



## MASTER EN DATA SCIENCE

Curso Académico 2016/2017

### **Trabajo Fin de Máster**

Aplicaciones de ciencia de datos para la detección de  
*phishing*

**Autor: José Vicente Mellado**

Director: José Felipe Ortega Soto



## **Agradecimientos**

En este apartado quiero nombrar especialmente al director de este trabajo, Don José Felipe Ortega Soto, por el interés y atención que ha puesto en mí durante estos meses de verano resolviendo mis dudas y buscando soluciones a las dificultades que iban surgiendo a lo largo de este trabajo.

Por supuesto, no puedo olvidarme de familiares, amigos y compañeros del máster (especialmente Jon Kobeaga y Jorge Navarro) que siempre me han apoyado y ayudado.



# **Índice**

|   |    |
|---|----|
| Aplicaciones de ciencia de datos para la detección de <i>phishing</i> ..... |    |
| Agradecimientos .....   | 2  |
| Resumen.....  | 3  |
| Capítulo 1. Introducción .....  | 5  |
| 1.1 Contexto del TFM y objetivos .....                                      | 5  |
| 1.2 Estructura del documento.....   | 7  |
| Capítulo 2. Infraestructura .....   | 7  |
| 2.1 Modelos matemáticos utilizados.....                                     | 7  |
| 2.2 Tecnologías utilizadas.....   | 8  |
| Capítulo 3. Desarrollo.....   | 9  |
| 3.1 Comprensión del problema.....   | 9  |
| 3.1.2 Exploración del conjunto de datos Sherlock.....                       | 9  |
| 3.2 Comprensión del conjunto de datos .....                                 | 10 |
| 3.3 Preparación de los datos.....   | 11 |
| 3.3.1 Técnicas de rebalanceo utilizadas .....                               | 11 |
| 3.4 Modelado .....  | 12 |
| 3.5 Evaluación y conclusiones .....   | 13 |
| 3.5.1 Resultados del clústering de los casos de phishing.....               | 15 |
| 3.5.2 Explicación de los modelos.....                                       | 18 |
| 3.6 Despliegue.....   | 19 |
| Capítulo 4. Uso de tecnologías Big Data en Machine Learning.....            | 21 |
| 4.1 Arquitectura .....  | 21 |
| 4.2 Ejecución.....  | 22 |
| Capítulo 5. Conclusiones .....  | 25 |
| 5.1 Conclusiones finales .....  | 25 |
| 5.2 Lecciones aprendidas.....   | 25 |
| 5.3 Líneas de trabajo futuro .....  | 26 |
| Referencias.....  | 27 |
| Anexo.....  | 29 |
| Estructura del proyecto .....   | 29 |



## **Resumen**

En un mundo en el que el número de ataques informáticos crece diariamente e impacta gravemente como a empresas o personas en su ámbito más personal, es necesario, no sólo la formación y contratación de especialistas en esta área sino también la creación de sistemas que, de manera automática, nos ayuden a prevenir y detectar ataques.

Este TFM está centrado en el tipo de ataque *phishing* que en estos últimos años ha podido tener un mayor alcance e impacto debido al uso corriente de los *smartphones* y la aparición de redes sociales.

El objetivo de este trabajo es aplicar técnicas de ciencia de datos para la detección de sitios Web maliciosos. Para el entrenamiento de los modelos se ha usado un conjunto de datos público y se han aplicado modelos como árboles de decisión, *gradient boosting*, SVMs y *random forest*, entre otros. También se ha usado la tecnología de cómputo distribuido Spark para la aceleración de los cálculos de algunos de estos modelos.

## 1.1 Contexto del TFM y objetivos

## **Capítulo 1. Introducción**

### **1.1 Contexto del TFM y objetivos**

Cada día se producen millones de ataques informáticos debido a la falta de concienciación o por agujeros de seguridad en los sistemas.

En los últimos 8 años se han expuesto más de 7.1 miles de millones de identidades en filtraciones de datos [1]. Estas filtraciones se producen en ocasiones debido a la sustracción de credenciales de cuentas con privilegios para acceder a estos datos. Desafortunadamente, hay varias técnicas para conseguir las credenciales de una cuenta personal: uno de ellos es el *phishing*.

El *Phishing* tiene como objetivo obtener información personal y de carácter sensible haciéndose pasar o imitando la apariencia de una entidad en la que se puede confiar. Los canales habituales son mediante correo electrónico, sitios Web, aplicaciones móviles y a veces incluso mediante llamadas telefónicas [2].

Los principales tipos de *phishing* son los siguientes:

- *Spear phishing*: Se trata de un ataque muy especializado para ciertos individuos, en ocasiones recopilando información personal de ellos para aumentar las probabilidades de éxito. En 2012, en un informe de la compañía Symantec se determinó que el *spear phishing* estuvo involucrado en un 91% de ataques exitosos [3].
- *Clone phishing*: Consiste en el envío de correos electrónicos con contenido u enlaces aparentemente idénticos a un contenido legítimo pero en realidad se trata de versiones modificadas con fines maliciosos.
- *Whaling*: Este ataque va dirigido a altos cargos de compañías y suele tener forma de un mensaje crítico por parte de una autoridad oficial y legítima.
- *Link manipulation*: Se trata de crear sitios Web idénticos a los originales pero con su URL modificada ligeramente para que la víctima no se dé cuenta de que no está visitando el sitio original.
- *Website forgery*: La idea es que el ataque no se acaba cuando la víctima visita el sitio Web malicioso, sino que ese comportamiento se perpetúa debido a que, por ejemplo, mediante código JavaScript, se altere en ciertas ocasiones la barra de direcciones del navegador para redirigir a la víctima a los sitios Web deseados por el atacante.

Aunque es un ataque conocido desde principios de los años noventa o incluso antes, hoy en día se sigue utilizando con éxito: En los últimos tres años, se han robado más de 3 mil millones de dólares mediante *spear-phishing emails* [4].

Este tipo de ataque no sólo afecta a empresas: también puede afectar al buen funcionamiento democrático de países enteros, como ocurrió en Estados Unidos en las últimas elecciones [5] o incluso a nivel personal debido a la filtración de fotografías íntimas de celebridades en el año 2014 [6].

## 1.1 Contexto del TFM y objetivos

Aunque muchos de estos ataques son mediante correo electrónico o aplicación móvil, por debajo suele haber un sitio Web, dado que se solicita presionar un enlace o incluso la aplicación de móvil utiliza tecnologías Web [7].

Por la historia de este tipo de ataque y los acontecimientos recientes, es preciso concienciar a la población del mismo y proponer sistemas que detecten el intento de suplantación de identidad. De esta manera, surge la idea de aplicar técnicas de ciencia de datos y *big data* para la detección de sitios Web maliciosos.

La ciencia de datos y *big data* van de la mano y juntos permiten obtener resultados óptimos en menos tiempo del habitual gracias a la computación distribuida.

Para la realización de este Trabajo de Fin de Máster se ha usado el conjunto de datos “Phishing Website” del “Machine Learning Repository” de la Universidad de Irving, California [8]. Este conjunto es fruto de la investigación de miembros de la Universidad de Huddersfield en la que disponen de una herramienta para obtener características de sitios Web y catalogarlas como legítimas o *phishing*.

Los objetivos de este trabajo son los siguientes:

1. Realizar un análisis exploratorio del conjunto de datos y una interpretación del mismo.
2. Aplicar modelos y otras técnicas de *machine learning* vistos en clase para la detección de sitios Web maliciosos.
3. Implementar el punto anterior con Apache Spark [9].

## 1.2 Estructura del documento

El contenido de este documento está organizado en los siguientes apartados:

- **Introducción:** el objetivo es exponer el contexto en el que se ubica este trabajo, las razones por las que tiene sentido la creación del sistema propuesto y los diferentes objetivos que se pretenden alcanzar en este TFG.
- **Estado del arte:** repaso de la actualidad sobre las bases de datos haciendo especial hincapié en las bases de datos no convencionales analizando también sus desventajas.
- **Planteamiento del problema:** en este apartado se exponen las características de los datos con los que vamos a trabajar, la elección del tipo de base de datos y la tecnología concreta en base al estado del arte contado anteriormente.
- **Desarrollo:** se continúa con el trabajo relativo al almacenamiento de productos, detallando el diseño de la base de datos y su arquitectura. Adicionalmente se expone la recopilación de información para nutrir de datos a la infraestructura.
- **Caso de estudio:** se muestra la creación de un sitio Web que utiliza la infraestructura desarrollada.
- **Conclusiones:** conclusiones de la realización de este trabajo, lecciones aprendidas y líneas de trabajo que podrían ser útiles para completar el trabajo aquí documentado.

## Capítulo 2. Infraestructura

### 2.1 Modelos matemáticos utilizados

Se ha realizado una exploración en la literatura para determinar qué métodos son los más utilizados en la detección de casos maliciosos y su rendimiento [10], además de revisar publicaciones que tienen como base este conjunto de datos [11] [12] [13].

De las lecturas realizadas se ha determinado que los siguientes modelos pueden ser óptimos para la resolver el problema que nos ocupa:

- Árboles de decisión
- Random Forest
- Extremely Randomized Trees
- SVM
- Gradient Boosting
- Naïve Bayes
- Redes Neuronales

Este conjunto de datos consta de 30 atributos. No es un número demasiado grande de características, pero dado que se debe intentar hacer los modelos lo más simples posible, se

## 2.2 Tecnologías utilizadas

aplicarán también técnicas para la reducción de la dimensionalidad como PCA (*Principal Component Analysis*) y LDA (*Linear Discriminant Analysis*).

Con PCA se intenta buscar una combinación de variables que explican la máxima cantidad de varianza posible y son independientes entre ellas. LDA sigue una filosofía similar pero está pensado para maximizar la diferencia entre clases.

### 2.2 Tecnologías utilizadas

Para la resolución del problema, se utilizará el lenguaje de programación Python, versión 3.5, junto con la librería Scikit Learn [14] que nos ofrece un conjunto de funciones para la realización de técnicas de *machine learning*. Para completar las carencias de Scikit Learn, se han usado también imbalanced-learn [15] para el balanceo de clases, scikit-plot [16] para la impresión por pantalla de gráficos que ayuden a explicar los modelos y Keras [17] para el desarrollo de redes neuronales.

También se ha usado Jupyter Notebook [18] de tal manera que se ha podido escribir código junto con explicaciones para un mejor seguimiento y legibilidad.

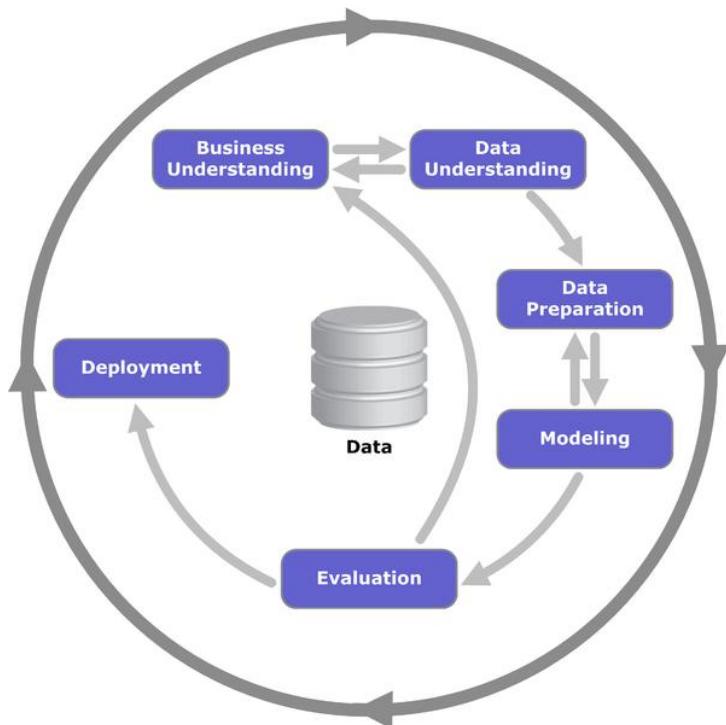
Además de estas, se han usado otras librerías muy recurrentes en un proyecto de ciencia de datos como son Pandas [19], Numpy [20], Seaborn [21] y Matplotlib [22].

La gestión de las librerías corre a cargo de Anaconda [23], haciendo posible la creación de diferentes entornos con diferentes versiones de Python y librerías, así como la compartición de estos entornos.

El motivo de haber elegido Python es el de explorar librerías de aprendizaje máquina más allá de las vistas en clase con R, y también la facilidad que existe con Python para trabajar con tecnologías como Spark.

## Capítulo 3. Desarrollo

Dado que se trata de un proyecto de ciencia de datos, se ha utilizado el estándar de flujo de trabajo *Cross Industry Standard Process for Data Mining* (CRISP-DM) [24] el cual se resume en seis fases y se trata de un proceso cíclico.



*Figura 1: Diagrama de las relaciones entre distintas fases en CRISP-DM*

### 3.1 Comprensión del problema

Estudio del problema a resolver, su impacto en la actualidad, los diferentes tipos de phishing existentes y sus técnicas para llevarlo a cabo.

En esta fase ha resultado especialmente útil la lectura de los informes que la empresa Symantec lanza anualmente [25] y la exploración del *dataset* Sherlock [26].

#### 3.1.2 Exploración del conjunto de datos Sherlock

Se trata de un conjunto de datos con datos sobre acciones realizadas por voluntarios con un teléfono móvil. Los móviles usados tienen un software que genera ataques informáticos. En estos datos se recogen muchos valores sobre el estado del móvil en determinados instantes, así como el momento en el que se realizan los ataques.

En la versión de los datos de prueba que se ha utilizado sólo había datos durante un período de dos semanas y de un único participante. Además todos los ataques eran de tipo phishing.

### 3.2 Comprensión del conjunto de datos

La mecánica del ataque consistía en crear cada cierto tiempo accesos directos a aplicaciones falsas de redes sociales, así como generar notificaciones si el usuario no las abría con el objetivo de que el individuo introdujese sus credenciales para un hipotético inicio de sesión.

En el análisis exploratorio realizado se encontraron patrones cuando se realizan ataques, especialmente en el uso de CPU y memoria, pues el uso de ambos recursos era notablemente inferior cuando se producían.

Como se ha mencionado anteriormente, se ha trabajado con la versión de prueba de conjunto de datos, por lo que a efectos prácticos había muy pocos registros con los que poder trabajar. Con la versión completa se podría, sin duda, hacer un estudio interesante sobre este conjunto de datos.

#### 3.2 Comprensión del conjunto de datos

Lectura de la documentación del conjunto de datos, así como su origen y *papers* asociados.

También se realizó un análisis exploratorio con gráficos para una rápida interpretación de los datos, así como la obtención de algunas conclusiones:

Hay algunos atributos cuyos valores permiten ayudar a distinguir claramente si un sitio Web es phishing o no. Algunos de esos atributos son:

- Prefix\_Suffix: Esta variable indica si hay un guión en la URL o no. Como vemos, siempre que no hay un guión en la URL se trata de un caso que no es phishing.
- having\_Sub\_Domain: Se puede apreciar que, para sitios benignos, no suele haber subdominios. Para los casos de phishing, predomina la presencia de un subdominio o varios.
- SSLfinal\_State: Especialmente el valor https&issuer (el sitio utiliza HTTPS y el proveedor de los certificados es confiable) es indicador de que el sitio Web no es un caso de phishing.
- URL\_of\_Anchor: Mide el porcentaje de etiquetas de enlaces HTML que redirigen a otros sitios con diferente nombre de dominio o que directamente no redirigen a ningún otro sitio. Cuando el porcentaje es menor al 31% suelen ser claramente sitios benignos pero cuando el porcentaje es mayor del 67% se suele tratar de casos de phishing.
- age\_of\_domain: En los sitios Web sanos es más común que el dominio tenga una antigüedad de 6 meses o más.
- DNSRecord: Suele haber registros del DNS en Whois o bien del propio dominio con más frecuencia de los sitios sin phishing.
- web\_traffic: La mayoría de sitios Web legítimos suelen estar en el top 100.000 del Ranking Alexa.

Para el resto de variables hay pequeñas diferencias pero no tan claras como en las variables descritas.

### 3.3 Preparación de los datos

Cabe destacar el rendimiento de la variable “Statistical Report”. Dicha variable contiene una predicción de si el sitio Web es legítimo o no en base al resultado que presentan los sitios <https://www.phishtank.com/> y <https://www.stopbadware.org/>. En el análisis exploratorio se demuestra que el resultado es pobre dado que se consigue un *accuracy* del 56.9%.

Así, se tendrá como objetivo mínimo mejorar la predicción contenida en esa variable.

#### 3.3 Preparación de los datos

Dado que los datos originales están codificados con valores de 1, 0 y -1, se han convertido estos valores en etiquetas de texto de tal manera que el análisis exploratorio sea más comprensible. Además, para la aplicación de modelos, se ha eliminado la columna “Statistical Report”.

Dentro de la preparación de los datos cabe destacar el particionado de los datos entre conjuntos de entrenamiento y test en una proporción 80/20.

Para un correcto aprendizaje, se trató de balancear el conjunto de entrenamiento con respecto a las etiquetas a predecir (las clases legítimo o phishing). Se han originado tantos conjuntos de entrenamiento como técnicas de rebalanceo se han aplicado.

Posteriormente, se dividieron los distintos conjuntos de entrenamiento resultantes 5 partes, teniendo a su vez cada una de ellas dos partes: conjunto de entrenamiento y conjunto de validación. A esta técnica se le llama Cross Validation [27].

Gracias a esto, luego será posible usar Grid Search. Esta técnica permite buscar la mejor combinación de valores de parámetros para un modelo dados unos valores previos de entrada. Se encuentra el mejor modelo usando los pliegues realizados anteriormente (se entrena con la parte de entrenamiento y se obtienen métricas con la parte de validación) y realizando la media de las métricas en cada uno.

##### 3.3.1 Técnicas de rebalanceo utilizadas

Las técnicas de rebalanceo usadas son todas de *under-sampling* (eliminar observaciones).

- *Random Under-Sampler* (USR): Consiste en eliminar, de manera aleatoria, observaciones de la clase mayoritaria hasta que se iguala el número de observaciones de todas las clases.

Para entender las siguientes técnicas de rebalanceo es preciso hablar de los siguientes términos [28]:

- Valores atípicos: Puntos que están rodeados de observaciones de otra clase.

- Prototipos: Conjunto mínimo de puntos requeridos en el conjunto de entrenamiento para que todos las observaciones que no son *outliers* sean clasificadas correctamente.
- Puntos absorbidos: Observaciones que no son *outliers* y serían correctamente clasificadas gracias al conjunto de puntos que son prototipos.

Estas técnicas son consideradas de selección de prototipos [29]:

- *Condensed Nearest Neighbour* (CNN): El propósito principal es disponer de un conjunto de entrenamiento con observaciones que son únicamente prototipos, dado que las observaciones absorbidas pueden ser clasificadas correctamente mediante KNN con  $K = 1$ .
- *One-Sided Selection* (OSS): Consiste en aplicar enlaces Tomek y después CNN. El hecho de aplicar enlaces Tomek implica que se eliminan casos de la clase mayoritaria que están cerca de la región de la clase minoritaria, obteniendo así un conjunto de datos con las clases mucho más separadas.

### 3.4 Modelado

Aplicación de modelos matemáticos para la predicción de si se trata de un caso de *phishing* o uno legítimo.

Esta parte se ha dividido a su vez en varias fases:

Una primera fase en la que se aplican modelos vistos en clase (Decision Tree, Random Forest, SVM, XGBoost) y otros cuya aplicación puede ser interesante según se recoge en la literatura (Extremely Randomized Trees, Naïve Bayes) usando un conjunto de entrenamiento balanceado de manera aleatoria.

En una segunda fase se entrenaron modelos que en la fase anterior no tenían un buen rendimiento en los conjuntos de validación de los diferentes pliegues y también aquellos que necesitaban mucho tiempo para ser entrenados. En esta ocasión, se utilizaron los conjuntos de entrenamiento fruto de usar las técnicas de rebalanceo *One-Sided Selection* y *Condensed Nearest Neighbour*.

Adicionalmente, y utilizando como base el conjunto de entrenamiento balanceado de manera aleatoria, se entrenaron algunos modelos con los resultados de transformar los datos mediante PCA (*Principal Component Analysis*) y LDA (*Linear Discriminant Analysis*), con el objetivo de reducir la dimensionalidad y aumentar la discriminación entre clases para obtener resultados mejores.

Se prosiguió utilizando técnicas de clústering sólo en las observaciones catalogadas como phishing para así caracterizar diferentes casos de sitios Web maliciosos en función de sus características. En un apartado posterior se detallará esta fase.

### 3.5 Evaluación y conclusiones

Finalmente se realizó con la librería Keras una red neuronal simple, modelo también visto en clase. En esta ocasión se utilizó el conjunto de entrenamiento al que se le aplicó *One-Sided Selection*.

### 3.5 Evaluación y conclusiones

Se evalúan los diferentes modelos con el conjunto de test, no utilizado hasta entonces.

En la Tabla 1 se pueden apreciar las distintas métricas de acierto para los distintos modelos, así como en la Tabla 2 (pag. 15) se aporta esa misma tabla ordenada de mayor a menor *Accuracy*. Dado que “legit” ha sido la clase positiva y “phishing” la clase negativa, se ha tenido muy en cuenta la medida *Precision* para controlar el número de falsos positivos (casos de phishing clasificados erróneamente como legítimos).

| <b>Modelo</b>            | <b>Accuracy</b> | <b>Precision</b> | <b>Recall</b> |
|--------------------------|-----------------|------------------|---------------|
| Decision Tree            | 0.901           | 0.90             | 0.93          |
| Random Forest            | 0.936           | 0.95             | 0.94          |
| Extra Trees Classifier   | 0.933           | 0.94             | 0.94          |
| Bernoulli Naïve          | 0.672           | 0.99             | 0.43          |
| Bayes                    |                 |                  |               |
| Multinomial Naïve        | 0.567           | 1.00             | 0.25          |
| Bayes                    |                 |                  |               |
| <u>Multinomial</u> Naïve | 0.924           | 0.92             | 0.94          |
| <u>Bayes - PCA</u>       |                 |                  |               |
| <b>XGBoost</b>           | <b>0.967</b>    | <b>0.98</b>      | <b>0.96</b>   |
| XGBoost - PCA            | 0.955           | 0.97             | 0.95          |
| XGBoost - LDA            | 0.933           | 0.95             | 0.93          |
| SVM Lineal               | 0.936           | 0.94             | 0.94          |
| SVM Polinomial           | 0.938           | 0.96             | 0.93          |
| SVM Polinomial -         | 0.918           | 0.89             | 0.97          |
| LDA                      |                 |                  |               |
| <u>SVM</u> Polinomial -  | 0.948           | 0.97             | 0.94          |
| <u>PCA</u>               |                 |                  |               |
| SVM RBF                  | 0.565           | 0.99             | 0.25          |
| <u>SVM RBF - OSS</u>     | 0.965           | 0.96             | 0.98          |
| SVM RBF - CNN            | 0.958           | 0.96             | 0.96          |
| SVM RBF - LDA            | 0.929           | 0.95             | 0.93          |
| SVM RBF - PCA            | 0.566           | 0.99             | 0.25          |
| <u>SVM Sigmoidal</u>     | 0.931           | 0.94             | 0.94          |
| <u>SVM Sigmoidal</u> -   | 0.931           | 0.93             | 0.95          |
| OSS                      |                 |                  |               |
| SVM Sigmoidal -          | 0.604           | 1.00             | 0.31          |
| CNN                      |                 |                  |               |
| LDA                      | 0.934           | 0.94             | 0.95          |
| <b>Perceptrón</b>        | <b>0.966</b>    | <b>0.96</b>      | <b>0.98</b>   |

Tabla 1- Modelos agrupados

En general, la mayoría de modelos dan muy buenos resultados siendo XGBoost (implementación de *Gradient Boosting* que intenta optimizar el uso de memoria y la velocidad de cómputo) y el perceptrón son los dos mejores modelos.

Estos dos modelos tardan bastante tiempo en ser entrenados, especialmente XGBoost. En el caso del perceptrón, se ha llegado a la conclusión de que puede ser ajustado tanto como se quiera, estando el tiempo de cómputo del mismo en función de esto. Para no originar sobreentrenamiento se ha decidido no ir más allá de un 96.6% de *accuracy*.

Las SVM son los siguientes modelos que mejores resultados arrojan. Tardaron bastante tiempo en ser entrenadas con el conjunto de entrenamiento de rebalanceo aleatorio simple, incluso con el kernel RBF no se obtienen buenos resultados. Sin embargo, esto cambia cuando se trabaja con los conjuntos de entrenamiento CNN, OSS y LDA: el tiempo de entrenamiento es muy notablemente inferior e incluso se consiguen resultados óptimos. En esto último destaca especialmente el caso de las SVM con el kernel RBF.

La reducción de tiempo de cómputo en algunos casos de SVM es lo que motivó usar el conjunto OSS como conjunto de entrenamiento para la red neuronal. La red neuronal implementada consta de una capa de entrada con 30 neuronas y una capa de salida. Se ha aplicado la técnica *dropout* [30] que consiste en desactivar un tanto por ciento de neuronas (20% en este caso) de una capa a otra para evitar el sobreajuste.

En general los modelos de Naïve Bayes no tienen buenos niveles de *accuracy*. Esto puede estar causado porque no se cumplen las premisas que Naïve Bayes asume que se cumplen: independencia entre características y que los datos estén distribuidos de una cierta manera (distribución multinomial para Multinomial Naïve Bayes y distribución multinomial de Bernoulli para Bernoullly Naïve Bayes).

| <b>Modelo</b>          | <b>Accuracy</b> | <b>Precision</b> | <b>Recall</b> |
|------------------------|-----------------|------------------|---------------|
| XGBoost                | 0.967           | 0.98             | 0.96          |
| Perceptrón             | 0.966           | 0.96             | 0.98          |
| SVM RBF - OSS          | 0.965           | 0.96             | 0.98          |
| SVM RBF - CNN          | 0.958           | 0.96             | 0.96          |
| XGBoost - PCA          | 0.955           | 0.97             | 0.95          |
| SVM Polinomial - PCA   | 0.948           | 0.97             | 0.94          |
| SVM Polinomial         | 0.938           | 0.96             | 0.93          |
| Random Forest          | 0.936           | 0.95             | 0.94          |
| SVM Lineal             | 0.936           | 0.94             | 0.94          |
| LDA                    | 0.934           | 0.94             | 0.95          |
| Extra Trees Classifier | 0.933           | 0.94             | 0.94          |
| XGBoost - LDA          | 0.933           | 0.95             | 0.93          |
| SVM Sigmoidal          | 0.931           | 0.94             | 0.94          |
| SVM Sigmoidal - OSS    | 0.931           | 0.93             | 0.95          |
| SVM RBF - LDA          | 0.929           | 0.95             | 0.93          |
| Multinomial Naïve      | 0.924           | 0.92             | 0.94          |
| Bayes - PCA            |                 |                  |               |
| SVM Polinomial - LDA   | 0.918           | 0.89             | 0.97          |
| Decision Tree          | 0.901           | 0.90             | 0.93          |
| Bernoulli Naïve        | 0.672           | 0.99             | 0.43          |
| Bayes                  |                 |                  |               |
| SVM Sigmoidal - CNN    | 0.604           | 1.00             | 0.31          |
| Multinomial Naïve      | 0.567           | 1.00             | 0.25          |
| Bayes                  |                 |                  |               |
| SVM RBF - PCA          | 0.566           | 0.99             | 0.25          |
| SVM RBF                | 0.565           | 0.99             | 0.25          |

Tabla 2 – Modelos ordenados por accuracy

### 3.5.1 Resultados del clústering de los casos de phishing

El objetivo de esta actividad ha sido agrupar los casos maliciosos en un número determinado de grupos para caracterizar los diferentes tipos de Webs maliciosas.

Para la búsqueda de grupos se ha usado el modelo KNN con un valor de K igual a 4. Este valor ha sido determinado mediante el cálculo y impresión por pantalla del índice Silhouette para diferentes valores de K. La función de distancia usada ha sido la distancia Euclídea.

A continuación se detallan las características más particulares de cada grupo:

Tipo 1 de Website con phishing:

- Request\_URL: requestUrl%\_lt\_22
- Links\_in\_tags: links%\_gt\_81
- popUpWindow: other
- web\_traffic: alexa\_gt\_100K

Este grupo de sitios Web maliciosos tiene menos de un 22% de objetos externos (vídeos, imágenes y sonidos) que proceden de dominios distintos. Esto contrasta con los enlaces en etiquetas de tipo “meta”, “script” y “link”, que contienen más de un 81% de enlaces a dominios distintos. También contienen algún tipo de ventana emergente que no solicita introducir texto. Este tipo de sitios suelen estar por debajo de la posición 100.000 del ranking Alexa [31].

Tipo 2 de Website con phishing:

- Favicon: external\_domain
- port: preferred\_status
- Links\_in\_tags: links%\_gte\_17\_lte\_81
- Submitting\_to\_email: mail\_or\_mailto
- on\_mouseover: statusbar\_change
- popUpWindow: yes&with\_form
- web\_traffic: alexa\_gt\_100K

Su “favicon” está cargado desde un dominio externo y su servidor tiene algunos puertos habituales en un estado no deseable. Además tiene un porcentaje de enlaces a otros dominios en etiquetas “meta”, “script” y “link” mayor o igual que el 17% o menor o igual que el 81%. Este grupo contiene instrucciones de tipo “mail” o “mailto:” para enviar información personal del visitante a un correo electrónico. Suelen incluir código que modifica la barra de estado del navegador así como incluir ventanas emergentes en las que introducir información personal a modo de formulario. No suelen tener una situación privilegiada en el ranking Alexa: más allá de la posición 100.000.

Tipo 3 de Website con phishing:

- Domain\_registration\_length: gt\_1\_year
- Links\_in\_tags: links%\_gt\_81
- popUpWindow: other
- web\_traffic: not\_in\_alexa

Su dominio tiene una longevidad de más de un año, así como incluir más de un 81% de enlaces en etiquetas “meta”, “script” y “link” de dominios externos. Contienen ventanas emergentes que no son de tipo formulario y ni siquiera se encuentran en el ranking Alexa.

Tipo 4 de Website con phishing:

- having\_IP\_Address: true
- Shortining\_Serice: true
- double\_slash\_redirecting: true
- HTTPS\_token: true
- Links\_in\_tags: links%\_lt\_17
- Abnormal\_URL: no\_hostname\_in\_url
- Redirect: 0\_or\_1
- popUpWindow: yes&with\_form
- age\_of\_domain: gte\_6\_months
- DNSRecord: not\_found
- web\_traffic: alexa\_lt\_100K
- Links\_pointing\_to\_page: gt\_2

Este grupo tiene muchas particularidades: Tiene como dirección una IP, utiliza servicios para acortar las URL y contiene la subcadena en la URL “//”, lo que indica que el usuario será redirigido a otros sitios Web. Contiene URLs con la subcadena “https” para engañar al usuario pensando que está o se dirige hacia un sitio seguro. Contiene menos de un 17% de enlaces a dominios distintos en etiquetas “meta”, “script” y “link” y no tiene su nombre de dominio en la URL. Son sitios Web que han sido redirigidos como mucho una vez, contienen ventanas emergentes con formularios, sus dominios datan de como mucho 6 meses atrás, su DNS no ha sido encontrado en WhoIs [32]. Sin embargo, están por encima del puesto 100.000 en el ranking Alexa y hay al menos dos sitios Web que enlazan a este tipo de páginas.

### 3.5.2 Explicación de los modelos

Exceptuando el caso del árbol de decisión, el resto de modelos son de caja negra (no se puede explicar cómo las variables de entrada afectan a la variable a explicar), por lo tanto sólo en unos pocos, concretamente en Random Forest y Extremely Randomized Trees, ha sido posible obtener la importancia de cada variable en el modelo. Para el resto de modelos Scikit Learn no implementa la función “importancia de variables”.

A continuación se presenta el árbol de decisión:

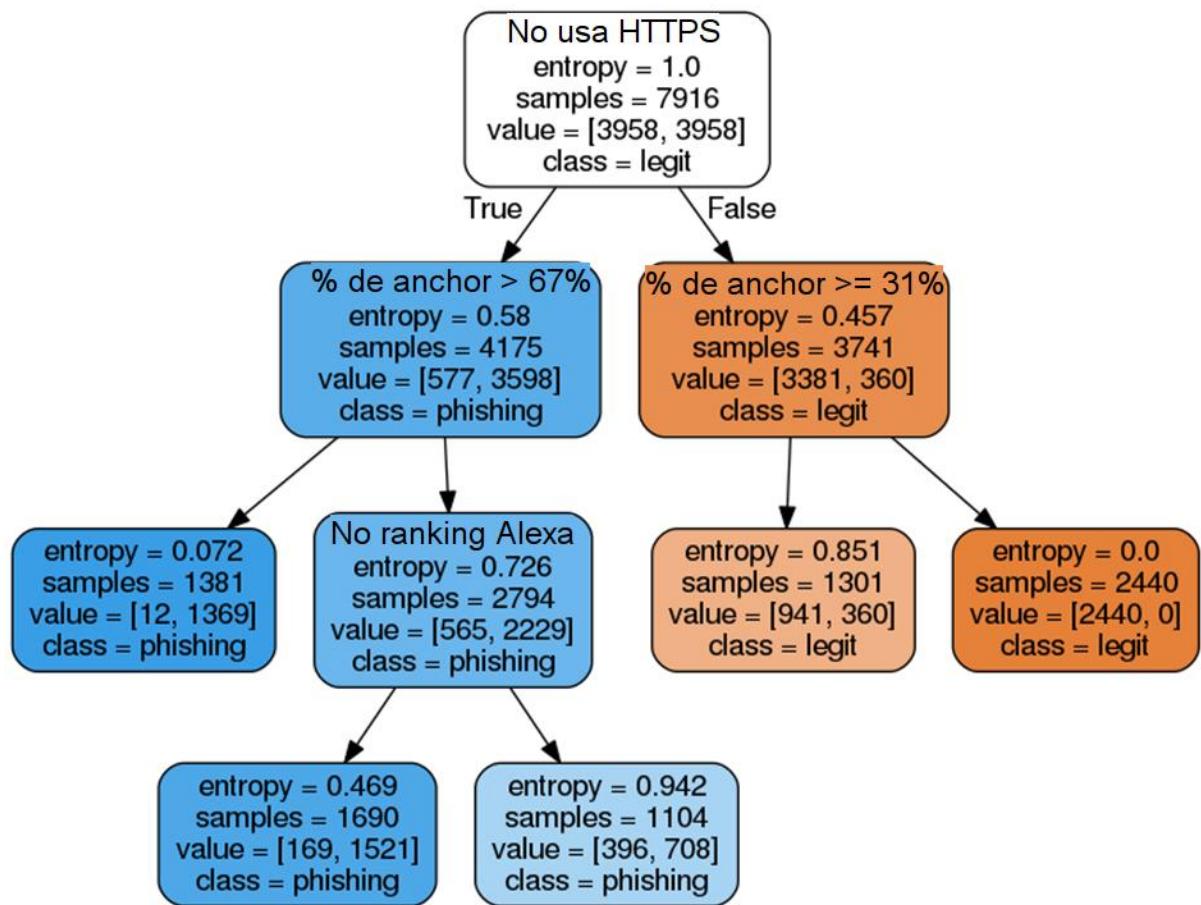


Figura 2: Árbol de decisión

Notar que los datos son categóricos y originalmente este árbol tenía valores numéricos. El gráfico anterior es una interpretación de esos valores.

### 3.6 Despliegue

Las variables que más importancia tienen según los modelos de tipo bosque de árboles son SSLFinal\_State, URL\_of\_Anchor, web\_traffic (posición del sitio Web en el ranking Alexa), Prefix\_Suffix (si la URL contiene un guión o no) y having\_Sub\_domain (número de subdominios que tiene el sitio).

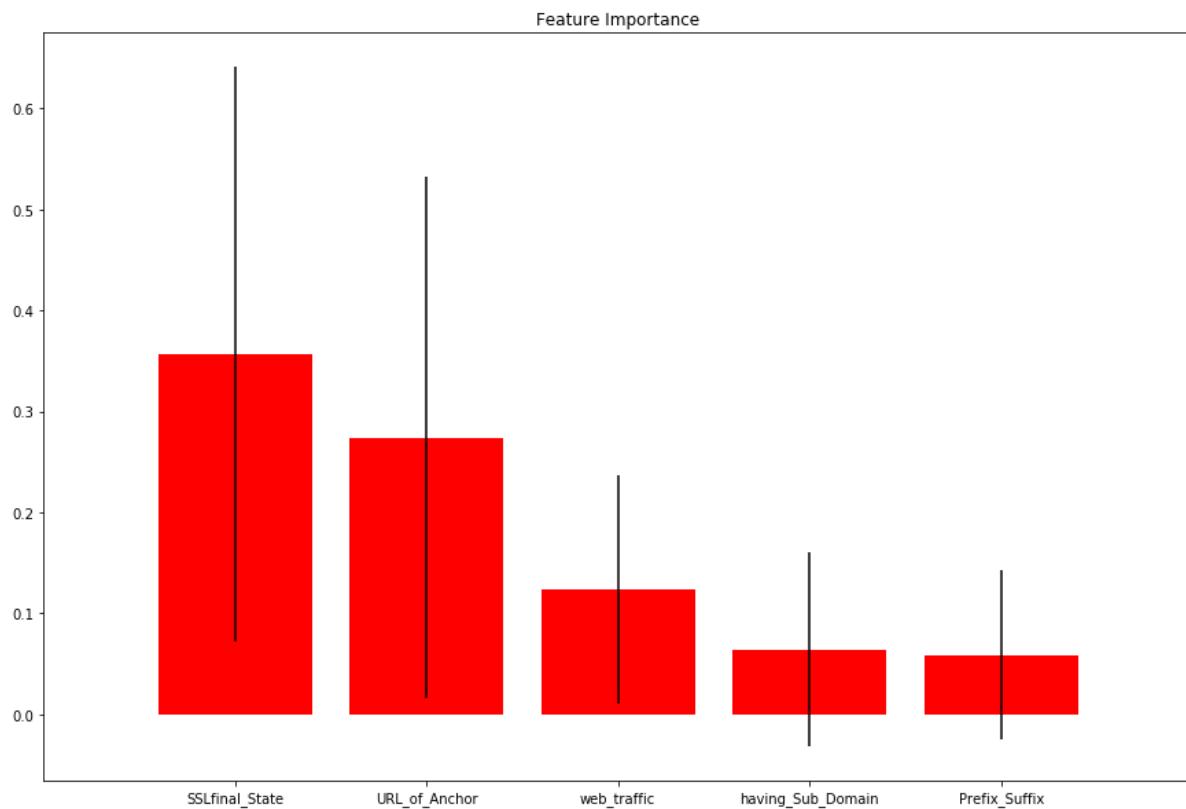


Figura 3: Importancia de las variables para Random Forest

Estas variables coinciden con aquellas que son más discriminatorias según el análisis exploratorio realizado al principio del desarrollo.

### 3.6 Despliegue

Únicamente se han guardado los modelos con los parámetros óptimos en ficheros para un posterior uso.

Por otra parte, se ha desplegado, en máquinas virtuales un clúster Spark para el cálculo de algunos modelos que en un ordenador convencional requieren mucho tiempo para entrenar. Esto último se verá en el siguiente capítulo de este documento.

### 3.6 Despliegue

## Capítulo 4. Uso de tecnologías Big Data en Machine Learning

Como se ha visto en el capítulo anterior, la ejecución de estos modelos en un computador convencional puede tomar mucho tiempo: desde varios minutos hasta varias horas.

Dado que es preciso realizar muchas pruebas para conseguir la mejor configuración del modelo, es realmente importante que cada prueba se realice en el menor tiempo posible. En esta situación, cobra especial importancia el uso de tecnologías *Big Data* de tal manera que el cálculo de operaciones se haga de manera distribuida entre varios computadores.

Por ello, se van a probar dos modelos costosos en tiempo de ejecución con las tecnologías Spark (vista en clase) y la librería de *Machine Learning* H2O [33].

### 4.1 Arquitectura

Para la ejecución del programa en Spark se han levantado tres instancias EC2 de Amazon Web Services. Dado que el presupuesto es limitado, sólo se han podido utilizar máquinas de tipo t2.medium, cuyas características principales son 2 CPU y 4 GB de memoria RAM.

De estos tres computadores, una actuará como *driver* (el que lanza la aplicación) y los otros dos como *workers* (encargados de almacenar los datos particionados y ejecutar las tareas).

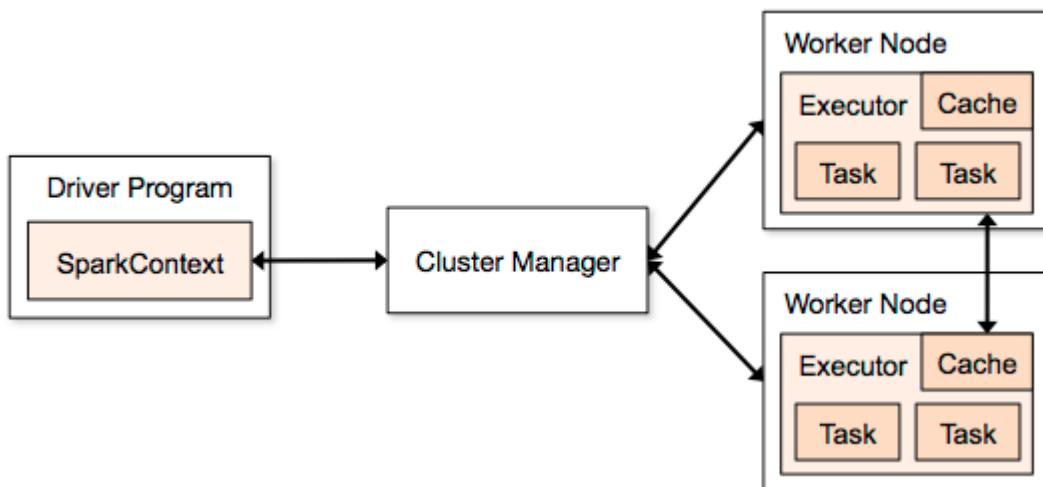


Figura 4: Diagrama del clúster Spark usado

Dado que se trata de un clúster pequeño, se utilizará el gestor de recursos que viene incorporado por defecto con Spark.

## 4.2 Ejecución

Se ha realizado la ejecución de dos modelos de costosa computación en un entorno local. Estos modelos serán Gradient Boosting y Random Forest.

Las tecnología usada ha sido Sparkling Water [34], gracias a la cual se combina la potencia de H2O, librería de *machine learning* con algoritmos muy optimizados, y la potencia de Spark, ejecutando así funciones de H2O de manera distribuida con Spark.

Esta librería tiene una herramienta muy interesante: Flow UI. Su finalidad es utilizar de manera gráfica y accesible desde un explorador de Internet todas las funcionalidades que ofrece.

Se ha intentado utilizar exactamente los mismos rangos de valores para los parámetros que se han usado con SciKit Learn, teniendo en cuenta limitaciones como que H2O no implementa exactamente los mismos parámetros que recoge SkLearn.

En el archivo “cluster commands.txt” de la carpeta “assets” se detallan tanto los mandatos de consola para el levantamiento de clúster Spark como el lanzamiento de la aplicación Spark H2O. Apréciese que tamaño elegido para el nodo *driver* es de 3 GB, lo que supone utilizar el 75% de la memoria. De esta manera se deja libre el 25% de la memoria para gestiones de Spark.

A continuación se adjuntan algunas capturas de la WebUI de Spark mientras se ejecutaba el entrenamiento de los modelos:

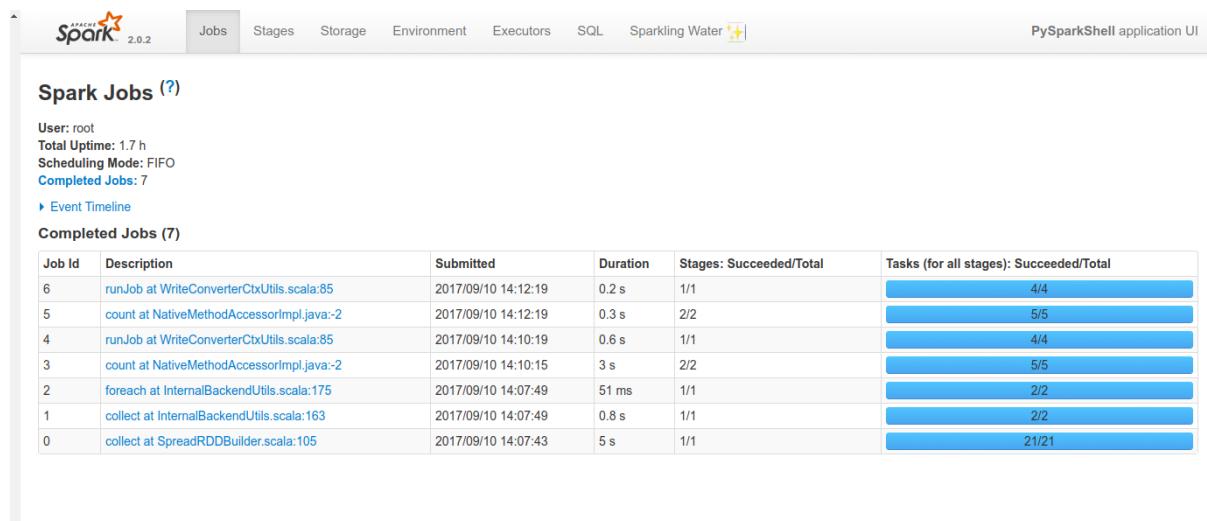


Figura 5: Spark WebUI

## 4.2 Ejecución

The screenshot shows the Apache Spark 2.0.2 WebUI. The top navigation bar includes links for Jobs, Stages, Storage, Environment, Executors, SQL, and Sparkling Water (which is currently selected). A sub-header "PySparkShell application UI" is visible. The main content area displays the "Sparkling Water" application details, including user information (User:root, Uptime:1.7 h), health status (Health:✓, Secured communication:✗), and node count (Nodes:2). It also provides links for Flow UI and Download H2O Logs, and a log entry for FATAL ERROR DEBUG HTTPD INFO WARN TRACE. Below this is a "Sparkling Water" configuration table and a "Sparkling Water Properties" table.

| Name    | Value   |
|---------|---|
| Flow UI | <a href="http://ec2-34-253-225-131.eu-west-1.compute.amazonaws.com:54321">http://ec2-34-253-225-131.eu-west-1.compute.amazonaws.com:54321</a> |
| Nodes   | 172.31.1.83:54321,172.31.5.233:54321  |

| Name                         | Value  |
|------------------------------|--|
| spark.ext.h2o.client.ip      | 172.31.15.58                                 |
| spark.ext.h2o.cloud.name     | sparkling-water-root_app-20170910140638-0009 |
| spark.ext.h2o.client.log.dir | /root/UCI/h2ologs/app-20170910140638-0009    |

Figura 6: Spark WebUI, Sparkling Water en funcionamiento

The screenshot shows the Apache Spark 2.0.2 WebUI displaying the master cluster status. It includes sections for URL, REST URL, Alive Workers, Cores in use, Memory in use, Applications, Drivers, and Status. Below this is a "Workers" table listing two active workers with their addresses, states, cores, and memory usage. Further down are sections for "Running Applications" and "Completed Applications", each with their respective tables.

| Worker Id                                | Address            | State | Cores      | Memory               |
|--|--------------------|-------|------------|----------------------|
| worker-20170909114445-172.31.1.83-57562  | 172.31.1.83:57562  | ALIVE | 2 (2 Used) | 2.6 GB (2.3 GB Used) |
| worker-20170909114445-172.31.5.233-49908 | 172.31.5.233:49908 | ALIVE | 2 (2 Used) | 2.6 GB (2.3 GB Used) |

| Application ID          | Name                | Cores | Memory per Node | Submitted Time      | User | State   | Duration |
|-------------------------|---------------------|-------|-----------------|---------------------|------|---------|----------|
| app-20170910140638-0009 | (kill) PySparkShell | 4     | 2.3 GB          | 2017/09/10 14:06:38 | root | RUNNING | 57 min   |

| Application ID          | Name        | Cores | Memory per Node | Submitted Time      | User | State    | Duration |
|-------------------------|-------------|-------|-----------------|---------------------|------|----------|----------|
| app-20170910140003-0008 | Spark shell | 4     | 2.3 GB          | 2017/09/10 14:00:03 | root | FINISHED | 22 s     |

Figura 7: Spark WebUI, estado del clúster

## 4.2 Ejecución

## **Capítulo 5. Conclusiones**

### **5.1 Conclusiones finales**

Con este trabajo se ha intentado seguir un flujo de trabajo típico de un proyecto de ciencia de datos. Cada uno de los pasos realizados ha sido muy importante para el correcto desarrollo de este ejercicio.

Además, se ha puesto hincapié en la reproducibilidad, usando *notebooks*, fijando semillas de números aleatorios y proveyendo las dependencias del proyecto.

Dada la creciente popularidad de *machine learning*, se está tratando de aplicar en múltiples campos. Uno de ellos es la ciberseguridad, asunto que puede afectar tanto al bienestar personal como al de países, organizaciones y empresas. En estos últimos se han realizado múltiples investigaciones para reforzar la seguridad informática con técnicas de minería de datos. Este trabajo aspira a aportar su grano de arena a esta tendencia.

Se han aplicado una gran cantidad de modelos, la mayoría con éxito. Sin embargo, no se ha buscado sólo obtener buenos resultados, sino que también sean eficientes en cuanto a tiempo de ejecución. Esto último es también importante, dado que la ciencia de datos en muchos casos consiste en prueba y error, y esperar un tiempo a obtener los resultados de un modelo es tiempo que se deja de invertir en realizar pruebas.

Afortunadamente este último punto se ha conseguido, no sólo demostrando la potencia de la computación distribuida sino también utilizando técnicas de tratamiento de los datos, como son las técnicas de rebalanceo *Condensed Nearest Neighbour* y *One-Sided Selection*. En este caso, aunque ambas técnicas tienen como objetivo utilizar posteriormente modelos de aprendizaje por similitud como KNN o KMeans, se ha logrado que el tiempo de entrenamiento de las SVM sea mucho menor, y en algún caso como con el kernel RBF. También se ha intentado utilizar técnicas de reducción de la dimensionalidad, que salvo en el caso de Multinomial Naïve Bayes y SVM con kernel polinomial, no han producido mejoras.

Este trabajo también sirve como broche final a una etapa en la que se han aprendido tanto técnicas de minería de datos como tecnologías de *Big Data* y se ha querido que esto quedase plasmado en este trabajo abarcando ambos enfoques.

### **5.2 Lecciones aprendidas**

Este trabajo me ha resultado muy enriquecedor por los siguientes puntos:

- Se han reforzado los conocimientos en minería de datos, tanto usando lo visto en clase como explorando en libros y en la Red nuevo contenido.

### 5.3 Líneas de trabajo futuro

- Aprendizaje de una librería de *machine learning* nueva como lo es Scikit Learn en un lenguaje de programación distinto al visto en clase.
- Conocimiento de nuevas técnicas para el rebalanceo de clases.
- Realización de un proyecto de ciencia de datos completo, pasando por todas las fases.
- Utilización de una nueva librería de *machine learning* para Spark como es SparklingWater (H2O).
- Se han afianzado los conocimientos en Spark vistos en clase.
- Concienciación y aprendizaje sobre ataques *phishing*.

### 5.3 Líneas de trabajo futuro

Dado que en el conjunto de datos se aportaba una predicción en base a algunos sitios Web y esta no era muy buena, se propone la creación de una herramienta que detecte los sitios Webs que son phishing teniendo como base los modelos entrenados en este trabajo.

Algunas maneras de aplicar los modelos entrenados podrían ser las siguientes:

- Realización de un sitio Web que se encargue de realizar un listado de páginas Web maliciosas.
- Implementación en los navegadores de Internet de un sistema que detecte si el sitio que se está visitando es phishing o no y en caso afirmativo se bloquee la navegación y se notifique al usuario de que el sitio que está visitando es peligroso.
- Eliminación de publicaciones con enlaces de sitios Web con phishing en redes sociales, foros o blogs. De la misma manera que se procesa parte del contenido Web para realizar vistas previas, se podría procesar para detectar sitios Web maliciosos.
- Inclusión de estos modelos en antivirus.
- Inclusión de los modelos entrenados en sistemas de detección de Spam para evitar casos de *spear phishing*.

## **Referencias**

- [1] Internet Security Threat Report, April 2017 - [https://s1.q4cdn.com/585930769/files/doc\\_downloads/lifelock/ISTR22\\_Main-FINAL-APR24.pdf](https://s1.q4cdn.com/585930769/files/doc_downloads/lifelock/ISTR22_Main-FINAL-APR24.pdf)
- [2] Phishing - <https://en.wikipedia.org/wiki/Phishing>
- [3] Spear Phishing: Who's Getting Caught? - <https://www.firmex.com/thedealroom/spear-phishing-whos-getting-caught/> (visitado el 8/9/2017)
- [4] Spear Phishing: Scam, Not Sport - <https://us.norton.com/spear-phishing-scam-not-sport/article> (visitado el 8/9/2017)
- [5] MEPs sound alarm on anti-EU propaganda from Russia and Islamist terrorist groups - [http://www.europarl.europa.eu/pdfs/news/expert/infopress/20161118IPR51718/20161118IPR51718\\_en.pdf](http://www.europarl.europa.eu/pdfs/news/expert/infopress/20161118IPR51718/20161118IPR51718_en.pdf)
- [6] iCloud leaks of celebrity photos - [https://en.wikipedia.org/wiki/ICloud\\_leaks\\_of\\_celebrity\\_photos](https://en.wikipedia.org/wiki/ICloud_leaks_of_celebrity_photos) (visitado el 8/9/2017)
- [7] Ionic Framework - <https://ionicframework.com/> (visitado el 8/9/2017)
- [8] Phishing Websites Data Set - <https://archive.ics.uci.edu/ml/datasets/phishing+websites> (visitado el 8/9/2017)
- [9] Apache Spark - <https://spark.apache.org/> (visitado el 8/9/2017)
- [10] Chapter 3, Data Mining and Machine Learning in Cibersecurity. Sumeet Dua and Xian Du
- [11] Mohammad, Rami, McCluskey, T.L. and Thabtah, Fadi (2012) An Assessment of Features Related to Phishing Websites using an Automated Technique
- [12] Mohammad, Rami, Thabtah, Fadi Abdeljaber and McCluskey, T.L. (2014) Predicting phishing websites based on self-structuring neural network.
- [13] Mohammad, Rami, McCluskey, T.L. and Thabtah, Fadi Abdeljaber (2014) Intelligent Rule based Phishing Websites Classification.
- [14] Scikit Learn - <http://scikit-learn.org/stable/> (visitado el 8/9/2017)
- [15] Imbalanced Learn - <http://contrib.scikit-learn.org/imbalance-learn/stable/index.html> (visitado el 9/9/2017)

## Referencias

- [16] Scikit Plot - <https://github.com/reiinakano/scikit-plot> (visitado el 9/9/2017)
- [17] Keras - <https://keras.io/> (visitado el 9/9/2017)
- [18] Jupyter - <http://jupyter.org/> (visitado el 9/9/2017)
- [19] Pandas - <http://pandas.pydata.org/> (visitado el 9/9/2017)
- [20] Numpy - <http://www.numpy.org/> (visitado el 9/9/2017)
- [21] Seaborn - <https://seaborn.pydata.org/> (visitado el 9/9/2017)
- [22] Matplotlib - <https://matplotlib.org/> (visitado el 9/9/2017)
- [23] Anaconda - <https://docs.continuum.io/anaconda> (visitado el 9/9/2017)
- [24] CRISP-DM - <http://crisp-dm.eu/> (visitado el 9/9/2017)
- [25] Threat Report - <https://www.symantec.com/security-center/threat-report> (visitado el 9/9/2017)
- [26] Sherlock dataset - <http://bigdata.ise.bgu.ac.il/sherlock/#/dataset> (visitado el 15/7/2017)
- [27] Geisser, Seymour (1993). Predictive Inference. New York, NY: Chapman and Hall.
- [28] Introduction to kNN Classification and CNN Data Reduction - [http://www.math.le.ac.uk/people/ag153/homepage/KNN/OliverKNN\\_Presentation.pdf](http://www.math.le.ac.uk/people/ag153/homepage/KNN/OliverKNN_Presentation.pdf)
- [29] Intelligent Data Engineering and Automated Learning, IDEAL 2013. Hujun Yin, Ke Tang, Yang Gao, Frank Klawonn, Minho Lee, Bin Li, Thomas Weise, Xin Yao - [https://books.google.es/books?id=EEq6BQAAQBAJ&pg=PA26&lpg=PA26&dq=one-sided+selection+cnn&source=bl&ots=KZvIFaDWt3&sig=mK\\_LKypUyflmslCbABClsyqDjgM&hl=en&sa=X&ved=0ahUKEwjE2bGMq5HWAhWDtRoKHdeBA60Q6AEISTAF#v=onepage&q=cnn&f=false](https://books.google.es/books?id=EEq6BQAAQBAJ&pg=PA26&lpg=PA26&dq=one-sided+selection+cnn&source=bl&ots=KZvIFaDWt3&sig=mK_LKypUyflmslCbABClsyqDjgM&hl=en&sa=X&ved=0ahUKEwjE2bGMq5HWAhWDtRoKHdeBA60Q6AEISTAF#v=onepage&q=cnn&f=false)
- [30] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting <http://www.jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf>
- [31] Alexa Ranking - <https://www.alexa.com/> (visitado el 10/9/2017)
- [32] Whois Database - <https://whois.icann.org/en> (visitado el 10/9/2017)
- [33] H2O - <https://www.h2o.ai/> (visitado el 10/9/2017)
- [34] Sparkling Water - <https://github.com/h2oai/sparkling-water> (visitado el 10/9/2017)

## **Anexo**

### **Estructura del proyecto**

Proyecto disponible en: <https://github.com/jvicentem/phishing-website-detection>

Directorios y algunos archivos:

- data: Carpeta con los datos utilizados así como su documentación.
- assets: Contiene archivos o scripts auxiliares
  - startup.sh: Script ejecutado al inicio del clúster Spark para instalar las librerías y versión de Python necesarias.
  - cluster command.txt: Comandos ejecutados en la consola para el lanzamiento del clúster Spark en AWS y el lanzamiento de H2O.
- final models: Directorio con los modelos finales serializados, para ser usados posteriormente.
- publications: Publicaciones relacionadas con el conjunto de datos.
- gs models: Contiene objetos Grid-Search de los diferentes modelos serializados.
- rebalanced data: Contiene objetos DataFrame serializados fruto de aplicar diferentes técnicas de rebalanceo.
- utils.py: Contiene funciones auxiliares de transformación de datos y métricas para la validación de modelos.
- environment.yml: Entorno de Anaconda con las dependencias del proyecto.

Notebooks:

- EDA.ipynb: Análisis exploratorio
- Data Splitting.ipynb: División del conjunto de datos entre entrenamiento y test, además de rebalanceo de clases.
- Modeling I.ipynb: Aplicación de modelos sobre el conjunto de entrenamiento rebalanceado de manera aleatoria simple.
- Modeling II.ipynb: Entrenamiento de modelos sobre conjuntos de entrenamiento con las clases rebalanceadas usando otras técnicas distintas a la usada en “Modeling I”. También se utilizaron técnicas de reducción de la dimensionalidad.
- Clustering.ipynb: Clusterización de las observaciones *phishing*.
- Neural Network.ipynb: Entrenamiento de red neuronal.
- Testing models.ipynb: Prueba de los modelos entrenados en el conjunto de test.
- Spark.ipynb: Entrenamiento de algunos modelos usando Spark y H2O.

## Anexo