Universitat Oberta de Catalunya
uoc.edu

Machine Learning aplicado a seguridad

Walter Rene Cuenca Guachamin

Máster Universitario en Seguridad de las tecnologías de la información y de las comunicaciones

Trabajo fin de Máster

Área de Análisis de datos

Enric Hernández Jimenez

Junio del 2019

FICHA DEL TRABAJO FINAL

Titulo del trabajo:	Técnicas de Machine Learning aplicadas a Seguridad
Nombre del autor:	Walter Rene Cuenca Guachamin
Nombre del consultor/a:	Victor Garcia Font
Nombre del PRA:	Enric Hernández Jimenez
Fecha de entrega(mm/aaaa):	06/2019
Titulación:	Máster Universitorio en Seguridad de las Tecnologías de la información y de las Comunicaciones.
Área del Trabajo Final:	Análisis de datos
Idioma del trabajo:	Español
Palabras claves:	Aprendizaje automático, Seguridad, Análisis de datos

Resumen del Trabajo (máximo 250 palabras): Con la finalidad, contexto de aplicación, metodología, resultados y conclusiones del trabajo.

El problema de la desinformación es un problema que siempre ha existido pero ha aumentado ha aumentado debido al crecimiento tecnológico y la aparición de diversas formas de comunicar la información en internet, por ejemplos, blogs o redes sociales. Prestando atención a las redes sociales esto ha provocado la aparición de bots con el objetivo de propagar la desinformación con un determinado objetivo. Teniendo en mente esto el proyecto se centra en realizar un análisis que permita generar un modelo de *Machine Learning* que se capaz de detectar contenido falso y/o bots en redes sociales. Por ello, se ha seleccionado como metodología *CRISP-DM* debido a su gran utilización en este tipo de proyectos. El proyecto consta de las siguientes: Obtención de datos, Generación de nuevas características, Análisis y selección de características, Modelado y y Evaluación.Cada una de estas etapas se encuentra desarrollado en fichero *TFM_Seguridad.ipynb*. Al finalizar el análisis se concluye que el modelo seleccionado permite abordar este problema de manera correcta.

Abstract (in English, 250 words or less):

The problem of misinformation is a problem that has always existed but has increased due to technological growth and the emergence of various ways of communicating information on the Internet, for example, blogs or social networks. Paying attention to social networks this has led to the appearance of bots with the aim of spreading misinformation with a certain objective. With this in mind, the project focuses on performing an analysis that allows generating a *Machine Learning* model capable of detecting fake content or bots in social networks. Therefore, it has been selected as a methodology *CRISP-DM* due to its great use in this type of projects. The project consists of the following:Obtaining data, Generating new features, Analysis and Selection of features, Modeling and Evaluation. Each of these stages is developed in file *TFM_Seguridad.ipynb*. At the end of the analysis, it is concluded that the selected model allows us to address this problem correctly.

Agradecimientos

Dar las gracias a mis hermanos y a mi mama
por el apoyo y su confianza.
También a mi tío Luis Anibal por ser como un padre
y tener siempre una palabra de aliento
cuando lo he necesitado.
Y por ultimo dar las gracias a mi tutor.

Gracias.

Índice general

Índice de figuras

Capítulo 1

Introducción

1.1. Contexto y justificación del trabajo

En la actualidad con el gran desarrollo tecnológico el problema de la seguridad informática se ha convertido en un problema a escala mundial. Antiguamente la protección se fundamentaba en emplear mecanismos que protejan las instalaciones en las que se encontraban los servidores y/o información sensible. A día de hoy las medidas anteriores han dejado de ser suficientes debido a la evolución de los formatos de almacenamiento y su acceso por medio de Internet. Esto ha provocado que las personas puedan acceder de forma rápida y desde cualquier parte del mundo a cualquier clase de información.

Teniendo en cuenta esto aparece el termino ciber-criminal, que se puede definir como aquel individuo que haciendo uso de la tecnología accede a información sensible o información al que no está autorizado para malos fines. Debido a esto se han desarrollado herramientas que permitan la detección de accesos no autorizados o posibles ataques a sistemas viéndose limitadas debido a que estos evolucionan con el tiempo o aparecen nuevos. Por ello, estos sistemas han ido incorporando mecanismos de aprendizaje automático (*Machine Learning*) con el objetivo de proporcionar al sistema de mecanismo que permitan mitigar estas limitaciones. El proyecto seleccionado está motivado sobre este ultima característica.

1.2. Objetivos del proyecto

El presente trabajo tiene como objetivo emplear diversas técnicas de *Machine Learning* en una determinada área de Ciber-seguridad. Como primer paso es necesario realizar un estudio sobre el estado del arte sobre el uso de técnicas de *Machine Learning* en esta área y de esta forma seleccionar el tema interés para el desarrollo del proyecto. Una vez se ha seleccionado el tema se pasa a seleccionar el dataset de trabajo. A continuación, se inicia el análisis de datos del dataset con el objetivo de obtener un modelo matemático que permita detectar una posible amenaza.

1.3. Enfoque y método seguido

Debido a la naturaleza del proyecto, se ha empleado la metodología *CRISP-DM*. Se trata de un modelo para el proceso de minería de datos que describe la forma de abordar este tipo de problemas. Existen otras metodologías similares a *CRISP-DM*, pero esta última ha conseguido convertirse en un marco de referencia para la implementación de este tipo de proyectos en empresas.

Se caracteriza por estar compuesto por 6 etapas (ver Figura 1). La etapa de *Entendimiento del negocio* se encarga de comprender como encaja el problema propuesto en el área de trabajo, la etapa de *Entendimiento de los datos* se encarga de la recolección de los primeros datos para entenderlos y plantear las primeras hipótesis. Tras la obtención de los datos la etapa de *Preparación de los datos* se encarga de adaptarlos por medio de técnicas de visualización, buscando relaciones entre los elementos para la etapa de *Modelado* en la que se selecciona el algoritmo que mejor se adapte al problema. Tras esto, en la etapa de *Evaluación* se evalúa el modelo seleccionado teniendo en cuenta los criterios marcados en la etapa inicial y por último, en la etapa de *Implementación* se integra el modelo en el negocio tras su creación y validación. Cabe destacar que algunas de estas etapas son bidireccionales, es decir, que en función del resultado obtenido se puede volver al paso anterior para realizar comprobaciones que se consideren oportunas.

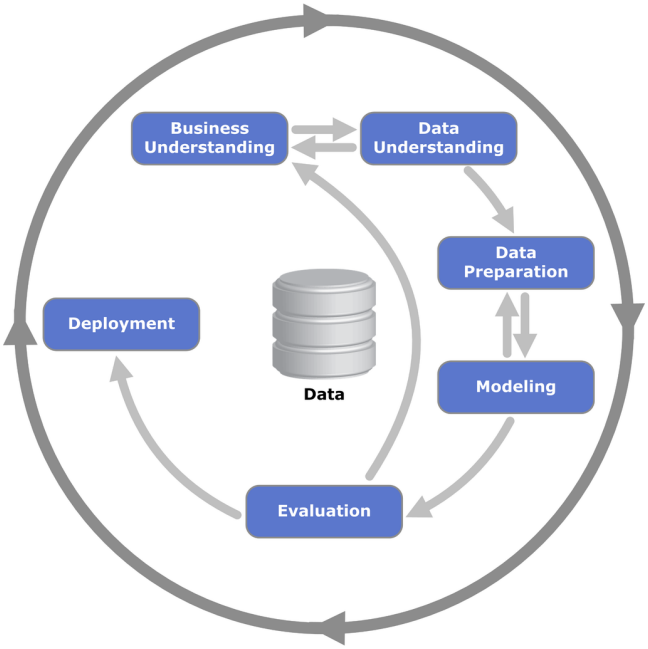

Figura 1: Diagrama de proceso que muestra la relación entre las diferentes fases de CRISP-DM.

1.4. Planificación del trabajo

El presente proyecto se divide en un conjunto de fases que se corresponden a los objetivos definidos en el proyecto. A continuación, se describe cada una de estas fases:

1. **Fase inicial:** tiene como objetivo adquirir el conocimiento necesario para hacer frente al proyecto. Para ello, es necesario revisar las distintas las técnicas de aprendizaje automático y estadísticas necesarias para el proyecto. Además, se revisa el estado del arte en el área de Ciberseguridad enfocado a las aproximaciones que utilicen técnicas de aprendizaje automático para resolver los problemas. De esta forma se puede seleccionar y situar el contexto en el que se desarrollara el proyecto. Por último, se realiza un estudio de los diferentes frameworks y librerías disponibles para abordar proyectos de este estilo.

2. **Fase de desarrollo:** tras seleccionar el contexto y el conjunto herramientas se pasa al desarrollo del proyecto. Esta labor se ha divido en el siguiente conjunto de etapas: *Obtención de datos*, *Generación de nuevas características*, *Análisis y selección de características*, *Modelado*, *Conclusiones* y *Comunicación / visualización de*

datos.

3. **Fase de presentación**: recoge la elaboración del material requerido para la entrega como es la memoria del proyecto y la grabación del vídeo que detalla los puntos más relevantes del proyecto.

Las fases descritas anteriormente presentan una inversión distinta de tiempo debido al conjunto de tareas que se debe realizar. Por ello, la Figura 2 indica de forma aproximada el tiempo en semanas invertido en cada una de las fases.

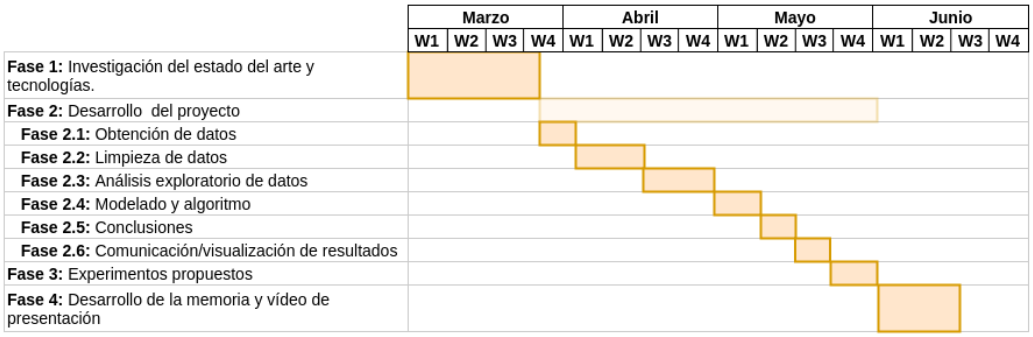

Figura 2: Diagrama de Gantt del plan de trabajo para el proyecto.

1.5. Herramientas empleadas

Para abordar este proyecto se han seleccionado las siguientes herramientas ya que se adaptan a los requerimientos establecidos.

- **Lenguaje de programación:** para realizar análisis de datos habitualmente se pueden emplear *Python* [12] o *RStudio* [13]. De estas alternativas se ha seleccionado *Python* ya que se tenía conocimientos previos sobre este lenguaje.
- **Entorno de trabajo:** como entorno de trabajo se utiliza *Anaconda* [1]. Se ha selecciona utilizar un entorno de trabajo para encapsular el proyecto y de esta forma evitar posibles fallos por incompatibilidad de versiones entre las librerías existentes en el ordenador.
- **Librerías:** para el tratamiento de datos se emplea *Scipy* [15], *Numpy* [9] y *Pandas* [10]. Para la visualización de los datos y resultados se emplea *Seanborn* [16]

y *Matplotlib* [7]. Por último, se utiliza *Scikit-Learn* [14] para acceder a los algoritmos de *Machine Learning* debido a que es una de las librerías más utilizada.

- **Generación del informe:** para generar un informe con el análisis realizado se emplea *Jupyter-Nootbook* [5] ya que permite trabajar con código implementado en Python.

1.6. Breve sumario de productos obtenidos

Los productos obtenidos del proyecto desarrollado se listan a continuación:

- Realizar una comparativa de algoritmos de *Machine Learning* con el objetivo de seleccionar el algoritmo más apropiado para detectar noticias falsas y bots en redes sociales.
- Presentación de informe en *Jupyter-Nootbook* con las características más relevantes del análisis desarrollado.

1.7. Breve descripción de los capítulos de la memoria

- En este primer capítulo de la memoria se hace una introducción al proyecto, su contexto, motivación y objetivos.
- En el capítulo 2, se presenta el estado del arte, esto es, un pequeño resumen de desarrollos previos centrados en el campo de estudio y la aportación que se desea realizar.
- Tras esto en el capítulo 2.3, se describe el desarrollo propuesto en este proyecto. Se explica en profundidad las distintas fases y algoritmos empleados.
- A continuación, en el capítulo 3 se describe las decisiones tomadas en cada una de las etapas del desarrollo propuesto en este proyecto.
- En el capítulo 4, se detallan las conclusiones obtenidas al finalizar el proyecto y se proponen líneas de trabajo futuras de acuerdo a los resultados obtenidos.
- Finalmente, en el capítulo 5 se listan los términos y acrónimos empleados a lo largo de la memoria del proyecto.

Capítulo 2

Estado del arte

En este capítulo se describen los distintos campos de estudio respecto al tema seleccionado para desarrollar el proyecto. En el apartado 2.1 se realiza una revisión de las técnicas de *Machine Learning* y sus características. El apartado 2.2 contiene un resumen de trabajos previos en el área de Ciber-Seguridad prestando especial atención a aquellas aproximaciones que emplean técnicas de aprendizaje automático para detectar desinformación en redes sociales.

2.1. Aprendizaje automático en análisis de datos

El *Machine Learning* es una rama de la Inteligencia Artificial (*AI*) que permite a los ordenadores tener la capacidad de aprender, sin haber programado las acciones para que esto suceda. El proceso de aprendizaje guarda cierto parecido al empleado en minería de datos. Mientras en la minería de datos se extraen los datos para la comprensión humana el *Machine Learning* los utiliza para detectar patrones y ajustar las acciones del programa correspondiente.

Los datos que se emplean pueden estar en diferentes formatos como dataset, imágenes, vídeos o audios. Como metodología básica se divide en 2 o 3 sub-conjuntos (ver Figura 3) llamados entrenamiento, validación y test respectivamente sobre el conjunto inicial. Esto es necesario para construir un modelo que generalice correctamente y no solo sea capaz de trabajar con los datos con los que se ha creado, esto recibe el nombre de *overfitting*. Por ello, los dos primeros conjuntos se emplean para diseñar del modelo y el último se emplea para evaluar su capacidad predicativa

con datos nuevos que no visto en etapas anteriores.

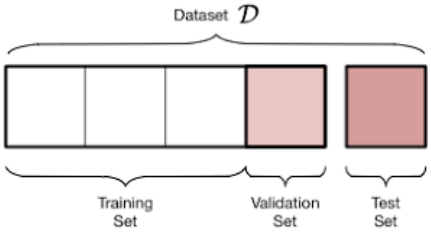

Figura 3: Diagrama de división en sub-conjuntos de los datos.

En función de las características disponibles de los datos iniciales y el enfoque que requiera el problema a tratar se puede encontrar los siguientes 3 tipos de aprendizaje [21]: *Supervisado* , *No supervisado* y *Profundo* que se detallan en los apartados 2.1.1, 2.1.2 y 2.1.3 respectivamente. Teniendo en cuenta esto existen *Cheat-sheets* (ver Figura 4) que intentan ayudar a seleccionar un algoritmo en función de los datos de los que se dispone.

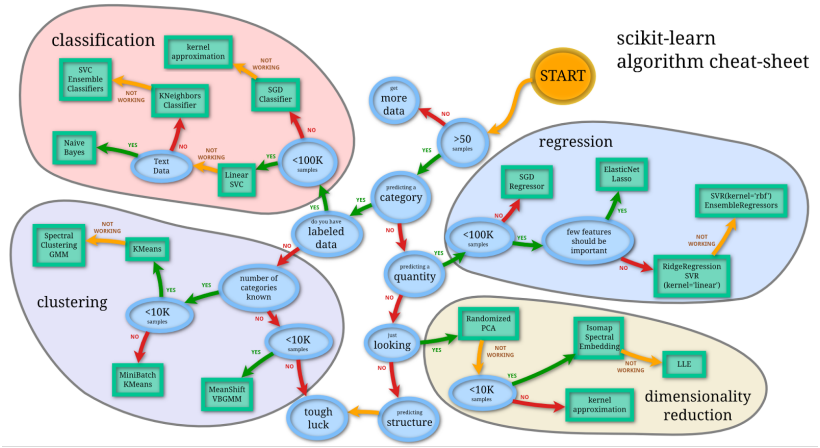

Figura 4: *Cheat-sheet* sobre los tipos de algoritmos de *Machine Learning* [14].

2.1.1. Aprendizaje supervisado

Se caracteriza por trabajar con datos etiquetados, es decir, además de los datos necesarios para realizar la predicción se necesita conocer para cada instancia la característica objetivo. Los datos que se emplean suelen ser valores históricos lo que permite aprender a etiquetar los datos de forma correcta. La etiqueta se emplea únicamente en el proceso de entrenamiento del modelo ya que en la etapa de test se

comprueba la calidad de la predicción. Alguno de los algoritmos que forman parte de este aprendizaje [21] son:

1. *K-NN:* este algoritmo se encarga de clasificar cada nuevo dato en el grupo correspondiente, según tenga k vecinos más cerca de un grupo o del otro. Para ello, calcula la distancia del nuevo elemento a cada uno de los ya existentes.

2. *Regresión lineal:* se encarga de trazar una linea recta a través de un conjunto de puntos intentando minimizar los residuos.

3. *Regresión logística:* es una manera estadística de modelar un resultado binomial con una o más variables explicativas. Se encarga de medir la relación entre la variable dependiente y una o más variables independientes estimando las probabilidades por medio función logística.

4. *SVM:* fue desarrollado en la década de los 90, dentro de la ciencia computacional. Se basa en el concepto del hiper-plano, es decir, consiste en encontrar un plano capas de separar las clases de interés. Es capaz de obtener una buena clasificación lineal o no lineal, regresión y en ocasiones la detección de outliers. Además, se suele emplear cuando las dimensiones del dataset es de tamaño pequeño o mediano.

5. *Decision Trees:* es una herramienta de apoyo a la toma de decisiones empleando un modelo similar a un árbol.

6. *Random Forests:* este algoritmo nace como mejora sustancial al algoritmo *Decision Trees*. Este método combina una gran cantidad de árboles de decisión independientes probados sobre un conjunto de datos.

2.1.2. Aprendizaje no supervisado

Se caracteriza por no disponer de datos etiquetados ya que se pretende descubrir nuevos patrones o resultados a partir de los datos. Este tipo de problemas suelen ser más complejos, ya que el modelo debe aprender sin conocer la característica objetivo de cada instancia. En este caso los algoritmos se pueden dividir [21] en los siguientes 2 grupos:

1. **Clustering:** se tratan de técnicas exploratorias de análisis de datos, que se usan para organizar la información en grupos similares sin tener conocimiento previo de sus estructuras. Cada uno de los grupos comparte características similares dando lugar a grupos diferentes en función de sus características (ver Figura 5). Alguno de los modelos que se emplean son *K-Means*, *Hierarchical Cluster Analysis (HCA)* y *Expectation Maximization*.

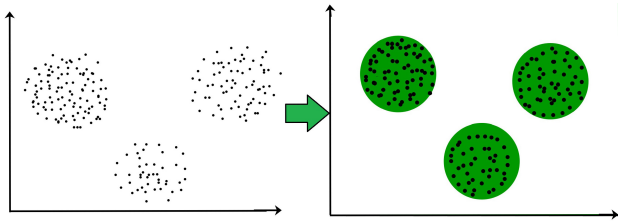

Figura 5: Ejemplo de algoritmos de clustering.

2. **Visualización y reducción de la dimensión:** en ocasiones se trabaja con datos que presentan un alto número de características lo que es un reto para la capacidad de procesamiento y rendimiento computacional de los algoritmos *Machine Learning*. Estas técnicas intentan mitigar este problema reduciendo la dimensión de los datos por medio de la correlación entre las variables y de esta forma eliminar ruido existente en los datos. Con esto se consigue comprimir los datos en un sub-espacio menor reteniendo la mayor información posible (ver Figura 6). Alguno de los modelos que se emplean son *Principal Component Analysis (PCA)*, *Locally-Linear Embedding (LLE)* y *t-distributed Stochastic Neighbor Embedding (t-SNE)*.

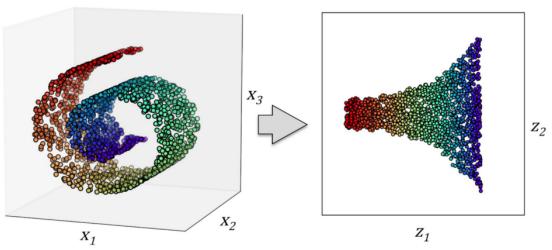

Figura 6: Ejemplo de algoritmos de reducción de la dimensión.

2.1.3. Aprendizaje profundo

El aprendizaje profundo, también llamado *Deep-Learing*, es una de ramas del *Machine Learning* que en la actualidad ha despertado especial interés por su versatilidad. Esto ha sido posible al desarrollo tecnológico especialmente al desarrollo de potentes GPUs que permiten un entrenamiento rápido de este tipo de modelos.

Este aprendizaje se fundamenta en investigaciones biológicas sobre el comportamiento de las neuronas en el cerebro humano. Teniendo en cuenta esto, una red neuronal está compuesta por un conjunto de neuronas interconectadas entre si mediante enlaces. Cada neurona toma como entradas las salidas de las neuronas antecesoras, multiplicando cada una de esas entradas por un peso y mediante una función de activación calculan una salida. La unión de todas estas neuronas interconectadas es lo que compone la red neuronal artificial. Estas redes no son más que redes interconectadas masivamente en paralelo organizadas jerárquicamente que intentan interactuar con los objetos del mundo real del mismo modo que lo hace el sistema nervioso biológico. En función de la topología se pueden clasificar en *Perceptrón simple*, *Perceptrón multicapa*, *Redes neuronales convolucionales*, *Redes neuronales recurrentes* y *AutoEncoders*.

En cuanto al *Perceptrón simple* es la unidad más básica que existe (ver Figura 7). Cada una de las neuronas está formado por un conjunto de datos de entrada $(X_1, X_2, ...X_n)$. A cada uno de los datos de entrada se les asigna su correspondiente peso $(W_1, W_2, ...W_n)$. Por último, la salida de la neurona se calcula por medio de una función de activación que otorga mucha flexibilidad y permite estimar relaciones complejas.

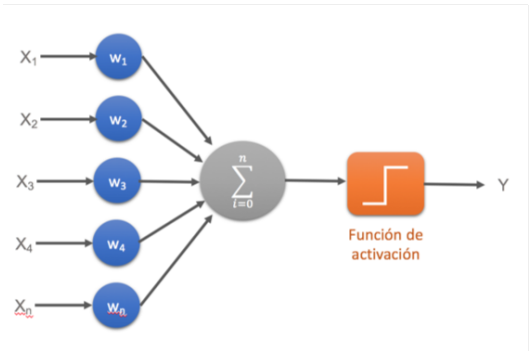

Figura 7: Esquema red neuronal mono-capa.

Con respecto al *Perceptrón multicapa* (ver Figura 8) es una generalización de la red neuronal mono-capa. La principal diferencia reside en que estas redes constan de un conjunto de capas intermedias (capas ocultas) entre la capa de entrada y la de salida. Dependiendo del número de conexiones que presente la red esta puede ser total o parcialmente conectada.

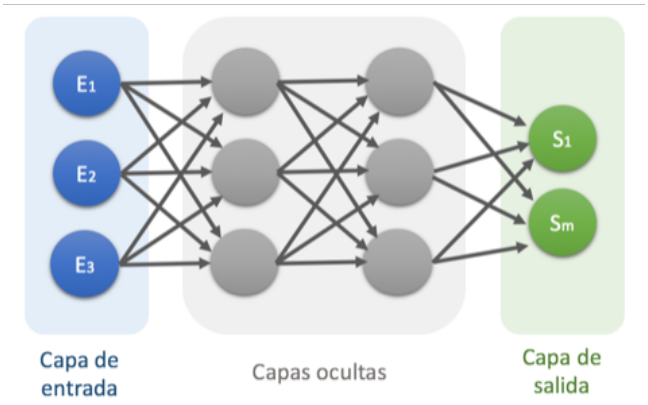

Figura 8: Esquema Perceptrón multicapa

Las *Redes neuronales convolucionales* se diferencian de las anteriores en que cada neurona no se une con todas y cada una de las capas siguientes sino solo a un solo subgrupo. Su arquitectura está formada por 3 capas (ver Figura 10). La primera capa (convolucional) se caracteriza por realizar operaciones de productos y sumas entre la capa de partida y los n filtros, generando mapas de características. Las características extraídas corresponden a cada posible ubicación del filtro en los datos de origen. Tras aplicar la convolución se aplica a los mapas de características una función de activación. La segunda capa (reducción) se encarga de disminuir la cantidad de parámetros al quedarse con las características más comunes. La última capa (clasificador) suelen ser capas completamente conectadas que contiene tantas neuronas como clases se quiere predecir.

Las *Redes neuronales recurrentes* se caracterizan por no tener una estructura fija de capas sino que permite conexiones arbitrarias entre las neuronas, incluso llegando a crear ciclos, con esto se consigue crear temporalidad dentro de la red, permitiendo que la red tenga memoria. Presentan un gran potencial en todo lo que tiene que ver con análisis de secuencias, como puede ser el análisis de texto, sonido o vídeo. Existen varios tipos de estas redes dependiendo del número de capas ocultas des-

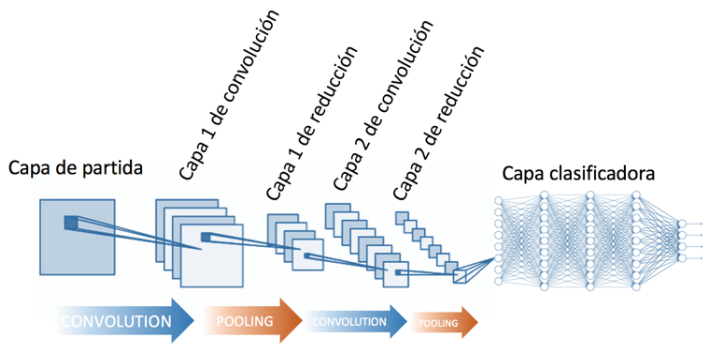

Figura 9: Esquema red neuronal convolucional.

tacando redes recurrentes simples (SRN), redes LSTM y redes RGU.

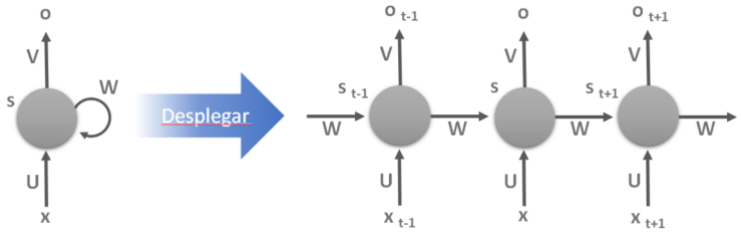

Figura 10: Esquema red neuronal recurrente.

Por último, los *AutoEncoders* son redes neuronales que tienen como objetivo generar nuevos datos comprimiendo la entrada en un espacio de variables latentes y luego reconstruyendo la salida en base a la información adquirida. Este tipo de redes están formadas por un encoder que se encarga de comprimir la entrada mediante una función de codificación y un decoder, que se encarga de reconstruir la entrada en base a la información recolectada previamente.

2.2. Ciberseguridad

Debido al crecimiento tecnológico y al fácil acceso a la información por medio de internet ha aparecido el área de Ciber-seguridad. Su objetivo es proveer de herramientas o mecanismos para defender ordenadores y servidores, dispositivos móviles, sistemas electrónicos, redes y datos de ataques maliciosos. Existen tres tipos de amenazadas que se intentan contrarrestar: el ciber-crimen, que son actos indi-

viduales o de grupos cuyos ataques tienen como objetivo un beneficio económico; la ciber-guerra, a menudo se trata de recopilar información de acuerdo a una motivación política; y el ciber-terrorismo, cuyo propósito es comprometer los sistemas electrónicos para causar pánico o temor. Por otra parte, la aparición de las redes sociales ha proporcionado un nuevo posible vector de ataque basado en ingeniería social. Este vector se caracteriza por proporcionar contenido de confianza a los usuarios, pero en realidad esconde código con código malicioso o que intente promover la desinformación entre los usuarios.

Como ya se ha mencionado el crecimiento tecnológico ha provocado que los mecanismos tradicionales no sean efectivos, ya que muchos de los ataques tienden a evolucionar en el tiempo o aparecen nuevos. Además, el estudio manual de los datos en busca de anomalías o ataques es inviable en la actualidad debido al enorme flujo que se genera cada segundo. Por ello, se ha visto la necesidad de emplear técnicas de *Machine Learning* por su capacidad para aprender patrones de ataques e identificarlos. Junto a esto, la aparición de métodos de procesamiento de grandes cantidades de datos (*Big Data*) permiten aumentar la eficacia a estos algoritmos. A continuación, se resumen trabajos previos sobre la detección de noticias falsas empleando este tipo de técnicas.

Una de las ramas de investigación sobre el tema de interés se centra en la detección de noticias falsas analizando información de diferentes plataformas. En [22] se enfoca el problema como una clasificación binaria para decidir si el contenido es falso o verdadero. Como fuente de datos se emplea un gran conjunto de tweets publicados en *Twitter* [18]. Las características de los datos anteriores son agrupadas en los siguientes 5 niveles: usuario, tweet, texto, tema central y sentimiento. Toda esta información permite entrenar varios modelos de aprendizaje automático como *Naive Bayes*, *Arboles de decisión*, *Machine Vector Suporte*, *Redes Neuronales*, *Xboosting* y *Random Forest*. Finalmente, tras validar los distintos modelos se selecciona *Xboosting*. En [23] también se realiza un enfoque de clasificación. De los datos empleados se centran en los metadatos, elementos multimedia y el valor sentimental asociado al texto. Toda esta información se utiliza para entrenar los algoritmos de *Machine Vector Suporte* y *Arboles de decisión (J48)*. Por último, en [28] se estudia la detección de noticias falsas en redes sociales por medio de la relación entre el contenido del

tweet y las características del perfil de los usuarios de Twitter.

Otro tema relacionado al anterior se basa en investigar la existencia de Bots en redes sociales que tienen como objetivo la propagación de información falsa o manipular la opinión de las personas. En [25] se realiza la detección de bots emplean dos conjuntos de datos previamente etiquetados (bot y no bot). De esta información se estudia los atributos más representativos y se generan nuevos atributos por medio de esta información, por ejemplo, *Likes_age_ratio* que indica la relación entre el número de likes y los días transcurridos desde la creación de la cuenta. Toda esta información se emplea para entrenar el algoritmo *RandomForest* obteniendo los atributos más representativos para realizar la detección. En [19] la detección de bots se centra en tweets escritos en árabe ya que existen pocas investigaciones del problema en este idioma. Para ello, se realiza un extracción por medio de la API REST [2] de *Twitter* y se etiqueta de manera manual las cuentas como reales y no reales. De la información extraída se realiza un estudio de las características estableciendo los siguientes 4 grupos: características del tweet, contenido del tweet, comportamiento de la cuenta y perfil de la cuenta. Por último, la información se utiliza como entrada de varios algoritmos de clasificación como *Random Forest*, *Naïve Bayes* y *Support vector machine (SVM)*. En el caso de [20] se estudia como la utilización de bots influye en la opinión pública. Para ello, se toma como tema centran las elecciones generales de Suiza a lo largo de tres meses. Al igual que en otras investigaciones los datos se obtienen de *Twitter* extrayendo solo aquellas publicaciones que traten del tema de interés. Con la información obtenida se desarrolla un modelo que permita la detección de cuentas falsas que generan contenido de forma automática y extraer los temas que emplean estas cuentas y la forma en la que se difunde la información por medio de los bots. Con la información obtenida se entrenan varios modelos de clasificación como *AdaBoost*, *Logistic regression*, *Support vector machines (SVM)* y *Naive Bayes*. Tras validar cada uno de los modelos anteriores se selecciona *Random Forest*. Por último, en [24] la detección de bot se realiza estudiando el texto publicado por los usuarios en *Twitter*. Las características que se utilizan son los metadatos y el texto de las publicaciones. Esta información sirve para entrenar técnicas de *Deep learning*, en particular LSTM. Debido al enfoque que proponen se modifica la arquitectura general de LSTM para incorporar los metadatos como información auxiliar,

en cuanto al análisis de texto emplean un embeding pre-entrenado (GloveE) de esta forma se obtiene un modelo más robusto al incorporar la información auxiliar.

2.3. Enfoque de la propuesta

Tras haber revisado el estado del arte sobre la detección de noticias falsas y bots se llega a las siguientes conclusiones: esta propuesta se centra en analizar la información de la red social de *Twitter* como se ha mencionado anteriormente. Dicho análisis empleara metadatos e información del tweet e información del usuario. Toda esta información se utilizara para entrenar varios modelos *Machine Learning* prestando especial atención a los *Supervisados* y *No supervisados*. Por último, se seleccionará el modelo que permita detectar noticias falsas e identificar bots.

chapterDiseño e implementación de la propuesta

En este capítulo se describe la implementación propuesta, partiendo del estado del arte descrito en el capítulo anterior que sirve como base para el planteamiento propuesto. La sección 2.4 detalla el alcance funcional de la propuesta, tras esto, la sección 2.5 trata en detalle cada una de las etapas de la implementación. Por último, la sección 2.6 indica la estructura del informe desarrollado con las distintas etapas del análisis realizado.

2.4. Alcance funcional de la propuesta

Esta sección se centra en describir el objetivo que se pretende alcanzar en el proyecto propuesto. Como objetivo principal se pretende obtener un modelo matemático que permita detectar información falsa en redes sociales y la existencia de bots. Para ello, se selecciona *PHEME* [11] y *Detecting Twitter bot* [3] como datasets para iniciar el análisis. En cuanto al objetivo principal se puede sub-dividir en los siguientes sub-objetivos:

- Generación de nuevas características en función de los datos disponibles que ayuden a alcanzar el objetivo del proyecto.
- Orientar el proyecto hacia el área *Data Sciences*.
- Realizar un análisis comparativo de diferentes modelos de *Machine Learning* para seleccionar el más adecuado. Para ello, se debe emplear la curva *ROC*.

2.5. Diseño de la implementación

En esta sección se presenta el diseño seleccionado para abordar la propuesta. Se indican las distintas etapas implementadas en el análisis para conseguir el objetivo del proyecto (ver Figura 11). Consta de 4 etapas cuyas características más relevantes se encuentra dentro del fichero *Jupyter-Nootbook* y de esta forma generar un informe con los pasos realizados en cada etapa.

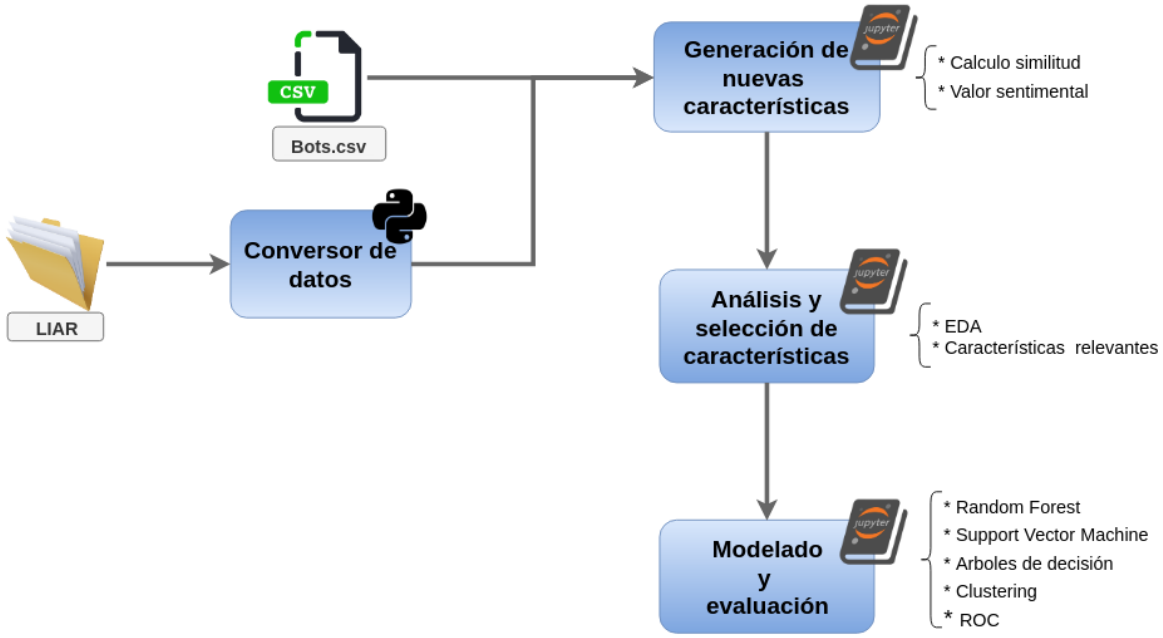

Figura 11: Etapas del análisis propuesto para el proyecto.

2.6. Detalle de la implementación

Esta sección se centra en detallar las etapas que forman parte de la implementación del proyecto. El apartado 2.6.1 detalla el proceso de obtención de datos. En el apartado 2.6.2 se explica la generación de nuevas características. Tras esto, en el apartado 2.6.3 se detalle el análisis del conjunto de datos y la selección de características. Por último, en el apartado 2.6.4 y 2.6.5 se desarrollado y evalúa el modelo final de *Machine Learning* respectivamente.

2.6.1. Obtención de datos

Debido al enfoque del proyecto la extracción de datos se realiza de la red social de *Twitter*. Esta plataforma permite a los usuarios registrados publicar información por medio de tweets (ver Figura 12), que no son más que publicaciones dentro de la red que pueden contener texto, imágenes, enlaces externos, menciones a otros usuarios y/o hastangs.

Teniendo en mente esto se han seleccionado los datasets *PHEME* y *Detecting Twitter bot* para la detección de noticias falsas y bots respectivamente. La informa-

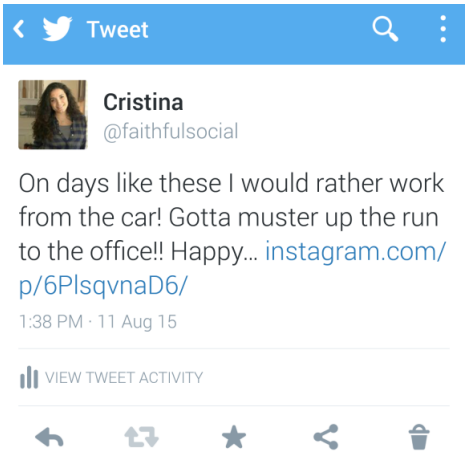

Figura 12: Ejemplo de un tweet de *Twitter*.

ción de *PHEME* se encuentra dividida en diferentes directorios en función de las 9 noticias a las que hace referencia (ver Figura 13). Cada directorio cuenta a su vez con dos subdirectorios, rumores y no rumores. El tweet en sí se puede encontrar en el directorio *origen-tweet* del tweet en cuestión, cuyo formato es *JSON* [4]. Debido a la estructura en la que se encuentra la información es necesario realizar una transformación de formato. Para ello, por medio de un fichero *Python* se realiza una lectura de cada uno de los ficheros *JSON* de los directorios. De cada fichero, se extraen las características más relevantes, por ejemplo, el número de seguidores, el texto de la publicación y nombre del usuario entre otros. Además, se valida que los datos estén codificados de forma correcta para evitar problemas en etapas posteriores. El resultado anterior se almacena en el fichero final *data_fake_news.csv*.

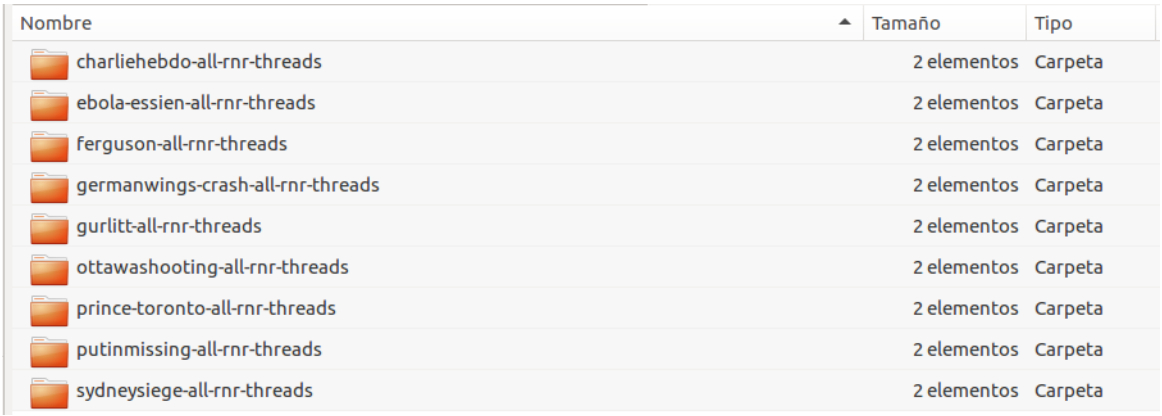

Figura 13: Estructura de directorios de *PHEME*.

Con respecto al conjunto de datos *Detecting Twitter bot* [3] se encuentra publicado en *Kaggle* [6]. La información también pertenece a *Twitter* y se encuentra divida en los ficheros *training_data_2_csv_UTF.csv* y *test_data_4_students*. Esto se debe a que el primer fichero se ha diseñado para ser empleado en la fase entrenamiento mientras que el otro para la fase de test. Por último, es necesario indicar que el fichero *test_data_4_students* no tiene los datos etiquetados.

2.6.1.1. Entendimiento de los datos

Antes de pasar a analizar la información obtenida es necesario entender las variables disponibles sobre los tweets y de esta forma utilizarlas de manera adecuada en procesos posteriores. Debido a que la extracción ha sido realizada por medio de la API de *Twitter* se dispone de información acerca del tweet y del usuario que lo ha publicado.

A modo de resumen se ha diseñado la tabla 2.1. Esta tabla identifica las variables que corresponden al tweet o al usuario que lo ha publicado. Además, se identifica el tipo de dato para cada una de las variables y una breve descripción sobre ellas. Debido a que se trabaja con dos dataset se incluye en la tabla la columna *Dataset* para indicar en qué conjunto de datos se pueden encontrar. Los valores que puede tomar esta columna son *PH* y *DTB* si se trata de *PHEME* y/o *Detecting Twitter bot* respectivamente.

Para finalizar este apartado, como se ha mencionado previamente la información en los dos dataset se encuentra clasificada previamente. Por ello, se realiza un estudio básico de la distribución (ver Figuras 27 y 15) de la variable objetivo en cada dataset. En la primera figura se puede ver como la distribución es muy diferente en cada categoría, por ejemplo, sobre el tema de *Charlie Hebdo* existe más información verdadera que falsa lo que puede repercutir en el diseño del modelo. En cuanto a la segunda figura la distribución es similar en las distintas categorías.

2.6.2. Generación de nuevas características

Este apartado se centra en la etapa de *Generación de nuevas características* que se encuentra disponible en el fichero *TFM_Seguridad.ipynb*. Para realizar las operaciones necesarias se han desarrollados varios ficheros auxiliares con el objetivo de de-

	Parámetro	Tipo de dato	Descripción	Dataset
Información Twitter	created_at	fecha	Fecha de publicación del tweet.	PH
	id	número	Identificador único asociada al tweet.	PH/DTB
	text	texto	Texto que contiene el tweet.	PH
	favourite_count	número	Número de favoritos dados al tweet.	PH/DTB
	coordinates	coordenadas	Coordenadas geográficas asociadas al tweet.	PH/DTB
	entities	–	Listado de hastangs, menciones, urls del tweet.	PH
	status	–	Respuesta del API REST de Twitter.	DTB
Información Usuario	id	número	Identificador único del usuario.	PH
	name	texto	Nombre del usuario.	PH/DTB
	screen_name	texto	Nombre del usuario en la red social.	PH/DTB
	location	texto	País y/o ciudad del usuario.	PH
	description	texto	Breve información del usuario.	PH/DTB
	followers_count	número	Número de seguidores del usuario.	PH/DTB
	friends_count	número	Número de amigos asociados al usuario.	PH/DTB
	created_at	fecha	Fecha de creación de la cuenta del usuario.	PH/DTB
	verified	boleano	Flag que indica si la cuenta ha sido verificada.	PH/DTB
	favourites_count	número	Número de favoritos.	PH/DTB

Cuadro 2.1: Resumen de la información sobre un tweet obtenida por la API de *Twitter*.

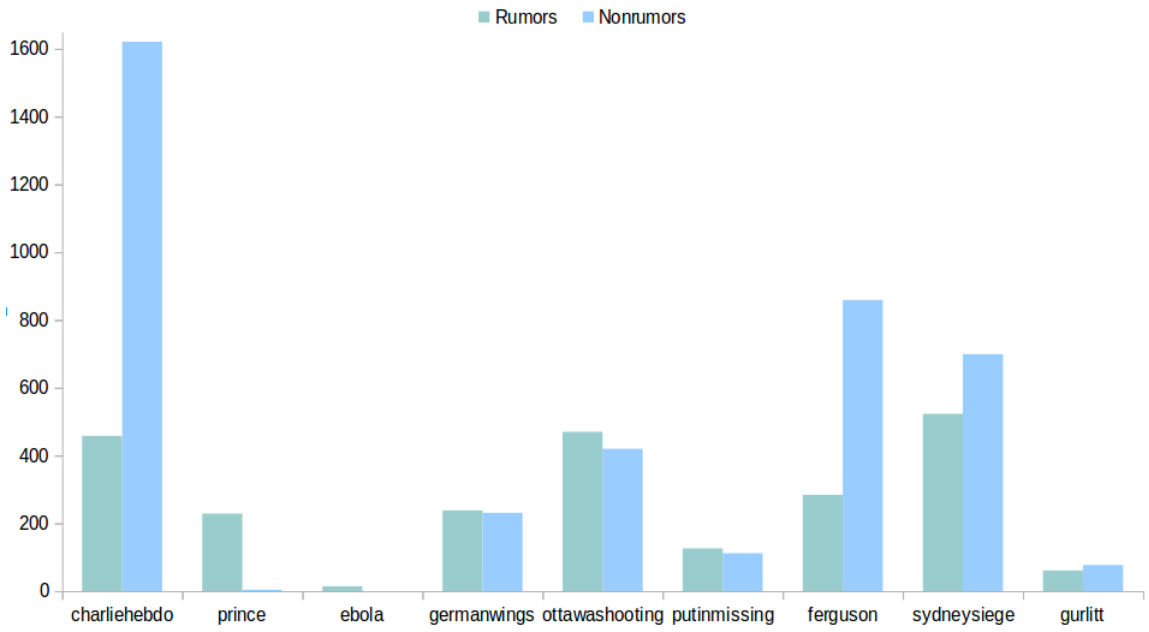

Figura 14: Distribución de la variable objetivo por cada directorio en el dataset *PHEME*.

jar el fichero anterior lo más entendible posible. Tras investigar las variables de los datasets se seleccionan las de tipo texto en particular *screen_name*, *description* y *text*.Debido al tipo de variables seleccionadas es necesario realizar un filtrado previo para seleccionar aquellos registros donde sus variables *description* y *text* estén

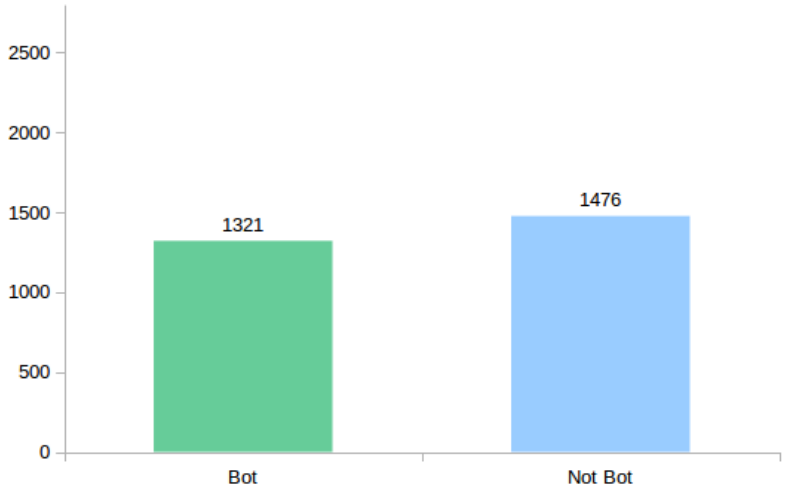

Figura 15: Distribución de la variable objetivo en el dataset *Detecting Twitter bot*.

en Ingles.

Tras esto, por medio de la variable *screen_name* se generan 3 nuevas variables que estudian su composición. La primera *len_screen_name* contiene el número de caracteres totales, a continuación *punt_screen_name* y *num_screen_name* contienen el número de signos de puntuación y números que forman parte de la variable *screen_name* respectivamente.

A continuación, se generan las variables *similarity_description* y *similarity_text* que contienen el valor de similitud dentro del dataset de la variable *description* y *text* respectivamente. Para obtener los valores de similitud es necesario emplear un procesador de texto, en este caso, NLTK [8]. El texto se trocea en palabras generando un listado de tokens. A este listado se le aplica un conjunto de filtros para eliminar signos de puntuación, palabras vacías y otros caracteres que no aportan información. A medida que se procesa el texto se calcula el TF e IDF con el objetivo de obtener la relevancia de las palabras. Por último, al tener los valores de TF-IDF se aplica la similitud coseno [26], cuyo valor está comprendido entre 0 y 1, dando como resultado el valor de similitud del contenido de los tweets entre sí. Este proceso es común para calcular los valores de estas nuevas variables.

Por último, se genera la variable *val_sentiment* a partir de la variable *text*. Para ello, es necesario emplear nuevamente *NLTK* para tratar el texto de la misma forma que se ha explicado anteriormente. Adicionalmente, se calcula la categoría grama-

tical y se aplica la lematización a cada token de la lista. Para obtener la polaridad de las palabras es necesario emplear un lexicón, en este caso, *SentiWornet* [17] cuyos valores se encuentran entre -1 y 1. Cada uno de los tokens se pasan por el lexicón obteniendo su valor sentimental, por lo que el valor sentimental de un tweet se obtiene como el valor medio que aporta cada palabra.

Las siguientes variables *len_screen_name*, *punt_screen_name*, *num_screen_name* y *similarity_description* se añaden en los dos datasets. En cuanto a las variables *similarity_text* y *val_sentiment* solo se añade al dataset *PHEME* (ver Figura 16). Por último, debido a las modificaciones realizadas se generan los ficheros *data_fake_news_prepro.csv* y *data_bots_prepro.csv*.

	Valores iniciales		Valores finales	
	Observaciones	**Caracteristicas**	**Observaciones**	**Caracteristicas**
PHEME	6425	17	6425	22
Dete. Twitter bot	2797	20	2442	24

Figura 16: Comparación de los datasets tras la generación de nuevas características.

2.6.3. Análisis y selección de características

Este apartado se centra en la etapa de *Análisis y selección de características* que se encuentra disponible en el fichero *TFM_Seguridad.ipynb*. Para ello, en el apartado 2.6.3.1 se detalla el proceso de *Preparación de los datos* y en el apartado 2.6.3.2 se detalla el proceso del *Análisis exploratorio de datos (EDA)*. Por último, en el apartado 2.6.3.3 se trata la *Selección de variables*.

2.6.3.1. Preparación de los datos

Este apartado se centra en explicar el estudio realizado sobre las variables de los datasets con el objetivo de detectar datos faltantes y/o datos que no estén codificados correctamente para evitar posibles fallos en etapas posteriores. Debido a que se trabaja con *dataframes* se emplea la librería *Pandas* de *Python* para esta tarea.

El estudio de las variables del dataset *PHEME* indica que no presenta errores de codificación ni falta de datos en sus variables. Esto se debe a que la información

ha sido pre-procesada en la etapa de *Obtención de datos* donde se han realizado las correcciones necesarias de los datos.

En cuanto al dataset *Detecting Twitter bot* se han detectado errores de codificación y la presencia de falta de datos en alguna de sus variables. Las variables con errores de codificación son: *id* que se transforma de *Float* a *Int* ya que se trata de un identificador, *created_at* que se cambia a formato *YYYY-MM-DD* para identificar de forma correcta el periodo temporal y por último, *has_extended_profile* que se transforma de texto a *Boolean* ya que indica la ausencia o no de este atributo en el tweet. Sobre la falta de datos para las variables *location* y *url* se ha detectado que este comportamiento es normal debido a la funcionalidad de la aplicación. En cuanto a la variable *status* contiene la respuesta de la API por lo que no se puede inferir los datos que faltan. Por último, sobre la variable *has_extended_profile* se puede inferir el valor de los datos que faltan por lo que se asigna el comportamiento por defecto de la aplicación. En cuanto al porcentaje de datos que faltan en cada una de las variables se puede ver en la Figura 17.

	Catacteristica	% null
0	location	32.6
1	url	41.9
2	status	7.8

Figura 17: Porcentaje de datos faltantes en *Detecting Twitter bot*.

2.6.3.2. Análisis exploratorio de datos (EDA)

Tras realizar las comprobaciones previas sobre las variables de los datasets se pasa a estudiar sus características. Para ello, este análisis se ha dividido en un estudio uni-variante y multi-variante con el objetivo de encontrar relaciones entre las variables. Esto se realiza por medio de la visualización de datos.

En el caso del análisis uni-variante se distinguen dos tipos de variables dicotómicas y continuas. En el primer caso se representan los valores por medio de un histograma (ver Figura 18) obteniendo la distribución de los valores. Como se puede ver los valores más comunes en las variables corresponde al 0. Con respecto a

las variables continuas se emplea la distribución de densidad de las variables. Esto permite por ejemplo ver que la variable *similarity_description* tiene una distribución bi-modal centrada en valores cercanos a 0 y cercanos a $0{,}1$. Aunque solo se muestren los valores correspondientes al dataset *Detecting Twitter bot* este proceso también se ha realizado para el dataset *PHEME*.

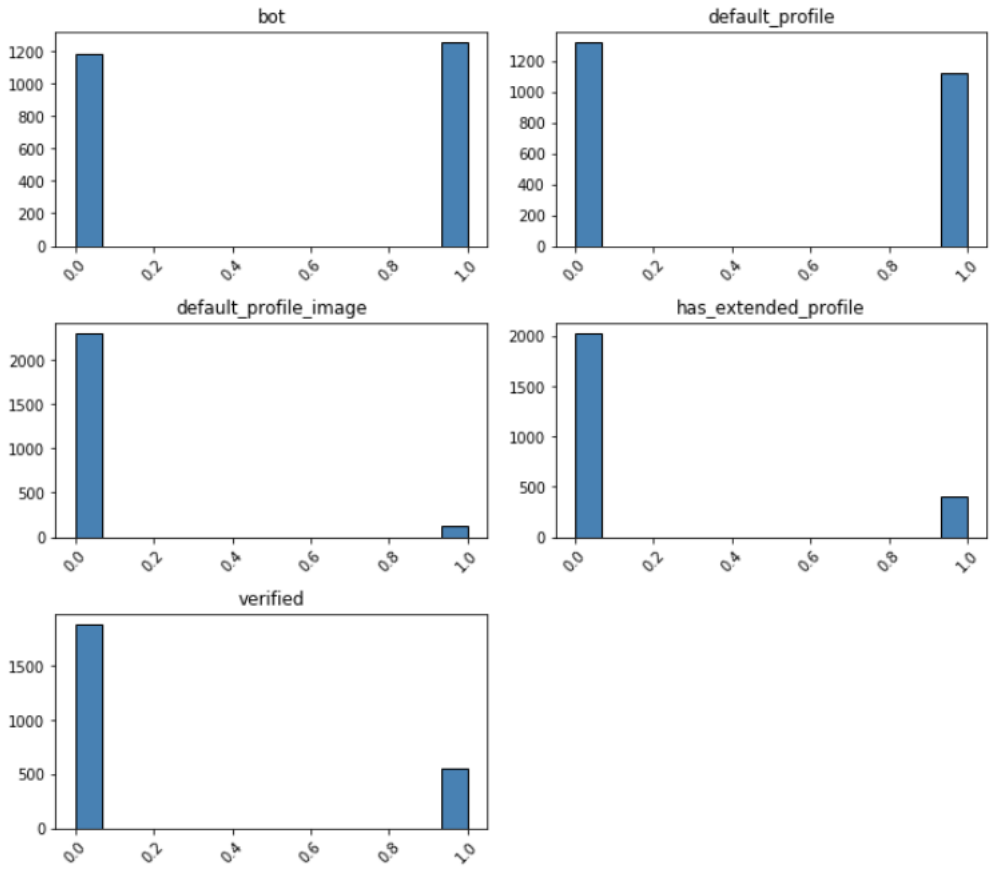

Figura 18: Distribución de las variables dicotómicas para el dataset *Detecting Twitter bot*.

En cuanto al análisis multi-variante se inicia realizando el cálculo de la correlación entre las variables de los datasets por medio de una matriz de correlación (ver Figuras 20 y 24). Para ello, se emplea el método de *Spearman* debido a su gran aplicabilidad en este campo. El valor de la correlación se muestra en cada uno de los elementos de la matriz. Además, se ha seleccionado como representación un mapa de calor para que de esta forma sea fácil identificar zonas con correlaciones elevadas. A la vista de los resultados las zonas con alta correlación hacen referencia a las variables que indican el comportamiento o actividad del usuario dentro de la red social, por ejemplo, las variables *verified* y *followers_count* tienen una fuerte

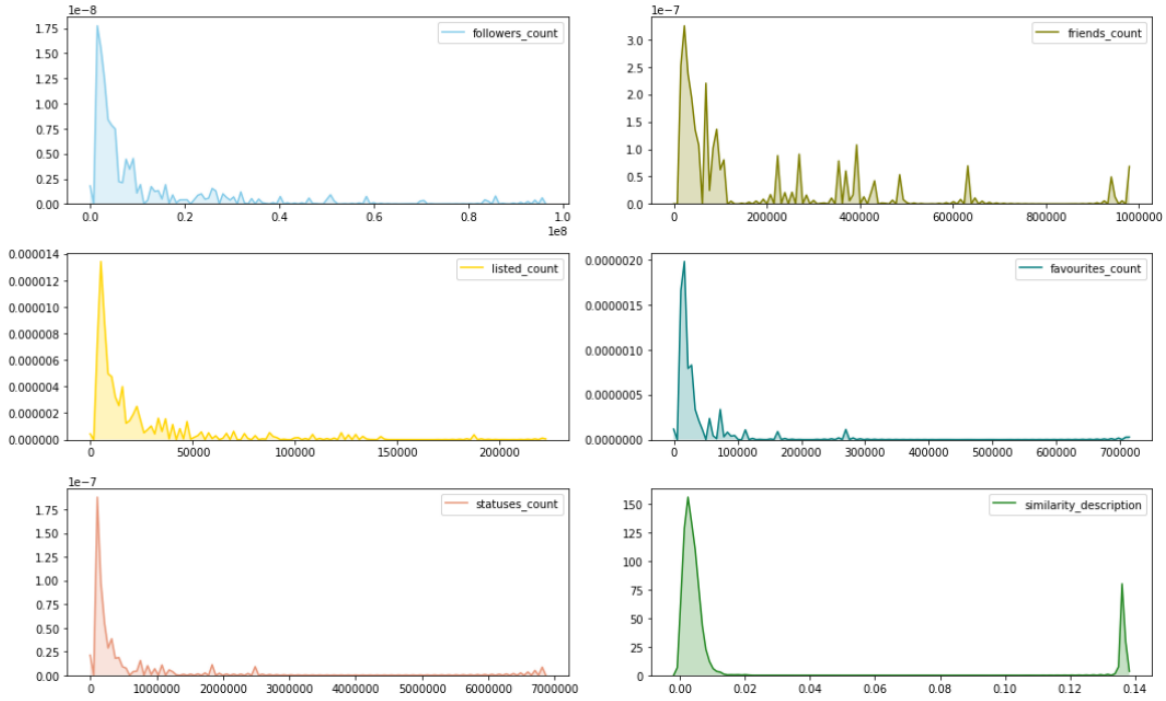

Figura 19: Distribución de las variables continuas para el dataset *Detecting Twitter bot*.

correlación.

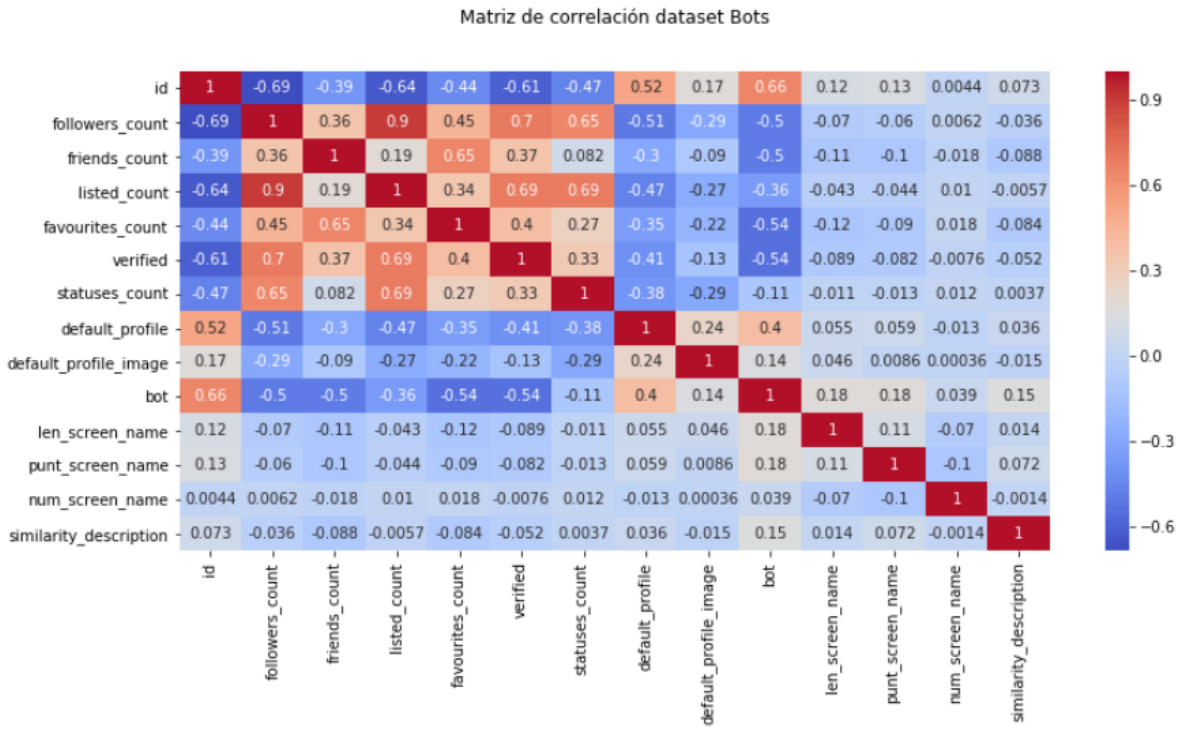

Figura 20: Matriz de correlación del dataset *Detecting Twitter bot*.

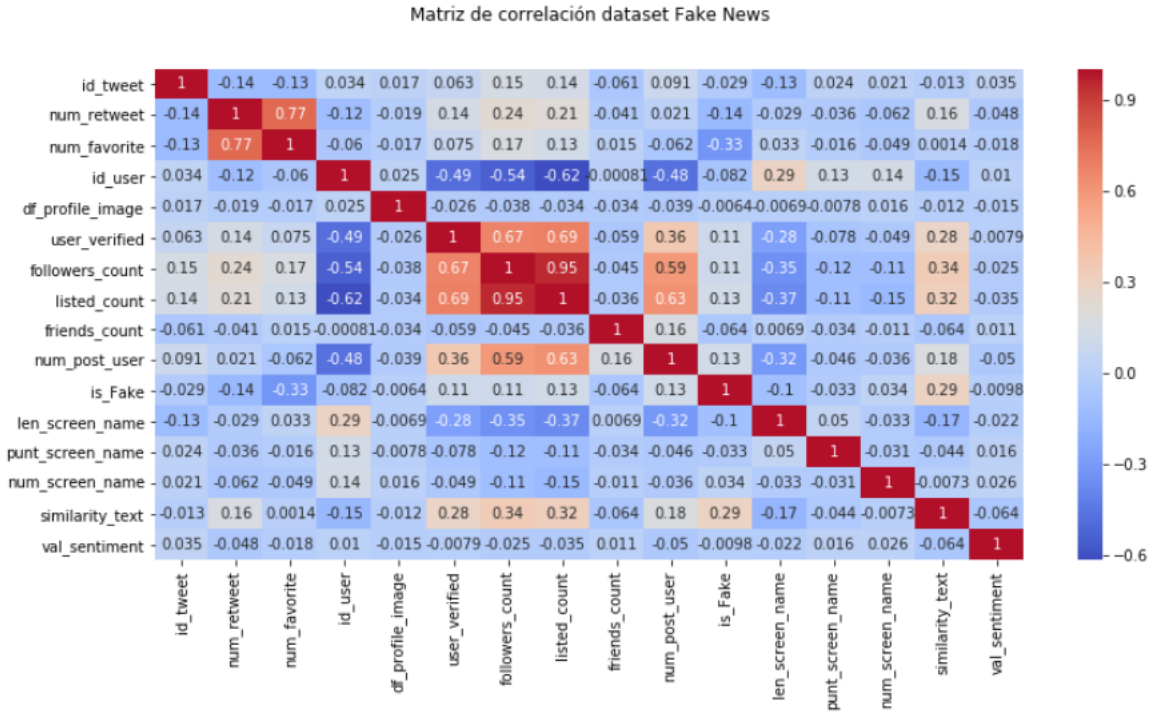

Figura 21: Matriz de correlación del dataset *PHEME*.

Tras esto, se selección las variables con mayor correlación para visualizar dicha correlación, con el objetivo de ver la linealidad que presentan (ver Figura 11). Como se esperaba a mayor valor de correlación la linealidad entre las variables implicadas aumenta.

Por último, se seleccionan variables con alta correlación para ver cómo se distribuyen en función de la variable objetivo (ver Figura 23). En este caso se seleccionan las variables *friends_count* y *followers_count* del dataset *Detecting Twitter bot*. A la vista del resultado se puede ver una gran diferencia en el comportamiento ya que los datos etiquetados como bots al tener pocos amigos tiene muchos seguidores sobre todo esto sucede con valores cercanos a 0. Mientras que los datos los etiquetados como no bots presentan unos valores más normales. Este estudio se realiza con el otro datasets también.

2.6.3.3. Selección de características

Con el objetivo de disponer solo de aquellas variables de interés para la etapa de modelado filtrado las variables en los dos conjuntos de datos.En el caso del data-

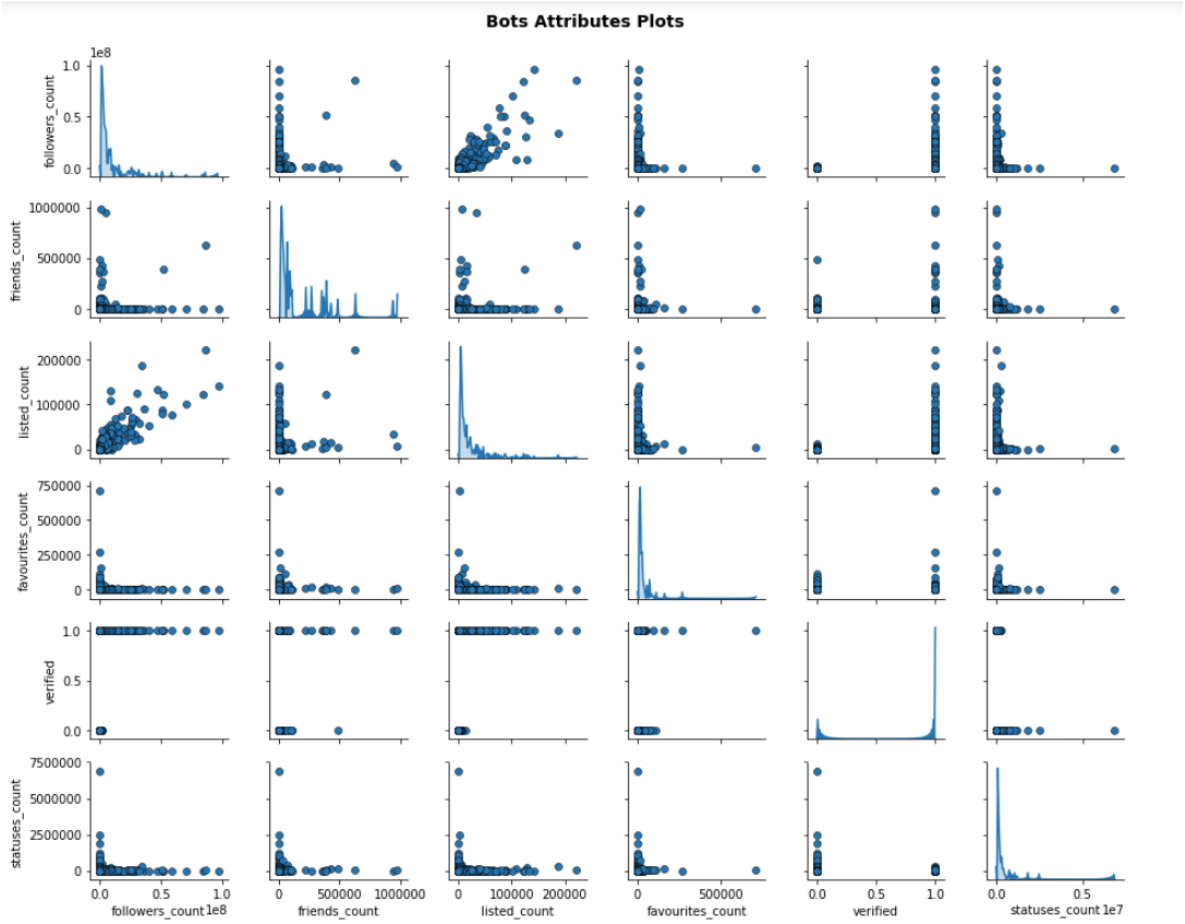

Figura 22: Representación variables con alta correlación del dataset *Detecting Twitter bot*.

set *Fake News Fake News* se eliminan las variables *text*, *description*, *date_create_tweet*, *date_user_create*, *name_user* y *screen_name* ya que no son útiles para el enfoque propuesto. En el caso de *Bots* se eliminan las variables *screen_name*, *location*, *description*, *url*, *created_at* y *name*. Por lo que las dimensiones de los dataset se reducen como se puede ver en la Figura 11. Con estos sub-conjuntos de datos se emplean en las etapas posteriores del análisis.

2.6.4. Modelado

Este apartado se centra indicar los distintos modelos que se han implementado para obtener el objetivo propuesto en el proyecto. Antes de iniciar el proceso de entrenamiento es necesario dividir los dataset en train-test asignando un porcentaje 70-30 (ver Figura 25) de los datos respectivamente. Con esta información se pasa

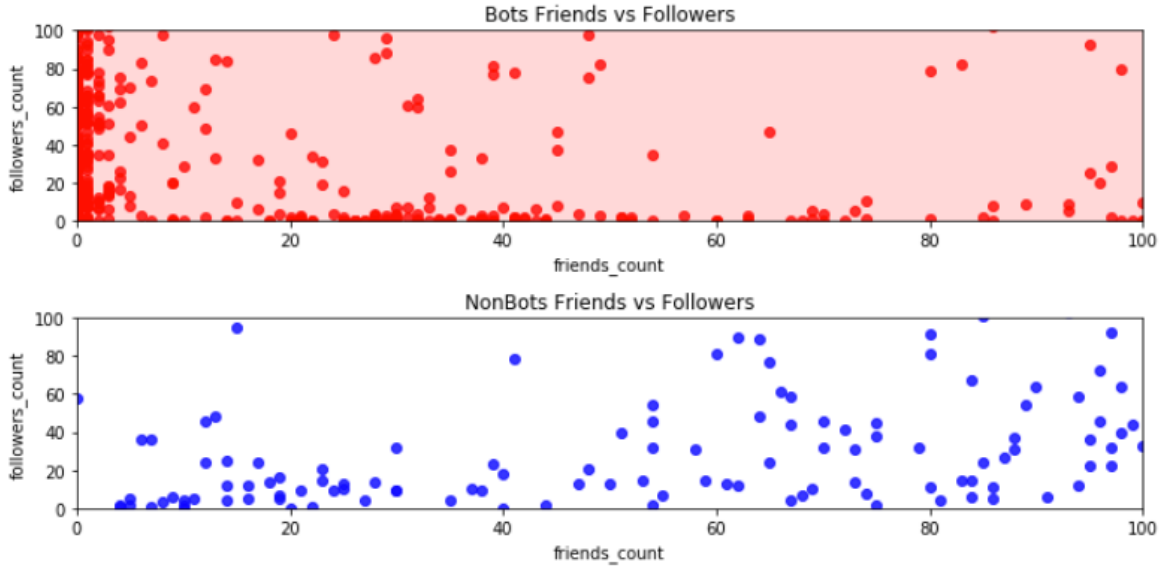

Figura 23: Relación entre las variables *friends_count* y *followers_count* en función de la variable objetivo para el dataset *Detecting Twitter bot*.

	Observaciones	Caracteristicas
PHEME	6425	16
Dete. Twitter bot	2442	14

Figura 24: Dimensiones de los datasets tras la selección de variables.

a entrenar los siguiente modelos: *Desiccion Tree*, *Random Forest*, *Support Vector Machine (SVM)* y *K-Means*. Debido a que los parámetros de estos modelos deben ser configurados por el desarrollador y su selección afectan directamente a su calidad se emplea el módulo *GridSearchCV* de la librería *Sklearn*.

El primer modelo entrenado es *Decision Tree* que se caracteriza por generar una estructura de árbol cuyas hojas corresponden a las variables del dataset para la toma de decisiones. El parámetro que se ha definido es *min_samples_split* con los valores 44 y 47 en el caso del dataset *PHEME* y *Detecting Twitter bot* respectivamente. Además, este modelo permite conocer las variables más importantes (ver Figura 26) que el modelo ha seleccionado permitiendo si fuese necesario generar un modelo más simple con aquellas variables que sean más importantes.

En el caso del modelo *Random Forest* es un modelo basado en *Decision Tree* que se caracteriza por generar muchos árboles y por votación se asigna a la clase a la que

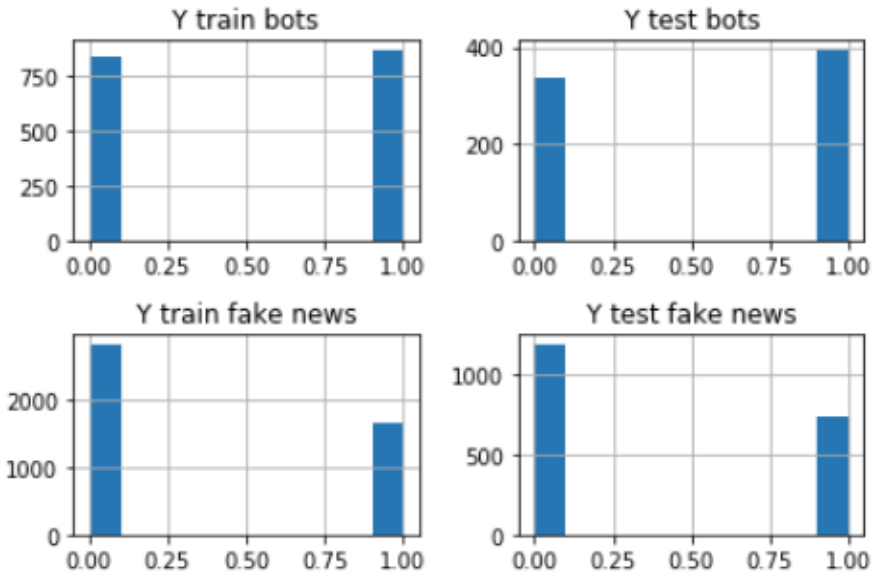

Figura 25: Distribución de la variable respuesta en train-test para *Detecting Twitter bot* y *PHEME*.

pertenece un determinado elemento. Dispone de varios parámetros configurables, por ejemplo, el parámetro como *max_depth* que hace referencia a la profundidad de cada uno de los árbol. De la misma forma que el modelo anterior se puede obtener las variables más representativas que el modelo a seleccionado (ver Figura 27).

Tras esto, se pasa a definir el modelo *Support Vector Machine (SVM)* que se caracteriza por intentar encontrar un hiper-plano capas de separar de las clases. Para ello, es necesario definir los parámetro *C* y *Gamma*. En este caso se empleado los valores *C* $0{,}1$ y *Gamma* $0{,}0001$ para el dataset *PHEME* y los valores *C* $0{,}1$ y *Gamma* $0{,}001$ para el dataset *Detecting Twitter bot*.

Por último, el modelo *K-means* se caracteriza por intentar agrupar a individuos con características similares sin tener conocimiento previo de la clase a la que pertenecen. Para ello, es necesario el cálculo del número óptimo de clusters para cada uno de los dataset. Aunque existen varios métodos uno de los más utilizados es el método de *Elbow* (ver Figura 28) que en nuestro caso indica que deberíamos seleccionar 2 o 3 cluster. Como método adicional se emplea *Silueta* obteniendo como resultado 2 clusters, por lo que se selecciona selecciona 2 clusters. Tras esto, se entrena el modelo y se pasa representar los resultados obtenidos (ver Figura 29) en cada uno de los datasets. Para obtener esta presentación se ha empleado *PCA* sobre los datos de entras para de esta forma realizar una representación en dos dimensiones

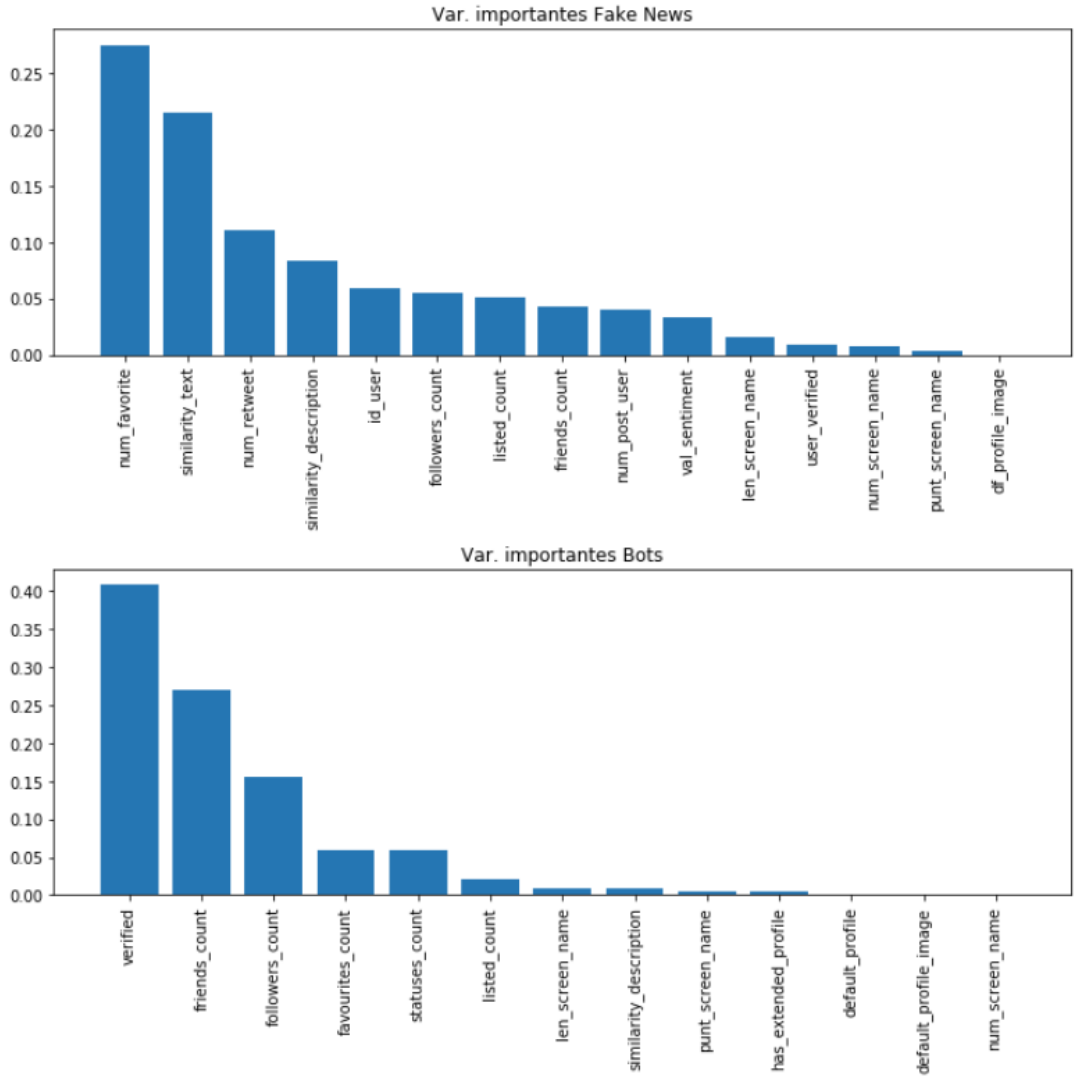

Figura 26: Representación de variables importantes del modelo *Random Forest* para *Detecting Twitter bot* y *PHEME*.

del dos grupos que establece el modelo y las variables.

Tras haber definido los modelos anteriores es necesario seleccionar el más adecuado para cada datasets. Por ello, se emplea el *Área bajo la curva (ROC)* como mecanismo para seleccionar de una forma objetiva el modelo que cumpla las necesidades del problema. En esta representación se compara el número de fallos al asignar un elemento a la categoría incorrecta y los aciertos al asignar un elemento a la categoría correcta . A la vista de la comparación del AUC (ver Figura 30) que proporciona cada modelo se puede concluir que: el mejor modelo para ambos datasets es *Random Forest* aunque el modelo de *Desicion Tree* ofrece valores similares. Además, el

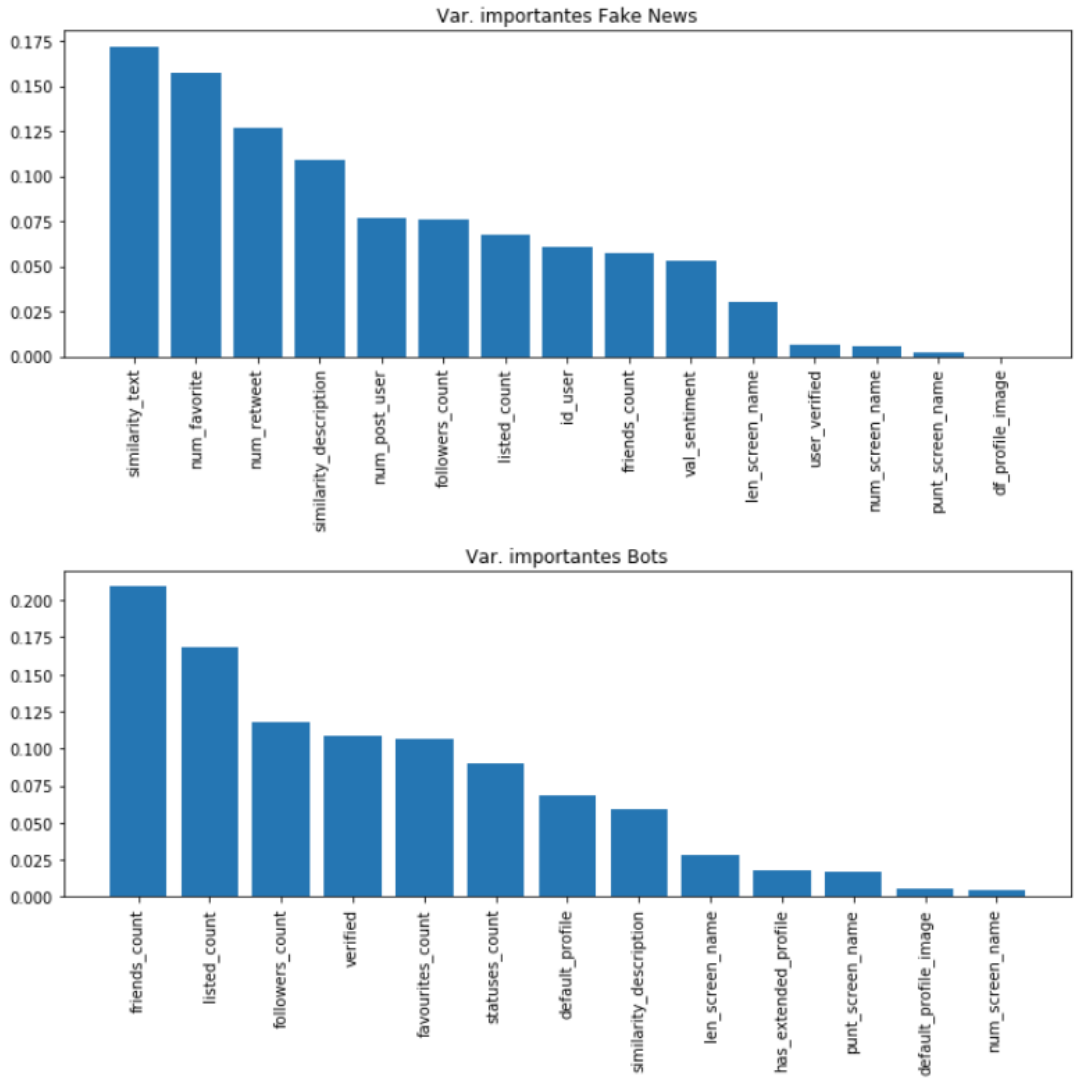

Figura 27: Representación de variables importantes del modelo *Random Forest* para *Detecting Twitter bot* y *PHEME*.

peor modelo es *K-means* ya que proporciona un valor muy bajo. En conclusión se selecciona como modelo definitivo *Random Forest* debido a su valor de AUC y las variables que considera importantes para realizar la clasificación.

2.6.5. Evaluación

Una vez se ha seleccionado el modelo adecuado con los datos disponibles en la fase de entrenamiento el siguiente paso es evaluar su comportamiento o capacidad predictiva frente a nuevos datos que no ha visto antes. Para ello, se emplea los conjuntos de tests separados previamente obteniendo su capacidad predictiva

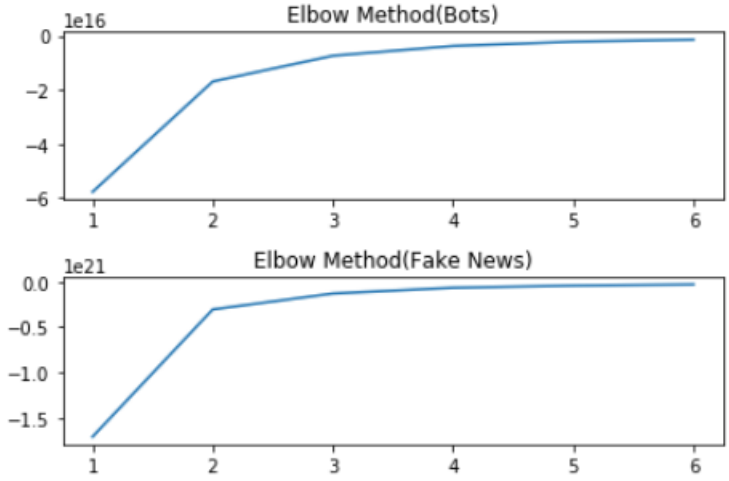

Figura 28: Método del *Elbow* para los dataset *Detecting Twitter bot* y *PHEME*.

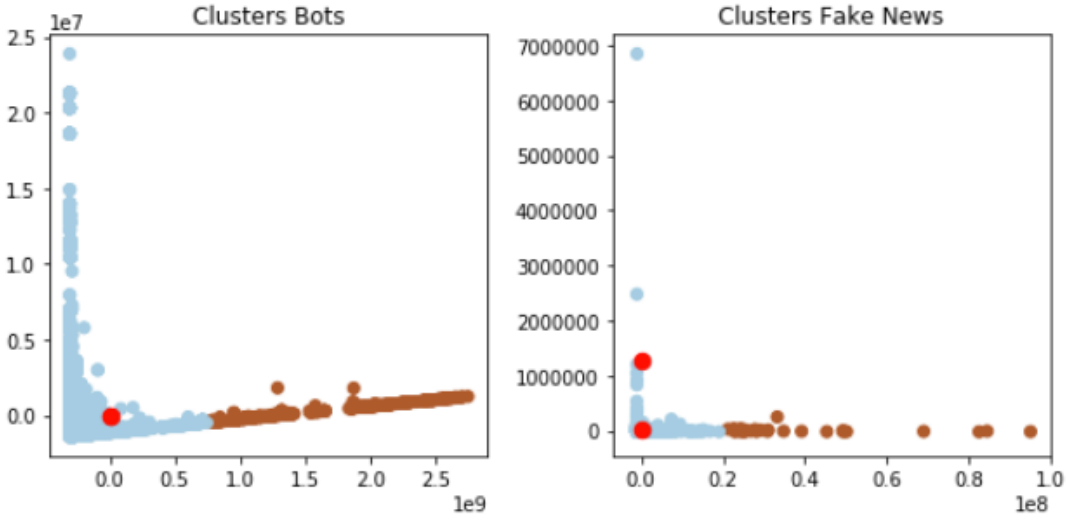

Figura 29: Representación *K-means* para *Detecting Twitter bot* y *PHEME*.

aplicando la matriz de confusión (ver Figura 31). Como se puede ver los falsos positivos y falsos negativos no son elevados para ambos dataset aunque el número es mayor en el dataset *PHEME*. Por último, se generar la *ROC* en la etapa de test (ver Figura 32) obteniendo valores aceptables por lo que el modelo generiza bien.

2.7. Comunicación/visualización de los resultados

Esta sección se encarga de mostrar el aspecto que tiene el documento generado para comunicar los resultados obtenidos en las diferentes etapas del análisis reali-

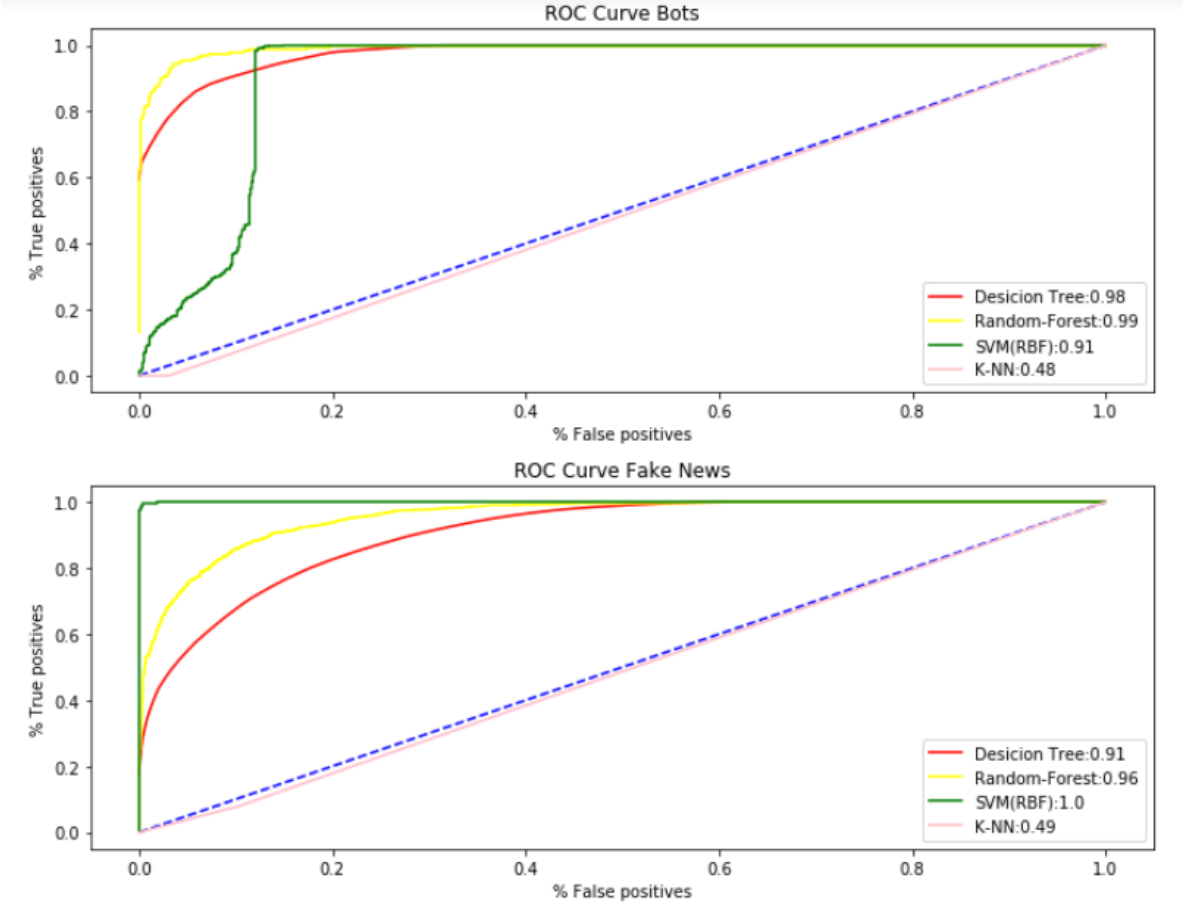

Figura 30: Curva ROC en train para los algoritmos diseñado para *Detecting Twitter bot* y *PHEME*.

	Matrix Bots		Matrix Fake News	
	0	1	0	1
0	292	47	1019	171
1	39	355	312	426

Figura 31: Matrices de confusión en test para el algoritmo *Random Forest*.

zado para el proyecto (ver Figura 33). Como se puede ver el documento consta de un índice que es navegable lo que permite ir a la sección que se quiera. Además, contiene el código generado en cada una de las etapas y el resultado obtenido de esta forma se permite la trazabilidad del proyecto.

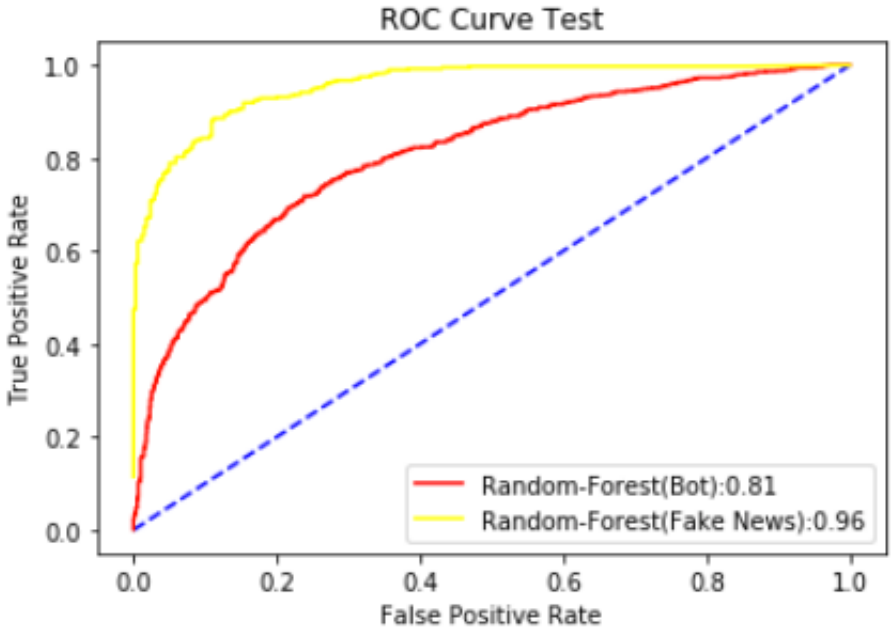

Figura 32: Curva ROC en test para el algoritmo *Random Forest*.

Tabla de contenido

Carga de datos

Se realiza la carga del conjunto de datos iniciales del proyecto. Se disponen de dos tipos: uno empleado para la detección de **Fake News** y el otro para la detección de **Bots**, ambos contienen información extraida de Twitter.

Fake News

```
data_fake = pd.read_csv("./data/data_fake_news.csv")
data_fake.head(1)
```

	id_tweet	screen_name	description	date_create_tweet	text	num_retweet	num_favorite	id_user	df_profile_image	user_ve
0	552785551930454016	BreakingNews	Get our free iOS or Android app to unlock more...	2015-01-07	Witness says multiple gunmen involved in shoot...	179	31	6017542	False	True

Bots

Figura 33: Curva ROC en test para el algoritmo *Random Forest*.

Capítulo 3

Desarrollo de la propuesta

Este capítulo se centra en explicar las decisiones que se han llevado a cabo durante el desarrollo de la propuesta. La sección 3.1 describen las etapas por las que ha pasado el desarrollo señalando los puntos más importantes de cada una.

3.1. Etapas del desarrollo

Esta sección describe las etapas de desarrollo de la propuesta. Se detalla el proceso que se ha llevado a cabo y las decisiones que se han tomado en cada una de ellas. Se muestran las etapas en orden cronológico, siendo la primera etapa el apartado 3.1.1 que trata la obtención de datos. Después, el apartado 3.1.2 se trata de entender los datos obtenidos. Tras esto, en el apartado 3.1.3 trata el proceso preparación de los datos. Finalmente, el apartado 3.1.4 y 3.1.5 trata de la creación del modelo y su evaluación respectivamente.

3.1.1. Obtención de datos

Tras haber investigado sobre la detección de noticias falsas y de bots en redes sociales se ha detectado como principal inconveniente la falta de variedad de datasets que permitan tratar los distintos formatos en los que se puede tratar la creación de noticias falsas [29]. Debido a esta limitación la comunidad de investigación ha publicado varios datasets para tratar los distintos formatos y de esta forma avanzar en esta área de investigación.

En la actualidad se ha publicado el dataset *FakeNewsNet* [27] que ha sido diseña-

do con el objetivo de obtener un solo dataset con todas las variantes. Debido a los objetivos marcados por el proyecto como primera aproximación se seleccionó este dataset. Para obtener los datos es necesario descargar el proyecto y ejecutarlo de forma local. Aunque la instalación y la puesta en marcha se realiza rápidamente la extracción de la información se convierte en un proceso muy costoso debido al gran número de datos que es necesario descargar. Además, a medida que la información se descargaba se realizaban comprobaciones detectando que la información en muchos casos estaba incompleta. Debido a estos problemas se tomó como alternativa emplear los datasets *PHEME* y *Detecting Twitter bot* para detectar noticias falsas y bots en redes sociales respectivamente.

El dataset *PHEME* fue seleccionado por ser acceso libre y la información pertenecen a la red social de *Twitter*. Esta información se encuentra en una carpeta comprimida. Dentro de ella, la información se encuentra divida en sub-carpetas en función de la temática de la noticia. Además, la información se encuentra clasificada en rumores y no rumores por especialistas. Como la información esta almacenada en directorios es necesario aplicar una transformación de formato. Para ello, se ha generado el fichero *transform_data.py* que se encarga de acceder a los distintos directorios *source-tweets* para obtener la información de los ficheros *JSON*. La información se filtra seleccionando las características más relevantes, por ejemplo, la variable *text* que contiene el texto del tweet. Por último, toda esta información se almacena en el fichero *data_fake_new.csv*.

En cuanto al dataset *Detecting Twitter bot* se encuentra publicado en *Kaggle*. La información que contienen se ha extraído de la red social de Twitter sobre cuentas etiquetas como bots. Dicha información se encuentra separada en los siguientes ficheros: *training_data_2_csv_UTF.csv* y *test_data_4_students.csv*. La principal diferencia entre estos ficheros es el etiquetada de los registros. Por ello, en el desarrollo del proyecto se emplea solo el primer fichero debido a esta limitación.

3.1.1.1. Entendimiento de los datos

Una vez se han obtenido los datos es necesario entender las variables disponibles con el objetivo tener una idea general sobre el dataset. Para realizar esta tarea se ha consultado la documentación de la API REST [2] de Twitter en la que se de-

talla cada una de las variables permitiendo identificar las características que están relacionadas con el tweet o el usuario que lo ha publicado.

3.1.2. Generación de nuevas características

Debido al estudio realizado sobre la detección de noticias falsas y/o bots se descubrió que la generación de nuevas características en función de las existentes en los datasets a portaban más información. Por ello, este módulo se diseña para generar 3 o 5 nuevas características en función de la información disponible, esto se debe a que las nuevas variables se basan en variables de tipo texto. Del conjunto de variables se seleccionan las variables *screen_name* y *description* ya que la generación de este contenido entre un usuario normal y un bot es diferente. Por último, se selecciona la variable *text* con el objetivo de estudiar la similitud entre los distintos valores de esta variable y de esta forma ayudar en la detección de contenido verídico o no verídico, además se calcula el valor sentimental de dicha variable ya que en estudios previos se ha visto la utilidad de esta información.

3.1.3. Análisis y selección de características

En etapa del proyecto el objetivo es estudiar la distribución de las variables y ver si existe alguna relación entre ellas por lo que se ha divido en un análisis univariante y multi-variante. En el primer caso se visualiza la distribución de cada una de las variables para conocer los valores más frecuentes y otras características de las variables. Para el segundo caso se calcula la correlación por medio de una matriz empleando el método de *Sperman* de esta forma los valores cercanos a 1 son de interés ya que muestran gran relación entre ellas. Los análisis empleados son básicos debido al conocimiento limitado sobre las técnicas que se suelen emplear en esta etapa.

3.1.4. Modelado

Una vez se ha realizado el análisis y se ha filtrado las variables con las que se va a trabajar se pasa a la etapa de modelado. Esta etapa se centra en diseñar diferentes algoritmos para seleccionar el algoritmo que se ajuste mejor a los objetivos y

presente mejores resultados. Por ello, se han seleccionado algoritmos ampliamente utilizados en el área de *Data Sciences* como son *Desicion Tree*, *Random Forest*, *Support Vector Machine (SVM)* y *K-means*. Debido a que a que es necesario definir una serie de parámetros para cada modelo y realizar esta tarea de forma manual no sería correcto se emplea el módulo *GridSearchCV* permitiendo automatizar esta tarea. Por último, para realizar una selección objetiva en función de las características de los diferentes modelos se empleado la curva *ROC*, seleccionando aquel modelo que presente mejores prestaciones. Hay que destacar que aunque el modelo *Support Vector Machine (SVM)* para el dataset *PHEME* presenta mayor accuracy no se ha seleccionado ya que puede ser que posea overfitting y computacionalmente es más complejo que *Random Forest*.

3.1.5. Evaluación

Tras haber selecciona el modelo más apropiado para cada uno de los datasets es necesario evaluar su capacidad para generalizar al intentar evaluar nuevos datos que previamente no había visto. Por ello, se emplean los conjuntos de test previamente obtenidos para conseguir el porcentaje de aciertos y fallos que proporcionan los modelos. Para esta tarea se genera una matriz de confusión ya que es forma cómoda e intuitiva para entender este tipo de información. Por último, también se ofrece la representación en esta etapa de la curva *ROC* de esta forma se conoce el accuracy de los modelos permitiendo identificar si funcionan correctamente.

Capítulo 4

Conclusiones y trabajo futuro

En este capítulo se muestran las conclusiones alcanzadas en el proyecto y se identifican posibles líneas de trabajo futuro que han sido detectadas a lo largo del desarrollo. En la sección 4.1 se recapitula el contexto del análisis desarrollado y expone las conclusiones obtenidas del proyecto. En la sección 4.2 se muestran las posibles líneas de trabajo futuro para mejorar la detección.

4.1. Conclusiones

La propuesta planteada en este proyecto ha sido el desarrollo de un modelo matemático o algoritmo que permita la detección de noticias falsas y bots en redes sociales. detectar noticias falsas a través del análisis de sentimiento del texto. Debido al enfoque del proyecto se ha seleccionado como metodología *CRIPS-DM*. Por lo que el desarrollo se realiza en una serie de etapas para abordar el análisis.

El análisis propuesto emplea como conjunto de datos *PHEME* y *BOTS* para abordar los objetivos del proyecto. Para los conjuntos de datos se generan nuevas características en función de las existentes. Para ello, se han empleado técnicas de procesamiento del lenguaje natural por medio procesador *NLTK* que permite trabajar el texto en busca de palabras relevantes y estudiar la similitud en función de la relevancia de las palabras. Además, se ha empleado análisis de sentimiento por por medio del lexicón *SentiWornet*. La generación de estas variables han sido de utilidad ya que el modelo final las emplea.

Respecto al análisis exploratorio de datos nos ha permitido conocer la distribu-

ción de las variables y encontrar aquellas que presentan mayor relación entre sí por medio de una matriz de correlación. Además, esto ha permitido ver que la distribución de estas variables en función de la variable objetivo es diferente.

Una vez, se ha terminado las etapas anteriores se ha pasado a la generación de varios modelos de *Machine Learning*. Para seleccionar el modelo más apropiado se ha empleado la ROC ya que de esta forma se realiza una selección objetiva. Por último, una vez se ha seleccionado el modelo para cada dataset que en este caso de *Random Forest* se pasa a evaluar el modelo con nuevos datos. Esto da como resultado que el modelo seleccionado generaliza bastante bien en ambas cosas.

Por lo tanto, se puede concluir que el análisis realizado en el proyecto se puede considerar como un primera aproximación para la detección de noticias falsas y bots por medio del desarrollo algoritmos de *Machine Learnig*. Además, se puede concluir que la generación de nuevas características ayuda en la construcción de los modelos. Al tratarse una primera aproximación se debería investigar sobre la generación de nuevas características y utilizar distintas aproximaciones.

4.2. Trabajos futuros

En esta sección se detallan las posibles líneas de trabajo futuro que permitan mejora el análisis propuesto. Tras terminar el desarrollo del proyecto se han detectado los siguientes puntos de mejora

- **Fuentes de datos:** se debería desarrollar nuevos conjuntos de datos que permitan detectar noticias falsas y bots en redes sociales de forma conjunta sin la necesidad de emplear conjuntos separados.
- **Exploración de los datos:** se debería realizar un análisis en mayor profundidad de las variables de los datasets ya que en esta primera aproximación se han empleado técnicas muy básicas.
- **Generación del informe :** se debería mejorar el informe generado ya que se debería crear informes parciales con el objetivo de tener informes más comprensibles y poco extensos para su análisis.
- **Mejora en la detección:** en cuanto a la detección se han empleado las técnicas

más comunes en este ámbito, por lo que sería conveniente intentar emplear otro tipo de algoritmos , por ejemplo, Redes Neuronales o combinando modelos entre sí.

Capítulo 5

Glosario

Machine Learning: también llamado aprendizaje automático. Es una de las ramas de la inteligencia artificial.
Dataset: se denomina al conjunto de datos de trabajo en el ámbito de *Machine Learning*.
Dataframes: estructura de datos similar a una tabla que permite trabajar por filas y columnas.
Overfitting: sobre-entrenamiento de un algoritmo de aprendizaje automático.
Frameworks: marco de trabajo.

Capítulo 6

Anexo: Código fuente de los ficheros auxiliares

Este capítulo muestra el código que ha sido desarrollado en ficheros auxiliares como apoyo en las distintas etapas del proyecto.

```
def format_date_twitter(v_date):
    parse_fecha = parser.parse(v_date)
    return str(parse_fecha.date())

def init_csv():
  with open('./data/data_fake_news.csv', 'w') as csvfile:
    writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
    writer.writeheader()

def select_features(json_data, label):
  date_create_tweet = format_date_twitter(json_data['created_at'])
  df_profile_image =  True if (json_data['user']["default_profile_image"]) else False
  df_profile =  True if (json_data['user']["default_profile"]) else False
  user_verified = True if (json_data['user']["verified"]) else False
  user_create = format_date_twitter(json_data['user']["created_at"])

  dict_data = {
    'id_tweet': json_data['id'],
    'screen_name': json_data['user']['screen_name'],
    'description': json_data['user']['description'],
    'date_create_tweet': date_create_tweet,
    'text': json_data['text'],
    'num_retweet': json_data['retweet_count'],
    'num_favorite':json_data['favorite_count'],
    'id_user': json_data['user']['id'],
    'df_profile_image':df_profile_image,
```

```
    'user_verified': user_verified,
    'followers_count': json_data['user']['followers_count'],
    'listed_count': json_data['user']["listed_count"],
    'friends_count': json_data['user']["friends_count"],
    'name_user': json_data['user']["name"],
    'date_user_create': user_create,
    'num_post_user': json_data['user']["statuses_count"],
    'is_Fake':label,
  }
  return dict_data


def save_data(dict_data):
    spamWriter = csv.DictWriter(open('./data/data_fake_news.csv', 'a'), delimiter= ',',
        quoting=csv.QUOTE_NONNUMERIC, fieldnames=fieldnames)
    spamWriter.writerow(dict_data)


def count_number_files(path,label):
  listing_data = os.listdir(path)
  num_files = 0
  for data in listing_data:
    if data != '.DS_Store':
      path_data = path+data+'/source-tweets/'
      files = os.listdir(path_data)
      num_files = num_files + len(files)
      data_raw = open(path_data+"/"+files[0]).read()
      json_data = json.loads(data_raw, encoding='utf8')
      data_filter = select_features(json_data,label)
      save_data(data_filter)
  #
  return num_files


def read_data():
  path_root = './data/data_pheme/'
  listing_folder = os.listdir(path_root)
  result_data = []
  # Por cada uno de los ficheros
  for folder in listing_folder:
    name_topic = folder.split("-")[0]
    if folder != '.DS_Store':
      num_file_rumors = count_number_files(path_root+folder+'/rumours/', 1)
      num_file_nonrumors = count_number_files(path_root+folder+'/non-rumours/', 0)
      result_data.append({'topic': name_topic, 'num_rumors':num_file_rumors, 'num_nonrumors
          ':num_file_nonrumors})
  ##
```

```
  return result_data

if __name__ == "__main__":
  fieldnames = ['id_tweet', 'screen_name', 'description', 'date_create_tweet','text', '
      num_retweet','num_favorite','id_user','df_profile_image','user_verified',
    'followers_count', 'listed_count', 'friends_count', 'name_user','date_user_create','
       num_post_user', "is_Fake"]
  init_csv()
  data_summarize = read_data()
```

Listing 6.1: Código empleado para transformar el formato de datos (transform_data.py)

```
class Operate_Similarity:

    def __init__(self, dim_1):
        self.dim_data = dim_1 #
        self.text_process = Process(False)
        self.idfDict = {}

    def calculate_worset(self, list_texts):
        wordSet = []
        for text in list_texts:
            token_data = self.text_process.analitc_text(text)
            wordSet.extend(token_data)
        self.wordSet = set(wordSet)

    ###
    def create_matrix(self):
        matrix_basic = np.zeros((self.dim_data, len(self.wordSet)))
        self.df_matrix_basic = pd.DataFrame(matrix_basic, columns=self.wordSet)

    ###
    def calculate_TF(self, list_texts):
        index_tweet = 0
        for text in list_texts:
            token_data = self.text_process.analitc_text(text)
            wordDict = dict.fromkeys(self.wordSet, 0)
            for word in token_data:
                wordDict[word]+=1
            tfDict = {}
            bowCount = len(token_data)
            if bowCount > 0:
                for word, count in wordDict.items():
                    tfDict[word] = count/float(bowCount)

            for word in token_data:
```

```
                self.df_matrix_basic.at[index_tweet, word] = tfDict[word]
            index_tweet += 1

    def calculate_IDF(self, list_texts):
        self.idfDict = dict.fromkeys(self.wordSet, 0)
        for text in list_texts:
            token_data = self.text_process.analitc_text(text)
            wordDict = dict.fromkeys(self.wordSet, 0)
            if len(token_data) > 0:
                for word in token_data:
                    wordDict[word]+=1

                for word, val in wordDict.items():
                    if val > 0:
                        self.idfDict[word] += 1

        for word, val in self.idfDict.items():
            self.idfDict[word] = math.log10(self.dim_data / float(val))

    def calculate_TF_IDF(self):
        for data in self.df_matrix_basic:
            self.df_matrix_basic[data] = self.df_matrix_basic[data] * self.idfDict[data]

    def calculate_similarity(self, name_feature):
        name_colum = 'similarity_'+ name_feature
        simi_cosine = cosine_similarity(self.df_matrix_basic).mean(axis=1)
        df_simil_description = pd.DataFrame(list(simi_cosine), columns = [name_colum])
        return df_simil_description

    def exe_process(self, list_texts , name_feature):
        self.calculate_worset(list_texts)
        self.create_matrix()
        self.calculate_TF(list_texts)
        self.calculate_IDF(list_texts)
        self.calculate_TF_IDF()
        return self.calculate_similarity(name_feature)
```

Listing 6.2: Clase empleada para calcular la similitud (calculate_similarity.py)

```
class Process:

    def __init__(self, need_sentiment):
        self. stops = stopwords.words('english')
        self.lemmatizer = WordNetLemmatizer()
        self.calculate_sent = need_sentiment
```

```
    def remove_special_characters(self, text):
        v_data =  text.replace('"',"")
        return v_data.lower()

    def remove_stop_words(self, tokens):
        list_tokens = []
        for token in tokens:
            if token[0] not in self.stops and len(token[0]) > 2:
                list_tokens.append((token[0], token[1]))
        return list_tokens

    def remove_punctuation(self, tokens):
        result = []
        for token in tokens:
            punct_removed = ''.join([letter for letter in token[0] if letter in string.
                ascii_letters])
            if punct_removed != '':
                result.append((punct_removed,token[1]))
        return result

    def wordnet_value(self, value):
        result = ''
        if value.startswith('J'):
            return wordnet.ADJ
        elif value.startswith('V'):
            return wordnet.VERB
        elif value.startswith('N'):
            return wordnet.NOUN
        elif value.startswith('R'):
            return wordnet.ADV
        return result

    def lemmatize(self, token_list):
        result = []
        for token in token_list:
            if len(token) > 0:
                pos = self.wordnet_value(token[1])
                if pos != '':
                    result.append((self.lemmatizer.lemmatize(str(token[0]).lower(), pos=pos
                        ), pos))
        return result

    def analitc_text(self, text):
        v_data =  self.remove_special_characters(text)
        tokens = nltk.word_tokenize(v_data)
        tokens = nltk.pos_tag(tokens)
        token_data = self.remove_punctuation(tokens)
```

```
        token_data = self.remove_stop_words(token_data)
        token_data = self.lemmatize(token_data)
        if not self.calculate_sent:
            token_data = [ token[0] for token in token_data]
        return token_data
```

Listing 6.3: Clase empleada para procesar el texto (process_text.py)

```
import pandas as pd
from nltk.corpus import sentiwordnet as swn
from process_text import Process

class sentiment_process():

    def __init__(self):
        self.text_process = Process(True)

    def calculte_sentiment(self, tokens):
        sentiment_tweet = 0.0
        count_words = 0
        for token in tokens:
            synsets = list(swn.senti_synsets(token[0],token[1]))
            if len(synsets) > 0:
                word_total = 0
                for synset in synsets:
                    word_sentiment = synset.pos_score() - synset.neg_score()
                    word_total = word_total + word_sentiment
                count_words += 1
                word_total = round(word_total / len(synsets),3)

                if word_total != 0:
                    sentiment_tweet = sentiment_tweet + word_total

        if sentiment_tweet != 0:
            sentiment_tweet = round(sentiment_tweet/count_words,3)
        return sentiment_tweet

    def exe_process(self, list_text):
        list_sentiment = []
        for text in list_text:
            token_data = self.text_process.analitc_text(text)
            val_sent = self.calculte_sentiment(token_data)
            list_sentiment.append(val_sent)
        return  pd.DataFrame(list_sentiment, columns = ['val_sentiment'])
```

Listing 6.4: Clase empleada para calcular el sentimiento (calculate_sentiment.py)

Bibliografía

[1] Anaconda. https://www.anaconda.com/. [Online: accessed 01-May-2019].

[2] API Twitter. https://developer.twitter.com/en/docs/api-reference-index.html. [Online: accessed 01-May-2019].

[3] Detecting-twitter-bot-data. https://www.kaggle.com/charvijain27/detecting-twitter-bot-data. [Online: accessed 01-May-2019].

[4] JSON. https://json.org/json-es.html. [Online: accessed 01-May-2019].

[5] Jupyter. https://jupyter.org/. [Online: accessed 01-May-2019].

[6] Kaggle. https://www.kaggle.com/. [Online: accessed 01-May-2019].

[7] Matplotlib. https://matplotlib.org/. [Online: accessed 01-May-2019].

[8] NLTK. https://www.nltk.org/. [Online: accessed 01-May-2019].

[9] Numpy. https://www.numpy.org/. [Online: accessed 01-May-2019].

[10] Pandas. https://pandas.pydata.org/. [Online: accessed 01-May-2019].

[11] Pheme dataset for rumour detection and veracity classification. https://figshare.com/articles/PHEME_dataset_for_Rumour_Detection_and_Veracity_Classification/6392078/1. [Online: accessed 01-May-2019].

[12] Python. https://www.python.org/. [Online: accessed 01-May-2019].

[13] Rstudio. https://www.rstudio.com/. [Online: accessed 01-May-2019].

[14] Scikit-learn. `https://scikit-learn.org/stable/`. [Online: accessed 01-May-2019].

[15] Scipy. `https://www.scipy.org/`. [Online: accessed 01-May-2019].

[16] Seaborn. `https://seaborn.pydata.org/`. [Online: accessed 01-May-2019].

[17] SentiWordNet. `http://sentiwordnet.isti.cnr.it/`. [Online: accessed 01-May-2019].

[18] Twitter. `https://twitter.com/`. [Online: accessed 01-May-2019].

[19] Bayan Boreggah, Arwa Alrazooq, Muna Al-Razgan, and Hana AlShabib. Analysis of arabic bot behaviors. In *2018 21st Saudi Computer Society National Computer Conference (NCC)*, pages 1–6. IEEE, 2018.

[20] Johan Fernquist, Lisa Kaati, and Ralph Schroeder. Political bots and the swedish general election. In *2018 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 124–129. IEEE, 2018.

[21] Aurélien Géron. *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems*. .ºReilly Media, Inc.", 2017.

[22] Stefan Helmstetter and Heiko Paulheim. Weakly supervised learning for fake news detection on twitter. In *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 274–277. IEEE, 2018.

[23] Saranya Krishnan and Min Chen. Identifying tweets with fake news. In *2018 IEEE International Conference on Information Reuse and Integration (IRI)*, pages 460–464. IEEE, 2018.

[24] Sneha Kudugunta and Emilio Ferrara. Deep neural networks for bot detection. *Information Sciences*, 467:312–322, 2018.

[25] James Schnebly and Shamik Sengupta. Random forest twitter bot classifier. In *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0506–0512. IEEE, 2019.

[26] Alok Sharma, Arun K Pujari, and Kuldip K Paliwal. Intrusion detection using text processing techniques with a kernel based similarity measure. *computers & security*, 26(7-8):488–495, 2007.

[27] Kai Shu, Deepak Mahudeswaran, Suhang Wang, Dongwon Lee, and Huan Liu. Fakenewsnet: A data repository with news content, social context and dynamic information for studying fake news on social media. *arXiv preprint arXiv:1809.01286*, 2018.

[28] Kai Shu, Suhang Wang, and Huan Liu. Understanding user profiles on social media for fake news detection. In *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 430–435. IEEE, 2018.

[29] Xinyi Zhou, Reza Zafarani, Kai Shu, and Huan Liu. Fake news: Fundamental theories, detection strategies and challenges. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 836–837. ACM, 2019.