UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Desarrollo de un paquete R para el análisis de sitios web de empresas

Trabajo Fin de Grado

Grado en Ciencia de Datos

AUTOR/A: Serrano Moliner, Arturo

Tutor/a: Doménech i de Soria, Josep

CURSO ACADÉMICO: 2021/2022

ÍNDICE

RESUMEN

La utilización de librerías en lenguajes de programación de alto nivel como R es una práctica que supone ahorros de tiempo y recursos a la hora de llevar a cabo proyectos de cualquier ámbito o escala. Es tal esta afirmación, que no se concibe el desarrollo de ningún proyecto sin la utilización de estas a lo largo del mismo. En este trabajo pretendemos aportar nuestro granito de arena creando uno de estos paquetes y enfocando el mismo hacia el análisis web de empresas, con el objetivo de poder emplear la información presente en estas para fines de análisis web, marketing digital, toma de decisiones de negocio o cualquier otra tarea relacionada con el mundo de la empresa.

Palabras clave: webscraping, R, business, programming libraries, web analysis

1 INTRODUCCIÓN

1.1 Motivación

El mundo de la empresa se ha visto afectado de lleno por el auge que vivimos de un tiempo a esta parte debido al *Data Science*. Ha supuesto el cambio de modelos de negocio que se mantenían inalterables durante décadas[1], y también de que el peso de la información más residual, repetitiva o considerada inútil cambie de ser unos y ceros llenando los discos duros a suponer un motor de negocio por explorar. Las webs de gran parte de empresas internacionales no suponían ningún tipo de herramienta útil hasta hace unos años, momento donde se empezó a ver el potencial de comunicación y ventas que supone y se invirtió potentemente en el desarrollo de las mismas[2]. Es por esto que ponemos en desarrollo esta idea de desarrollar un paquete de análisis web, siguiendo la línea de investigación comercial de los últimos años, pretendiendo aportar nuestro granito de arena a este auge de exprimir hasta el último ápice de información.

1.2 Contexto del trabajo: Origen

Antes de entrar en detalle sobre qué pretendemos desarrollar y cómo, es conveniente comentar de dónde nace la idea de este proyecto y cuál es su estado al inicio de este TFG. El tutor dio forma a la idea de explotar las webs de diferentes empresas hace unos 8-10 años, buscando aportar información extra al análisis de competencia en base a todo lo que se pudiese extraer de una web asociada a la empresa en cuestión. Para esto, desarrolló una serie de *scripts* en *shell* donde cada uno de estos cumplía una función de extracción de información básica, como obtener el texto, contar el número de palabras, detectar el idioma de la web, ver las etiquetas de html más repetidas…Estos scripts son el comienzo teórico de este proyecto y los que enmarcan un poco qué queremos llegar a hacer, sin dibujar límites en cuanto a la ampliación de los mismos ni a su mejora de calidad o rendimiento.

```
cat $1 | tr  '[:punct:] ' '\n' | tr '[:upper:]' '[:lower:]' | sed -n "s/^\([a-z]*\)$/\1/p" | grep -v ^$ | ./freq.sh
```

Figura 1. Ejemplo de código en shell. Fuente: Josep Domenech

Para muestra un botón, la estructura de la mayoría de estos es simple: una sucesión de ejecuciones, llamadas a funciones nativas de shell o, inclusive, a funciones definidas en el mismo marco de desarrollo, para finalmente extraer exactamente la información que buscamos dentro del código en *html*.

Ya con esto en mente, durante el resto del documento mostraremos qué y cómo hemos empleado esta base para construir nuestro trabajo, qué partes se han aprovechado, cuales se han eliminado y qué ampliaciones se han hecho sobre los originales.

1.3 Objetivos

Partiendo de la idea de buscar una solución eficaz al análisis de webs asociadas a empresas, y con la finalidad de poder extraer la mayor cantidad de información útil en base a la información presente en estas, hemos pretendido enmarcar los objetivos generales de este trabajo hacia el desarrollo de una biblioteca en el lenguaje de programación R, la cuál extraiga las principales características no evidentes de una web asociada a una empresa cualquiera, y el cuál será evaluado en la correcta clasificación de estas mismas empresas en base al contenido de sus webs corporativas, clasificadas como exportadoras o no exportadoras, todo esto en base a las diferentes métricas calculadas observando tanto el contenido orientado al usuario (texto, imágenes, cantidad de links y demás elementos) como a la forma en la que la web está escrita y estructurada en html, teniendo en cuenta la cantidad de elementos que posee de cada tipo (como etiquetas, cabeceras o tablas), la distribución de los mismos… entre otros muchos posibles factores. En este proyecto no se tiene en cuenta la posible visualización implementada con CSS o similares, sólo el contenido de la misma albergado en html.

La parte del análisis puro se realizará mediante los diferentes códigos en R que desarrollaremos a lo largo del proyecto, tomando como referencia los códigos en Shell que comentamos en la contextualización del trabajo, mientras que los modelos de clasificación que pretendemos utilizar para ver la utilidad del paquete en una tarea de clasificación concreta están programados y visualizados íntegramente en R, teniendo como entrada una batería de métricas seleccionadas de todas las que generamos en primera instancia, a fin de no sobreentrenar el modelo con individuos redundantes y repetitivos o con una cantidad excesiva de variables.

Además del desarrollo comentado, otro objetivo que buscamos cubrir al realizar este trabajo es ver la utilidad del paquete desarrollado en un entorno práctico, como el que nuestro hipotético futuro usuario final experimentará. Para ello, decidimos evaluar cuán útiles pueden llegar a ser los datos extraídos de un *pool* de webs previamente seleccionadas utilizando una serie de modelos predictivos con un objetivo muy marcado: intentar, en base a la web de la empresa consultada, saber si esta misma exporta sus servicios o productos de alguna manera fuera del territorio nacional (independientemente del volumen de estas exportaciones). Nuestro set de urls de entrenamiento estará formado por una selección de empresas españolas extraídas de la base de datos SABI, cuyo acceso será facilitado por la UPV, y de la que tomamos los valores de su dirección web y si tiene algún tipo de actividad exterior o no.

De todo este trabajo pretendemos ver si realmente existe información útil y explotable de los datos presentes en la página web de una empresa, de cómo está construida esa misma web y más factores que aplican tanto a la empresa dicha como a su página corporativa, resultados los cuales serán sintetizados y debidamente segmentados en el apartado de conclusiones.

Para sintetizar ideas, nuestro objetivo principal es el de preparar la librería mencionada, de forma que suponga una herramienta de utilidad en tareas de análisis web, y de forma secundaria, evaluar las métricas extraídas por medio de esta

mediante una tarea de clasificación simple, entrenando varios modelos y validando sus resultados con los diferentes conjuntos de datos creados para esta tarea.

1.4 Impacto esperado

Si bien nuestro objetivo es proveer de una herramienta competente y fiable a todo aquel que quiera hacer uso de ella, lo cierto es que siempre cabe la posibilidad de no llegar a ningún público objetivo, o que aquello para la tarea que se prepare y dote a la librería no cumpla ninguna exigencia real que tengan los usuarios. Nuestro objetivo con el paquete es añadir una herramienta más a todo aquel que quiera analizar la web de una empresa, teniendo así como meta entrar en un conjunto de bibliotecas de nicho y bastante específicas, dificultando de esta forma llegar a un público más grande.

Por esto mismo, creemos que el producto final no llegará a un grupo de gente muy amplio, pero que igualmente cumplirá las expectativas de aquellos usuarios que la gasten. De igual forma, nuestro trabajo también puede servir como comienzo de otros proyectos que sigan una línea de estudio, investigación o negocio similar, o que directamente pretendan ampliar las funciones creadas para así depurar y ampliar los resultados obtenidos con el paquete.

Comentar también que empresas como similarweb o Antara ya hacen negocio con este tipo de estrategias o métodos, pero dando por hecho que nuestro usuario final no se corresponderá al reducido grupo de desarrolladores de empresas de esta índole, asumimos los riesgos comentados en el apartado anterior.

1.5 Metodología

Para cumplir con nuestros objetivos y tener un desarrollo segmentado, organizado y secuencial, decidimos plantear el desarrollo del trabajo en 4 partes:

- Obviando la toma de decisiones inicial como los procesos a seguir, la decisión de soluciones propuestas o el planteamiento de objetivos, empezamos el trabajo revisando en profundidad las funciones en shell presentes, decidiendo cuáles queríamos mantener en nuestro proyecto, cuáles eliminaremos por falta de interés o de utilidad y qué otras funciones nuevas sería interesante implementar de cara a mejorar lo presente.
- En segundo lugar, ya con las funciones debidamente redactadas, prepararemos las diferentes baterías de entrenamiento y los diferentes modelos de clasificación, empleando como *inputs* estas mismas baterías generadas gracias a las funciones definidas para extraer los parámetros.
- Realizaremos un testeo y validación de los modelos propuestos en base a estas dos baterías de webs de empresas previamente preparadas

siguiendo ciertos criterios de agrupación de su población, como la localización o el sector donde se desempeñan.
- Finalmente, empaquetaremos las líneas de código necesarias en un paquete de libre distribución para así darle la mayor visibilidad posible.

Durante la realización del proyecto y de forma regular se irá realizando la redacción de esta memoria. No es un cuaderno de bitácora o un diario al uso, las partes redactadas con anterioridad a las actuales están sujetas a cambios, variaciones o a ser eliminadas si el proyecto lo demanda por causas del desarrollo. De igual forma, algunas partes han sido redactadas antes de apenas comenzar con el desarrollo del mismo (como estas mismas líneas), fruto de la investigación y preparación del proyecto, y otras surgen del desarrollo mismo de este, como todos los comentarios referentes a funciones en R, la preparación y validación de los distintos modelos o la generación de la biblioteca, entre otros.

1.6 Estructura

La estructura general de este TFG se puede dividir a grandes rasgos en 3 apartados: preámbulo al desarrollo, que comprende desde el resumen hasta el apartado del Plan de Trabajo, y que a grandes rasgos da toda la información necesaria para conocer las características principales del trabajo; Desarrollo y Despliegue del Paquete, formado por todos los apartados referentes al desarrollo de código, las funciones, el modelo y el paquete final; y por último el cierre, comprendido por el glosario y la bibliografía, dotando de información y fuentes adicionales al resto del documento. Entrando más en detalle de cada apartado, el desglose general quedaría:

1.6.1 Preámbulo al desarrollo

- **Abstract:** breve párrafo donde se sintetiza en qué consiste el proyecto y dentro de qué marco de estudio se sitúa.
- **Introducción:** Primera parte del documento. Explicación del origen del proyecto, su finalidad, las ideas planteadas para su desarrollo y demás información general empleada para contextualizar al lector.
    - **Motivación:** motivos por los que se ha llevado a cabo el trabajo.
    - **Contexto del trabajo:** contextualización del origen del proyecto y su fase previa al inicio.
    - **Objetivos:** conjunto de metas propuestas por el alumno a alcanzar al finalizar el desarrollo del trabajo.
    - **Impacto esperado:** repercusión sobre el mundo real que se espera tengan los frutos dados durante este proyecto.
    - **Metodología:** planteamiento general del desarrollo completo del proyecto, segmentado por partes y sus respectivas tareas o bien por cumplimiento de objetivos aislados o sencillos.

    - **Estructura:** apartado que está leyendo en este momento. Segmentación literal de las distintas partes del documento junto a una breve descripción de las mismas.

- **Estado del arte:** contextualización breve de la situación general contemporánea del área de estudio del proyecto.

    - **Webscraping:** comentarios al respecto del estado del arte dentro del Webscraping.

    - **R y librerías en R:** comentarios al respecto del estado del arte dentro de R, su uso actual y las librerías del mismo.

    - **Modelos:** comentarios al respecto del estado del arte dentro del machine learning y los modelos que emplea, de forma muy genérica y reducida.

    - **Crítica al estado del arte:** opinión general dada con respecto al estado actual del tema de estudio, tomando como referencia los trabajos comentados y las áreas de estudio afectadas.

    - **Propuesta:** línea que se pretende abordar dentro de la documentación que ya existe referente al proyecto.

- **Análisis del problema:** párrafo donde se enfatiza en el problema a abordar, contextualizando el mismo con las herramientas disponibles, los objetivos propuestos y la problemática que puede llegar a suponer en ámbitos de seguridad, legalidad, ética o eficiencia, entre otros.

- **Soluciones posibles:** breve discusión con las principales vías de solución al problema, argumentadas respectivamente.

    - **Solución escogida:** argumentación de cuál ha sido la solución escogida.

- **Plan de trabajo:** se hace hincapié en la distribución del tiempo y los recursos del proyecto a lo largo del mismo.

    - **Presupuesto:** nos cuestionamos sí, en base a nuestros conocimientos, los recursos disponibles, nuestra experiencia y los objetivos y metodología propuestos, seremos capaces de completar todas nuestras expectativas al finalizar el trabajo.

1.6.2 Desarrollo del proyecto

- **Preparación y Comprensión de Datos:** extenso apartado introductorio al desarrollo del código donde se cuenta la historia de cómo se ha planteado el problema, de dónde hemos extraído los datos, qué parámetros hemos extraído de los mismos y de qué forma y cuáles han sido los modelos escogidos para

llevar a cabo la experimentación con las diferentes baterías de entrenamiento desarrolladas.

- **Conocimiento Extraído y Evaluado de Modelos:** demostración del uso de diferentes librerías para el entrenamiento de los modelos propuestos a fin de evaluar la utilidad de las métricas desarrolladas en el apartado anterior, con sus correspondientes comentarios asociados a los resultados obtenidos en cada experimentación.

- **Legado:** huella dejada una vez finalizado el proyecto sobre el área de estudio trabajada durante el mismo.

- **Conclusiones:** último apartado principal del trabajo, breve resumen de todos los resultados obtenidos de nuestra experimentación y síntesis general de las ideas y conocimientos extraídos a lo largo de todo el trabajo.

1.6.3 Cierre

- **Glosario:** conjunto de definiciones de conceptos específicos de la materia, a fin de que el lector pueda comprender la totalidad de estos.

- **Bibliografía:** referencias bibliográficas empleadas tanto para el preámbulo como para el desarrollo del proyecto, yendo desde artículos académicos utilizados para dar información general al respecto de las ideas comentadas hasta links a páginas web empleadas durante el desarrollo del proyecto[4].

- **Anexos:** restante documentación donde encontramos los Objetivos de Desarrollo Sostenible (ODS).

Los apartados que acaba de leer pueden no corresponderse de forma literal con los encontrados en el documento, ya que algunos de los mismos se han comentado en conjunto con su apartado principal o entre ellos a fin de reducir la lectura y sintetizar ideas.

1.7 Relación del trabajo desarrollado con los estudios cursados

Debido a la naturaleza del proyecto y los objetivos que nos marcamos al comienzo del mismo, hemos tratado materias muy diversas vistas durante el grado, tales como el tratamiento de los datos en sus facetas de comprensión, extracción, limpieza y visualización, uso de expresiones regulares, creación de código orientado al web scraping en R apoyado en diversos paquetes, implementación de modelos de clasificación con diferentes objetivos… Estas tareas han sido respaldadas gracias a software y librerías externas a nuestro código, material citado correspondientemente en cada uno de los apartados donde se ha empleado.

Además de todos los conocimientos empleados que ya habíamos visto previamente en el grado, comentar la importancia de realizar tareas o proyectos de

cierta envergadura, trabajos donde hemos tenido también la necesidad de estructurarlos, dedicarles una investigación en profundidad y un desarrollo a medida, sin los cuales la familiarización del alumno con este tipo de trabajos habría sido más farragosa, sobre todo en ciertos apartados. Destacar las asignaturas de proyecto u optimización donde todas estas fases se pueden apreciar claramente en la redacción de los documentos que se nos demandaban y en las cuales se pedían cosas como comentarios de papers relacionados con la materia, tarea que también se da en este tipo de trabajos.

2 ESTADO DEL ARTE

En el ámbito general de lo que entendemos como extracción de información utilizando como fuente principal internet, el web scraping es la técnica por antonomasia empleada por empresas, académicos y organizaciones gubernamentales[2] debido a su robustez ante todo tipo de fuentes de información, sin dejar de ser susceptible a cambios en la estructura de la fuente de datos (reestructuración de una web,por ejemplo) . Esta afirmación no está extraída de ningún documento o publicación en particular, sino que es fruto de la investigación realizada referente a la extracción de información de internet, siendo prácticamente imposible encontrar documentación fiable que haga referencia a cualquier metodología que no emplee el web scraping en alguna de sus etapas o formas. Puede estar sustentado por un web crawler como extractor de información preliminar, aunque este tipo de bot se emplea mucho más en indexación web orientada a buscadores[5].

2.1 Webscraping

Como bien explica B. Zhao citando a otros dos autores, "*web scraping is widely acknowledged as an efficient and powerful technique for collecting big data (Mooney et al. 2015; Bar-Ilan 2001)*"[6]. Esta afirmación la ponemos de manifiesto a la hora de documentarnos en cuanto a la extracción de información de internet, siendo esta técnica la mayor exponente en su género. Su uso en sectores de investigación está más que contrastado[7], acto que de facto demuestra su utilidad frente a problemas de una complejidad elevada, como puede ser extraer información desde una fuente de datos que no está diseñada para ello. En general, es un campo de investigación que se encuentra en su apogeo de vida útil, teniendo ya un historial de uso, una evolución de la técnica y con margen de mejora de cara a futuras tecnologías web o evolución de las existentes[8].

2.2 R y librerías en R

De forma muy similar a Python y a diferencia de lenguajes de menor nivel, R es un lenguaje de programación con una cantidad de librerías muy considerable y de una variedad sorprendente[9], además de estar muy orientado a funcionar mediante el uso de las mismas. Al ser un lenguaje de alto nivel, su sintaxis es agradable a la vista y sus funciones, si dedicas el tiempo necesario a la documentación, fáciles de implementar en cualquier proyecto o desarrollo que lleves a cabo. A lahora de entrar en materia de desarrollo específico de librerías, hay que destacar que R cuenta con unos formatos de datos diferentes a los de la mayoría de lenguajes de programación orientados a objetos, ampliando los mismos y siendo, en ocasiones, obtuso y hermético para poder trabajar con los datos en crudo.

Esta serie de factores hace que, si comparamos el estimado de 1.200 librerías desarrolladas en R[10] con las más de 137.000 que posee python[11], se deduce de forma

bastante rápida cómo el desarrollo de código utilizando principalmente las herramientas proporcionadas por el core del lenguaje es mucho más accesible en un lenguaje que en otro.

2.3 Modelos empleados

Dada la experimentación que queremos hacer con las métricas generadas y los datasets descargados gracias a nuestra librería, hemos orientado esta parte del proyecto de forma general, sin especificar demasiado en nuestro problema en particular sino más bien en el concepto general de tarea de clasificación mediante machine learning[12]. El uso de estos está muy extendido en tareas académicas de todo tipo y sus parámetros optimizados hacia cada una de estas. Teniendo en cuenta que los modelos empleados durante este trabajo servirán principalmente para evaluar el resultado final de nuestro paquete y no como desarrollo principal del mismo, el detalle que debemos conocer de los mismos va acorde a su importancia dentro del proyecto. En sus respectivos apartados se hacen especificaciones técnicas y referencias.

2.4 Crítica al estado del arte

A pesar de que el *data mining* es un concepto que existe desde hace tiempo y está bastante explorado, su variante enfocada al mundo de internet denominada *web data mining* carece de esa profundidad en cuanto a usabilidad e investigación se refiere[2]. Asentando gran parte de nuestro trabajo sobre esta, buscamos añadir más valor al prematuro análisis web que existe hoy en día relacionado con el mundo de la empresa, en cuanto a librerías de código libre se refiere. No pretendemos crear un motor de análisis propio, sino añadir una herramienta más a aquellas personas que busquen en las páginas web de cualquier empresa, información extra a la que se muestra explícitamente. Es un área que en ciertas ramas, como los exploradores, tiene un desarrollo mucho más amplio que en sectores más clásicos como la economía o la banca. Por este motivo pretendemos empaquetar los resultados en formato de librería, para así hacer más fácil la difusión del paquete y facilitar su implementación en otros proyectos o consultas.

3 ANÁLISIS DEL PROBLEMA

Extraer información de una web es una forma de parafrasear el concepto de desestructurar el texto en formato html que conforma la misma con la finalidad de explotar el mismo. Para poder llevar a cabo un trabajo fiable y sin errores de concepto, debemos conocer a fondo cómo se estructuran las webs internamente: qué partes tienen, cómo se ordenan, qué funcionalidades poseen, qué partes se encapsulan dentro de otras…

Para ello, se ha realizado una experimentación previa sobre webs de importantes multinacionales como inditex (las cuales tienen páginas amplias y complejas) mediante peticiones simples, a fin de tener una primera impresión sobre las etiquetas más comunes, la cantidad de las mismas, detección de elementos ajenos al estándar de html como scripts de JavaScript y demás tipos de elementos.

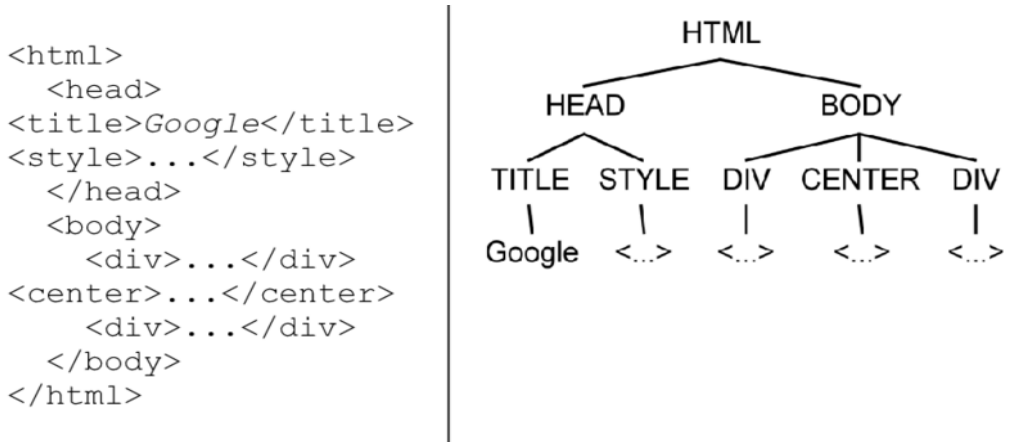


Figura 2. Estructura genérica de un html, izq. versión literal y dcha. versión jerárquica. Fuente: ResearchGate

Viendo la orientación de nuestros objetivos y los medios necesarios para cumplirlos, tuvimos presente desde el comienzo la necesidad de emplear expresiones regulares para indagar más a fondo sobre estos documentos html, siendo un factor clave tanto en la comprensión de la información presente como en el desarrollo del código empleado para crear la librería (especificaciones en apartados posteriores). En el desarrollo de ese mismo código iban a tener un papel importante tanto el modelo clasificador como la creación del paquete a distribuir, dos elementos que no deberían suponer tantos recursos dedicados como la parte de las funciones, pero que tienen igual o mayor importancia que estas al reflejar el producto final.

Con todo esto en cuenta, teníamos claras las partes del trabajo y sus prioridades, faltando por determinar cómo abordar cada una de estas, sus problemáticas particulares o las variaciones que podrían sufrir con respecto a nuestra idea original.

3.1 Análisis energético o de eficiencia algorítmica

Teniendo en cuenta que no vamos a emplear volúmenes de datos muy elevados ni procesos muy complejos, tanto el consumo energético como la eficiencia de nuestro proyecto serán casi insignificantes. Un aspecto a comentar es el tiempo que emplea el paquete para descargar, procesar y clasificar el link que le proporcionamos, teniendo en el proceso que realizar bastantes ejecuciones y demorándose un tiempo razonable en completar todo el proceso, factor difícil de reducir o suavizar debido a los diferentes cálculos y dependencias que lleva a cabo nuestra librería. La eficiencia no ha sido un factor que se haya tenido como meta en la creación de la librería, aunque más de una función tiene presente esta problemática en su creación y trabaja sobre elementos ya calculados por otras funciones o aprovecha cálculos previos para no repetir trabajo.

La máquina sobre la que se ejecute nuestro paquete no debería condicionar ni el consumo energético ni la velocidad de proceso, debido al poco coste temporal que conlleva cada llamada y la dependencia de un servidor externo para acceder a la web deseada.

3.2 Análisis del marco legal y ético

Descargar y analizar información 100% pública en internet no debería suponer ningún problema, a pesar de ser un campo de trabajo poco estudiado en el marco legal[13], aunque sí que podemos llegar a encontrar alguna especie de *disclaimer* explícito por parte de la web. Existen diversas páginas, sobre todo de entidades públicas o que contienen información de cierta sensibilidad, que especifican en su política de privacidad la no permisión de técnicas de web scraping o similares sobre webs publicadas bajo su dominio a fin de evitar la explotación de la información que publican, siendo este grupo el minoritario. Siguiendo esta línea, todas las webs empleadas durante la realización de este proyecto han sido debidamente estudiadas y clasificadas para evitar problemas como este.

Normalmente, estas mismas empresas se acogen al Robots Exclusion Standard (denominado comúnmente robots.txt)3, siendo este un protocolo estándar de comunicación entre las webs y los bots que ejecutan estos algoritmos de webscraping, a fin de lidiar con ellos y, sobretodo, buscando cubrir la correcta detección de los mismos.

En cuanto al apartado ético, sí que existe cierta información personal publicada de forma dudosamente lícita en internet como teléfonos, direcciones, códigos postales o correos electrónicos. La usabilidad de esta información para según qué propósitos es muy grande, y su potencial de mercado más aún, pero para este trabajo ese tipo de información va a tener un espacio mínimo o inexistente por el target de webs que vamos a emplear en entrenar y comprobar la efectividad del modelo.

3.3 Análisis de riesgos

Preparar una librería para su exportación hacia el mayor número de público posible conlleva problemas de compatibilidad entre sistemas operativos y/o versiones del lenguaje. Esta problemática no es exclusiva de nuestro proyecto, sino que es un factor de riesgo común al desarrollo de cualquier tipo de librería sobre cualquier lenguaje de programación. Industrias que no se pueden permitir este tipo de fallos, como la bancaria o las empresas relacionadas con el ámbito de la salud, emplean lenguajes o librerías que eliminan casi por completo este factor, debido a que en su sector el menor de los problemas podría suponer un auténtico desastre. En nuestro caso es algo completamente factible, más aún cuando no tenemos conocimiento de qué máquina, sistema operativo o entorno empleará nuestro usuario final.

Además de las características propias del proyecto, el otro factor que podría atentar a la realización del mismo sería la imposibilidad de realizar alguna de sus partes, por motivos relacionados con incompatibilidad de las tecnologías empleadas con la finalidad propuesta o similares. Esto es bastante improbable, ya que empleamos un lenguaje bastante popular y las librerías sobre las que nos respaldamos son de las más utilizadas entre la comunidad, pero siempre cabe la posibilidad de no poder hacer todo aquello que queramos.

4 IDENTIFICACIÓN Y ANÁLISIS DE POSIBLES SOLUCIONES

4.1 Discusión de soluciones planteadas

Una vez delimitado el contexto general y los objetivos que pretendemos cubrir, nos planteamos cómo abarcar de la mejor forma el proyecto teniendo en cuenta aquello que pretendemos conseguir y el formato final que queremos darle. La primera idea que surgió fue directamente implementar los diferentes archivos de *shell* (.sh) en R por medio de funciones que invoquen un kernel propio del sistema operativo donde se ejecutase el código, como podrían ser system, system2 o shell.execute (todas ellas pertenecientes al paquete base de R). Con este formato, conseguiríamos implementar pelo a pelo la base sobre la que partíamos, pero de igual manera dependeríamos directamente de una función en particular para el desarrollo, práctica poco recomendable en cuanto a funcionalidad se refiere, ya que el cuello de botella que nos auto imponemos es muy limitante.

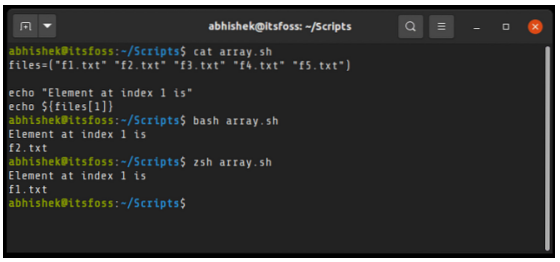


Figura 3. Consola de Linux (shell)

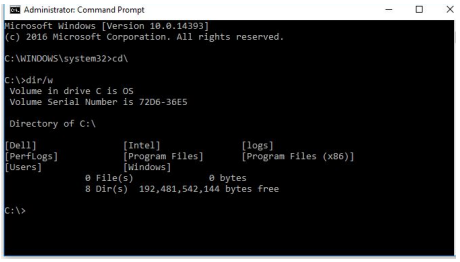


Figura 4. Consola de Windows (cmd)

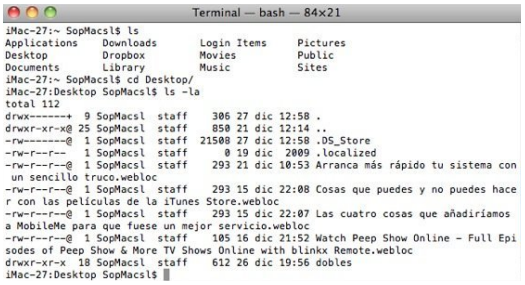


Figura 5. Consola de MAC OS (terminal.app)

Fuentes: Aulafacil, Wikipedia y Apple respectivamente.

Por otra parte, en el *brain storming* inicial surge la idea de implementar los *scripts** ya desarrollados por el tutor en código nativo de R, aumentando de esta forma la fiabilidad del mismo y preparándolo de cara a prevenir futuros errores que podrían surgir del hecho de depender directamente de llamar a una única función para poder ejecutar nuestra librería. Esta opción también cogía fuerza en términos de rendimiento, siendo mucho más óptima la idea de ejecutar código dentro de un mismo archivo que no ir realizando llamadas a archivos externos para la ejecución de los mismos. La problemática de la dependencia es palpable para cualquiera de las dos propuestas, si

bien en la primera esta misma es mayor en cuanto a peligrosidad de depender única y exclusivamente de una función, y la segunda teniendo un árbol de dependencias más amplio al realizarse los procesos internamente en R con librerías propias del lenguaje.

Ya con los resultados extraídos, los modelos empleados para ver cómo de útiles son las métricas desarrolladas serían indistintos de la opción escogida, debido a que la salida de nuestras funciones tendría continuidad en cuanto a formato se refiere (siempre sacarían la misma información independientemente de la solución escogida o de la web consultada). Estos modelos siguen una línea bastante clara en cuanto a su desarrollo: emplearemos diversos paquetes para entrenar modelos de regresión logística, redes neuronales y random forest, usando como inputs los resultados obtenidos del webscraping y, como objetivo secundario para evaluar la utilidad del paquete, veremos si estos modelos pueden llegar a clasificar las webs según si estas tienen exportaciones de sus productos o no.

4.2 Solución propuesta

Viendo la problemática ya comentada con la opción de realizar llamadas a la función *system* o similares, sumado al hecho de que estas mismas tienen comportamientos diferentes dependiendo del sistema operativo donde se ejecuten, la idea que llevaremos a cabo será adaptar los códigos de shell a R, aglutinando en un mismo proceso la ejecución de todos ellos en base a la *url* suministrada, y generando así todos los parámetros que definiremos

Cabe recalcar que se ha utilizado de base el proyecto comentado en el apartado de contextualización, pero este mismo ha sufrido cambios de estructura, ampliación de funcionalidades y diferentes modificaciones en este trabajo en comparación a su primera versión, adaptando de esta forma el análisis realizado al objetivo que nos hemos marcado y, a su vez, preparando el mismo para una posible escalabilidad o mejora futura como agregación de funciones paralelas, ampliación de los parámetros creados o empleabilidad para otro tipo de fines de clasificación o predicción.

5 PLAN DE TRABAJO

5.1 Planificación

Como ya hemos comentado en la metodología y en la estructura, la realización general del trabajo se segmenta en dos ejes fundamentales: preámbulo al desarrollo, y el desarrollo propiamente dicho en conjunto a la resolución y a las conclusiones. La línea a seguir para llevar a cabo todos los objetivos propuestos sin descuidar la calidad del producto final sigue un guión similar al de la estructura:

- La redacción de las diferentes partes del documento se realiza conforme el propio proyecto las demanda en su evolución natural: ciertas partes como el Estado del Arte, el Impacto Esperado, o estas mismas líneas deben ser redactadas previo desarrollo del código, para poder tener una perspectiva competente y llevar una hoja de ruta a seguir que esté fundamentada en argumentos y no en necesidades puntuales o esporádicas del alumno.

- El desarrollo del código, al ser la parte más *core* del proyecto, supone la mayor inversión de tiempo en cuanto a investigación y desarrollo nos referimos. Tendrá lugar desde el inicio hasta el final, pasando por la selección de funciones y funcionalidades a implementar, el desarrollo y validación del modelo escogido y, finalmente, el empaquetado en una librería para su posterior publicación.. Durante el desarrollo de estas distintas partes, este documento sufrirá cambios, ampliaciones o modificaciones según devengan los hechos.

- De forma contínua, tanto el Glosario como la Bibliografía irán siendo aumentados en función de las fuentes, palabras y expresiones que empleemos tanto en la redacción misma del documento como en la investigación del mismo. Dejarlos para el final podría dar lugar a olvidos o incoherencias con el resto del documento, motivo por el cual su desarrollo será contínuo.

Este guión es la ruta a seguir para completar correctamente todas las partes del proyecto, pero en ningún caso es inviolable ni invariable, pues su redacción parte del supuesto de no llegar a sufrir cambios en la estructura del trabajo, ni en el desarrollo del mismo, ni en el aumento o disminución de sus partes. Si en apartados futuros sucediese alguno de los hechos mencionados, este sería comentado, justificado y remarcado al lector para no perder detalle del desarrollo en su totalidad, teniendo la idea original como referencia y el desarrollo y la solución como resultado de nuestro trabajo.

5.2 Presupuesto

Analizando los requisitos técnicos y de conocimiento que demandan los objetivos planteados, la única problemática real que podría llegar a variar la hoja de ruta del proyecto serían los dos puntos señalados en el análisis de riesgos. Quitando de estos, tanto los requerimientos técnicos como de medios de trabajo son asequibles,

pudiendo realizar todo el proyecto con un ordenador cualquiera, sin potencia mínima requerida, y conexión a internet de manera constante.

Si entramos a evaluar el factor temporal, la conclusión a la que llegamos es la misma, ya que no es un factor limitante en nuestro caso por el amplio márgen que tenemos desde la adjudicación del tema a tratar. Aún teniendo partes del desarrollo, como la creación de los conjuntos de entrenamiento, que demandarán varios días de proceso para poder tener un bagaje considerable de población, sigue sin suponer un problema de cara a poder llevar a cabo con éxito nuestro paquete.

Reseñar que la participación activa del tutor durante todo el proceso ha supuesto una ayuda considerable en ciertos puntos críticos del proyecto como la adquisición de los datos en crudo o el uso de expresiones regulares en R, y sin la cual no podríamos hablar de una holgura de tiempo tan amplia.

6 PREPARACIÓN Y COMPRENSIÓN DE LOS DATOS

Como hemos comentado en líneas anteriores, el conocimiento en profundidad de los elementos que forman una página web es clave para poder sacar el máximo partido a nuestro análisis. Preguntas como qué elementos son los más frecuentes, cuáles son diferenciadores entre distintas páginas web, qué palabras del texto pueden dar mayor información y demás, son claves a la hora de saber cómo orientar nuestro código y cuáles son los *inputs* más interesantes y con mayor potencial para nuestro modelo.

6.1 Paquetes empleados

Durante distintas fases del desarrollo del código, se ha probado la implementación de las funcionalidades, el módelo, el parser de html y demás partes con diversas librerías distintas. El objetivo de esta práctica era poder tener una referencia directa y comprar qué opción se ajustaba mejor a nuestras demandas, para así mantener la línea de fiabilidad que buscamos desde el inicio del proyecto. Una vez terminado el código, las dependencias existentes son las siguientes:

- **cld3**: Google´s Compact Language Detector. Librería empleada para detectar el idioma de la web consultada empleando una red neuronal. Si bien se compone de dos funciones de detección, una de detección del lenguaje más probable, y otra de detección de los lenguajes más probables, sólo haremos uso de la primera.

- **gsubfn**: Paquete con mayor protagonismo dentro del proyecto.Si bien las funciones del paquete base de R como grep o regex, no nos permiten seleccionar todas las coincidencias del patrón dispuesto bien porque solo devuelven la primera coincidencia, bien porque nos devuelven los índices de inicio y fin de las coincidencias y no el texto en sí, o bien porque trabajan sobre coincidencias en elementos de un vector, y no de un texto plano (el cual es nuestro input). Es por esto que recurrimos a la función *strapplyc* de este paquete para cubrir nuestra necesidad de extraer la mayor cantidad de información posible del html.

- **rvest:** Paquete empleado para realizar las consultas http dada una url utilizando la función *read_html*

- **httr:** librería utilizada para poder establecer un tiempo máximo de espera a las peticiones http realizadas, ya que el paquete **rvest** no posee forma nativa de realizar un control en detalle de las peticiones realizadas, y esta librería sí tiene mayores opciones de parámetros para poder controlar con mayor detalle cómo y bajo qué condiciones se realizan las solicitudes.

```
tryCatch(html <- url %>% GET(., timeout(15)) %>% read_html,
         error = function(e) {error <<- 1 })
```

Figura 6. Línea de código donde combinamos ambos paquetes. Fuente: propia

- **htm2txt:** Empleado para la extracción del texto presente en el html en crudo resultante de nuestra petición. No consigue capturar correctamente la codificación de ciertas páginas, pero como el procesamiento que hacemos del texto en los parámetros sobre todo es a la hora de contar número de palabras o veces que se repita la palabra más común, no le damos mayor importancia. Como bien comentamos en las conclusiones, este apartado se deja como una posible ampliación al trabajo original, a fin de mejorar el paquete.

- **MKL:** Optimizador de procesos cuya implementación nace a raíz de los elevados tiempos de carga experimentados durante el entrenamiento de nuestros modelos. Justificación de su uso en detalle en su correspondiente apartado.

- **DMwR:** Por medio de la función **SMOTE** de este paquete, conseguimos resolver el problema de desbalanceo de clases presente en nuestras baterías de test[13]. Esta librería sólo es empleada para eso, sirviendo de puente entre la lectura de las métricas creadas y la entrada a los diferentes modelos creados.

Comentar que durante el desarrollo temprano del proyecto, empleamos una librería distinta para la correcta extracción del html desde la página web empleando el paquete **httr2**. Adjuntamos bajo estas líneas la redacción original del paquete para, en caso de ser necesario, contextualizar al lector en el desarrollo y evolución de la programación para montar las diversas baterías y la correcta definición de las funciones finales:

- **httr2**: Paquete utilizado para la realización de peticiones a los servidores web. Gracias a este conseguimos obtener un objeto de tipo response, del que podemos extraer desde la respuesta dada por el servidor hasta el html. Durante nuestra experimentación nos hemos encontrado con la problemática de que ciertas webs, utilizando este paquete en particular, deniegan de forma directa la solicitud con su correspondiente código 403. Dada esta particularidad, y que ciertas variables empleadas (como el tamaño en memoria del body) en el modelo salen directamente de ese *response* generado, decidimos añadir un comprobante de respuesta a nuestro código mediante la función *resp_status* para que, en caso de dar error o denegar la solicitud, la repita o pruebe a obtener el html de la web utilizando la función *read_html* del paquete **rvest** respectivamente.

6.2 Problemática con las Expresiones Regulares

Durante la implementación de las funciones propuestas, basadas en expresiones regulares, surge un impedimento bastante inesperado: estas mismas expresiones no funcionan exactamente igual en todos los lenguajes, teniendo

variaciones menores en carácteres, simbología o funcionalidades de *matching* en algunos de ellos. Ha supuesto un reto conseguir descifrar, a base de prueba y error, qué podíamos y qué no podíamos hacer en R, viendo que algunas expresiones funcionaban perfectamente en comparación a el uso de la misma en otro entorno, otras necesitaban variaciones menores para llegar a nuestro objetivo, y seguido de estas un extenso grupo de funcionalidades que o bien no están implementadas en los motores de *re**** que utiliza R, o bien su sintaxis es diferente. Llama la atención agrupaciones como [:punct:], las cuales deberían detectar todo un conjunto de caracteres (en este caso signos de puntuación) pero que al utilizarlos con las funciones básicas de R que utilizan patrones de expresiones regulares, hacen un uso literal de los corchetes en vez de interpretarlo como una agrupación compleja.

[:alnum:]	Caracteres alfanuméricos [:alpha:] y [:digit:]	A, B, c, d, 1, 2, …
[:alpha:]	Caracteres: [:lower:] y [:upper:]	A, B, c, d, …
[:blank:]	Caracteres blancos	Espacio, Tabulador, …
[:cntrl:]	Caracteres de control	
[:digit:]	Dígitos	0, 1, 2, 3, …
[:graph:]	Caracteres gráficos [:alnum:] y [:punct:]	A, B, c, d, 1, 2, #, %, …
[:lower:]	Todas las letras minúsculas	a, b, c, …
[:print:]	Caracteres gráficos [:alnum:] y [:punct:]	A, B, c, d, 1, 2, #, %, …
[:punct:]	Caracteres de puntuación	! » # $ % & ‘ ( ) * + , − . / : ; < = > ? @ [\ ] ^ _ ` { | } ~
[:space:]	Caracteres de espaciado	Espacio, tabulador, nueva linea, …
[:upper:]	Todas las letras mayúsculas	A, B, C, …
[:xdigit:]	Dígitos hexadecimales	0, 1, 2, 3, A, B, e, f, …

Figura 7. Expresiones regulares sobre un conjunto de carácteres específicos. Fuente: Diego Calvo

De forma gráfica, en vez de relacionar el patrón comentado con símbolos como comas, punto y coma, puntos o dos puntos, busca literalmente por “:”, “p”, “u”, “n”, “c” y “t”, omitiendo el significado que se le asocia en el 95% de documentación oficial que hay disponible en las páginas oficiales tanto de las expresiones regulares como de las librerías. No es exclusivo de esta expresión en particular, sino de todas las que siguen la misma estructura de corchetes con dos puntos, los *look-behind* como “(?<=ABC)” o los *look-ahead* como “(?=ABC)”, bastante similares estos dos últimos entre sí. Esto ha supuesto un desarrollo algo más extenso de lo debido por el sistema de prueba y error que se ha adoptado, pero no ha impedido el llevar a cabo las tareas que nos propusimos en un principio.

También nos hemos encontrado con que, durante el desarrollo de los parámetros a implementar en el modelo y testeando tanto los patrones como las variables empleadas, la respuesta recibida en la consola dificultaba seriamente el análisis de los resultados, ya fuera por tiempos de carga elevados, fallos de codificación entre las diferentes funciones empleadas o impresiones por pantalla que dejaban cientos de líneas vacías entre respuesta y respuesta.

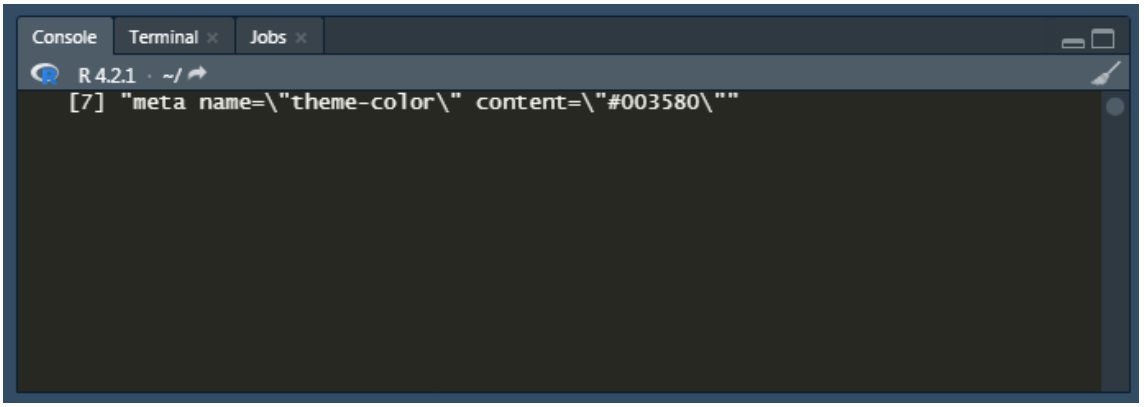


Figura 8. Ejemplo de fallo de impresión de los resultados en la consola de RStudio. Fuente: propia

6.3 Extracción del texto

En primera instancia, y siguiendo las indicaciones de nuestro tutor y de la mayoría de publicaciones en foros, probamos a extraer el "texto" propiamente dicho del html en cuestión por medio de funciones predefinidas, como pueden ser *html_text* del paquete *rvest* o *htm2txt*, del paquete con idéntico nombre. Lo cierto es que, probando con un test de webs considerable, estas funciones no devolvían el resultado esperado al completo, sino dando una serie de resultados que no eran los esperados.

Dependiendo de la forma en que extraigamos el html del response que nos ofrece la petición al servidor donde esté alojada la web, algunas de estas funciones nos devuelven un objeto de tipo character en blanco. Si les pasamos el html en crudo (esto es, como un vector carácter) en vez de como un objeto con separación entre las etiquetas de head y body, nos topamos con que la codificación de ciertas páginas tampoco las capta bien por diferencias entre ASCII y UTF-8. También se daba la situación de obtener el texto, sin problemas de tildes o signos, pero añadiendo etiquetas, atributos y valores de elementos de la web que nada tienen que ver con el texto plano que buscamos

Por estos motivos, y queriendo seguir la línea de trabajo empleada para construir nuestras métricas en R empleando expresiones regulares en vez de funciones que nos cocinen el resultado al completo, decidimos probar a extraer el texto visible por el usuario sin utilizar funciones predefinidas. Gracias a algunas de las funciones previamente comentadas, sabíamos a grandes rasgos el resultado que deberíamos obtener en cuanto a títulos, frases y demás, por lo que a base de prueba y

error, y utilizando la web https://regexr.com/ como tester con el html completo, definimos cuál sería la sucesión de expresiones regulares que consiguiesen extraer la mayor cantidad de texto posible para su posterior análisis por palabras.

Lo cierto es que, al igual que con nuestras primeras versiones del código, los resultados de esta experimentación no eran robustos: para ciertas páginas extraía más información de la buscada ya que no diferenciaba etiquetas de texto, para otras perdía mucha información de la presente en el html y, sólo para un muy reducido grupo de webs, cumplía nuestros criterios de calidad en cuanto a texto extraído de forma correcta se refiere. Probando con patterns distintos los resultados no mejoraban, y además de lo añadido, resultaba computacionalmente bastante ineficiente.

En vista de tal situación y con la experimentación realizada, decidimos mantener el procedimiento de extracción de texto mediante la función **htm2txt** del modo en que hemos comentado en el apartado de parámetros, manteniendo así la práctica que hacíamos con anterioridad.

6.4 Parámetros definidos

Para el entrenamiento de cualquier modelo que deba resolver un problema de clasificación, predicción, regresión o agrupación, es imprescindible tener un conjunto de datos de entradas o *inputs* iguales a todos los individuos de la población, bien definidos y que puedan ser igualmente extraídos de cada caso.

En este problema en particular, nuestra mayor fuente de información será la respuesta que nos dé el servidor a nuestra petición de una *url* concreta, teniendo en esta el código de respuesta del servidor, el tamaño en memoria de la web y su html. De estos tres elementos el que nos va a dar juego será este último, del cual haremos diferencias entre dos elementos:

- Todo aquello que se encuentre dentro de las etiquetas, teniendo en este grupo el tipo de la etiqueta, sus atributos y los valores de estos mismos
- Los elementos que podemos encontrar entre las distintas etiquetas, pudiendo encontrar aquí el texto visible en la web junto a saltos de línea y demás elementos relacionados con el texto

La diferenciación entre qué hay dentro y fuera de una etiqueta viene dada por los símbolos menor y mayor que (“<” y “>”), siendo lo que se encuentre entre el menor y el mayor todo aquello perteneciente a una etiqueta, y todo lo que esté entre un mayor y un menor el texto restante. Sabiendo esto, la mayor parte de variables proporcionadas al modelo salen de la extracción directa de estas cadenas de caracteres, seleccionadas mediante expresiones regulares. Dicho todo esto, los parámetros para entrenar el modelo son los siguientes:

- **export:** variable dependiente en los modelos empleados. Binaria, toma valor 0 para las empresas no exportadoras y 1 para las exportadoras. Es extraída directamente de la columna Importador/Exportador presente en la base de datos del SABI.

Figura 9. Columna fuente de la variable export. Fuente: SABI

Si el valor era “Exportador” o “Importador/Exportador”, el valor de la variable export tomaba valor 1, y en caso contrario (“Importador” o “No realiza actividad exterior”) tomaba valor 0

- **size_html:** número de caracteres que tiene el html asociado a la web. Factor que nos indica si nos encontramos ante una web de mayor o menor tamaño, y también empleado en la generación de las baterías de test para hacer criba de ciertas páginas excesivamente grandes (y, por lo tanto, *outlayers* de nuestra población).

- **lang:** idioma en el que está escrita la página. Extraído gracias al paquete cld3 de Google. Si bien todas las webs empleadas en este trabajo son españolas, algunas tienen sus páginas corporativas predefinidas en inglés, por lo que ese factor puede devenir diferencias en el modelo.

- **attribs:** número de atributos dentro de las etiquetas. Parámetro que da un nivel de detalle de, en relación a la cantidad de etiquetas presentes, cuán detallada ha sido la web en cuanto a estas, pudiendo especificar desde su posición, color, dimensiones o demás parámetros de, por ejemplo, una imagen o cualquier elemento gráfico.

- **div_tags:** número de etiquetas del tipo “div” presentes en el html.

- **hrefs_tags:** número de etiquetas de enlace o anclaje encontradas en el html.

- **headers:** número de etiquetas de tipo título dispuestas en el html. No se hace diferenciación entre h1, h2, h3… Diferenciar de la etiqueta <head> presente al inicio de los documentos html: en este parámetro nos centramos en las etiquetas de tipo cabecera de las distintas partes del html, indistintamente de si se encuentran dentro del <head> o del <body>.

- **scripts:** número de etiquetas asociadas a scripts (como ejecutables de JavaScript) presentes en el html. El número de estas no es directamente proporcional al número de scripts que la página emplea, ya que diferentes etiquetas a lo largo de la web pueden hacer referencia al mismo script.

- **classes:** número de etiquetas del tipo “class” encontradas en el html.
- **tables:** número de objetos de tipo tabla presentes en la web.
- **table_tr:** número de etiquetas del tipo fila de tabla que aparecen en el html.
- **table_td:** número de etiquetas del tipo columna de tabla presentes en el html.
- **most_rep_tag:** nombre de la etiqueta con más apariciones en el html
- **n_most_rep_tag:** número de veces que aparece la etiqueta más repetida.
- **n_tags:** número total de etiquetas que posee la web.
- **n_words:** número de palabras presentes en el texto extraído del html. El factor utilizado para, del texto procesado, contar las palabras, ha sido aplicar la función *split()* sobre todo el texto utilizando el espacio en blanco como separador.
- **most_rep_word:** palabra más repetida en el texto. Para este parámetro no hemos hecho limpieza de conjunciones y demás *stopwords* debido a que normalmente son estas palabras las más repetidas en el texto de una web (y de cualquier texto en general), y si eliminásemos este conjunto de palabras del texto original
- **n_most_rep_word:** número de veces que aparece la palabra más repetida. Dentro de las variables que hemos definido,
- **span:** número de veces que se repite la etiqueta “span”. Asociada al formateo de datos de texto (color, tipografía, ubicación en la página…).

** Destacar que tanto “table_td” como “table_tr” pueden aparecer sin necesidad de que exista una etiqueta de tipo “table” o un objeto tabla propiamente dicho en el html*

La elección de estos parámetros no es en ningún caso arbitraria. El criterio seguido para saber qué información pasarle al modelo ha sido variado: nos hemos fijado en las etiquetas asociadas a recursos más complejos que otros o que denotan modernidad, como *script* o *classes*, porque consideramos que esto es un factor clave en la diferenciación de una empresa grande frente a una pequeña, por los recursos dedicados a la creación de su web o por lo actualizada que estuviese la misma.

La mayor limitación vista en este apartado, y archiconocida a lo largo de todo el grado en los entrenamientos de modelos, ha sido el *overfitting* a raíz de exceder el número óptimo de inputs. Para suavizar este efecto, y en vista de tener muchas páginas con parámetros similares, decidimos reducir el número de variables a emplear y aumentar el número de elementos en la población, para así buscar la mayor diversidad de población en ambas baterías y primar la calidad del modelo ante el volumen de información procesado.

6.5 Baterías de test

Ya con todos los parámetros que queremos emplear definidos, nos pusimos a preparar dos conjuntos de urls de cara al entrenamiento de los modelos y así evaluar la utilidad de las métricas. Teníamos en mente crear dos poblaciones relacionadas entre sí por su naturaleza (en este caso, todo serían empresas) pero al mismo tiempo que se diferenciasen en algún factor clave, como los beneficios, el número de trabajadores, el área de acción o la gama de productos. Como todos estos factores no están disponibles ni para nosotros al realizar los test, ni para el usuario futuro a la hora de consultar una web cualquiera, decidimos que el factor de agrupación fuese el sector económico al que se dedica la empresa, variable que puede llegar a ser deducida sin mayor dificultad por el usuario a la hora de consultar una web y que nosotros, por nuestras fuentes de datos, poseemos.

Especificadas las exigencias previas para generar nuestros datos, estos han sido los dos grupos propuestos para entrenar nuestros modelos:

- El primer grupo estaría conformado por grandes empresas, multinacionales, pymes, organizaciones, asociaciones deportivas, comercios locales y demás conglomerado de webs, buscando ver cómo se comportan los modelos frente a tanta variedad de fuentes de información. Por la naturaleza misma de algunas de estas páginas, hemos primado que en todas ellas se dé algún tipo de servicio o venta, para así poder clasificar correctamente si la empresa u organización asociada a la web exportaba productos fuera de su país o no.

- El segundo, ya más específico, lo hemos creado siguiendo el criterio de que lo formasen empresas relacionadas con el sector agrícola o alimentario. Este grupo está formado por productoras de abonos, bodegas, empresas de frutos secos y pipas, pequeñas distribuidoras de bebida, fruta, verdura o pescado y demás empresas relacionadas con el mundo agrícola y alimentario. Ya con este lo que pretendemos es distinto: si bien tener un conjunto de individuos genérico enriquece y hace más robusto el modelo que pretendamos entrenar al recibir información más dispar entre sí, experimentar sobre un subconjunto de individuos que tienen un grado de similitud más próximo entre sí puede llevarnos a crear un modelo menos robusto, pero sí con mayor precisión.

Para la creación de ambos grupos, y dado que lo que buscamos es poder tener una comparación competente, hemos recurrido a la base de datos de SABI, cuya licencia está disponible para los miembros de la UPV. Gracias a esta, hemos tenido acceso a toda la información que hemos necesitado para desarrollar el proyecto, desde la dirección url de la empresa a saber si esta misma presentaba exportaciones en sus actividades económicas. También nos ha sido posible filtrar esas mismas empresas por localización, actividad y demás factores empleados para mantener una cierta homogeneidad en los test.

A pesar de que pueda ser contraproducente para el entrenamiento del modelo, a la hora de encontrar webs pertenecientes al mismo sector en la segunda batería (en nuestro caso, el de la alimentación) no hemos hecho una criba en cuanto al sector específico al que pertenece cada empresa. De esta forma, tenemos desde productoras

de abonos, fitosanitarios, frutos secos, infusiones, algas… a distribuidoras locales de fruta del día o ultracongeladoras de pescado, como ya hemos comentado antes. Si bien cuanto más similares sean las empresas entre sí más acertado debería ser el modelo por su inducida similitud en las webs, a su vez limitaría la efectividad del mismo frente a webs de empresas que se alejaran de ese grupo entrenado. Buscando mantener la línea de fiabilidad y robustez que buscamos desde el principio del proyecto, tomamos esta decisión de diversificación aún sabiendo que de esta forma sacrificamos un posible porcentaje de acierto mayor.

Inicialmente, nuestra primera batería estaba formada por más de 180.000 empresas. Debido a las limitaciones de presencia de urls en la base del SABI, las que realmente nos son de utilidad para nuestro trabajo son 18.188, que son todas aquellas con una web asociada. En nuestra segunda batería pasa algo bastante similar, ya que al filtrar en la web del SABI todas las empresas españolas cuya actividad estuviera relacionada con el sector agrícola o alimenticio, inicialmente contábamos con 28.826 empresas, de las cuales únicamente tenían una página web asociada 1.658, lo que limitaba bastante el número de individuos hábiles para entrenar.

Además de la no presencia de webs en los datos originales, más adelante comentaremos la criba existente debido a fallos en las peticiones http que realizamos desde R, factor que nos ha limitado aún más el número de individuos prácticos para poder entrenar nuestros modelos.

6.6 Optimización de tiempos de proceso

Sabiendo que R es un lenguaje *single thread*, y en vista de los tiempos de ejecución medios que teníamos durante el testeo de algunas combinaciones de modelos, decidimos aumentar el número de hilos de proceso de los que podía disponer el kernel utilizando la librería **MKL** de Intel, la cual nos permitía cambiar esta configuración de serie y de esta forma reducir los tiempos de entrenamiento y validación de los modelos. Nos fue realmente útil, sobre todo con la batería de empresas genéricas, debido a que su tamaño y tiempos de proceso eran considerablemente mayores a los de la batería de empresas alimentarias. Esta medida, como es evidente, no afectó en mayores aspectos a los resultados del trabajo, simplemente redujo los tiempos de proceso necesarios para llevar a cabo los entrenamientos necesarios.

6.7 Análisis exploratorio

Una vez tuvimos ambas baterías preparadas, con sus respectivas limpiezas previas y métricas montadas gracias al algoritmo desarrollado, pasamos a comenzar con el entrenamiento de los modelos propuestos para así ya poder aplicar los parámetros a predicciones sobre una empresa cualquiera deseada. Antes de ponernos con estos, y en vista de lo opaco que fue el montaje y extracción de la información desde las páginas webs, decidimos explorar un poco más en profundidad

Nuestra primera aproximación una vez creados nuestros conjuntos de datos y realizado las limpiezas y conversiones oportunas fue utilizar la función summary de R para explorar los valores que recibía cada variable, ver su comportamiento aislado, su media y demás factores:

```
> summary(metricas)
     url                export          syze_html          lang               n_words
 Length:9422        Min.   :0.0000   Min.   :     0   Length:9422        Min.   :    0.0
 Class :character   1st Qu.:0.0000   1st Qu.: 36786   Class :character   1st Qu.:  206.0
 Mode  :character   Median :0.0000   Median : 75474   Mode  :character   Median :  418.0
                    Mean   :0.1984   Mean   : 99072                      Mean   :  596.1
                    3rd Qu.:0.0000   3rd Qu.:131450                      3rd Qu.:  754.0
                    Max.   :1.0000   Max.   :498153                      Max.   :11280.0

 most_rep_word      n_most_rep_word       ntags          n_attribs      most_rep_attrib
 Length:9422        Min.   :   1.00   Min.   :   3   Min.   :    2   Length:9422
 Class :character   1st Qu.:  14.00   1st Qu.: 302   1st Qu.:  610   Class :character
 Mode  :character   Median :  28.00   Median : 519   Median : 1068   Mode  :character
                    Mean   :  42.29   Mean   : 640   Mean   : 1354
                    3rd Qu.:  52.00   3rd Qu.: 819   3rd Qu.: 1794
                    Max.   :1163.00   Max.   :6924   Max.   :11044
                    NA's   :173
 n_most_rep_attrib most_rep_tag       n_most_rep_tag      headers          hrefs_tags
 Min.   :   1.0    Length:9422        Min.   :   1.0   Min.   :  0.00   Min.   :   0.0
 1st Qu.: 157.0    Class :character   1st Qu.:  85.0   1st Qu.:  5.00   1st Qu.:  49.0
 Median : 298.0    Mode  :character   Median : 159.0   Median : 13.00   Median :  78.0
 Mean   : 403.8                       Mean   : 214.4   Mean   : 16.88   Mean   : 100.8
 3rd Qu.: 524.0                       3rd Qu.: 280.0   3rd Qu.: 22.00   3rd Qu.: 121.0
 Max.   :7323.0                       Max.   :3653.0   Max.   :405.00   Max.   :1538.0

   div_tags          span             tables           tables_tr        tables_td
 Min.   :   0   Min.   :   0.00   Min.   :  0.0000   Min.   :  0.00   Min.   :   0.000
 1st Qu.:  76   1st Qu.:  12.00   1st Qu.:  0.0000   1st Qu.:  0.00   1st Qu.:   0.000
 Median : 149   Median :  33.00   Median :  0.0000   Median :  0.00   Median :   0.000
 Mean   : 199   Mean   :  57.86   Mean   :  0.3988   Mean   :  1.41   Mean   :   1.906
 3rd Qu.: 267   3rd Qu.:  69.00   3rd Qu.:  0.0000   3rd Qu.:  0.00   3rd Qu.:   0.000
 Max.   :3430   Max.   :2616.00   Max.   :188.0000   Max.   :768.00   Max.   :1464.000

   scripts
 Min.   :  0.00
 1st Qu.: 12.00
 Median : 23.00
 Mean   : 28.78
 3rd Qu.: 37.00
 Max.   :190.00
```

Figura 10. Sumario de los parámetros extraídos de la batería 1. Fuente: propia

Como podemos ver en la imagen superior, y siguiendo la línea de que la mayoría de parámetros tienen relación entre sí, al ser referidos muchos de estos a etiquetas html, el valor mínimo de estas variables es mayoritariamente 0 mientras que el máximo es bastante variable. Las medias tampoco siguen una proporción entre sí, teniendo desde valores muy bajos, como la media de tablas o de scripts, hasta la media de etiquetas como los divisores o los “span”, teniendo bastante sentido esta proporción por el uso dado a cada uno en el diseño web y la necesidad específica de cada uno.

En cuanto a las variables generales del html, vemos que las medias de palabras y de etiquetas totales van bastante a la par entre ellas, dando a entender que ambos parámetros tienen una relación lineal o similar. En cuanto a los más repetidos, existe bastante diferencia entre atributos, etiquetas y palabras, presentando medias mucho más altas estos primeros y viendo que las palabras más repetidas no son ni comparables a proporción sobre el total respectivo. El tamaño de las webs, al estar limitadas a 500.000 carácteres como máximo para evitar errores y tiempos de carga,

pone el máximo muy cerca de este valor, a pesar de que su media sea cinco veces menor.

```
> summary(metricas_agro)
     url                export          syze_html          lang              n_words
 Length:849         Min.   :0.0000   Min.   :     0   Length:849         Min.   :   0.0
 Class :character   1st Qu.:0.0000   1st Qu.: 32156   Class :character   1st Qu.: 182.0
 Mode  :character   Median :0.0000   Median : 70153   Mode  :character   Median : 354.0
                    Mean   :0.2839   Mean   : 95717                      Mean   : 483.8
                    3rd Qu.:1.0000   3rd Qu.:117361                      3rd Qu.: 609.0
                    Max.   :1.0000   Max.   :859297                      Max.   :7748.0

 most_rep_word      n_most_rep_word      ntags          n_attribs     most_rep_attrib
 Length:849         Min.   :  1.00   Min.   :   6.0   Min.   :   5   Length:849
 Class :character   1st Qu.: 13.00   1st Qu.: 263.0   1st Qu.: 531   Class :character
 Mode  :character   Median : 25.00   Median : 456.0   Median :1018   Mode  :character
                    Mean   : 35.55   Mean   : 549.3   Mean   :1264
                    3rd Qu.: 44.00   3rd Qu.: 712.0   3rd Qu.:1609
                    Max.   :611.00   Max.   :3316.0   Max.   :8975
                    NA's   :12
 n_most_rep_attrib most_rep_tag       n_most_rep_tag      headers         hrefs_tags
 Min.   :   1.0    Length:849         Min.   :   1.0   Min.   :  0.00   Min.   :  0.00
 1st Qu.: 137.0    Class :character   1st Qu.:  70.0   1st Qu.:  4.00   1st Qu.: 44.00
 Median : 257.0    Mode  :character   Median : 133.0   Median : 10.00   Median : 72.00
 Mean   : 353.4                       Mean   : 184.4   Mean   : 14.04   Mean   : 85.17
 3rd Qu.: 471.0                       3rd Qu.: 237.0   3rd Qu.: 19.00   3rd Qu.:108.00
 Max.   :2893.0                       Max.   :1526.0   Max.   :206.00   Max.   :662.00

    div_tags           span            tables          tables_tr         tables_td
 Min.   :   0.0   Min.   :  0.00   Min.   : 0.0000   Min.   : 0.0000   Min.   :  0.0000
 1st Qu.:  64.0   1st Qu.: 11.00   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.:  0.0000
 Median : 124.0   Median : 28.00   Median : 0.0000   Median : 0.0000   Median :  0.0000
 Mean   : 170.4   Mean   : 47.86   Mean   : 0.3251   Mean   : 0.7727   Mean   :  0.9458
 3rd Qu.: 231.0   3rd Qu.: 62.00   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.:  0.0000
 Max.   :1300.0   Max.   :769.00   Max.   :40.0000   Max.   :94.0000   Max.   :216.0000

    scripts
 Min.   :  0.00
 1st Qu.: 13.00
 Median : 24.00
 Mean   : 30.33
 3rd Qu.: 39.00
 Max.   :171.00
```

Figura 11. Sumario de los parámetros extraídos de la batería 2. Fuente: propia

Los comentarios al respecto de las webs agroalimentarias siguen la misma línea que los referidos al dataset genérico. Si bien aquí, y debido a que el volumen de individuos era sustancialmente menor, la limitación del tamaño de las webs fue aumentado a 900.000 caracteres, la media de tamaño es incluso menor que la encontrada en la primera batería. La media de tablas encontradas también baja en un 50%, así como las etiquetas de tipo span, div o headers. SI nos fijamos en la media de la variable export, existe un desbalanceo de clases menor que con su contraparte más genérica, pero cabe destacar que en ambos conjuntos se observa una predominancia de población hacia el sector no exportador, factor que suavizaremos en los modelos gracias a la función **SMOTE**[14] comentada en el apartado de librerías empleadas.

7 CONOCIMIENTO EXTRAÍDO Y EVALUADO DE LOS MODELOS

En las siguientes páginas, y de forma segmentada, pasaremos a comentar cuáles han sido los modelos empleados en la experimentación referente a predecir la variable export con los distintos parámetros definidos en apartados anteriores. Nuestro principal objetivo es, con estos datos, poder llegar a predecir con la mayor precisión posible cuáles de nuestras empresas exportan y cuáles no. Vamos a emplear 3 modelos diferentes, sobre ambos datasets, y de cada modelo se comentará la experimentación realizada, qué parámetros han sido los más influyentes, qué combinaciones de las experimentadas han resultado más interesantes y cuáles han sido los resultados obtenidos.

7.1 Random Forest

Los modelos predictivos basados en árboles de decisión son uno de los modelos de machine learning más populares y utilizados tanto en investigación como en desarrollo, junto a las regresiones lineales y logísticas, por su combinación de antigüedad, robustez y fiabilidad. Definidos como una evolución del bagging, este método combina la selección aleatoria de atributos con el bagging propiamente dicho[16], dando como resultado un modelo realmente fiable con conjuntos de datos considerablemente grandes, y cuyo mayor inconveniente puede ser el sobreajuste del modelo desarrollado, factor influenciado a más sobre todo por el número de variables y de individuos.

Ya con el concepto más claro, pasamos a poner en marcha este mismo paradigma sobre los dos conjuntos de datos que teníamos preparados

```
> print(rf)

Call:
 randomForest(formula = export ~ ., data = train, importance = TRUE,      proximity = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 4

        OOB estimate of  error rate: 20.97%
Confusion matrix:
     0   1 class.error
0 4651 140  0.02922146
1 1118  90  0.92549669
> test.predicted <- predict(rf,test)
> table(test$export, test.predicted)
   test.predicted
       0    1
  0 1618   29
  1  339    8
> sum(test.predicted==test$export)/nrow(test)
[1] 0.8154463
```

Figura 12. Random forest sobre batería 1, seed(1) y train-test 75-25. Fuente: propia

```
> rf <- randomForest(export ~ ., data=train, importance=TRUE, proximity=TRUE)
> print(rf)

Call:
 randomForest(formula = export ~ ., data = train, importance = TRUE,      proximity = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 4

        OOB estimate of  error rate: 32.7%
Confusion matrix:
    0  1 class.error
0 420 29  0.06458797
1 177  4  0.97790055
> test.predicted <- predict(rf,test)
> table(test$export, test.predicted)
   test.predicted
      0   1
  0 150   1
  1  60   0
> sum(test.predicted==test$export)/nrow(test)
[1] 0.7109005
```

Figura 13. Random forest sobre batería 2, seed(1) y train-test 75-25. Fuente: propia

Como podemos ver en las dos imágenes superiores, ambos modelos sufren un sobreentrenamiento tal que únicamente clasifican bien las empresas que no exportan, ya no por ser muy finos en esa tarea particular, sino porque clasifican casi la totalidad de la población sobre esa clase. En estos casos es recomendable acompañar el accuracy de alguna otra métrica u observación, ya que puede dar lugar a conclusiones erróneas.

Ya que esta es la clase mayoritaria en ambas baterías, y viendo que los modelos probados no clasifican de manera eficiente, decidimos aplicar técnicas de desbalanceo de clases. Por medio de la función **SMOTE** del paquete **DMwR**, pasamos a rellenar el *dataframe* con nuevas entradas, en base a la proximidad de los individuos de la clase seleccionada con sus congéneres (valor establecido en *k* vecinos, como en *clustering* o similares).

Una vez recalibrados los conjuntos de entrenamiento, los individuos de cada clase han aumentado de forma inversamente proporcional al porcentaje de los mismos sobre el total de su población, siendo mayor el aumento cuantos menos individuos tuviesen. Para que el algoritmo no sólo afectase a la clase minoritaria, sino que su efecto fuese más homogéneo a todo el conjunto de datos, también se han creado individuos nuevos pasa la clase mayoritaria, siendo este aumento considerablemente menor al de su contraparte.

Ya con las nuevas baterías, procedimos a evaluar de nuevo los modelos, para así poder deducir si el algoritmo aplicado para suavizar el desbalanceo tiene un efecto positivo en la clasificación final sobre el conjunto de test.

```
prop.table((table(test_agro$export, test_agro.p
 test_agro.predicted
           0         1
0 0.06470588 0.39411765
1 0.04411765 0.49705882
```

Figura 14. Train con SMOTE sobre la nueva batería 2. Fuente: propia

Sobre las empresas agroalimentarias sucede más de lo mismo. Las clasificaciones realizadas durante el entrenamiento dejan ver unos aciertos por clase superiores al 90% en ambas. Pero al validar el modelo con los datos de test, vuelve a clasificar a la mayoría de individuos en exportadores. Al ver tanta similitud en la clasificación de ambos modelos, y a fin de intentar ver diferencias entre las diferentes fuentes de datos, decidimos desgranar la importancia de cada una de las variables sobre cada uno de los modelos, para así poder deducir si había algún conjunto de parámetros donde se concentra el poder de decisión o, si por el contrario, los modelos repartían los pesos de cada uno de estos de manera equilibrada.

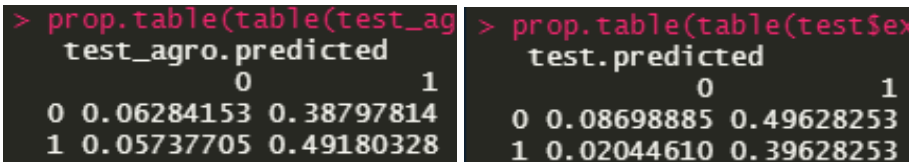
```
> importance(rf)
                        0         1 MeanDecreaseAccuracy MeanDecreaseGini
syze_html           86.96281  15.79823             87.80682          398.0213
lang                78.97720  18.76969             75.12635          526.1849
n_words             75.79223  38.82882             76.46288          483.8325
most_rep_word       78.38028  14.42485             73.94133          590.2641
n_most_rep_word     89.95753  11.57800             91.34147          452.1506
ntags               57.44554  32.51093             72.03142          335.7502
n_attribs           66.23644  31.58716             78.90755          346.8544
most_rep_attrib     94.84119  13.17595             88.07265          562.9792
n_most_rep_attrib   61.41581  29.61300             72.59977          340.2182
most_rep_tag        96.12395 -11.65504             93.27871          565.7994
n_most_rep_tag      55.63723  25.21078             59.58095          390.1085
headers             57.90647  35.22987             57.34609          351.1735
hrefs_tags         100.64270  41.79654            103.04638          459.4805
div_tags            67.31728  24.03105             76.53833          470.8578
span                69.79140  44.59608             78.04455          310.6953
tables              36.44488  26.38011             36.56452          240.9842
tables_tr           24.40585  11.27790             25.03990          106.5964
tables_td           27.23625  10.51515             30.31491           43.1599
scripts             86.13577  46.40386             90.41102          369.9632
```

```
> importance(rf_agro)
                        0         1 MeanDecreaseAccuracy MeanDecreaseGini
syze_html           36.92885 11.805723             36.86052         56.562879
lang                29.46363  7.363109             28.31968         38.448634
n_words             41.76723 25.458324             44.71784         63.333891
most_rep_word       58.85389 18.051434             57.17639         85.548767
n_most_rep_word     53.63546 12.592495             51.94434         75.575958
ntags               35.92785 13.976895             42.05026         52.355479
n_attribs           45.43339 21.440586             47.97339         60.671848
most_rep_attrib     28.30636  5.130196             27.77552         23.637367
n_most_rep_attrib   37.17245 19.345149             43.58157         53.260084
most_rep_tag        56.29472 16.982158             54.96829         88.271226
n_most_rep_tag      34.26129 21.126112             40.78091         54.787930
headers             40.09664 27.069449             42.01988         65.473359
hrefs_tags          51.27836 23.918795             56.23734         65.522851
div_tags            38.96986 17.998743             43.90666         66.620151
span                47.50252 25.585956             53.71117         63.531486
tables              19.78460 11.206724             18.91854         17.851235
tables_tr           20.93438  9.313482             21.36645         14.423721
tables_td           15.02534  7.930048             16.06691          6.508577
scripts             41.53124 25.085155             42.37285         58.973476
```

Figura 15 y 16. Importancia de las variables en random forest para las baterías 1 y 2 respectivamente Fuente: propia

Una vez yendo al detalle de los modelos sobre la influencia aislada de cada uno de los input utilizados, sí que podemos apreciar diferencias claras entre ambas experimentaciones. Si nos fijamos en la columna de *MeanDecreaseAccuracy*, cuyo valor nos da a conocer cuánta precisión perdería el modelo al prescindir de cada variable, se observa cómo las variables en nuestro primer modelo tienen más peso individual que las del segundo, suceso que se repite de igual forma con los valores para *MeanDecreaseGini*, cuyo valor indica cuánto aporta esa variable a la homogeneidad de las hojas del árbol de decisión. Añadiendo a nuestra lectura el hecho de que, de forma categórica y sin excepción, todos los parámetros en ambos modelos presentan mayor importancia para clasificar a las empresas no exportadoras que a las que sí, lo más probable es que el efecto de SMOTE sobre los datos haya provocado un sobreentrenamiento de los modelos empleados, fenómeno que seguramente venga influenciado por todos los individuos nuevos generados durante el balanceo de poblaciones de ambos conjuntos.

A fin de confirmar nuestras sospechas y poder discernir con objetividad a qué se debían estos resultados tan decepcionantes, decidimos hacer una normalización de clases menos agresiva, reduciendo los valores de los parámetros empleados en las mismas de k = 5 a k = 2, y también reduciendo el número máximo de nuevos individuos generados por SMOTE de 600 por clase a 200, para así poder realizar una comparativa entre las diferentes configuraciones de poblaciones.



Figuras 17 y 18. Segunda experimentación con random forest. Fuente: propia

Sin entrar demasiado en detalles, y propiciado por la similitud de los resultados obtenidos, el cambio de parámetros en el balanceo del dataset no mejora las predicciones sobre el conjunto de test, clasificando nuevamente al 90% de los individuos como empresas exportadoras. Esto nos hace pensar que tal vez sea problemática de la naturaleza de los random forest, y que cabe la posibilidad de que estos modelos no sean los adecuados para nuestro problema, o bien que los datos de entrada proporcionados no son los adecuados para llevar a cabo esta tarea de clasificación. A fin de despejar esta incógnita, experimentaremos con varios modelos más para poder llegar a conclusiones más contrastadas y fundamentadas.

7.2 Redes Neuronales

Las redes neuronales son un modelo de inteligencia artificial estructurado por capas de proceso, formadas a su vez por lo que se conoce por neuronas. Básicamente es un modelo que intenta imitar la estructura del cerebro humano, pasando la información desde los inputs establecidos a través de la capa o capas

diseñadas de n neuronas, y variando los pesos (o weights) de éstas en base a las iteraciones internas que sufre con los diferentes inputs que recibe[17].

Para llevar a cabo nuestra experimentación con las redes neuronales, y teniendo en cuenta las características de las mismas, sus fortalezas a la hora de evaluar conjuntos de datos y algunos requerimientos más, hemos tomado una serie de decisiones comunes a todas las experimentaciones realizadas

- Los conjuntos de datos, tanto de train como de test, han sido normalizados en base a la media numérica de cada variable. Esta decisión viene dada por consenso general en base a la experimentación, ya que es sabido que las redes neuronales simples tienen mejores resultados sobre conjuntos de datos normalizados que sobre datasets donde la varianza sea mayor, sobre todo por la convergencia más temprana de los pesos de las neuronas hacia sus valores óptimos, lo que conlleva menos iteraciones o “epochs”.

- Las variables de tipo factor han sido transformadas a tipo numérico, y las variables categóricas han sido eliminadas de estos modelos de red neuronal. Estas dos decisiones vienen dadas porque las redes neuronales sólo son capaces de procesar números para así poder variar sus pesos correctamente, y por ende no saben trabajar con otra cosa que no sea un valor numérico (tanto enteros como decimales).

Si bien podríamos haber transformado las variables categóricas a variables *dummy* para llevar a cabo la experimentación, y viendo en la práctica la eficacia de estas mismas en modelos complejos independientemente de la codificación empleada[18], decidimos no utilizarlas en esta experimentación con la finalidad de intentar diferenciar los resultados de los obtenidos con random forest. De esta forma, si con redes neuronales obtenemos unos resultados sustancialmente mejores que con random forest, podremos discernir diferencia evidente entre las variables de un tipo y las de otro, e incluso deducir que los parámetros categóricos definidos tienen relevancia nula a la hora de clasificar a las webs.

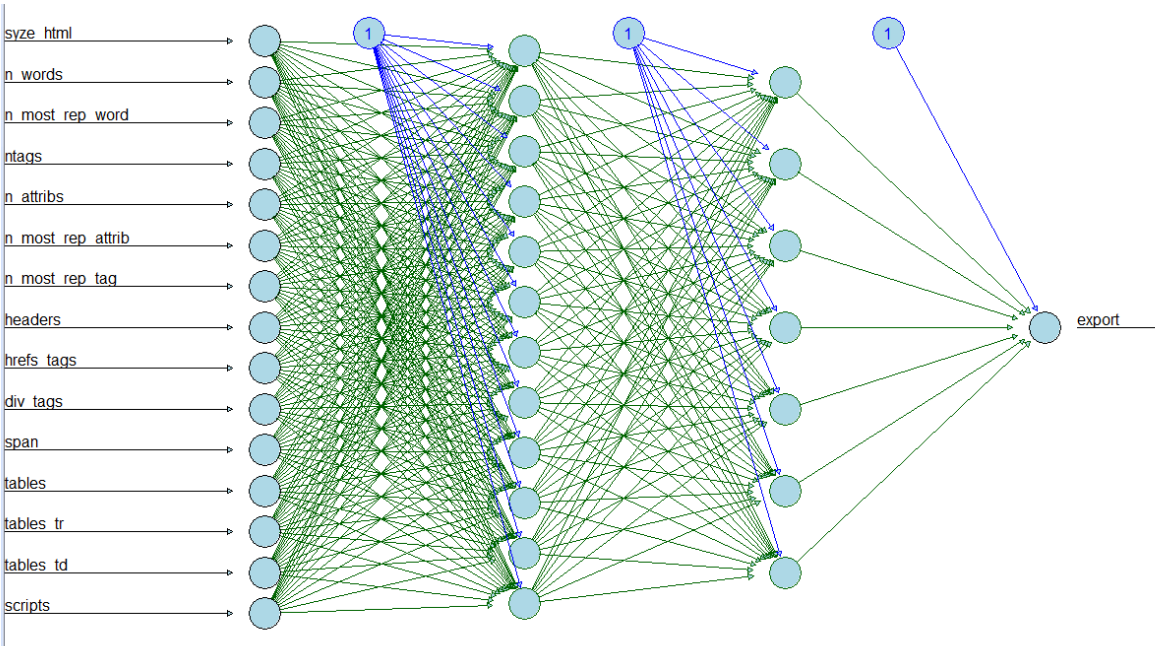


Figura 19. Red neuronal de dos capas con las variables numéricas. Fuente: propia

Como se aprecia en la imagen superior, la red neuronal propuesta es un perceptrón multicapa, siendo este modelo uno de los más populares dentro del machine learning. Teniendo en cuenta la cantidad de variables empleada y que las baterías no son necesariamente amplias, no nos planteamos ir a redes neuronales más complejas, ya que suelen estar indicadas a problemas de mayor complejidad u orientadas a conjuntos de datos con más bagaje a fin de tener un training lo más robusto posible.

Una vez definida, pasamos a entrenar la red con ambas baterías y validarla con sus respectivas particiones de test. El resultado obtenido fue el siguiente:

```
> table(pred)
pred
   0    1
1539 1272
> table(testtarget$export)

   0    1
2286  525
```

```
> table(pred)
pred
  0   1
 36 219
> table(testtarget$export)

  0   1
188  67
```

Figuras 20 y 21. Predicciones frente a valores reales utilizando la red neuronal propuesta en batería 1 y 2 respectivamente. Fuente: propia

En vista de las predicciones realizadas sobre ambas baterías con la red neuronal explicada, ahora nuestra clase mayoritaria en cuanto a la clasificación obtenida se refiere es la no exportadora. Si bien ahora las particiones de test son ligeramente superiores que las utilizadas en random forest (30% frente al 25% anteriormente empleado), las validaciones de los modelos siguen sin dar unos resultados buenos, ya que en todas las experimentaciones ha clasificado a la mayoría

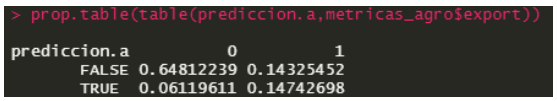
de individuos en una clase, indistintamente de si las clases del dataset estaban balanceadas o no.

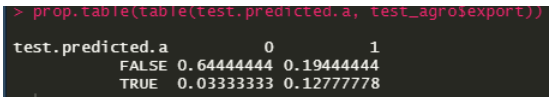
Ya con dos modelos entrenados, y en vista de las diferentes combinaciones de limpiezas, formatos, distribuciones y parámetros empleados, empezamos a creer que sencillamente no estamos haciendo una experimentación adecuada o los parámetros definidos no son útiles para esta finalidad. A fin de intentar despejar al máximo estas dudas, procedemos a realizar la última experimentación.

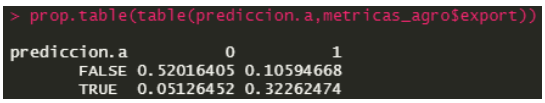
7.3 Regresión

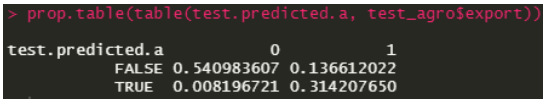
La regresión en estadística, de forma general, es un proceso en el que se pretende estimar las relaciones entre variables de un conjunto de datos, gastando una de ellas como variable dependiente y una o más como variables predictoras[3]. Si bien su complejidad de cómputo es menor que la de los dos modelos previamente empleados, eso no conlleva implícitamente que sus resultados deban de ser peores a los ya obtenidos.

Por medio de la función **glm** del paquete **stats**, entrenamos dos regresiones sobre las dos baterías desarrolladas, en primera instancia aplicando SMOTE sobre los datos como hemos hecho en anteriores experimentaciones, y en la misma experimentación sin aplicarle ninguna medida en contra del desbalanceo, para ver la influencia de este sobre un modelo más simple que los anteriores









Figuras 22, 23, 24 y 25. Predicción de entrenamiento y validación sobre batería 2 sin SMOTE (imágenes superiores) y con SMOTE (imágenes inferiores) respectivamente. Fuente: propia

A diferencia de las experimentaciones previas, con regresión comenzamos a tener valores considerablemente más buenos. Apreciando las imágenes superiores, vemos que tanto los datos originales como la población normalizada con SMOTE tienen porcentajes de acierto considerablemente buenos, superando en ambos casos el 80% de acierto en la validación con los conjuntos de test.

Al ver estos resultados, empezamos a desconfiar de los mismos por distar tanto de los vistos previamente con otros modelos, pero tras diversas comprobaciones de origen de datos, formato, parámetros y porcentajes de entrenamiento, llegamos a la

conclusión que sencillamente la regresión entrenaba mejor los parámetros creados por el paquete que los modelos vistos hasta ahora.

A raíz de experimentar con las regresiones simples, observamos algo llamativo: el formato de los datos afectaba y mucho a la clasificación que hacían estos modelos de los datos, teniendo variaciones sobre todo con las variables categóricas al representarlas como una cadena de caracteres, una variable dummy o un objeto de tipo factor.

8 CREACIÓN DEL PAQUETE

8.1 Especificaciones generales

Tras poder apreciar toda la experimentación realizada, ver qué modelos tenían mayor robustez ante las webs de entrenamiento empleadas y ver el posible potencial que pueden tener las métricas que hemos desarrollado, en este apartado procederemos al montaje del mismo. Todo este proceso se ha realizado siguiendo el artículo de In Song Kim, Phil Martin, Nina McMurry y Andy Halterman (18 de Marzo de 2018) sobre montaje de paquetes en R empleando RStudio, en particular la parte redactada para el sistema operativo de Windows[19].

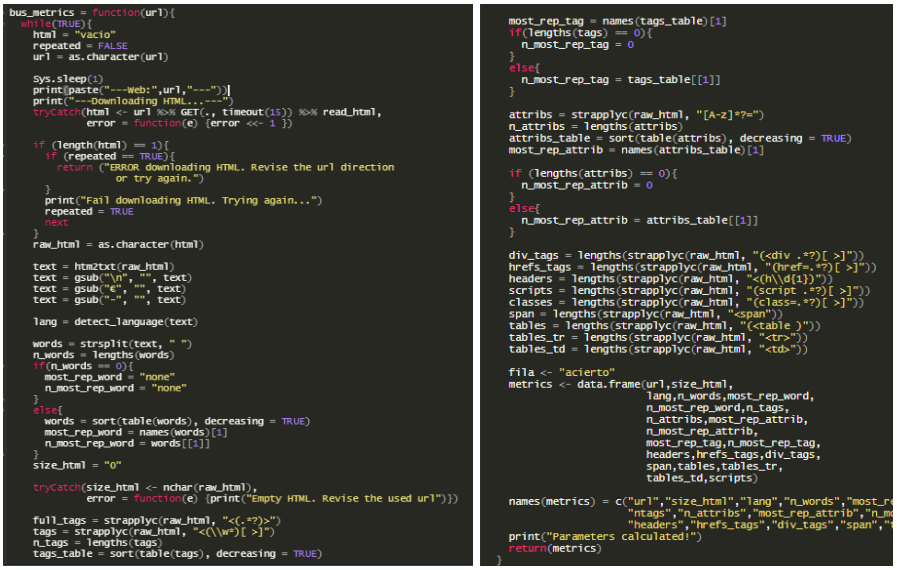
Dentro de este paquete, implementamos todo el código desarrollado, que comprende el algoritmo de extracción de información de una web dada por el usuario, y el posterior montaje de todas las variables extraídas en base a la url y su html asociado, en una única función llamada **busmetrics**.

Siguiendo las instrucciones, creamos el paquete con nuestra función y subimos el mismo a nuestro repositorio de GitHub, a fin de permitir el acceso al mismo a cualquier usuario que se preste a utilizarlo. Para su descarga y puesta en uso en cualquier ordenador, bastará con acceder a GitHub, buscar el repositorio asociado a la librería y descargar los archivos asociados a la misma para poder emplearlos. En este mismo repositorio se explica de forma breve el origen, motivo y utilidad del paquete, así como algunas especificaciones generales para su instalación.

8.2 Función “busmetrics”

Esta función, como es de esperar en cualquier desarrollo que maneje buenas prácticas de trabajo, tiene excepciones y mensajes preparados para los diferentes errores que pueda llegar a dar la web asociada. Si bien no cuenta con su propio menú de debugging o con una traza compleja de los errores que se pueden llegar a generar, los mensajes que hemos preparado le dan una idea clara al usuario de porqué la web que ha consultado no es válida, no ha podido ser extraída, no tiene texto asociado o demás factores que pueden llegar a suceder.

La estructura del algoritmo es totalmente secuencial, con apenas una llamada recursiva en caso de fallo de descarga del html. Las métricas se definen por grupos, siendo las relacionadas con el texto las primeras en ser generadas debido a su mayor tendencia a dar errores que el resto, y yendo inmediatamente después todas aquellas relacionadas con las etiquetas.



```
bus_metrics = function(url){
  while(TRUE){
    html = "vacio"
    repeated = FALSE
    url = as.character(url)

    Sys.sleep(1)
    print(paste("---Web:",url,"---"))
    print("---Downloading HTML...---")
    tryCatch(html <- url %>% GET(., timeout(15)) %>% read_html,
             error = function(e) {error <<- 1 })

    if (length(html) == 1){
      if (repeated == TRUE){
        return ("ERROR downloading HTML. Revise the url direction
                or try again.")
      }
      print("Fail downloading HTML. Trying again...")
      repeated = TRUE
      next
    }
    raw_html = as.character(html)

    text = htm2txt(raw_html)
    text = gsub("\n", "", text)
    text = gsub("€", "", text)
    text = gsub("-", "", text)

    lang = detect_language(text)

    words = strsplit(text, " ")
    n_words = lengths(words)
    if(n_words == 0){
      most_rep_word = "none"
      n_most_rep_word = "none"
    }
    else{
      words = sort(table(words), decreasing = TRUE)
      most_rep_word = names(words)[1]
      n_most_rep_word = words[[1]]
    }
    size_html = "0"

    tryCatch(size_html <- nchar(raw_html),
             error = function(e) {print("Empty HTML. Revise the used url")})

    full_tags = strapplyc(raw_html, "<(.*?)>")
    tags = strapplyc(raw_html, "<(\\w*)[ >]")
    n_tags = lengths(tags)
    tags_table = sort(table(tags), decreasing = TRUE)
```

```
    most_rep_tag = names(tags_table)[1]
    if(lengths(tags) == 0){
      n_most_rep_tag = 0
    }
    else{
      n_most_rep_tag = tags_table[[1]]
    }

    attribs = strapplyc(raw_html, "[A-z]*?=")
    n_attribs = lengths(attribs)
    attribs_table = sort(table(attribs), decreasing = TRUE)
    most_rep_attrib = names(attribs_table)[1]

    if (lengths(attribs) == 0){
      n_most_rep_attrib = 0
    }
    else{
      n_most_rep_attrib = attribs_table[[1]]
    }

    div_tags = lengths(strapplyc(raw_html, "(<div .*?)[ >]"))
    hrefs_tags = lengths(strapplyc(raw_html, "(href=.*?)[ >]"))
    headers = lengths(strapplyc(raw_html, "<(h\\d{1})"))
    scripts = lengths(strapplyc(raw_html, "(script .*?)[ >]"))
    classes = lengths(strapplyc(raw_html, "(class=.*?)[ >]"))
    span = lengths(strapplyc(raw_html, "<span"))
    tables = lengths(strapplyc(raw_html, "(<table )"))
    tables_tr = lengths(strapplyc(raw_html, "<tr>"))
    tables_td = lengths(strapplyc(raw_html, "<td>"))

    fila <- "acierto"
    metrics <- data.frame(url,size_html,
                          lang,n_words,most_rep_word,
                          n_most_rep_word,n_tags,
                          n_attribs,most_rep_attrib,
                          n_most_rep_attrib,
                          most_rep_tag,n_most_rep_tag,
                          headers,hrefs_tags,div_tags,
                          span,tables,tables_tr,
                          tables_td,scripts)

    names(metrics) = c("url","size_html","lang","n_words","most_r
                       "ntags","n_attribs","most_rep_attrib","n_m
                       "headers","hrefs_tags","div_tags","span","
    print("Parameters calculated!")
    return(metrics)
  }
```

Figuras 26 y 27. Recortes de pantalla de la función busmetrics en RStudio. Fuente: propia

Como se puede llegar a apreciar en las imágenes, se utilizan diversas variables de apoyo en formato de lista, desde las cuales se van extrayendo de forma segmentada, y por medio de las expresiones regulares, las métricas definidas, para finalmente darles formato de tipo **data.frame()** y devolverle al usuario una fila con todos los resultados. El formato de salida hemos decidido empaquetarlo como una tabla antes que como un vector, una matriz o una simple lista de valores porque de cara a realizar visualizaciones, trabajar con modelos o hacer formateos a demanda, estos citados objetos tienen limitaciones o especificaciones que los *dataframes* suelen o bien evadir o solucionar mediante *built-in functions*.

9 CONCLUSIONES

9.1 Conclusiones

Para finalizar con este proyecto, en estas líneas recogeremos las ideas principales, los resultados más significativos, nuestra opinión con respecto al desarrollo y, en resumen, las conclusiones extraídas del trabajo realizado. Tomando como partida los objetivos que nos propusimos, deducimos que:

- Nuestra mayor meta, que era poner en producción un paquete en R, la hemos llevado a cabo con éxito. El paquete es fiable en cuanto a descargas se refiere, robusto frente a diferentes equipos y direcciones web, cumple con su cometido en cuanto a parámetros calculados y formato de los mismos y, lo que más nos importaba, resulta de utilidad real a la hora de ver las características de una web.

- La disponibilidad del paquete, gracias a la plataforma GitHub, hace que consigamos mantener las máximas de accesibilidad y robustez que buscábamos en nuestro desarrollo, ya que al ser código público y con

- En vista de los resultados obtenidos en la experimentación, las métricas generadas por la librería sí tienen un potencial real en técnicas de análisis o clasificación, como hemos podido comprobar en la experimentación realizada con regresión logística.

Ya entrando en detalles más específicos, todo el desarrollo, documentación y experimentación realizados nos ha hecho ver en qué no hemos acertado, cuáles han sido los puntos menos brillantes del trabajo y qué partes no se han abordado de la manera más productiva o eficaz (como las expresiones regulares). De forma complementaria a nuestros objetivos, y en base al trabajo realizado y a la experiencia vivida durante el desarrollo, principalmente en la experimentación, nos deja dos ideas que destacan sobre el resto.

- Los modelos más complejos, como random forest o redes neuronales, no dan resultados sólidos para datasets pequeños o entradas de variables poco numerosas, mientras que otros modelos de complejidad menor brillan mejor en estas situaciones de mayor sencillez.

- Si bien los resultados obtenidos de la mayoría de experimentaciones que hemos realizado no son realmente buenos, el potencial de la información que extraemos de cada una de las empresas sigue latente, de forma que los mismos parámetros que hemos desarrollado pueden ser traspuestos a otros proyectos o estudios.

9.2 Legado

Después del desarrollo comentado, la creación y validación de los modelos propuestos y la implementación del código de montaje de parámetros en un paquete descargable por cualquier usuario de R, hemos conseguido proporcionar de forma pública y consistente una herramienta de análisis web para todo aquel usuario, empresa u organización que quiera emplearla. No hemos previsto un mantenimiento a largo plazo de la librería, motivo por el cual podría quedar desactualizada para futuras versiones de R o de alguna de las librerías que empleamos internamente, pero siempre debería de funcionar con las versiones de librerías empleadas durante el desarrollo.

9.3 Futuros trabajos

Vistos los resultados obtenidos, la construcción completa del paquete, la utilidad extraída al mismo y la problemática solucionada durante todo el desarrollo, es evidente que hay margen de mejora en todos los apartados. Una vez terminado el proyecto al completo, hemos querido hacer una revisión general de qué cosas habríamos podido mejorar y cuáles han sido las ideas que o bien se han quedado en el tintero, o bien podrían ser una ampliación.

- **Ampliación de los parámetros definidos.** Durante todo el desarrollo hemos hecho alusión de forma recurrente al concepto del sobreaprendizaje y cómo pretendíamos evitar este en la medida de lo posible, y una de estas medidas era mediante el uso de pocas variables. A la vista queda que los resultados no han sido los mejores para la tarea propuesta, y uno de los factores que ha influido con alta seguridad es el número de variables y la naturaleza de las mismas. Aumentar el número de métricas calculadas por el paquete sería una propuesta muy acertada, ya que siempre es posible emplear sólo aquellas que te sean útiles en el estudio que pretendas hacer.

- **Extracción de información del texto.** Si bien hemos tenido en cuenta factores del texto como el número de palabras, y en general factores genéricos sin especificar en ningún apartado, para tareas tan particulares como el estudio que hemos llevado a cabo de las exportaciones, hacer análisis de palabras relacionadas con la temática como “exportaciones”, “internacional” o “asia” podría haber llevado a conclusiones y modelos más precisos.

- **Modelos preentrenados.** No le hemos dado un peso real en el paquete, pero el mismo estudio que hemos hecho, suponiendo que tuviese unos valores de precisión aceptables, podría haber supuesto la creación de un buen modelo para predecir características de las webs consultadas (en nuestro caso, las exportaciones de las mismas). De esta forma, en el propio paquete podríamos ofrecer, además de las métricas en crudo, unas predicciones factibles sobre la web consultada.

GLOSARIO

Librería: en programación, se denomina erróneamente al concepto de biblioteca informática como, simplemente, librería (a causa del false-friend library en inglés)[wikipedia]. Durante la redacción del trabajo se alude tanto al término de librería como de biblioteca para hacer referencia al mismo término.

re: regular expressions, expresiones regulares. Patrones de texto empleados en programación.

Data Mining: minado de información, concepto que hace referencia a cualquier forma de extracción de datos de una fuente en particular

Disclaimer: descargo de responsabilidad

Url: Uniform Resource Locator (Localizador de Recursos Uniforme). Dirección que es dada a un recurso único en la World Wide Web.

Brain Storming: lluvia de ideas, fase de un proyecto donde se busca aportar el mayor número de formas de abordar un problema.

Core: núcleo, parte principal de aquello a lo que se refiere. En programación, también se gasta el término de origen alemán **kernel**.

S.O.: Sistema Operativo. Conjunto de programas informáticos que gestiona los recursos software y hardware de un ordenador.

Outlayers: datos anómalos, individuos que por alguna de sus características se alejan del resto de la población.

Epochs: épocas. En machine learning, se denomina con este anglicismo a cada iteración de una red neuronal sobre el conjunto de datos empleado como input. Un término equivalente podría ser las generaciones de los algoritmos genéticos.

RStudio: software empleado para la edición y ejecución de código en R así como para el desarrollo del paquete comentado.

built-in functions: en programación, funciones “construidas” sobre el núcleo del lenguaje. Suelen estar asociadas a un paquete base del lenguaje, o a otro principal que se instala en conjunto con la versión del lenguaje actual.

BIBLIOGRAFÍA

1. Provost, F., & Fawcett, T. (2013). *Data Science for Business: What you need to know about data mining and data-analytic thinking*. " O'Reilly Media, Inc.".

2. Blazquez, D., & Domenech, J. (2018). Web data mining for monitoring business export orientation. *Technological and economic development of economy*, *24*(2), 406-428.

3. Wikipedia, the Free Encyclopedia https://en.wikipedia.org/wiki/Main_Page (Consultada 26 de Agosto de 2022).

4. García Folgado, María José; Rodríguez Gonzalo, Carmen (2014). Aspectos bibliográficos del TFG. Cómo citar y componer la bibliografía. En: Investigació i bones pràctiques al voltant del Treball de Fi de Grau. València: Neopàtria, pp. 97-118.

5. Khder, M. A. (2021). Web Scraping or Web Crawling: State of Art, Techniques, Approaches and Application. International Journal of Advances in Soft Computing & Its Applications, 13(3).

6. Zhao, B. (2017). Web scraping. Encyclopedia of big data, 1-3.

7. Glez-Peña, D., Lourenço, A., López-Fernández, H., Reboiro-Jato, M., & Fdez-Riverola, F. (2014). Web scraping technologies in an API world. Briefings in bioinformatics, 15(5), 788-797.

8. R. Diouf, E. N. Sarr, O. Sall, B. Birregah, M. Bousso and S. N. Mbaye, "Web Scraping: State-of-the-Art and Areas of Application," 2019 IEEE International Conference on Big Data (Big Data), 2019, pp. 6040-6042, doi: 10.1109/BigData47090.2019.9005594.99

9. Joo, R., Boone, M. E., Clay, T. A., Patrick, S. C., Clusella‐Trullas, S., & Basille, M. (2020). Navigating through the R packages for movement. Journal of Animal Ecology, 89(1), 248-267.

10. Introducción a R https://estadistica-dma.ulpgc.es/cursoR4ULPGC/5-librerias.html (Consultada 1 de septiembre de 2022).

11. Everything you need to crack your Next Tech Interview https://www.interviewbit.com/blog/python-libraries/ (Consultada 22 de julio de 2022).

12. Mahesh, B. (2020). Machine learning algorithms-a review. International Journal of Science and Research (IJSR).[Internet], 9, 381-386.

13. Krotov, V., & Silva, L. (2018). Legality and ethics of web scraping. Emergent Research Forum (ERF), Twenty-fourth Americas Conference on Information Systems, New Orleans, 2018.

14. Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, *16*, 321-357.

15. Ho, T. K. (1995, August). Random decision forests. In Proceedings of 3rd international conference on document analysis and recognition (Vol. 1, pp. 278-282). IEEE.

16. Matich, D. J. (2001). Redes Neuronales: Conceptos básicos y aplicaciones. Universidad Tecnológica Nacional, México, 41, 12-16.

17. Potdar, K., Pardawala, T. S., & Pai, C. D. (2017). A comparative study of categorical variable encoding techniques for neural network classifiers. International journal of computer applications, 175(4), 7-9

18. Instructions for Creating Your Own R Package, In Song Kim, Phil Martin, Nina McMurry, Andy Halterman; March 18, 2018

ANEXO

OBJETIVOS DE DESARROLLO SOSTENIBLE

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No Procede
ODS 1. **Fin de la pobreza.**			X	
ODS 2. **Hambre cero.**				X
ODS 3. **Salud y bienestar.**			X	
ODS 4. **Educación de calidad.**				X
ODS 5. **Igualdad de género.**			X	
ODS 6. **Agua limpia y saneamiento.**				X
ODS 7. **Energía asequible y no contaminante.**			X	
ODS 8. **Trabajo decente y crecimiento económico.**		X		
ODS 9. **Industria, innovación e infraestructuras.**		X		
ODS 10. **Reducción de las desigualdades**.				X
ODS 11. **Ciudades y comunidades sostenibles**.			X	
ODS 12. **Producción y consumo responsables.**				X

ODS 13. **Acción por el clima.**				**x**
ODS 14. **Vida submarina.**				**X**
ODS 15. **Vida de ecosistemas terrestres.**				**X**
ODS 16. **Paz, justicia e instituciones sólidas.**				**X**
ODS 17. **Alianzas para lograr objetivos.**	**X**			

En lo que respecta a la influencia del paquete desarrollado sobre los Objetivos de Desarrollo Sostenible (ODS) propuestos, nuestro proyecto tiene influencia sobre todos los objetivos que tienen un factor económico directo y principal, y en contraparte puede llegar a tener una utilidad real en los objetivos relacionados con el mundo de la empresa, la producción, la industria o el desarrollo económico debido a su naturaleza empresarial. A consecuencia de la predicción empresarial que realizamos mediante el modelo de predicción desarrollado, objetivos como la lucha contra la pobreza, el hambre y la desigualdad tienen cero o muy poca relación.

Siguiendo la misma línea de ideas, y al ser gestión de las administraciones públicas, la educación tampoco es un área de influencia de nuestro trabajo ni directa ni indirecta, ya que las empresas potenciales afectadas no tendrían una influencia real sobre los factores educativos. Si podría llegar a tener un ápice de influencia sobre la educación privada, pero hablar de factores tan aislados e improbables sería incoherente con el resto de objetivos comentados.

La influencia sobre la vida, tanto marina como en ecosistemas terrestres, es inexistente en nuestro proyecto de nuevo acusando la misma justificación de antes, por su naturaleza empresarial y la falta de influencias más amplias de nuestro paquete.

Por lo que podemos referirnos al trabajo de calidad, el crecimiento económico, el desarrollo de las ciudades o la innovación en industria e infraestructuras (objetivos 8, 9 y 11), nuestra librería vuelve a tener un papel que podría llegar a ser relevante en la toma de decisiones al respecto por su naturaleza, ya que en estos tres objetivos juega un papel clave la figura de la empresa, sector sobre el cual hemos basado la totalidad de pruebas de nuestro proyecto. El crecimiento económico es fruto de una serie de buenas decisiones o de factores provechosos que pueden estar influidos por nuestro proyectos, y a raíz de este crecimiento, pueden devenir mejoras en la industria, en la calidad del trabajo, en los beneficios netos de una entidad o en el desarrollo a largo plazo de la población donde se afinquen las empresas.

El punto de interés debemos colocarlo, principalmente, en el decimoséptimo objetivo de nuestra lista, las Alianzas para lograr objetivos. La posibilidad real de que nuestra librería sea relevante a la hora de tomar decisiones en un estudio, trabajo, empresa, nicho de mercado u oportunidad de negocio de cualquier ámbito son factores a tener en cuenta a la hora de considerarlo influyente en este objetivo. Si bien no es una herramienta de toma de decisiones directa, ni tampoco aporta unas previsiones muy complejas de la web o webs analizada/s, el hecho de poder saber la capacidad exportadora de una empresa de la que se desconocen sus datos a través de un recurso tan público como su propia página corporativa puede ser un factor a tener en cuenta cuando no tenemos otra información como sus ingresos, beneficios o porcentajes de margen.