de forma eficiente toda la información que se podría extraer del conjunto de entrenamiento. Con el objetivo de resolver esta posible deficiencia surge la regla de los k vecinos más cercanos (k -NN), que es una extensión en la que se utiliza la información suministrada por los k ejemplos del conjunto de entrenamiento más cercanos. En problemas prácticos donde se aplica esta regla de clasificación se acostumbra tomar un número k de vecinos impar para evitar posibles empates (aunque esta decisión solo resuelve el problema en clasificaciones binarias). En otras ocasiones, en caso de empate, se selecciona la clase que verifique que sus representantes tengan la menor distancia media al ejemplo que se está clasificando. En última instancia, si se produce un empate, siempre se puede decidir aleatoriamente entre las clases con mayor representación.

Redes Neuronales: Las redes neuronales en general han sido propuestas en numerosas ocasiones como instrumentos útiles para la Recuperación de Información y también para la clasificación automática. Coincidiendo con el auge general, en todos los campos, de las redes neuronales, especialmente en los primeros años 90. De una manera genérica, una de las principales aplicaciones de las redes neuronales es el reconocimiento de patrones. Podemos distinguir tres tipos de aprendizaje, las redes neuronales con el Aprendizaje supervisado consisten en que la red dispone de los patrones de entrada y los patrones de salida que deseamos para esa entrada y en función de ellos se modifican los pesos de las sinapsis para ajustar la entrada a esa salida.

Capítulo 5

Análisis De Paquetes

5.1. Random Forest

Random forest es una combinación de árboles predictores en la que cada árbol depende de los valores de un vector aleatorio probado independientemente y con la misma distribución para cada uno de estos. El algoritmo para inducir un random forest fue desarrollado por Leo Breiman y Adele Cutler, siendo Random forests su marca de fábrica.

Este algoritmo mejora la precisión en la clasificación mediante la incorporación de aleatoriedad en la construcción de cada clasificador individual. Esta aleatorización puede introducirse en la partición del espacio (construcción del árbol), así como en la muestra de entrenamiento. El Random Forest comienza con una técnica de aprendizaje automático estándar llamada árbol de decisiones, que, en cuanto al conjunto, corresponde a un aprendizaje. En un árbol de decisión, una entrada se introduce en la parte superior y hacia abajo a medida que atraviesa el árbol de los datos, los cuales se acumulan en conjuntos más pequeños.

El Método Random Forest es fácil de aprender y de usar, tanto por profesionales como por personas con menos experiencia. Con poca investigación y tiempo de desarrollo, puede ser utilizado por personas con poca base estadística. En pocas palabras, este método nos permite hacer de manera segura predicciones más precisas y sin la mayoría de los errores básicos comunes a otros métodos.

Principales beneficios:

- Precisión.
- Funciona de manera eficiente con bases de datos de gran tamaño.
- Maneja miles de variables de entrada sin necesidad de borrado.
- Da estimaciones de qué variables son importantes en la clasificación.
- Genera una estimación objetiva interna de la generalización de errores.
- Proporciona métodos eficaces para estimar datos que faltan.
- Los bosques generados se pueden guardar para uso futuro en otros datos.

- Los prototipos se calculan de manera que proporcionan información acerca de la relación entre las variables y la clasificación.
- Calcula proximidades entre pares de casos que se pueden utilizar en la agrupación, la localización de los valores extremos, o (en escala) dan interesantes vistas de los datos
- Ofrece un método experimental para la detección de interacciones de variables.

Fuente: <http://www.bigdatahispano.org/noticias/algoritmo-random-forest/>

Paquete Random Forest RandomForest implementa el algoritmo Random Forest de Breiman (basado en el código Fortran original de Breiman y Cutler) para clasificación y regresión. También se puede utilizar en clasificación no supervisada para evaluar proximidades entre los puntos de datos.

- Función Randomforest

Uso

S3 method for class 'formula' randomForest(formula, data=NULL, ..., subset, na.action=na.fail) Default S3 method: randomForest(x, y=NULL, xtest=NULL, ytest=NULL, ntree=500, mtry=if (!is.null(y) && !is.factor(y)) max(floor(ncol(x)/3), 1) else floor(sqrt(ncol(x))), replace=TRUE, classwt=NULL, cutoff, strata, sampsize = if (replace) nrow(x) else ceiling(.632*nrow(x)), nodesize = if (!is.null(y) && !is.factor(y)) 5 else 1, maxnodes = NULL, importance=FALSE, localImp=FALSE, nPerm=1, proximity, oob.prox=proximity, norm.votes=TRUE, do.trace=FALSE, keep.forest=!is.null(y) && is.null(xtest), corr.bias=FALSE, keep.inbag=FALSE, ...) S3 method for class 'randomForest' print(x, ...)

Argumentos

data Data frame opcional que contiene las variables en el modelo. Por defecto, las variables se toman del entorno del que el Randomforest es llamado.

subset Vector de índices que indican que filas se deben usar.

na.action Función para especificar la acción a llevar a cabo si se encuentran NAs

x, formula Un data frame o una matriz de predictores, o una formula describiendo el modelo para ser equipado.

y Un vector de respuesta. Si hay un factor, la clasificación es asumida, de la otra forma la clasificación es fictia. Si se omite, randomForest se ejecutar en modo sin supervisión.

xtest Un data frame o matriz (como x) que contiene predictores para el set de prueba.

ytest Respuesta para el set de prueba

ntree Nmero de árboles a utilizar. No se debe establecer un número demasiado pequeno, para garantizar que cada fila de entrada se ha predecido por lo menos dos veces.

mtry Número de variables mostradas al azar como candidatos en cada división. Tenga en cuenta que los valores por defecto son diferentes para la clasificación. (\sqrt{p}) cuando p es el número de variables en x) y regresión ($p/3$)

replace Pueden las muestras de los casos realizarse con o sin sustitución?

classwt Priors de clases. Ignorado para la regresión.

cutoff Sólo clasificación. Un vector de longitud igual al número de clases. La clase ganadora para una clasificación es la que tiene la máxima relación de votos para cortar. El valor predeterminado es $1/k$, donde k es el número de clases (es decir, la mayoría voto gana).

strata Un factor (variable) que se utiliza para el muestreo estratificado.

sampsize Tamaño de muestra/s a dibujar. Para la clasificación, si sampsize es un vector de la longitud del número de estratos, entonces la muestra se estratifica por estratos, y los elementos de sampsize indican los números que pueden extraerse de los estratos.

nodesize Tamaño mínimo de los nodos terminales.

maxnodes Número máximo de nodos terminales de árboles que el bosque puede tener.

importance Se debe registrar la importancia de los predictores?

localImp Se debe calcular la medida de importancia por casos?

nPerm Número de veces que los datos OOB se permutaron por rbo1 para la evaluación de importancia de las variables.

proximity La medida de proximidad entre filas debe ser calculada?

oob.prox La proximidad debe ser calculada solo en datos out-of-bag?

norm.votes Si es verdadero, el resultado final de los votos se expresan como fracciones. Si es falso el recuento de votos se devuelven. Ignorado para la regresión.

do.trace Si se establece como True, da una salida más detallada mientras que el RandomForest se ejecuta. Si se establece algún entero, entonces la salida es imprimida por todos los do.trace rboles.

keep.forest Si se establece en False, el bosque no será retenido en el objeto de salida. Si xtest es dado, entonces por defecto es falso.

corr.bias realizar la corrección de sesgo para la regresión?

keep.inbag debe ser devuelta una n por una matriz ntree?

Ejemplo

```
#install.packages("randomForest") library(randomForest) modelo1 - randomForest(class=.,data=datos2, ntree=500,importance=TRUE,maxnodes=10,mtry=25) #Creamos un objeto con las 'importancias' de las variables importancia=data.frame(importance(modelo1)) library(reshape) importancia-sort_df(importancia,vars='MeanDecreaseGini') importancia
```

Implementación en R Lo primero que debemos hacer es instalar el paquete RandomForest en Rstudio. Para ello introducimos el siguiente comando:

```
install.packages("RandomForest")
```

Para comprender el funcionamiento de RandomForest, vamos a utilizar los datos de la Base de Datos IRIS, que viene por defecto en la instalación de R. Iris cuenta con las variables:

- Longitud Sépalo
- Anchura Sépalo
- Longitud Pétalo
- Anchura Pétalo

- Especies

De tres especies de flores: Iris Setosa, versicolor y virgínica.

En este ejercicio vamos a trabajar con 150 muestras. Cien de estas muestras van a ser de entrenamiento, y cincuenta de test.

Para obtener las muestras de entrenamiento, debemos introducir el siguiente comando:

iris_train - iris

s,

```
> iris_train <- iris [s,]
>
> iris_train
   Sepal.Length Sepal.width Petal.Length Petal.width
33           5.2         4.1         1.5         0.1
51           7.0         3.2         4.7         1.4
46           4.8         3.0         1.4         0.3
49           5.3         3.7         1.5         0.2
145          6.7         3.3         5.7         2.5
104          6.3         2.9         5.6         1.8
110          7.2         3.6         6.1         2.5
76           6.6         3.0         4.4         1.4
55           6.5         2.8         4.6         1.5
20           5.1         3.8         1.5         0.3
89           5.6         3.0         4.1         1.3
14           4.3         3.0         1.1         0.1
129          6.4         2.8         5.6         2.1
42           4.5         2.3         1.3         0.3
..          ..          ..          ..          ..
```

Para obtener las 50 muestras restantes que utilizaremos como Test introducimos: iris_test - iris [-s,]

Procedemos a realizar la función RandomForest con el siguiente comando: rfm j- randomForest (Species ~., iris_train) En el indicamos que vamos a usar todas nuestras variables con Species y también especificamos los datos que vamos a utilizar, en este caso van a ser nuestros valores de entrenamiento aadidos anteriormente en Iris_train.

```
> rfm <- randomForest (Species ~., iris_train)
> rfm

Call:
randomForest(formula = Species ~ ., data = iris_train)
Type of random forest: classification
Number of trees: 500
No. of variables tried at each split: 2

OOB estimate of error rate: 8%
Confusion matrix:
      setosa versicolor virginica class.error
setosa      34          0          0 0.00000000
versicolor   0         30          3 0.09090909
virginica    0          5         28 0.15151515
```

En el podemos comprobar que nos indica que el campo ntree es = 500. Por lo que el número de árboles creados es de 500.

Con los datos obtenidos de la muestra de entrenamiento, procedemos a predecir los 50 restantes con el siguiente comando:

p - predict(rfm,iris.test)

Pasamos a mostrar el resultado en forma de tabla para poder comprenderlo de una manera más eficaz:

```
> table(iris_test[,5],p)
```

	p		
	setosa	versicolor	virginica
setosa	16	0	0
versicolor	0	16	1
virginica	0	0	17

```
> |
```

Para comprobar la exactitud de los datos mostrados introducimos el siguiente comando: `mean(iris_test[,5]==p)`

En el que nos indica que la exactitud de los calculos es del 94 %

Para comprobar la importancia de las variables en los cálculos introducimos el comando `importance` de la siguiente forma:

```
importance(rfm)
```

```
> importance(rfm)
```

	MeanDecreaseGini
sepal.Length	7.036293
sepal.width	3.003451
Petal.Length	28.568069
Petal.width	27.293478

En la imagen podemos comprobar que el valor más importante es el de la longitud del pétalo, seguido de la anchura del pétalo.

Por último utilizamos la variable `getTree`, con este comando vamos a obtener 6 columnas en el caso de que la variable `labelVar=True`

El significado de las columnas es el siguiente:

- left daughter: La fila donde el nodo de la izquierda es la hija; 0 si el nodo es terminal
- right daughter: La fila donde el nodo de la derecha es la hija; 0 si el nodo es terminal
- split var: Variable se utiliz para dividir el nodo; 0 si el nodo es terminal
- split point: Lugar en el que la división es la mejor
- status: Es el terminal de nodo (-1) o no (1)
- prediction: la prediccin para el nodo; 0 si el nodo no es terminal

En nuestro caso procedemos a obtener los resultados: `getTree(rfm,500,labelVar=TRUE)`


```
> getTree(rfm, 500, labelVar=TRUE)
      left daughter right daughter      split var split point status prediction
1         2              3 Petal.Length      4.75      1      <NA>
2         4              5 Petal.width      0.75      1      <NA>
3         6              7 Petal.Length      4.85      1      <NA>
4         0              0      <NA>      0.00     -1      setosa
5         0              0      <NA>      0.00     -1 versicolor
6         8              9 Sepal.Length      5.95      1      <NA>
7        10             11 Sepal.width      2.75      1      <NA>
8         0              0      <NA>      0.00     -1 versicolor
9         0              0      <NA>      0.00     -1 virginica
10        12             13 Sepal.width      2.65      1      <NA>
11        0              0      <NA>      0.00     -1 virginica
12        0              0      <NA>      0.00     -1 virginica
13        14             15 Petal.width      1.75      1      <NA>
14        0              0      <NA>      0.00     -1 versicolor
15        0              0      <NA>      0.00     -1 virginica
```

En el caso de querer comprobar más datos como `nrnodes`, `maxcat`, `xlevels`, etc.

Slo hara falta introducir el comando `Str(rfm)`

Referencias

<https://cran.rproject.org/web/packages/randomForest/randomForest.pdf>

Paquete Rborist Implementación acelerada del algoritmo Random Forest. En sintonía con el multineo y e hardware del GPU. Funciona con la mayoría de los lenguajes de numricos front-end, adems de R. La utilizacin es similar a la proporcionada por el paquete `randomForest`”

■ Función Rborist

Uso

Default S3 method: `Rborist(x, y, nTree=500, withRepl = TRUE, predProb = ifelse(!is.factor(y), 0.4, sqrt(ncol(x))/ncol(x)), predWeight = rep(1.0, ncol(x)), nSamp = ifelse(withRepl, nrow(x), round((1-exp(-1))*nrow(x))), minNode = ifelse(!is.factor(y), 6, 2), nLevel = 0, minInfo = 0.01, sampleWeight = NULL, quantVec = NULL, quantiles = !is.null(quantVec), pvtBlock = 8, pvtNoPredict = FALSE, ...)`

Argumentos

x Matriz que contiene nuevos datos. Puede ser tanto una matriz numérica como un dataframe con columnas numéricas o categóricas.

Y Vector de respuesta. Puede ser tanto numérico comoo categórico. Por lo tanto X e Y.

nTree Número de árboles que se van a usar.

withRepl Si el muestreo por fila es por sustitución

predProb La probabilidad de que una predicción arbitraria sea elegida como el candidato de una división. Similar en espíritu al intento por opción de Random Forest, el cual es un número fijo de predictores sobre los que tratan de dividir cada nodo.

predWeight Vector numérico de pesos para la selección de los predictores individuales como la selección de candidatos.

nSamp El número de muestras a dibujar cuando se trabaja con un árbol individual.

minNode Anchura mínima del conjunto sobre la cual intentar la división. Similar al parámetro de tamaño de nodo de RandomForest.

nLevel Número de niveles del árbol. Al menos un nivel del árbol de decisión está construido. El valor 0 está reservado.

minInfo Una relación umbral de información de valores entre un subnodo y su matriz, por debajo de la cual la división no se trató.

sampleWeight Vector numérico de pesos para la elaboración de filas individuales como candidatos muestra.

quantVec Vector de cuantiles sobre el que reportar la regresión por cuantiles.

quantiles Si `quantVec` no es especificada pero cuantiles han sido especificados, los cuantiles empleados son por defecto. `quantiles has been specified, then quartiles are employed by default.`

pvtBlock Bloqueo de parámetros para la optimización del rendimiento. Actualmente no compatible.

pvtNoPredict Omite predicción. Para testear únicamente.

Implementación en R Procedemos a probar el paquete Rborist. Para ello vamos a implementar paso a paso el algoritmo que nos ofrece el paquete de ejemeplo. En primer lugar debemos descargarlo e instalarlo en R.

```
install.packages(Rborist) library (rborist)
```

Como siguiente paso establecemos que en número de filas que va a tener nuestro entrenamiento va a ser de 5000:

```
nRow <- 5000
```

Establecemos los datos aleatorios que van a contener nuestros Data frames X e Y

Para ello en el caso de X utilizamos Replicate con Rnorm para generar desviaciones aleatorias, mientras que en Y utilizamos una función.

```
x <- data.frame(replicate(6, rnorm(nRow))) y <- with(x, X1^2 + sin(X2) + X3 * X4)
```

El siguiente paso es elaborar la función genérica de Rborist e indicarle un límite en nuestro caso de 300 árboles `rb <- Rborist(x, y)`

```
rb <- Rborist(x, y, nTree = 300)
```

Con el comando STR podemos obtener la información del resultado de la utilización Rborist

```
> str (rb)
List of 5
 $ forest      :List of 4
  ..$ forestNode:<externalptr>
  ..$ origin    : num [1:300] 0 2213 4440 6677 8894 ...
  ..$ facorig    : num [1:300] 0 0 0 0 0 0 0 0 0 0 ...
  ..$ facsplit   : num(0)
  ..- attr(*, "class")= chr "Forest"
 $ leaf        :List of 6
  ..$ origin    : num [1:300] 0 1107 2221 3340 4449 ...
  ..$ node       :<externalptr>
  ..$ bagRow     :<externalptr>
  ..$ rowTrain   : num 5000
  ..$ rank       : num [1:948660] 4638 4196 1702 224 1922 ...
  ..$ yRanked    : num [1:5000] -6.02 -5.85 -5.3 -5.01 -4.75 ...
  ..- attr(*, "class")= chr "LeafReg"
 $ signature   :List of 2
  ..$ predMap: int [1:6] 0 1 2 3 4 5
  ..$ level    : list()
  ..- attr(*, "class")= chr "signature"
 $ training    :List of 1
  ..$ info: num [1:6] 2252 2395 2240 2303 2254 ...
 $ validation:List of 4
  ..$ yPred: num [1:5000] 0.866 1.383 0.812 1.482 0.849 ...
  ..$ mse   : num 3.51
  ..$ rsq    : num -0.0348
  ..$ qPred: num[0 , 0 ]
  ..- attr(*, "class")= chr "ValidReg"
 - attr(*, "class")= chr "Rborist"
```

Referencias

<https://cran.rproject.org/web/packages/Rborist/Rborist.pdf>

Paquete Ranger Paquete que permite una implementación rápida de Random Forest. Particularmente desarrollado para trabajar con grandes cantidades de datos. Permite realizar tareas de clasificación, regresión, probabilidad y árboles de decisión.

- Función Ranger

Uso

```
(rangerformula = NULL, data = NULL, num.trees = 500, mtry =
NULL, importance = "none", write.forest = FALSE, probability = FALSE,
min.node.size = NULL, replace = TRUE, sample.fraction = ifelse(
replace, 1, 0.632), splitrule = NULL, case.weights = NULL, split.select.weights
= NULL, always.split.variables = NULL, respect.unordered.factors = FALSE,
scale.permutation.importance = FALSE, keep.inbag = FALSE, num.threads
= NULL, save.memory = FALSE, verbose = TRUE, seed = NULL, dependent.variable.name
= NULL, status.variable.name = NULL, classification = NULL)
```

Argumentos

formula Objeto de clase o carácter que describe el modelo a ajustar

data Conjunto de datos de Data.frame o gwaa.darra (GenABEL)

num.trees Número de árboles

mtry Número de variables que permiten dividir cada nodo. Por defecto es la raíz cuadrada de las variables numéricas.

importance Modos de la variable importancia, uno de ninguno, impureza, permutación. La medida de impureza es el índice de Gini para la clasificación y la varianza de respuestas para la regresión.

write.forest Guarda el objeto ranger.forest, el cual se necesita para la predicción.

probability Probabilidad de crecimiento de un bosque como en Malley et al. (2012)

min.node.size Mínimo tamaño de nodo. Por defecto es 1 para la clasificación, 5 para la regresión, 3 para la supervivencia y 10 para la probabilidad.

replace Muestra con reemplazo

sample.fraction Muestra de una fracción para la observación. Por defecto es 1 para la muestra con reemplazo y 0.632 para la muestra sin reemplazo.

splitrule Regla de división, para la supervivencia sólo. La regla de la división puede ser escogida de 'longrank' y 'c' con valor predeterminado 'longrank'.

case.weights Pesos para las muestras de observaciones de entrenamiento. Las observaciones con grandes pesos serán seleccionadas con una alta probabilidad en muestras bootstrap para los árboles.

split.select.weights Vector de números con pesos entre 0 y 1, representando las probabilidad de seleccionar variables para la división. Alternativamente, una lista del tamaño num.tress, contiene la división de los pesos de los vectores por cada árbol que puede ser usado.

always.split.variables Vector de caracteres con el nombre de las variables que van a ser siempre tratadas para la división.

respect.unordered.factors Considerar los factores de covariables no ordenadas como variables categóricas no ordenadas. Si es falso, todos los factores se consideran ordenados.

scale.permutation.importance La importancia de la escala de permutación por error estándar en (Breiman 2001). Solamente aplicable si ha seleccionado el uso de permutación variable de importancia.

keep.inbag Guardar la frecuencia de observaciones de cada árbol.

num.threads Número de amenazas. El valor predeterminado es el número de CPU disponibles.

save.memory Utilizar el ahorro de memoria (es más lento) en el modo de división. No tiene efecto para los datos de GWAS.

verbose Salida detallada encendido o apagado.

seed Semilla aleatoria. EL valor predeterminado es NULL, que genera la semilla de R.

dependent.variable.name Nombre de la variable dependiente, necesaria si hay una fórmula dada. Para los bosques de supervivencia esta es la variable tiempo.

status.variable.name Nombre de la variable de estado, solo se aplica a los datos de supervivencia y es necesaria si hay una fórmula dada. Utilice 1 para el evento y 0 para censurar.

classification Sólo es necesaria si los datos están en una matriz. Establecida en TRUE para crear el bosque de una clasificación.

Detalles El tipo de árbol es determinado por el tipo de variable de dependencia. El índice Gini es usado como una regla de división de la clasificación, la respuesta estimada para las varianzas de regresión y el log-rank test para la supervivencia. Para la supervivencia de la prueba de log-rank o una regla de reparto basada en el AUC están disponibles. tenga en cuenta que para los nodos de clasificación y regresión con el tamaño más pequeño que `min.node.size` pueden ocurrir, como en el bosque original al azar. Para la supervivencia todos los nodos contienen como mínimo muestras de `min.node.size`. Variables seleccionadas con `always.split.variables` se trataron adicionalmente a las variables `mtry` seleccionados al azar. En `split.select.weights` variables ponderadas con 0 Nunca se seleccionan y variables con 1 siempre se seleccionan. El uso de `split.select.weights` puede aumentar los tiempos de cálculo para grandes bosques. Para un gran número de variables y tramas de datos como datos de entrada, la interfaz puede ser lenta o imposible de usar. Alternativamente `dependent.variable.name` (y `status.variable.name` para la supervivencia) puede ser usado. Considere establecer `save.memory = TRUE` si se encuentra con problemas de memoria. Para los datos de GWAS considerar la combinación de guardabosques con el paquete GenABEL. Vea la sección de ejemplos abajo para la demostración del uso de los datos Plink. Todos los SNPs en el objeto GenABEL serán utilizados para la división. Para utilizar solo el SNP sin sexo o de otras variables del archivo de fenotipo, utilice 0 en el lado derecho de la fórmula. Tenga en cuenta que los valores que faltan son tratados como una categoría adicional, mientras se divide. Ver <https://github.com/mnwright/ranger> para la versión de desarrollo. notas: El multithreading actualmente no se admite para las plataformas Microsoft Windows.

Valores

Forest Bosque guardado (Si `write.forest` está en true).

Predictions Clases/valores predichos. (clasificación y regresión solamente).

Ejemplo `require(ranger) ## Classification forest with default settings`
`ranger(Species ~., data = iris) ### Prediction train.idx <- sample(nrow(iris),`
`2/3 * nrow(iris)) iris.train <- iris[train.idx,] iris.test <- iris[-train.idx,] rg.iris`
`<- ranger(Species ~., data = iris.train, write.forest = TRUE) pred.iris <- pre-`
`dict(rg.iris, data = iris.test) table(iris.test$Species, pred.iris$predictions)`
`## Variable importance rg.iris <- ranger(Species ~., data = iris, impor-`
`tance = "impurity") rg.iris$variable.importance ## Survival forest re-`
`quire(survival) rg.veteran <- ranger(Surv(time, status) ~., data = veter-`
`an) plot(rg.veteran$unique.death.times, rg.veteran$survival[1,]) ## Al-`
`ternative interface ranger(dependent.variable.name = "Species", data =`
`iris) ## Not run: ## Use GenABEL interface to read Plink data into`
`R and grow a classification forest ## The ped and map files are`
`not included library(GenABEL) convert.snp.ped("data.ped", "data.map",`
`"data.raw") dat.gwaa <- load.gwaa.data("data.pheno", "data.raw") phda-`
`ta(dat.gwaa)$trait <- factor(phdata(dat.gwaa)$trait) ranger(trait ~., data`
`= dat.gwaa) ## End(Not run)`

Implementación en R Procedemos a seguir los pasos que nos marca Ranger para elaborar un RandomForest.

Lo primero que debemos realizar es descargar el paquete e instalarlo:

```
install.packages("ranger") Library (ranger)
```

Una vez instalado, vamos a proceder a realizar el algoritmo en la Base de Datos Iris que contiene R.

Este paquete se caracteriza por su rápida implementación, para realizar la clasificación por defecto, sólo deberíamos introducir esta sentencia:

```
ranger(Species ~., data = iris)
```

```
> ranger(Species ~., data = iris)
Ranger result

call:
ranger(Species ~., data = iris)

Type:                Classification
Number of trees:      500
Sample size:          150
Number of independent variables: 4
Mtry:                 2
Target node size:     1
variable importance mode: none
OOB prediction error: 4.00 %
> |
```

Los pasos para realizar una predicción son los siguientes:

1. Con la siguiente sentencia establecemos un conjunto de números aleatorios

```
train.idx <- sample(nrow(iris), 2/3 * nrow(iris))
```

```
> train.idx <- sample(nrow(iris), 2/3 * nrow(iris))
> train.idx
[1] 21 120 23 31 145 150 46 132 7 25 146 116 110 53 1 64 6
[18] 45 41 12 76 98 78 49 66 56 72 93 70 147 144 67 47 86
[35] 82 48 124 27 89 92 74 142 11 149 109 79 133 39 134 14 126
[52] 80 55 138 16 36 33 26 139 81 94 68 121 118 135 113 102 100
[69] 24 115 8 122 127 28 129 117 9 131 140 20 95 3 44 136 17
[86] 90 71 34 43 10 99 108 29 111 40 32 22 96 83 125
```

La base de datos de Iris la forman 150 elementos.

Nos quedamos con el índice aleatorio de los 100 primeros

```
iris.train - iris[train.idx, ]
```

Los restantes que no han sido aadidos en la Base de Datos de entrenamiento, los aadimos a la de test para poder realizar la predicción:

```
iris.test - iris[-train.idx, ]
```

Elaboramos el algoritmo de RandomForest con la Base de Datos de entrenamiento

```
rg.iris - ranger(Species ~ ., data = iris.train, write.forest = TRUE)
```

```
> rg.iris
Ranger result

Call:
ranger(Species ~ ., data = iris.train, write.forest = TRUE)

Type:                                Classification
Number of trees:                      500
Sample size:                          100
Number of independent variables:      4
Mtry:                                 2
Target node size:                     1
Variable importance mode:             none
OOB prediction error:                 2.00 %
```

Esta función nos indica que el número de árboles utilizados ha sido de 500, nos indica que la muestra iris.train contiene 100 elementos y que el número de predictores es 4.

Por último realizamos la predicción de la muestra iris.test

```
> pred.iris <- predict(rg.iris, dat = iris.test)
> table(iris.test$Species, pred.iris$predictions)
```

	setosa	versicolor	virginica
setosa	13	0	0
versicolor	0	19	2
virginica	0	1	15

Mostramos la importancia de las variables que han llevado al resultado de la clasificación:

```
rg.iris - ranger(Species ~ ., data = iris, importance = 'impurity') rg.iris$variable.importance
```

```
> rg.iris <- ranger(Species ~ ., data = iris, importance = "impurity")
> rg.iris$variable.importance
Sepal.Length Sepal.Width Petal.Length Petal.Width
10.461555 2.533183 46.889661 39.346602
```

Podemos comprobar que la variable que ha tenido más peso en la decisión es la longitud del pétalo.

Referencias

<https://cran.rproject.org/web/packages/ranger/ranger.pdf>

5.2. Decision Tree

Un árbol de decisión es una forma gráfica y analítica de representar todos los eventos (sucesos) que pueden surgir a partir de una decisión asumida en cierto momento. Nos ayudan a tomar la decisión más acertada, desde un punto de vista probabilístico, ante un abanico de posibles decisiones. Permite desplegar visualmente un problema y organizar el trabajo de cálculos que deben realizarse. Cuenta con una terminología para representarlos:

- **Nodo de decisión:** Indica que una decisión necesita tomarse en ese punto del proceso. Está representado por un cuadrado.
- **Nodo de probabilidad:** Indica que en ese punto del proceso ocurre un evento aleatorio. Está representado por un círculo.
- **Rama:** Nos muestra los distintos caminos que se pueden emprender cuando tomamos una decisión o bien ocurre algún evento aleatorio. Está representado por una flecha.

Fuente: http://www.dmae.upct.es/~mcruiz/Telem06/Teoria/arbol_decision.pdf

Un ejemplo de implementación de Decision Tree puede ser la siguiente:

Desde una Base de Datos de clientes extraemos el nombre, altura, sexo y nacionalidad.

Name/Weight (lbs.)/Sex/Nationality

Larry 195 M American

Jerry 190 M American

Carrie 160 F American

Cheri 165 F American

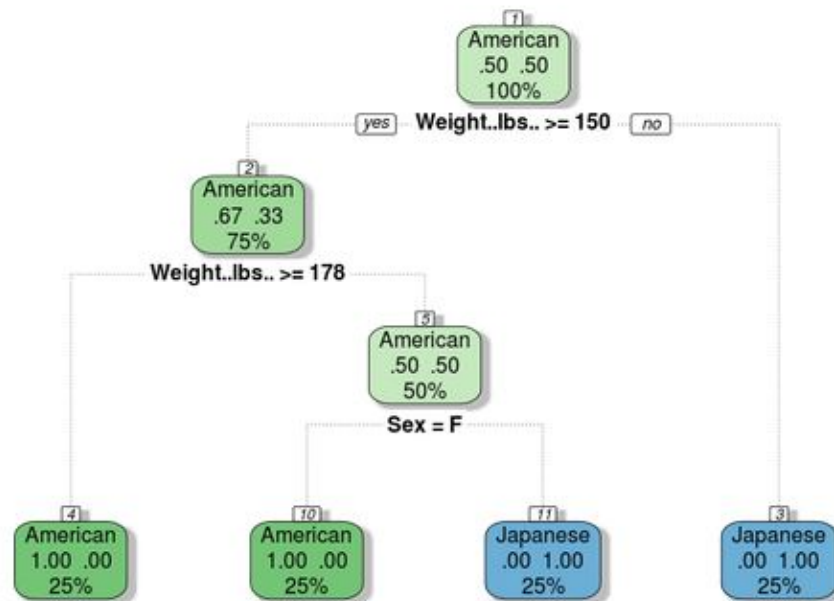
Yoshi 165 M Japanese

Yasuo 160 M Japanese

Michiko 130 F Japanese

Noriko 140 F Japanese

Al introducir el algoritmo de decisión en R obtendríamos una estructura parecida a la siguiente:



En primer lugar el algoritmo pregunta si la estatura es menor de 150, en caso de ser sí nos quedamos sólo con la posibilidad de que sean japoneses. En caso de ser no preguntamos si la estatura es mayor de 178 y si es sí sólo obtenemos americanos. Para predecir el resto necesitamos la variable de Sexo. En caso de ser hombre clasificamos como americanos y en caso de ser mujer clasificamos como Japoneses.

Paquete C50 Con C50 puede implementar árboles de decisión y modelos basados en reglas, para el reconocimiento de patrones. Su autores son: Max Kuhn, Steve Weston, Nathan Coulter, Mark Culp. La fecha de publicación en R Cran es la siguiente: 20150309.

■ Función C5.0.default

Uso

```

C5.0(x, ...) ## Default S3 method: C5.0(x, y, trials = 1, rules= FALSE,
weights = NULL, control = C5.0Control(), costs = NULL, ...) ## S3
method for class 'formula' C5.0(formula, data, weights, subset, na.action
= na.pass, ...)

```

Argumentos

x Data frame o matriz de predictores.

y Vector de factores con 2 o más niveles

trials Número entero que especifica el número de iteraciones. El valor 1 indica que se utiliza un único modelo

rules Es un número lógico. Debe ser el árbol descompuesto en una regla basada en un modelo?

weights Vector numérico opcional de ponderación de casos. Nota: Tenga en cuenta que los datos utilizados para las ponderaciones de los casos no serán utilizados como una variable de división en el modelo (ver <http://www.rulequest.com/see5-win.html#CASEWEIGHT> para las notas Quinlan en las ponderaciones de los casos)

control Una lista de parámetros de control, (ver <https://cran.r-project.org/web/packages/C50/C50.pdf>)

costs Matriz de costes asociada con los errores posibles. La matriz debe tener C columnas y filas, en el que C es el número de clases por niveles.

formula Una formula, con respuesta y al menos un predictor.

data Un dataframe opcional para interpretar las variables llamadas en la fórmula.

subset Expresión opcional que indica que sólo un subconjunto de las filas de datos puede ser usado en el ajuste

na.action Una función que indica que ocurre cuando los datos contienen NAs. Por defecto se incluyen los valores perdidos.

Detalles Este modelo extiende los algoritmos de clasificación C4.5 descritos en Quinlan (1992). Todos los detalles de extensión no están documentados. El modelo puede tomar la forma de un árbol de decisión completo o una colección de reglas. Cuando se utiliza el método de la fórmula, los factores y otras clases se conservan (es decir, son variables ficticias no se crean automáticamente). Este modelo en particular maneja los datos no numéricos de algunos tipos. La matriz de costos debe ser $C \times C$, donde C es el número de clases. Los elementos de la diagonal son ignorados. Las columnas deben corresponder a las clases verdaderas y las filas son las clases previstas. Por ejemplo, si $C = 3$ con clases de rojo, azul y verde (en ese orden), un valor de 5 en el (2,3) elemento de la matriz indicaría que el costo de la predicción de una muestra verde como azul es cinco veces el valor normal (de uno). Tenga en cuenta que cuando se utilizan los costos, las probabilidades de clase no pueden ser generados usando predict.C5.0.

Valores

boostResults Versión analizada de la mesa de impulso mostrada en la salida.

call Función de llamada

caseWeights No soportado actualmente

control Un eco de las especificaciones de Control C5.0

cost La versión en texto de la matriz de costes

costMatrix Un eco del modelo de argumentos

dims Dimensiones originales de la matriz de predicción o el data frame

levels Un vector de caracteres de los niveles de factor para el resultado

names Versión de texto del fichero de nombres

output Versión en texto de la línea de comandos de salida.

predictors Vector en texto de los nombres predictores

rbm Número lógico para reglas

rules Versión de caracteres del fichero de reglas

size Vector de enteros del tamaño del árbol/regla

tree Versión en texto del fichero árbol

trials Vector con elementos solicitados (un eco de la llamada de función) y Real(el número de modelo utilizado)

Ejemplo `data(churn) treeModel - C5.0(x = churnTrain[, -20], y = churnTrain$churn) treeModel summary(treeModel) ruleModel - C5.0(churn - ., data = churnTrain, rules = TRUE) ruleModel summary(ruleModel)`

- Función C5.0Control

Uso

```
C5.0Control(subset = TRUE, bands = 0, winnow = FALSE, noGlobalPruning = FALSE, CF = 0.25, minCases = 2, fuzzyThreshold = FALSE, sample = 0, seed = sample.int(4096, size = 1) - 1L, earlyStopping = TRUE, label = ".outcome")
```

Argumentos

subset Elemento lógico. Debe el modelo evaluar grupos de predictores discretos para las divisiones? Nota: Por defecto el valor está en Falso.

bands Número entero entre 2 y 1000. Si es verdadero, el modelo ordena las reglas por en la tasa de error y los grupos de reglas sobre el número especificado de bandas. Esto modifica la salida de modo que el efecto sobre la tasa de error puede ser vista por los grupos de reglas con bandas. Si esta opción es elegida y reglas= Falso, se emitirá una advertencia y las reglas serán cambiadas a Verdadero.

winnow Elemento lógico. Debe ser usado el predictor de aventar?

noGlobalPruning Lógica para simplificar la reducción del árbol

CF Un número (0,1) para el factor de confianza

minCases Un entero para el número menor de muestras que deben ser puestas en al menos dos de las divisiones.

fuzzyThreshold Una palanca lógica para evaluar posibles divisiones avanzadas de los datos. Ver Quinlan (1993) para más detalles y ejemplos.

sample Valor entre (0, .999) que especifica la proporción aleatoria que los datos deben usar para el modelo de entrenamiento. Por defecto, todas las muestras son usadas para el modelo de entrenamiento. Las muestras no usadas para el entrenamiento son usadas para evaluar la precisión del modelo en el valor de salida.

seed Un entero del número aleatorio de semillas dentro del código C

earlyStopping Una lógica para alternar si el método interno para detener/impulsar debería ser usado.

label Una etiqueta de caracteres del resultado utilizado en la salida.

Ejemplo `data(churn) treeModel = C5.0(x = churnTrain[, -20], y = churnTrain$churn, control = C5.0Control(winnow = TRUE)) summary(treeModel)`

■ Función C5imp

Uso

`C5imp(object, metric = 'usage', pct = TRUE, ...)`

Argumentos

object Un objeto de la clase C5.0

metric O bien 'uso' o 'divisiones' (ver detalles más abajo)

pct Se deben convertir los valores de importancia para estar entre 0 y 100?

Detalles De forma predeterminada, C5.0 mide la importancia del predictor determinando el porcentaje de muestras de conjuntos de entrenamiento que caen en todos los nodos terminales después de la división (esto es usado cuando la métrica es 'Uso'). Por ejemplo, el predictor en la primera división tiene automáticamente una medición de importancia de 100 %. Otros predictores pueden utilizarse con frecuencia en las divisiones, pero si los nodos terminales cubren sólo un puado de muestras del conjunto de entrenamiento, las puntuaciones de importancia pueden ser cercanas a cero. La misma estrategia es aplicada a una regla basada en modelos, así como a las correspondientes versiones impulsadas del modelo. Hay una diferencia en el número de uso de atributos entre esta salida y el comando nominal de la línea de salida. Aunque los cálculos son casi exactamente los mismos (no añadimos 1/2 a todo), el código C no muestra que se utilizó un atributo si el porcentaje de muestras de entrenamiento cubierto por las divisiones correspondientes es muy baja. Aquí, el umbral se redujo y el uso fraccional es mostrado. Cuando la métrica es 'Divisiones', el porcentaje de divisiones asociadas a cada predictor es calculada.

Ejemplo `data(churn) treeModel = C5.0(x = churnTrain[, -20], y = churnTrain$churn) C5imp(treeModel) C5imp(treeModel, metric = 'splits')`

Implementación en R Lo primero que debemos hacer es instalar el paquete C50 en Rstudio. Para ello introducimos el siguiente comando:

```
install.packages("C50")
```

Para mostrar el funcionamiento de este paquete vamos a utilizar una Base de Datos que trae por defecto R. Se trata de la Base de Datos Iris que contiene las medidas en centímetros de las variables de longitud y anchura del sépalo y pétalo de 50 flores de cada una de las tres especies de Iris. Las especies son: Iris Setosa, versicolor y virginica.

```
data(iris)
```

```
head(iris)
```

```
> data("iris")
> head(iris)
```

	Sepal.Length	Sepal.width	Petal.Length	Petal.width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa

Con head mostramos los primeros 6 elementos de nuestra Base de Datos.

Con la función STR vamos a conocer un poco más el archivo.

```
> str(iris)
'data.frame': 150 obs. of 5 variables:
 $ Sepal.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.width : num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.width : num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species : Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 1 1 1 1 1 1 ...
```

Contiene 5 variables:

- Longitud Sépalo
- Anchura Sépalo
- Longitud Pétalo
- Anchura Pétalo
- Especies

Las cuatro primeras variables son de tipo número mientras que la última es de tipo texto. Con la función table de R podemos comprobar que la variable Species contiene 3 tipos, con 50 objetos cada una.

```
table(iris$Species)
```

```
> table(iris$Species)
```

setosa	versicolor	virginica
50	50	50

Para poder replicar este análisis y tener los mismo resultados. Aplicamos la función set.seed

```
set.seed(9850)
```

Ahora utilizamos una función de R que sirve para generar números pseudo-aleatorios de forma automática. Esta asociada a la distribución uniforme y funciona dentro del paquete Stats que viene por defecto.

```
g - runif (nrow(iris))
```

Nos creamos una nueva Base de Datos con los datos aleatorios llamada `irisr`. Es básicamente igual que `iris` excepto en los datos que contienen las flores. Ya que van a tener unos datos y un orden basado en los números aleatorios, los cuales han sido generados en la función anterior.

```
irisr - iris [order(g),]
```

Estas tres líneas nos van a garantizar obtener los mismos resultados si volvemos a replicar el análisis

Mostramos los nuevos datos

```
> str(irisr)
'data.frame': 150 obs. of 5 variables:
 $ Sepal.Length: num 6.7 6.4 6.4 6.5 4.7 5.4 6.1 4.9 5 6.7 ...
 $ Sepal.Width : num 3.1 2.7 2.9 3 3.2 3.4 2.8 3.6 3.3 3.1 ...
 $ Petal.Length: num 4.7 5.3 4.3 5.8 1.6 1.5 4 1.4 1.4 4.4 ...
 $ Petal.Width : num 1.5 1.9 1.3 2.2 0.2 0.4 1.3 0.1 0.2 1.4 ...
 $ Species : Factor w/ 3 levels "setosa","versicolor",...: 2 3 2 3 1 1 2 1 1 2 ...
```

Ya estamos preparados para utilizar el análisis C50. Nuestro objetivo es predecir para clasificar las especies de flores.

Con el campo C5.0 indicamos que queremos utilizar el algoritmo del paquete anteriormente descargado.

De la Base de Datos `irisr` indicamos 1:100 ya que queremos analizar los 100 primeros registros de 150. Indicamos -5 seguido del 100 para eliminar el campo 5 (1.Sepal.Length, 2.Sepal.Width, 3.Petal.Length, 4.Petal.Width, 5.Species) que corresponde al campo Species por ser la etiqueta. Seguido indicamos las etiquetas de los predictores: `irisr [1:100,5]`

Por lo que quedará de la siguiente forma:

```
> m1 <- c5.0(irisr[1:100,-5], irisr[1:100,5])
> m1
```

Call:

```
c5.0.default(x = irisr[1:100, -5], y = irisr[1:100, 5])
```

Classification Tree

Number of samples: 100

Number of predictors: 4

Tree size: 3

Non-standard options: attempt to group attributes

En el resultado C.50 nos indica el número de objetos que es 100 junto con el número de predictores y el tamaño del árbol. Para obtener más información del proceso que ha realizado C.50 utilizamos el comando `Summary(m1)`


```

Call:
C5.0.default(x = irisr[1:100, -5], y = irisr[1:100, 5])

Classification Tree
Number of samples: 100
Number of predictors: 4

Tree size: 3

Non-standard options: attempt to group attributes
> summary (m1)

Call:
C5.0.default(x = irisr[1:100, -5], y = irisr[1:100, 5])

C5.0 [Release 2.07 GPL Edition]          Mon May 23 19:06:26 2016
-----

Class specified by attribute `outcome`

Read 100 cases (5 attributes) from undefined.data

Decision tree:

Petal.width <= 0.5: setosa (32)
Petal.width > 0.5:
...Petal.width <= 1.7: versicolor (39/1)
    Petal.width > 1.7: virginica (29/1)

Evaluation on training data (100 cases):

      Decision Tree
      -----
      Size      Errors
      3      2( 2.0%)  <<

```

Con Summary podemos ver el rbol, que nos indica lo siguiente:

Si el ancho del pétalo es menor o igual a 0.5: La predicción es de Setosa de 32 sobre 100.

Si el ancho del pétalo es entre 0.6 y 1.7 entonces la predicción es de Versicolor de (39/1) sobre 100

Si el ancho del pétalo es mayor de 1.7 entonces la predicción es de Virginica de (29/1) sobre 100

Referencias

<https://cran.r-project.org/web/packages/C50/C50.pdf>

Paquete Party El paquete Party es una caja de herramientas computacional para la partición recursiva. El núcleo del paquete es `ctree()`, una implementación de rboles de inferencia condicional que se incrusta en la estructura de rbol mediante modelos de regresión. En una teoría bien definida de condicionales procedimientos de inferencia. La clase no paramétrica de regresión es aplicable a todo

tipo de problemas de regresin, incluyendo nominal, ordinal, numrica, censurados, respuesta multivariante y escalas de medicin arbitraria de las covariables. Basado en rboles de inferencia condicional, `cforest()` proporciona una implementacin de bosques al azar de Breiman. La funcin `mob()` implementa un algoritmo recursivo de particin basada en modelos paramtricos (por ejemplo, modelos lineales, MLG o supervivencia pruebas de inestabilidad de parmetros empleando regresin) para split seleccin. Funcionalidad ampliable para la visualizacin de estructura de rbol.

- Función Conditional Inference Trees Sirve para realizar un particionamiento recursivo para las variables continuas, censuradas, ordenadas, nominales y variables de respuesta multivariable en un marco de inferencia condicional.

Uso

```
ctree(formula, data, subset = NULL, weights = NULL, controls =
ctree_control(), xtrafo = ptrfo, ytrafo = ptrfo, scores = NULL)
```

Argumentos

formula Descripción simbólica del modelo a desarrollar.

data Data frame que contiene las variable en el modelo.

subset Vector opcional que especifica un subconjunto de observaciones para ser usado en el proceso adecuado

weights Vector opcional de pesos que es usado en el proceso de adecuación. Sólo los pesos con valores no negativos son admitidos.

controls Objeto de la clase `TreeControl`, el cual puede ser obtenido usando `ctree_control`

xtrafo Funcin que puede ser aplicada a todas las variables de entrada. Por defecto, la función `ptrfo` es aplicada.

ytrafo Funció para ser aplicada a todas las variables de respuesta. Por defecto la función, `ptrfo` es aplicada.

scores Lista opcional de puntuaciones para atribuir a los factores ordenados.

Detalles Los árboles de inferencia condicional estiman una relación de la regresión por partición recursiva binaria en un marco de inferencia condicional. El algoritmo funciona de la siguiente forma:

1. Testeo de la hipótesis de independencia con valores nulos entre cualquiera de los valores de entrada y respuesta (los cuales pueden ser multivariante también). Se para si no se puede rechazar esta hipótesis. De lo contrario hay que seleccionar la variable de entrada con una asociación más fuerte. Esta asociación es medida por un pvalor correspondiente a una prueba para la hipótesis nula parcial de una sola variable de entrada y respuesta.
2. Poner en práctica una división binaria en la variable de entrada seleccionada.
3. Repetir recursivamente las etapas 1) y 2).

La implementación utiliza un marco unificado para la inferencia condicional, o pruebas de permutación desarrollado por Strasser y Weber (1999). El criterio de parada del paso 1) se basa tanto en la multiplicidad de pvalores ajustados (`TestType == "Bonferroni"` o `TestType == "MonteCarlo"` en los árboles de inferencia condicional `ctree_control`), sobre los valores de p univariantes (`TestType == "univariate"`), o en los valores de la prueba estadística (`TestType == "Teststatistic"`). En ambos casos, el criterio se maximiza, es decir, 1 - pvalue se utiliza. Una división se lleva a cabo cuando el criterio es superior al valor dado por `mincriterion` como se especifica en `ctree_control`. Por ejemplo, cuando `mincriterion = 0.95`, pvalor debe ser menor de 0.05 con el fin de dividir este nodo. Este enfoque estadístico asegura que el tamaño adecuado del árbol está crecido y ninguna forma de poda o la validación cruzada es necesaria. La selección de la variable de entrada para dividir es basada en los valores p univariantes evitando un sesgo de selección de variables hacia las variables de entrada con muchos puntos de corte posibles. El ajuste por multiplicidad de los valores p Monte Carlo se calculan siguiendo un enfoque "min-p". p-valores basados en la distribución límite (chi-cuadrado o normal) se calculan para cada una de las permutaciones aleatorias de los datos. Esto significa que uno debe utilizar una prueba estadística cuadrática cuando hay factores en juego (porque la evaluación de la distribución normal multivariante correspondiente es pérdida de tiempo). De forma predeterminada, las puntuaciones para cada factor ordinal `x` son `1:length(x)`, esto puede modificarse usando `puntuación = lista(x = c(1,5,6))`, por ejemplo. Las predicciones se pueden computar usando `predecir` o `treeresponse`. La primera función acepta argumentos `tipo = c("respuesta", "nodo", "prob")` donde el `tipo = "respuesta"` devuelve medias predictivas, clases predictivas, o tiempos de supervivencia media predicha. `tipo = "nodo"` devuelve nodo terminal IDs (idénticos a `donde`) y `tipo = "prob"` proporciona más información acerca de la distribución condicional de la respuesta, es decir, probabilidades de clase o predicciones de curvas Kaplan-Meier, es idéntico al `treeresponse`. Para observaciones con cero pesos, se calculan las predicciones del árbol equipado cuando `newdata = NULL`. Para una descripción general de la metodología ver Hothorn, Hornik, Zeileis (2006) y Hothorn, Hornik, van de Wiel y Zeileis (2006). Introducción para los principiantes se puede encontrar en Strobl et al (2009).

y en <http://github.com/christophM/overview-ctrees.git>.

Referencias bibliográficas Helmut Strasser y Christian Weber (1999). En la Teora asintótica de la estadística de la permutación.

Métodos matemáticos de estadística, 8, 220–250. Torsten Hothorn, Kurt Hornik, Mark A. van de Wiel y Achim Zeileis (2006).

Un sistema de Lego para la inferencia condicional. El estadístico americano, 60(3), 257-263. Torsten Hothorn, Kurt Hornik y Achim Zeileis (2006).

Particionamiento recursivo imparcial: Marco de inferencia Condicional. Revista de estadística computacional y gráfica, 15.3, 651-674.

Preprint disponible desde <http://statmath.wu-wien.ac.at/zeileis/papers/Hothorn+Hornik+Zeileis-2006.pdf>

Carolin Strobl, Jacobo de Malley y Gerhard Tutz (2009). Una introducción a la partición recursiva: Fundamentos, aplicación y características de clasificación y árboles de regresión, embolsado, y Bosques al azar. Métodos psicológicos, 14(4), 323-348.

Ejemplos `set.seed(290875)`

```
### regression
```

```
airq - subset(airquality, is.na(Ozone))
```

```
airct - ctree(Ozone ~ ., data = airq, controls = ctree_control(maxsurrogate = 3))
```

```
airct
```

```
plot(airct)
```

```
mean((airq$Ozone - predict(airct))^2)
```

```
### extract terminal node ID, two ways all.equal(predict(airct, type = "node"), where(airct))
```

```
### classification irisct - ctree(Species ~ ., data = iris)
```

```
irisct
```

```
plot(irisct)
```

```
table(predict(irisct), iris$Species)
```

```
### estimated class probabilities, a list tr - treeresponse(irisct, newdata = iris[1:10,])
```

```
### ordinal regression
```

```
data('mammoexp', package = 'TH.data')
```

```
mammoct - ctree(ME ~ ., data = mammoexp)
```

```
plot(mammoct)
```

```
### estimated class probabilities
```

```
treeresponse(mammoct, newdata = mammoexp[1:10,])
```

```
### survival analysis
```

```

if (require('TH.data') && require('survival')) data('GBSG2', package =
'TH.data') GBSG2ct - ctree(Surv(time, cens) ~ ., data = GBSG2) plot(GBSG2ct)
treeresponse(GBSG2ct, newdata = GBSG2[1:2,])
### if you are interested in the internals:
### generate doxygen documentation
## Not run:
### download src package into temp dir tmpdir - tmpdir() tgz i- down-
load.packages('party', destdir = tmpdir)[2]
### extract
untar(tgz, exdir = tmpdir) wd - setwd(file.path(tmpdir, 'party'))
### run doxygen (assuming it is there) system('doxygen inst/doxy-
gen.cfg') setwd(wd)
### have fun browseURL(file.path(tmpdir, 'party', 'inst', 'documenta-
tion', 'html', 'index.html'))
## End(Not run)

```

Implementación en R Para implementar Party lo primero que debemos de hacer es instalarlo en RStudio, para ello introducimos el siguiente comando:

```
install.packages("party") library(party)
```

```

> install.packages("party")
warning in install.packages :
  downloaded length 0 != reported length 233479
Installing package into 'C:/Users/asus/Documents/R/win-library/3.2'
(as 'lib' is unspecified)
warning in install.packages :
  package 'party' is not available (for R version 3.2.2)
>
> library(party)
Loading required package: grid
Loading required package: mvtnorm
Loading required package: modeltools
Loading required package: stats4
Loading required package: strucchange
Loading required package: zoo

Attaching package: 'zoo'

The following objects are masked from 'package:base':

```

Una vez descargado el paquete Party procedemos a realizar un árbol de decisión con R en la Base de Datos Iris utilizada anteriormente en otros paquetes.

Si utilizamos el comando `str(iris)` podemos comprobar las características de la misma.

Llamamos a la función `Ctree` de Party en el que le pasamos la variable que queremos analizar, en nuestro caso es `especies` junto a una lista de variables independientes que serán las características de las flores:

- Longitud Sépalo

- Longitud Pétalo
- Anchura Sépalo
- Anchura Pétalo

La función quedaría de la siguiente manera:

```
iris_ctree <- ctree(Species ~ Sepal.Length + Sepal.Width + Petal.Length +
Petal.Width, data=iris)
```

Una vez ejecutada procedemos a visualizar el contenido con la función Print:

```
print(iris_ctree)
```

```
> iris_ctree <- ctree(Species ~ Sepal.Length + Sepal.Width + Petal.Length +
Petal.Width, data=iris)
> print(iris_ctree)

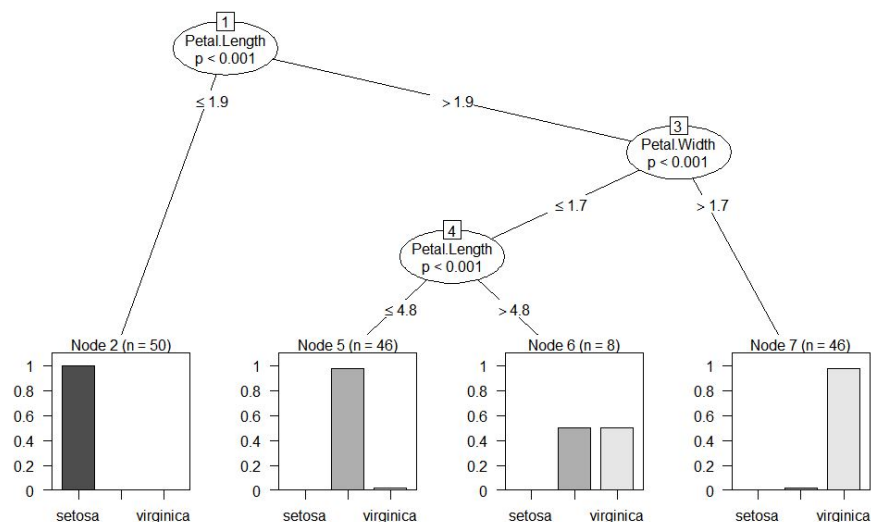
Conditional inference tree with 4 terminal nodes

Response: Species
Inputs: Sepal.Length, Sepal.Width, Petal.Length, Petal.Width
Number of observations: 150

1) Petal.Length <= 1.9; criterion = 1, statistic = 140.264
2)* weights = 50
1) Petal.Length > 1.9
3) Petal.Width <= 1.7; criterion = 1, statistic = 67.894
4) Petal.Length <= 4.8; criterion = 0.999, statistic = 13.865
5)* weights = 46
4) Petal.Length > 4.8
6)* weights = 8
3) Petal.Width > 1.7
7)* weights = 46
> |
```

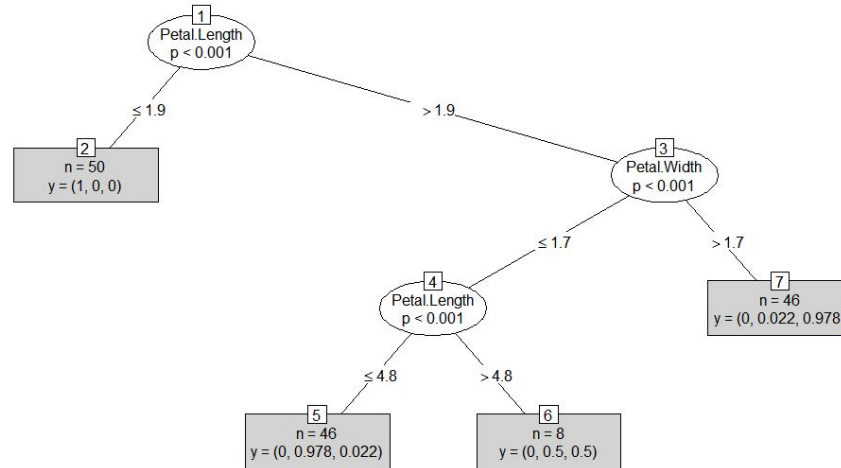
Para obtener los resultados de una forma más visual y cercana podemos mostrarlo en forma de gráfico con el comando por defecto de R Plot:

```
plot(iris_ctree)
```



Incluso de una manera mucho más fácil de interpretar con el siguiente comando:

```
plot(iris_ctree, type='simple')
```



Referencias

<https://cran.rproject.org/web/packages/party/party.pdf>

Paquete Tree El paquete Tree nos permite realizar árboles de clasificación y regresión. Con los árboles de decisión podemos realizar modelos de predicción para poder organizar gráficamente la información sobre las opciones posibles, las consecuencias y el valor final. Gracias a ellos podremos decidir entre diversos cursos de acción. Creando una representación visual de los variados riesgos, las recompensas y los valores potenciales de cada opción.

Las ventajas que vamos a conseguir utilizando Tree son las siguientes:

- Resume los ejemplos de partida, permitiendo la clasificación de nuevos casos siempre y cuando no existan modificaciones sustanciales en las condiciones bajo las cuales se generaron los ejemplos que sirvieron para su construcción.
- Facilita la interpretación de la decisión adoptada.
- Explica el comportamiento respecto a una determinada tarea de decisión.
- Reduce el número de variables independientes.
- Control de la gestión empresarial.
- Tree Sirve para realizar un particionamiento recursivo usando la respuesta de la fórmula especificada y eligiendo las divisiones a partir de los términos de la mano del lado derecho del árbol.

Uso

```
tree(formula, data, weights, subset, na.action = na.pass, control = tree.control(nobs,
...), method = 'recursive.partition', split = c('deviance', 'gini'), model =
FALSE, x = FALSE, y = TRUE, wts = TRUE, ...)
```

Argumentos

formula Una expresión de la fórmula. La mano del lado izquierdo (respuesta) debe ser o bien un vector numérico cuando un árbol de regresión se monta o bien un factor, cuando el árbol de clasificación es producido. La mano del lado derecho debe ser una serie de números o variables de factores separadas por +.

data Un data frame con el cual podemos interpretar la fórmula, los pesos y subgrupo.

weights Vector de ponderaciones de observación no negativos; se permiten pesos fraccionarios.

subset Una expresin que especifica el subconjunto de casos para usar.

na.action Una función para filtrar los datos que faltan desde el marco del modelo. El valor predeterminado es na.pass (No hace nada), como árbol trata los valores que faltan (dejándolos caer del árbol tan lejos como sea posible).

control Una lista que devuelve tree.control.

method Cadena de caracteres que da el método a utilizar. El único otro valor que se puede usar es "model.frame".

split Criterio de reparto par usar.

model Si este argument es en sí mismo un modelo de datos, entonces la fórmula y los argumentos de datos son ignorados, y el modelo es uso para definir el modelo. Si el argumento es lógico y verdadero, el modelo de datos es almacenado como componente del modelo en el resultado.

x Valor lógico. Si es verdad, la matriz de variables para ese caso es devuelta.

y Valor lógico. Si es verdad, la matriz de respuesta es devuelta.

wts Valor lógico. Si es verdad los pesos son devueltos.

Detalles Un árbol se cultiva por particionamiento recursivo binario usando la respuesta en la fórmula especificada y la elección de las divisiones de los términos de la mano del lado derecho. Las variables numéricas se dividen en X_j y $X_{j,a}$; Los niveles de un factor desordenado son divididos en dos grupos no vacíos. La división la cual maximiza la reducción en impureza es elegida, la división del conjunto de datos y el proceso repetido. División continúa hasta que los nodos terminales son demasiado pequeños o demasiado pocos para dividir. El crecimiento de los árboles está limitada a una profundidad de 31 por el uso de número enteros para etiquetar los nodos. El valor de las variables predictoras pueden tener hasta 32 niveles. Este límite se impone por la facilidad del etiquetado, pero desde su uso en árboles de clasificación con tres o más niveles en una respuesta implica una búsqueda a través de $2^{(k-1)}$ agrupaciones para los niveles de K , el límite práctico es mucho menor.

Valores El valor es un objeto de la clase "Tree" el cual tiene componentes.

frame Un data frame con una fila para cada nodo, y row.names dando los números del nodo. Las columnas incluyen var, la variable utilizada en la división (o «hojas» para un nodo terminal), n, el (ponderado) número de casos que llegan a ese nodo, Dev la desviación del nodo, yval, el valor ajustado en el nodo (la media para la regresión de árboles, una clase de la mayoría de los árboles de clasificación) y Split, una matriz de dos columnas de las etiquetas de las divisiones de izquierda y derecha en el nodo. Los árboles de clasificación también tienen yprob, una matriz de probabilidades para cada nivel de respuesta ajustado.

where Un vector de enteros que indica el número de filas de la trama que detalla el nodo al que se asigna cada caso.

terms Los terminos de la fórmula

call La llamada emparejada al árbol

model Si el modelo = Verdadero, es el marco del modelo.

x Si x = Verdadero, la matriz de modelo.

y Si y = Verdadero, la respuesta

wts Si wts es = Verdadero, los pesos

Referencias Bibliográficas L. Breiman, Friedman H. J. Olshen r. A. y piedra, C. J. (1984) Árboles de clasificación y regresión. Wadsworth.
 Ripley, B. D. (1996) Reconocimiento de patrones y redes neuronales. Prensa de la Universidad de Cambridge, Cambridge. Captulo 7

Ejemplos `data(cpus, package='MASS') cpus.ltr - tree(log10(perf) - syct+mmin+mmax+cach+chmin
 cpus) cpus.ltr summary(cpus.ltr) plot(cpus.ltr); text(cpus.ltr) ir.tr - tree(Species
 -, iris) ir.tr summary(ir.tr)`

Implementación en R En primer lugar para poder implementar el árbol de decisión con Tree debemos instalar el paquete en Rstudio. Para ello incluimos el siguiente comando:

`install.packages("tree") library(tree)`

```
> install.packages("tree")
Installing package into 'C:/Users/asus/Documents/R/win-library/3.2'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.2/tree_1.0-37.zip'
Content type 'application/zip' length 120474 bytes (117 KB)
downloaded 117 KB

package 'tree' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
C:/Users/asus/AppData/Local/Temp/RtmpqkppyG/downloaded_packages
> library(tree)
warning message:
package 'tree' was built under R version 3.2.5
```

Una vez instalado procedemos a realizar el árbol de decisión sobre la Base de Datos por defecto de R Iris.

Para ello utilizamos el siguiente comando:

`stree= tree(Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,data=iris)`

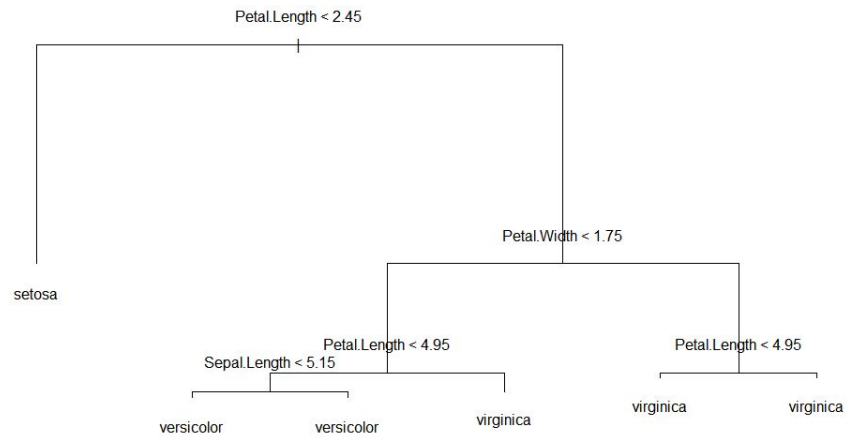
Comprobamos los resultados ofrecidos por Stree

```
> stree
node), split, n, deviance, yval, (yprob)
* denotes terminal node

1) root 150 329.600 setosa ( 0.33333 0.33333 0.33333 )
2) Petal.Length < 2.45 50 0.000 setosa ( 1.00000 0.00000 0.00000 ) *
3) Petal.Length > 2.45 100 138.600 versicolor ( 0.00000 0.50000 0.50000 )
6) Petal.Width < 1.75 54 33.320 versicolor ( 0.00000 0.90741 0.09259 )
12) Petal.Length < 4.95 48 9.721 versicolor ( 0.00000 0.97917 0.02083 )
24) Sepal.Length < 5.15 5 5.004 versicolor ( 0.00000 0.80000 0.20000 )
*
25) Sepal.Length > 5.15 43 0.000 versicolor ( 0.00000 1.00000 0.00000 )
*
13) Petal.Length > 4.95 6 7.638 virginica ( 0.00000 0.33333 0.66667 ) *
7) Petal.Width > 1.75 46 9.635 virginica ( 0.00000 0.02174 0.97826 )
14) Petal.Length < 4.95 6 5.407 virginica ( 0.00000 0.16667 0.83333 ) *
15) Petal.Length > 4.95 40 0.000 virginica ( 0.00000 0.00000 1.00000 ) *
```

Para poder verlos graficamente y as poder comprenderlos de una forma más rápida y eficiente los pintamos mediante el comando Plot.

`plot(stree) text(stree)`



En el podemos comprobar los distintos caminos del árbol según los valores de la longitud del pétalo / sépalo y anchuras de pétalo / sépalo

Referencias

<https://cran.rproject.org/web/packages/tree/tree.pdf>

- Rpart Con Rpart podemos implementar un particionamiento recursivo para la clasificación, regresión y árboles de supervivencia. Esta implementación se basa en el libro de Breiman, Friedman, Olshen and Stone publicado en 1984.

Uso

```
rpart(formula, data, weights, subset, na.action = na.rpart, method, model = FALSE, x = FALSE, y = TRUE, parms, control, cost, ...)
```

Argumentos

formula Es una fórmula, con respuesta pero sin terminos de interacción. Si está en una trama de datos, esta es cogida como el modelo de la estructura.

data Una trama de datos opcional en la cual se interpretan las variables llamadas en la fórmula.

weights Ponderaciones de los pesos opcionales.

subset Es una expresin opcional que indica que sólo un subconjunto de las filas de los datos debe ser utilizado en el ajuste.

na.action La opción por defecto borra todas las acciones por la cual Y se está perdiendo, pero mantiene aquellas en las cuales una o más predictores faltan.

method Uno de 'ANOVA', 'poisson', 'clase' o 'CAD'. Si el método no se encuentra entonces la rutina intenta hacer una sugerencia. Si y es un objeto de supervivencia entonces, método=.^{exp.es} asumido, si y tiene dos columnas entonces el método='Poisson' es asumido, si y es un fator entonces el método =^{class.es} asumido, en otros casos el método='ANOVA' es asumido. Es ms sabio especificar el mtodo directamente,especialmente en el caso de que más criterios puedan ser aadidos a la función en el futuro. Como alternativa, el método puede ser una lista de funciones con nombre init, split y eval. Se dan ejemplos en el archivo 'pruebas / usersplits.R' en las, y en el vietas 'Funciones de Split escritas por el usuario '

model Si es valor lógico: Guardar una copia del modelo de la trama en el resultado?Si el valor de entrada para el modelo es un modelo de datos (probablemente de una llamada previa a la funcin rpart), entonces ese frame se utiliza en lugar de la construcción de nuevos datos.

x Copia de la matriz X en el resultado.

y Para guardar una copia de la variable dependiente en el resultado. Si falta y el modelo es suministrado entonces el valor predeterminado es Falso.

parms ptional parámetros para la función de división. La división ANOVA no tiene parámetros. La división Poison tiene un único parámetro, el coeficiente de variación de la distribución de las tasas previas. El valor por defecto es 1. La división exponencial tiene el mismo parámetro que Poisson. Para la división de clasificación la lista puede contener: Vector de probabilidades previas, matriz de pérdida o el índice de división. Las distribuciones previas deben ser positivas y sumar 1. La matriz de pérdida debe tener ceros en los elementos fuera de la diagonal. El índice de división puede ser Gini o información. Los priores por defecto son proporcionales con los recuentos de datos, el valor por defecto de pérdidas es 1, y la división por defecto de Gini

control Una lista de opciones que controlan detalles del algoritmo rpart.

cost Un vector de costes no negativos, uno para cada variable en el modelo. Por defecto es uno para todas las variables. Estas son escaladas para ser aplicadas al considerar las divisiones, por lo que la mejora en la división de una variable se divide por su coste para decidir la división que se escoge.

Detalles Este difiere en la función `Tree` en S principalmente en su manejo de variables de sustitución. En la mayoría de detalles sigue a Breiman et. al (1984) de muy de cerca. El paquete de R `Tree` proporciona una reimplementación del árbol.

Referencias Bibliográficas Breiman L., J. Friedman H., Olshen R. A., y Stone, C. J. (1984) Árboles de clasificación y regresión. Wadsworth.

Ejemplos `fit - rpart(Kyphosis - Age + Number + Start, data = kyp-
hosis)`

`fit2 - rpart(Kyphosis - Age + Number + Start, data = kyphosis, parms
= list(prior = c(.65,.35), split = "information"))`

`fit3 - rpart(Kyphosis - Age + Number + Start, data = kyphosis, control
= rpart.control(cp = 0.05))`

`par(mfrow = c(1,2), xpd = NA) # otherwise on some devices the text is
clipped`

`plot(fit)`

`text(fit, use.n = TRUE)`

`plot(fit2)`

`text(fit2, use.n = TRUE)`

Implementación en R Para implementar `RPart` lo primero que debemos hacer es descargar e instalar el paquete en Rstudio. Para ello introducimos el siguiente comando:

`install.packages("rpart") library(rpart)`

```
> install.packages("rpart")
Installing package into 'C:/Users/asus/Documents/R/win-library/3.2'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.2/rpart_4.1-10.zip'
Content type 'application/zip' length 921892 bytes (900 KB)
downloaded 900 KB

package 'rpart' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
C:\Users\asus\AppData\Local\Temp\Rtmpcf2uhw\downloaded_packages
> library(rpart)
warning message:
package 'rpart' was built under R version 3.2.5
> |
```

Para realizar el algoritmo introducimos la sentencia. Con este algoritmo vamos a realizar un árbol de decisión para la Base de Datos que viene por defecto `IRIS`.

```
tree - rpart(Species ~ ., data = iris, method = 'class')
```

Una vez introducimos el comando vemos el resultado:

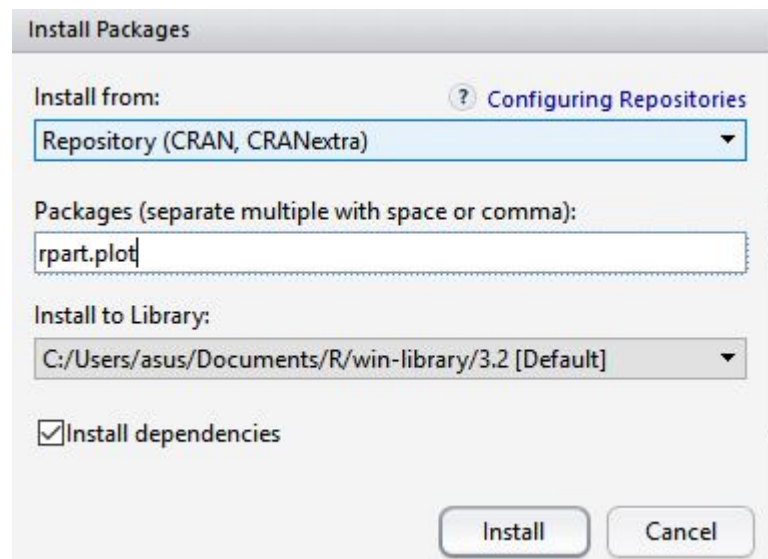
```
> tree
n= 150

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 150 100 setosa (0.3333333 0.3333333 0.3333333)
 2) Petal.Length< 2.45 50 0 setosa (1.0000000 0.0000000 0.0000000) *
 3) Petal.Length>=2.45 100 50 versicolor (0.0000000 0.5000000 0.5000000)
   6) Petal.width< 1.75 54 5 versicolor (0.0000000 0.9074074 0.0925925) *
   7) Petal.width>=1.75 46 1 virginica (0.0000000 0.0217391 0.9782608) *
```

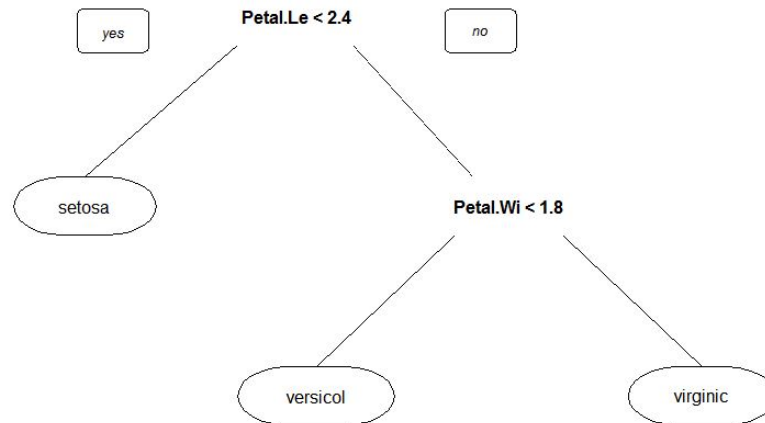
Como podemos comprobar nos indica que el número de elementos en IRIS es de 150. Y según la longitud y anchura del pétalo y sépalo podremos llegar a una de las especies por alguna de las ramas del árbol.

Por último para verlo de una forma más clara, vamos a visualizar el resultado con el paquete `rpart.plot`. Para ello debemos descargarlo e instalarlo en Rstudio:



El resultado de la implementación nos indica:

- Si la longitud del pétalo es menor de 2,4 entonces la especie es Setosa
- Si la longitud el pétalo es mayor de 2,4 entonces si la anchura del pétalo es menor que 1,8 puede ser Versicol o Virginic



Referencias

<https://cran.rproject.org/web/packages/rpart/rpart.pdf>

5.3. Support Vector Machines

Las máquinas de vectores de soporte (SVM) fueron desarrolladas en los años 90 por Vladimir Vapnik y su equipo en los laboratorios AT&T. En un primer lugar las SVM fueron pensadas para resolver problemas de clasificación binaria, pero en estos momentos también se utilizan para resolver otros tipos de problemas como pueden ser (regresión, multclasificación o agrupamiento).

Los campos en los que ha sido utilizado con éxito son tales como:

- Visión artificial
- Reconocimiento de caracteres
- Categorización de texto
- Categorización de hipertexto
- Clasificación de proteínas
- Procesamiento de lenguaje
- Análisis de series temporales

Clasificación es un amplio rango. Podríamos clasificar a las máquinas de vectores de soporte dentro de los clasificadores lineales, debido a que introducen hiperplanos lineales o hiperplanos.

Mientras que la mayoría de los métodos de aprendizaje se centran en minimizar los errores cometidos por el modelo generado a partir de los ejemplos de entrenamiento, el sesgo inductivo asociado a las SVMs radica en la minimización del denominado riesgo estructural. La idea es seleccionar un hiperplano de

separación que equidista de los ejemplos más cercanos a cada clase para de esta forma conseguir un margen máximo a cada lado del hiperplano. Además, a la hora de definir el hiperplano, sólo se consideran los ejemplos de entrenamiento de cada clase que caen justo en la frontera de dichos márgenes. Estos ejemplos reciben el nombre de vectores de soporte. Desde un punto de vista práctico, el hiperplano separador de margen máximo ha demostrado tener una buena capacidad de generalización, evitando en gran medida el problema de sobreajuste a los ejemplos de entrenamiento.

Desde un punto de vista algorítmico, el problema de optimización cuadrático con restricciones lineales que puede ser resuelto mediante técnicas estándar de programación cuadrática. La propiedad de convexidad exigida para su resolución garantizan una solución única, en contraste con la no unicidad de la solución producida por una red neuronal artificial entrenada con un mismo conjunto de ejemplos.

Paquete e1071 Este paquete permite realizar funciones para análisis de clases latentes, Transformación de Fourier en escaso tiempo, máquinas de vector de soporte, camino más corto en computación, clusters, Clasificación de Bayes, etc.

- SVM SVM se utiliza para entrenamiento con máquina de vectores de soporte. Se puede utilizar para llevar a cabo la regresión general y clasificación, así como de estimación de densidad.

Uso

```
## S3 method for class 'formula' svm(formula, data = NULL, ..., subset, na.action = na.omit, scale = TRUE)
```

```
## Default S3 method: svm(x, y = NULL, scale = TRUE, type = NULL, kernel = "radial", degree = 3, gamma = if (is.vector(x)) 1 else 1 / ncol(x), coef0 = 0, cost = 1, nu = 0.5, class.weights = NULL, csize = 40, tolerance = 0.001, epsilon = 0.1, shrinking = TRUE, cross = 0, probability = FALSE, fitted = TRUE, ..., subset, na.action = na.omit)
```

Argumentos

formula Descripción del modelo que va a ser desarrollado.

data Trama de datos opcional que contiene las variables en el modelo. Por defecto, las variables se toman del medio ambiente del cual 'svm' es llamado.

x Matriz de datos, un vector o una matriz dispersa

y Vector de respuesta con una etiqueta por cada fila/componente de **x**. Puede ser tanto un factor (para tareas de clasificación) como un vector numérico (para regresión)

scale Vector lógico que indican las variables que van a ser escaladas. Si la escala es de longitud 1, el valor se recicla tantas veces como sea necesario. Por defecto, los datos se escalan internamente (tanto **x** como **y**) a media cero y unidad de varianza.

type SVM puede utilizarse como una máquina de clasificación, como una máquina de regresión, o para detección de novedad. Dependiendo de si **y** es un factor o no, la configuración predeterminada para el tipo es C-clasificación o eps-regresión, respectivamente, pero puede ser sobrescrito por el establecimiento de un valor explícito.

Opciones válidas son:

- C-classification
- nu-classification
- one-classification
- eps-regression
- nu-regression

Kernel El kernel es utilizado en la formación y la predicción

degree Parámetro necesitado por el kernel. Es de tipo polinomial y por defecto 3

gamma Parámetro necesitado para todos los kernels excepto para el lineal. Por defecto: $(1/\text{dimensión de los datos})$

coef0 Parámetro necesario para los kernels de tipo polinómicos. (Por defecto: 0)

cost Es la 'C' constante de la regularización

nu Parámetro necesitado para nu-classification, nu-regression, and one-classification

class.weights Vector de pesos para las diferentes clases, usada para tamaños de clases asimétricas. No todos los niveles de los factores tienen que ser suministrados (Por defecto: 1). Todos los componentes tienen que ser identificados.

cache_size Memoria caché en MB (defecto:40)

tolerance La tolerancia del criterio de terminación (por defecto: 0.001)

epsilon Épsilon en la función de pérdida insensible (por defecto: 0,1)

shrinking Opción por si desea utilizar la función-heurística (por defecto: verdadero)

cross Si se especifica un valor entero $k \geq 0$, una validación cruzada k en la formación de datos es realizada para evaluar la calidad del modelo: la tasa de precisión para la clasificación y el error cuadrático medio para la regresión.

fitted Lógica que indica si los valores ajustados deben ser calculados e incluidos en el modelo o no (por defecto: Verdadero)

probability Lógica que indica si el modelo debe permitir predicciones de probabilidad.

subset Vector de índice que especifica los casos para ser usados en la muestra de entrenamiento.

na.action Función para especificar que se adopte la medida si se encuentran NAs. La acción por defecto es `na.omit`, que conduce al rechazo de los casos con valores perdidos en cualquier variable requerida. Una alternativa es `na.fail`, que causa un error si se encuentran casos de NA.

Detalles Para clasificación multiclase con k niveles, $k \geq 2$, `libsvm` utiliza el enfoque 'uno contra uno' en el cual $k(k-1)/2$ clasificadores son entrenados; la clase apropiada es considerada por un esquema de votación.

`Libsvm` internamente utiliza una representación de datos dispersos, que también son de alto nivel y soportado por el paquete `SparseM`.

Si las variables de predicción incluyen factores, la interfaz de fórmula debe ser utilizada para obtener un modelo de matriz correcta.

`plot.svm` permite una visualización gráfica simple de los modelos de clasificación.

El modelo de probabilidad para la clasificación se ajusta a una distribución logística utilizando la máxima verosimilitud a los valores de decisión de todos los clasificadores binarios, y calcula las probabilidades de la clase a-posteriori de el problema multi-clase mediante la optimización cuadrática. El modelo de regresión probabilística asume errores de Laplace para las predicciones, y estima el parámetro de escala utilizando máxima verosimilitud.

Valores

Objeto de la clase "SVM" que contiene el modelo ajustado, incluyendo:

SV Vectores de soporte resultantes

index El índice de los vectores de soporte resultantes en la matriz de datos. Tenga en cuenta que este índice se refiere a los datos preprocesados (después del posible efecto de na.omit)

coefs Los coeficientes correspondientes a las etiquetas de entrenamiento.

rho El intercepto negativo.

sigma En el caso de un modelo de regresión probabilística, el parámetro de escala de la hipótesis de distribución laplace estimado por máxima verosimilitud.

probA, probB Vectores numéricos de longitud $k(k-1)/2$, el número de clases k , que contienen los parámetros de la distribución logística ajustados a los valores de decisión de los clasificadores binarios ($1 / (1 + \exp(a + b x))$).

Autores David Meyer (based on C/C++ code by Chih-Chung Chang and Chih-Jen Lin) David.Meyer@Rproject.org

Referencias bibliográficas

- Chang, Chih-Chung and Lin, Chih-Jen: LIBSVM: a library for Support Vector Machines <http://www.csie.ntu.edu.tw/~cjlin/libsvm> 52 svm
- Formularios exactos de modelos, algoritmos, etc. Pueden ser encontrados en: Chang, Chih-Chung and Lin, Chih-Jen: LIBSVM: a library for Support Vector Machines <http://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.ps.gz>
- Más detalles de implementación se pueden encontrar en: Rong-En Fan and Pai-Hsueh Chen and Chih-Jen Lin: Working Set Selection Using the Second Order Information for Training SVM <http://www.csie.ntu.edu.tw/~cjlin/papers/quadworkset.pdf>

Ejemplos `data(iris)`
`attach(iris)`
`## classification mode`
`# default with factor response:`
`model - svm(Species ~ ., data = iris)`
`# alternatively the traditional interface:`

```
x - subset(iris, select = Species)
y - Species
model - svm(x, y)
print(model)
summary(model)
# test with train data
pred - predict(model, x)
# (same as:)
pred - fitted(model)
# Check accuracy:
table(pred, y)
# compute decision values and probabilities:
pred - predict(model, x, decision.values = TRUE)
attr(pred, 'decision.values')[1:4,]
# visualize (classes by color, SV by crosses):
plot(cmdscale(dist(iris[,5])),
col = as.integer(iris[,5]), pch = c('o','+')[1:150 %in% model$index + 1])
### try regression mode on two dimensions
# create data
x - seq(0.1, 5, by = 0.05)
y - log(x) + rnorm(x, sd = 0.2)
# estimate model and predict input values
m - svm(x, y)
new - predict(m, x)
# visualize
plot(x, y)
points(x, log(x), col = 2)
points(x, new, col = 4)
### density-estimation
# create 2-dim. normal with rho=0:
X - data.frame(a = rnorm(1000), b =
rnorm(1000))
attach(X)
# traditional way:
m - svm(X, gamma = 0.1)
# formula interface:
m - svm(., data = X, gamma = 0.1)
# or:
```

```

m - svm(- a + b, gamma = 0.1)
# test:
newdata - data.frame(a = c(0, 4), b = c(0, 4))
predict (m, newdata)
# visualize:
plot(X, col = 1:1000 %in % m$index + 1, xlim = c(-5,5), ylim=c(-5,5))
points(newdata, pch = '+', col = 2, cex = 5) # weights: (example not
particularly sensible)
i2 ¡- iris
levels(i2$Species)[3] - 'versicolor'
summary(i2$Species)
wts - 100 / table(i2$Species)
wts
m - svm(Species - ., data = i2,
class.weights = wts)

```

Implementación en R En primer lugar debemos de descargar el paquete e1071 a Rstudio.

Para ello utilizamos el siguiente comando: `install.packages('e1071')`

Una vez instalamos lo arrancamos para poder empezar a utilizarlo con el comando:

```
library (e1071)
```

```

> install.packages("e1071")
Installing package into 'C:/Users/asus/Documents/R/win-library/3.2'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.2/e1071_1.6-7.zip'
Content type 'application/zip' length 806889 bytes (787 KB)
downloaded 787 KB

package 'e1071' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
  C:\Users\asus\AppData\Local\Temp\RtmpI7VGE6\downloaded_packages
> library(e1071)
Warning message:
package 'e1071' was built under R version 3.2.5
> |

```

Vamos a realizar el algoritmo del paquete con una Base de Datos de código abierto, que cuenta con la longitud y anchura de los pétalos y sépalos de un conjunto de especies de flores.

Con el comando `head` mostramos los cinco primeros elementos para comprobar las columnas con la que cuenta la Base de Datos y el tipo de dato de cada una de las variables.

```
> head(iris,5)
  Sepal.Length Sepal.width Petal.Length Petal.width Species
1          5.1          3.5          1.4          0.2  setosa
2          4.9          3.0          1.4          0.2  setosa
3          4.7          3.2          1.3          0.2  setosa
4          4.6          3.1          1.5          0.2  setosa
5          5.0          3.6          1.4          0.2  setosa
> |
```

El siguiente paso es crear el modelo SVM y mostrar los resultados:

Para realizar el modelo debemos introducir la función SVM

```
svm_model - svm(Species ~ ., data=iris)
```

En la que le pasamos las clases de los datos Iris.

Por último mostramos los resultados obtenido con el comando Summary

```
summary(svm_model)
```

```
> svm_model <- svm(Species ~ ., data=iris)
> summary(svm_model)
```

```
Call:
svm(formula = Species ~ ., data = iris)
```

```
Parameters:
  SVM-Type:  C-classification
  SVM-kernel: radial
    cost:  1
  gamma:  0.25
```

```
Number of support vectors:  51
```

```
( 8 22 21 )
```

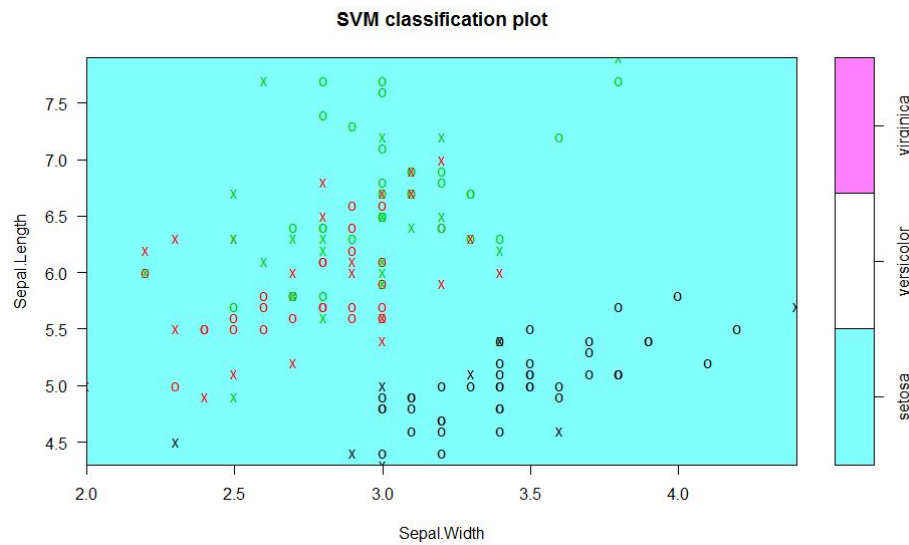
```
Number of classes:  3
```

```
Levels:
  setosa versicolor virginica
```

Para ver los resultados más claros lo mostramos en forma de gráfica.

Nota: En la gráfica sólo podemos mostrar dos predictores por lo que seleccionamos anchura y longitud del Sépalo.

```
plot(svm_model,iris, Sepal.Length ~ Sepal.Width)
```



Referencias

<https://cran.r-project.org/web/packages/e1071/e1071.pdf>

Paquete Kernlab Kernlab incluye métodos de aprendizaje automático basado en kernel para clasificación, regresión, clustering, detección de novedades, regresión cuantil y reducción de dimensionalidad. Incluye Support Vector Machines(SVM), Kernel PCA, Procesos Gaussianos y un solucionador de QP.

- KSVM Support Vector Machine es una excelente herramienta para la clasificación, detección de la novedad, y regresión.

KSVM tiene soporte para el conocido C-csv, nu-svc (para clasificación), clase única SVC para novedades, eps-svr y nu-svr para regresión.

KSVM también es compatible con la salida de la clase de probabilidades y los intervalos de confianza para la regresión.

Uso

```
## S4 method for signature 'formula' ksvm(x, data = NULL, ..., subset,
na.action = na.omit, scaled = TRUE) ## S4 method for signature 'vector'
ksvm(x, ...) ## S4 method for signature 'matrix' ksvm(x, y = NULL,
scaled = TRUE, type = NULL, kernel = 'rbfdot', kpar = 'automatic', C =
1, nu = 0.2, epsilon = 0.1, prob.model = FALSE, class.weights = NULL,
cross = 0, fit = TRUE, cache = 40, tol = 0.001, shrinking = TRUE, ...,
subset, na.action = na.omit)
```

```
## S4 method for signature 'kernelMatrix' ksvm(x, y = NULL, type =
NULL, C = 1, nu = 0.2, epsilon = 0.1, prob.model = FALSE, class.weights
= NULL, cross = 0, fit = TRUE, cache = 40, tol = 0.001, shrinking =
TRUE, ...)
```

```
## S4 method for signature 'list' ksvm(x, y = NULL, type = NULL,
kernel = "stringdot", kpar = list(length = 4, lambda = 0.5), C = 1, nu =
0.2, epsilon = 0.1, prob.model = FALSE, class.weights = NULL, cross =
0, fit = TRUE, cache = 40, tol = 0.001, shrinking = TRUE, ..., na.action
= na.omit)
```

Argumentos

x Descripción simbólica del modelo a desarrollar. Cuando no se use la fórmula, **x** puede ser una matriz o un vector que contenga los datos de entrenamiento o una matriz kernel de la clase `kernelMatrix` de los datos de entrenamiento o una lista de caracteres.

Nota: Tenga en cuenta, que la intersección se excluye siempre, ya sea dada en la fórmula o no

data Es un data frame opcional que contiene los datos de entrenamiento cuando se usa la fórmula. Por defecto, los datos son extraídos del entorno por el cual KSVM es llamado.

y Vector de respuesta con una etiqueta para cada fila / componente de **x**. Puede ser un factor (para tareas de clasificación) o un vector numérico (para la regresión).

scaled Vector lógico que indica las variables a ser escaladas.

Si el escalado es de longitud 1, el valor es utilizado tantas veces como sea necesario y todas las variables no binarias se escalan. Por defecto, los datos se ajustan a escala interna (ambas variables **x** e **y**). Los valores del centro y de escala se devuelven y se utilizan para las predicciones posteriores

type KSVM se puede utilizar para clasificación, para la regresión, o para la detección de la novedad. Dependiendo de si **Y** es un factor o no. La configuración por defecto para el tipo es C-SVC o eps-SVR, respectivamente, pero se puede sobrescribir mediante el establecimiento de un valor explícito. Las opciones válidas son:

- C-svc C
- Nu-sv
- C-bsvc
- Spoc-svc
- Kbb-svc
- One-svc

- Eps-svr
- Nu-svr
- Eps-bsvr

Kernel La función Kernel usada en entrenamiento y predicción. Este parámetro se puede ajustar a cualquier función de las clases de Kernel.

Kernlab proporciona las funciones Kernel más populares las cuales se pueden utilizar con parámetros Kernel como las de esta lista:

- Rbfdot - Kernel de Base Radial 'Gausiano'
- Polydot - Kernel polinómico
- Vanilladot - Kernel lineal
- tanhdot - Tangente hiperbólica
- laplacedot - Kernel Laplacian
- besseldot - Kernel de Bessel
- anovadot - ANOVA RBF kernel
- splinedot - Kernel de ranura
- stringdot - Kernel de cadena de texto

Al establecer el parámetro del kernel para 'matriz', x trata de llamar a la interfaz kernelMatrix.

El parámetro de Kernel puede también ajustarse a una función definida de una clase Kernel mediante el pase del nombre de la función como argumento.

kpar La lista de hiper-parámetros. Esta es una lista que contiene los parámetros que van a ser usados en la función Kernel.

C Coste de incumplir las limitaciones (por defecto: 1). Esta es la 'C' constante del término de regularización en la formulación de Lagrange.

nu Parámetro necesario para nu-SVC, de un SVC, y nu-SVR. El parámetro nu establece el límite superior en el error de entrenamiento y el límite inferior de la fracción de puntos de datos para convertirse en vectores soporte (por defecto: 0,2).

epsilon Épsilon en la función de pérdida. Utilizado para eps-SVR, nu-SVR y el EPS-BSVM (por defecto: 0,1)

prob.model Si es verdadero construye un modelo para el cálculo de las probabilidades de clase o en el caso de la regresión, calcula el parámetro de escala de la distribución de Laplace compatible con los residuos. Montaje se realiza en los datos de salida creados mediante la realización de 3 veces la validación cruzada de los datos de entrenamiento. Para más detalles ver referencias. (Por defecto: FALSO)

class.weights Vector llamado de pesos para las diferentes clases, que se utiliza para el tamaño de las clases asimétricas. No todos los niveles de los factores tienen que ser suministrados (peso por defecto: 1). Todos los componentes tienen que ser llamados.

cache Memoria caché en MB (Por defecto:40)

tol Tolerancia del criterio de terminación (por defecto: 0.001)

shrinking Opción por si desea utilizar métodos heurísticos (por defecto: Verdadero)

cross Si el valor entero de K es $K \geq 0$, una validación cruzada k-veces en los datos de entrenamiento se lleva a cabo para evaluar la calidad del modelo: la tasa de precisión para la clasificación y el error cuadrático medio para la regresión

fit Indica si los valores ajustados deben ser calculados e incluidos en el modelo o no (por defecto: Verdadero)

subset Un vector de índice para especificar los casos que se utiliza en la muestra de entrenamiento. (NOTA: Si se da, este argumento debe ser nombrado.)

na.action Función para especificar la acción a tomar si NAs son encontrados. La acción por defecto es na.omit.

Detalles KSVM utiliza el algoritmo SMO de John Platt para resolver el problema SVM QP y la mayoría de las formulaciones de SVM.

En la SPOC-SVC, KBB-SVC, C-bsvc y el EPS-bsvr formulaciones se utiliza un algoritmo de fragmentación basado en el solucionador de TRON QP.

Para clasificación-multiclase con las clases k_1, k_2, \dots, k_m , ksvm utiliza el 'enfoque de uno contra uno', en el que $k(k-1) / 2$ clasificadores binarios están capacitados; la clase apropiada es considerada por un esquema de votación; la clase apropiada es considerada por un esquema de votación, SPOC-SVC y las formulaciones KBB-SVC se ocupan de los problemas de clasificación multiclas. Mediante la resolución de un problema único cuadrático incorporando a todas las clases.

Si las variables de predicción incluyen factores, la interfaz de fórmula debe ser utilizada para obtener una matriz de modelo correcto. En clasificación cuando prob.model es Verdadero una validación cruzada se realiza 3 veces sobre los datos y una función sigmoide es utilizada en la decisión resultante de valores f .

Los datos se pueden pasar a la función KSVM en una matriz o una hoja de datos, además KSVM también es compatible con la entrada en forma de una matriz de núcleo de la clase `kernelMatrix` o como una lista de vectores de caracteres, donde un núcleo de cadena de caracteres tenga que usarse.

La función `plot` para clasificación binaria `ksvm`, muestra un gráfico de contorno de los valores de decisión con los vectores de soporte correspondientes resaltados.

La función de predicción puede devolver probabilidades de clases para problemas de clasificación mediante el ajuste del tipo de parámetro a `'probabilidades'`.

El problema de la selección del modelo se aborda parcialmente por una observación empírica de los Kernel (RBF de Gauss, Laplace), donde los valores óptimos de la anchura de sigma son mostrados entre el cuantil 0.1 y 0.9.

Cuando se utiliza un kernel RBF y el establecimiento `kPar` `“automático”`, SVM utiliza la función `Sigest` para estimar los cuantiles y utiliza la mediana de los valores.

Valores

alpha El resultado del vector de soporte

alphaindex El índice de los vectores de soporte resultantes en la matriz de datos. Tenga en cuenta que este índice hace referencia a datos pre-procesados.

coef Veces que los coeficientes corresponden a las etiquetas de entrenamiento

b Intercepto negativo

nSV Número de vectores de soporte

obj El valor de la función objetivo

error Error de entrenamiento

cross Error de validación cruzada

prob.model Contiene la anchura del ajuste Laplaciano.

Autores Alexandros Karatzoglou (SMO optimizers in C++ by Chih-Chung Chang & ChihJen Lin) jalexandros.karatzoglou@ci.tuwien.ac.at

Referencias bibliográficas

- Chang Chih-Chung, Lin Chih-Jen LIBSVM: a library for Support Vector Machines <http://www.csie.ntu.edu.tw/~cjlin/libsvm> Chih-Wei Hsu, Chih-Jen Lin BSVM <http://www.csie.ntu.edu.tw/~cjlin/bsvm/ksvm> 59
- J. Platt Probabilistic outputs for support vector machines and comparison to regularized likelihood methods
- Advances in Large Margin Classifiers, A. Smola, P. Bartlett, B. Schoelkopf and D. Schuurmans, Eds. Cambridge, MA: MIT Press, 2000. <http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.41.1639>
- H.-T. Lin, C.-J. Lin and R. C. Weng A note on Platts probabilistic outputs for support vector machines <http://www.csie.ntu.edu.tw/~htlin/paper/doc/plattprob.pdf>
- C.-W. Hsu and C.-J. Lin A comparison on methods for multi-class support vector machines IEEE Transactions on Neural Networks, 13(2002) 415-425. <http://www.csie.ntu.edu.tw/~cjlin/papers/multisvm.ps.gz>
- K. Crammer, Y. Singer On the learnability and design of output codes for multiclass problems Computational Learning Theory, 35-46, 2000. <http://webee.technion.ac.il/people/koby/publications/ecoc-mlj02.pdf>
- J. Weston, C. Watkins Multi-class support vector machines In M. Verleysen, Proceedings of ESANN99 Brussels, 1999 <http://citeseer.ist.psu.edu/8884.html>

Ejemplos ## simple example using the spam data set data(spam)

create test and training set

index - sample(1:dim(spam)[1])

spamtrain - spam[index[1:floor(dim(spam)[1]/2)],]

spamtest - spam[index[((ceiling(dim(spam)[1]/2)) + 1):dim(spam)[1]],]

train a support vector machine

filter - ksvm(type=.,data=spamtrain,kernel='rbfdot',kpar=list(sigma=0.05),C=5,cross=3)

filter

predict mail type on the test set

mailtype - predict(filter,spamtest[,58])

Check results

table(mailtype,spamtest[,58])

Another example with the famous iris data

data(iris)

Create a kernel function using the build in rbfdot function

rbf - rbfdot(sigma=0.1)

```

rbf
## train a bound constraint support vector machine
irismodel - ksvm(Species~.,data=iris,type='C-bsvc', kernel=rbf,C=10,prob.model=TRUE)
irismodel
## get fitted values
fitted(irismodel)
## Test on the training set with probabilities as output
predict(irismodel, iris[,-5], type='probabilities')
## Demo of the plot function
x - rbind(matrix(rnorm(120),,2),matrix(rnorm(120,mean=3),,2))
y - matrix(c(rep(1,60),rep(-1,60)))
svp - ksvm(x,y,type='C-svc')
plot(svp,data=x)
### Use kernelMatrix
K - as.kernelMatrix(crossprod(t(x)))
svp2 - ksvm(K, y, type='C-svc')
svp2
# test data
xtest - rbind(matrix(rnorm(20),,2),matrix(rnorm(20,mean=3),,2))
# test kernel matrix i.e. inner/kernel product of test data with
# Support Vectors
Ktest - as.kernelMatrix(crossprod(t(xtest),t(x[SVindex(svp2), ])))
predict(svp2, Ktest)
#### Use custom kernel
k - function(x,y) (sum(xy) +1)exp(-0.001*sum((x-y)^2))
class(k) j- 'kernel'
data(promotergene)
## train svm using custom kernel
gene - ksvm(Class~.,data=promotergene[c(1:20, 80:100),],kernel=k,gene
##### Use text with string kernels
data(reuters)
is(reuters)
tsv - ksvm(reuters,rlabels,kernel='stringdot', kpar=list(length=5),cross=3,C=10)
tsv
## regression
# create data
x j- seq(-20,20,0.1)
y j- sin(x)/x + rnorm(401,sd=0.03)

```

```
# train support vector machine
regm - ksvm(x,y,epsilon=0.01,kpar=list(sigma=16),cross=3) plot(x,y,type='l')
lines(x,predict(regm,x),col='red')
```

Implementación en R Lo primero que debemos hacer es instalar Kernlab en RStudio. Para ello utilizamos el siguiente comando:

```
install.packages('kernlab')
```

Una vez instalado lo ejecutamos con el comando library para poder empezar a utilizarlo como vemos en la siguiente imagen:

```
> install.packages("kernlab")
Installing package into 'C:/Users/asus/documents/R/win-library/3.2'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.2/kernlab_0.9-24.zip'
Content type 'application/zip' length 2060852 bytes (2.0 MB)
downloaded 2.0 MB

package 'kernlab' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
C:\Users\asus\AppData\Local\Temp\RtmpI7VGE6\downloaded_packages
> library(kernlab)
warning message:
package 'kernlab' was built under R version 3.2.4
> |
```

Vamos a realizar el algoritmo SVM sobre la Base de Datos de código libre de pétalos y sépalos de flores que trae por defecto R.

Para ello vamos a utilizar la función Kernel rbfot basado en Kernel Gaussiano.

```
rbf - rbfdot(sigma=0.1) rbf
> rbf <- rbfdot(sigma=0.1)
> rbf
Gaussian Radial Basis kernel function.
Hyperparameter : sigma = 0.1
> |
```

Una vez realizado ese paso le indicamos todos los datos que nos pide la función: Type, tipo de kernel, C y prob.model

```
> irismodel
Support Vector Machine object of class "ksvm"

SV type: C-bsvc (classification)
parameter : cost C = 10

Gaussian Radial Basis kernel function.
Hyperparameter : sigma = 0.1

Number of Support vectors : 32

Objective Function value : -5.8442 -3.0652 -136.9786
Training error : 0.02
Probability model included.
> |
```

Referencias

<https://cran.r-project.org/web/packages/kernlab/kernlab.pdf>

Paquete klaR klaR se compone de un conjunto de funciones auxiliares para la clasificación y visualización desarrolladas en la facultad de estadística de la Universidad técnica de Dortmund

■ svmlight

Función de clasificación R. La clasificación de grupo múltiple se realiza con la partición de uno contra el resto de los datos.

Uso

```
svmlight(x, ...) ## Default S3 method:
svmlight(x, grouping, temp.dir = NULL, pathsvm = NULL, del = TRUE,
type = 'C', class.type = .oaa", svm.options = NULL, prior = NULL, out
= FALSE, ...)
## S3 method for class 'data.frame'
svmlight(x, ...) ## S3 method for class 'matrix'
svmlight(x, grouping, ..., subset, na.action = na.fail)
## S3 method for class 'formula'
svmlight(formula, data = NULL, ..., subset, na.action = na.fail)
```

Argumentos

x Matriz o trama de datos que contiene las variables explicativas

grouping Factor de especificación de la clase para cada observación

formula Fórmula que indica la forma de los grupos - $x_1 + x_2 + \dots$. Es decir, la respuesta es la agrupación de los factores y la derecha especifica los discriminadores (no atribuibles a factores).

data Trama de datos a partir de la cual las variables especificadas en la fórmula son preferentes para ser tomadas

temp.dir Directorio de archivos temporales

pathsvm Ruta de acceso a los archivos binarios SVMlight (obligatorio, si la ruta es desconocida por el sistema operativo).

del Variable de tipo lógico: Para eliminar archivos temporales.

type Para realizar C- Clasificación o R- Regresión

class.type Esquema a utilizar en la multiclase

svm.options Parámetros opcionales para usar en SVMlight

prior Probabilidades a priori de clases.

out Variable de tipo lógico: Para decidir si la salida SVMlight debe estar impresa en la consola (sólo para Windows)

subset Un vector de índice de la especificación de los casos que se utiliza en la muestra de entrenamiento.

na.action Especificación de la acción a realizar en el caso de encontrar NAs. La acción por defecto es para que el procedimiento falle. Se puede utilizar na.omit, lo cual sirve para rechazar los valores en los que se encuentre NAs.

Detalles Función para llamar SVMlight desde R para clasificación (Tipo='C').

SVMlight es una implementación de la máquina de vectores de soporte Vapnik. Está escrito en C por Thorsten Joachims.

En la página de inicio (véase más adelante) el código fuente y los binarios para SVMlight están disponibles.

Si no se da m—as que dos clases de SVM entonces se resuelve por el esquema de uno contra todos (class.type = "AA"). Eso significa que cada clase está resuelta contra las otras clases (K-1). La clase con la función de decisión m—as alta de la SVM gana. Por lo que K SVM tiene que ser aprendida. Si class.type = "AO" cada clase se comprueba frente a todas las demás y la última clase es la elegida por mayoría de votos. Si type = R, entonces una regresión SVM se lleva a cabo.

Autores Karsten Luebke, karsten.luebkefom.de, Andrea Preusser

Referencias bibliográficas <http://svmlight.joachims.org/>

Ejemplos `## Only works if the svmLight binaries are in the path.`

```
data(iris)
```

```
x - svmLight(Species ~ ., data = iris)
```

```
## Using RBF-Kernel with gamma=0.1:
```

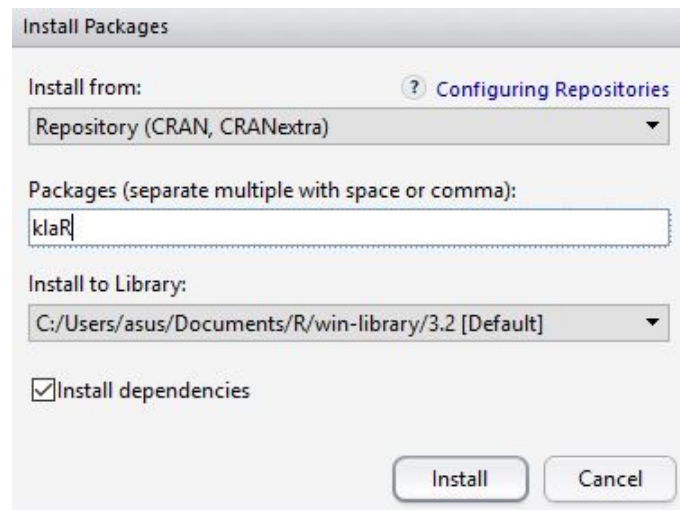
```
data(B3)
```

```
x - svmLight(PHASEN ~ ., data = B3, svm.options = '-t 2 -g 0.1')
```

```
## End(Not run)
```

Implementación en R Lo primero que debemos de realizar es instalar nuestro paquete `klaR` en Rstudio.

Para ello debemos de ir a herramientas - Instalar paquetes



Una vez descargado e instalado procedemos a realizar el algoritmo `svmLight` sobre una Base de Datos de código de abierto que tiene R llamada Iris.

```
> library(klaR)
Loading required package: MASS
Warning message:
package 'klaR' was built under R version 3.2.5
> data("iris")
> iris
```

	Sepal.Length	Sepal.width	Petal.Length	Petal.width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa
9	4.4	2.9	1.4	0.2	setosa
10	4.9	3.1	1.5	0.1	setosa

Como podemos comprobar en la imagen. Esta Base de Datos contiene los campos de longitus y anchura tanto del sépalos como del pétalo de un conjunto de especies de flores.

Por último para desarrollar el algoritmo deberíamos introducir la siguiente sentencia:

```
x - svmlight(Species - ., data = iris)
```

Con el comando summary (x) podremos obtener mucha más información sobre el resultado.

Nota: En la librería klaR no viene incluido el algoritmo SVM Light por lo que debes entrar en:

<http://svmlight.joachims.org/>

Desde esa web debes desarcargarte los binarios para poder realizar el algoritmo.

Una vez descargado el los binarios, te guían por una serie de pasos para poder realizarlo.













Para ello debes abrir CMD y crearte una carpeta con el nombre SVM_light con el comando mkdir

```
mkdir svm_light
```

Una vez creada debes dejar los binarios de SVMLight en dicha carpeta y descromprimirla

```
gunzip -c svm_light.tar.gz — tar xvf -
```

Una vez realizado los pasos ejecutas y creas los dos archivos ejecutables svm_learn (learning module) svm_classify (classification module)

Nombre	Fecha de modifica...	Tipo	Tamaño
 kernel	14/05/2004 18:48	C/C++ Header	2 KB
 LICENSE	14/05/2004 18:48	Documento de tex...	3 KB
 Makefile	08/10/2008 21:38	Archivo	4 KB
 svm_classify	08/10/2008 21:05	C Source	7 KB
 svm_common	08/10/2008 23:00	C Source	25 KB
 svm_common	08/10/2008 22:34	C/C++ Header	13 KB
 svm_hideo	03/09/2004 22:00	C Source	29 KB
 svm_learn	27/08/2004 23:56	C Source	133 KB
 svm_learn	16/07/2004 17:38	C/C++ Header	9 KB
 svm_learn_main	08/10/2008 22:51	C Source	18 KB
 svm_light.tar	18/06/2016 16:17	Archivo WinRAR	50 KB
 svm_loqo	14/07/2004 23:45	C Source	7 KB

Capítulo 6

Conclusiones

Con este Proyecto de Fin de Grado nos hemos sumergido en los paquetes de clasificación supervisada de R.

R es uno de los grandes lenguajes de análisis de datos, muchos profesionales del sector lo utilizan por ser un lenguaje de código abierto con una gran comunidad de desarrollo a sus espaldas.

La mayoría de desarrolladores destacan su apartado de visualizaciones, por la alta calidad de paquetes que la conforman. Grandes empresas como Google o Facebook utilizan las gráficas conformadas por R para entender mejor los datos extraídos de sus sistemas.

R cuenta también con un reconocido prestigio dentro del mundo educativo. En los estudios realizados mostrábamos como en EEUU la mayoría de Universidades confían en este lenguaje como proceso de aprendizaje de sus alumnos.

En cuanto a la clasificación supervisada estamos hablando de que es un tipo de clasificación muy importante en estos momentos para las empresas y que en el futuro lo va a seguir siendo por el alto potencial de diferenciación que produce.

En un mundo en el que las diferencias y márgenes en los sectores tienen a reducirse es muy importante mantener la fidelidad de los clientes, por lo que es crucial poder clasificarlos (segmentarlos) según variables de negocio.

Enrique Lowe: 'El precio se olvida, la calidad permanece'

En este apartado R cuenta con inmensas librerías de algoritmos de clasificación supervisadas de la mayoría de tipologías.

Analizando y clasificando los paquetes de cada una de estas librerías, podemos descubrir que muchos de estos paquetes son muy parecidos y que la mayoría son fáciles de implementar y desarrollar.

Hay algunas tipologías como puede ser Decision Tree en las que la comunidad y número de paquetes es muy elevada y otras tipologías como SVM que se reduce bastante el número de paquetes e incluyen paquetes difíciles de instalar y desarrollar ya que no incluyen todos los binarios necesarios para su desarrollo.

Top Technology Spending Priorities — The Nexus Has Taken Up Residence

Rank	Technology	2015	2014
1.	BI/Analytics	41%	50%
2.	Infrastructure and Data Center	31%	37%
3.	Cloud	27%	32%
4.	ERP	26%	34%
5.	Mobile	24%	36%
6.	Digitalization/Digital Marketing	17%	11%
7.	Security	13%	11%
8.	Networking, Voice and Data Communications	12%	12%
9.	Customer Relationships	11%	8%
10.	Industry-Specific Applications	9%	10%
11.	Legacy Modernization	7%	7%
12.	Enterprise Applications	6%	2%

Source: Gartner 2015 CIO Survey, n = 2,793

Percentage of CIOs identifying each as a top three new spending priority

© 2014 Gartner, Inc. vendor is affiliated. All rights reserved.

Gartner

Como idea final cabe destacar que el análisis de datos está en la mente de los directivos. Según el último estudio de Gartner en 2016, la mayor prioridad para los CIOs es el Business Intelligence y Analytics y que debido a este aumento en las necesidades empresariales, el número de personas que se desarrollen en este sector va a aumentar altamente, con conocimientos de diversas herramientas ya que como hemos visto en el análisis comparativo inicial, todas tienen algo que las diferencia por lo que no va de escoger una o otra, sino de saber utilizar varias y exprimir las mejores características de cada una.

Bibliografía

- [1] KPMG EN LOS ÚLTIMOS 2 YEARS SE HAN GENERADO MÁS DATOS QUE EN TODA LA HISTORIA DE LA HUMANIDAD. SITIO WEB: [HTTP://VOZPOPULI.COM/ECONOMIA-Y-FINANZAS/34420-EN-LOS-ULTIMOS-2-ANOS-SE-HAN-GENERADO-MAS-DATOS-QUE-EN-TODA-LA-HISTORIA-DE-LA-HUMANIDAD](http://VOZPOPULI.COM/ECONOMIA-Y-FINANZAS/34420-EN-LOS-ULTIMOS-2-ANOS-SE-HAN-GENERADO-MAS-DATOS-QUE-EN-TODA-LA-HISTORIA-DE-LA-HUMANIDAD) (2013)
- [2] JULIO SERGIO SANTANA Y EFRAÍN MATEOS FARFÁN *Qué es R? El arte de programar en R: un lenguaje para la estadística*. Instituto Mexicano de Tecnología del Agua (2014)
- [3] LIBROS CIENTÍFICOS SOFTWARE CIENTÍFICO R. LENGUAJE DE PROGRAMACIÓN Y APLICACIONES AL CÁLCULO NUMÉRICO. CREATESPACE, 2015, ISBN-10: 1512017752
- [4] BRETT LANTZ *Machine Learning with R*. Packt Publishing, 2013, ISBN-10: 1782162143
- [5] OSCAR LAMIGUEIRO *Sobre Software, documentación y ciencia*. Sitio web: <https://procomun.wordpress.com/2011/02/23/que-es-r/> (2011)
- [6] BBVA OPEN4U *Taller para novatos en R: ventajas, instalación y paquetes*. Sitio web: <http://www.bbvaopen4u.com/es/actualidad/taller-para-novatos-en-r-ventajas-instalacion-y-paquetes> (2015)
- [7] BBVA OPEN4U *Data Scientist, el unicornio de los datos*. Sitio web: <http://www.bbvaopen4u.com/es/actualidad/data-scientist-el-unicornio-de-los-datos-que-es-que-hace-y-como-cambiara-el-mundo> (2015)
- [8] EDSGER W. DIJKSTRA *Recursive programming*. *Numerische Mathematik*, 2(1):312 318, 1960.
- [9] KDNUGETS *Analytics, Data Mining, Data Science software/tools used in the past 12 months*. Sitio web: <http://www.kdnuggets.com/polls/2015/analytics-data-mining-data-science-software-used.html> (2015)
- [10] RSTUDIO *Open Source Edition*. Sitio web: <https://www.rstudio.com/products/RStudio/#Desktop>
- [11] DATA SCIENTISTS COUNT *Distribution in Data Science: Sector and Geography*. Sitio web: <http://data-scientists-count.silk.co/>

- [12] BOB MUENCHEN *SAS Dominates Analytics Job Market; R up 42 %* Sitio web: <http://r4stats.com/2013/05/29/sas-dominates-analytics-jobs/> (2013)
- [13] MICHIEL HAZEWINKEL *Encyclopaedia of Mathematics (Encyclopaedia of Mathematics)*. Springer, 2001. ISBN 978-1-55608-010-4
- [14] ANDY FIELD *Discovering Statistics Using R*. SAGE Publications Ltd, 2012, ISBN-10: 1446200469
- [15] FAST COMPANY *The 9 Best Languages For Crunching Data*. Sitio web: <http://www.fastcompany.com/3030716/the-9-best-languages-for-crunching-data>
- [16] - W. N. VENABLES AND B. D. RIPLEY
Modern Applied Statistics with S. Springer, New York, NY, 4th edition, 2002. URL <http://www.stats.ox.ac.uk/pub/MASS4>. ISBN 0-387-95457-0
- [17] REVOLUTION ANALYTICS *Microsoft uses R for Xbox matchmaking*. Sitio web: <http://blog.revolutionanalytics.com/2014/05/microsoft-uses-r-for-xbox-matchmaking.html> (2014)
- [18] SIMON N. WOOD *Generalized Additive Models: An Introduction with R*. Chapman & Hall/CRC, Boca Raton, Florida, first edition, 2006. ISBN 1-58488-474-6
- [19] MICROSOFT RESEARCH *A Bayesian Skill Rating System*. Sitio web: <http://research.microsoft.com/apps/pubs/default.aspx?id=67956> (2007)
- [20] RAFAEL ARTZY *Linear geometry*. Addison-Wesley series in mathematics. AddisonWesley Pub. Co., 1974. ISBN 0-201-00362-7