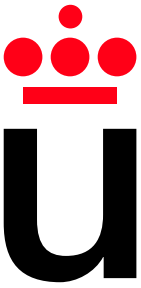


Universidad
Rey Juan Carlos

Escuela Técnica Superior
de Ingeniería Informática

Máster Ciberseguridad y Privacidad

Curso 2021-2022

Trabajo Fin de Master

IA/MACHINE LEARNING APLICADA A CIBERSEGURIDAD

Autor: Carlos Luzzatti Vázquez
Tutor: Raul Martín Santamaria

Agradecimientos

Breves agradecimientos o dedicatoria.

Resumen

¿Cómo la IA y las aplicaciones de machine learning pueden ser útiles?
Las IA´s actualmente se encuentran integradas en casi todos los entornos gracias a las distintas soluciones que dan en las diversas áreas de la sociedad, por lo cual la ciberseguridad no es una excepción ya que las amenazas de ciberseguridad informáticas ocasiones millones en daños.

Los algoritmos de machine learning o aprendizaje realizan tareas inteligentes sin que nadie las deba programar para esa tarea, esto lo realiza una aplicación dentro de la IA que realiza un modelo matemático que proviene de datos de entrenamiento para poder realizar un modelo con el que predecir los resultados cuando se introduzcan nuevos datos automático

La IA puede utilizarse para detección de amenazas, contraseñas y autentificación, prevención y detección de phishing entre otras.

Breve introducción al proyecto

Utilizaremos sets de datos de Kaggle para poder analizarlos y realizar las predicciones.

Nosotros utilizaremos técnicas de machine learning para predecir mediante un dataset de datos cuantos correos son spam y cuantos son correos seguros para así evitar posibles fraudes o infecciones del sistema, realizaremos esta acción mediante dos maneras diferentes para poder comprobar la exactitud que empleamos en ambos modelos, en uno de ellos utilizaremos un algoritmo que previamente compararemos con los mas utilizados y lo elegiremos y la otra manera sera mediante una red neuronal para ver que método puede resultar mas efectivo para esta acción, después de este proceso generaremos un fichero con los datos de los spam.

Para comparar los algoritmos mas utilizados lo haremos calculando su exactitud y su tiempo de ejecución y una vez analizados podremos valorar cual de ellos es el mas conveniente, nosotros en este caso después de analizarlo hemos optado por la opción ComplementNB

Para la red neuronal hemos optado por hemos utilizado el algoritmo de regresión logística que es un aprendizaje supervisado, pero contrariamente a su nombre, no es una regresión, sino un método de clasificación. Asume que los datos pueden ser clasificados (separados) por una línea o un plano de n dimensiones.

Los datos del dataset debemos preprocesarlos para eliminar características especiales y tokenizar las palabras para convertir esos datos en datos que el proceso soporte y pueda realizar las predicciones.

Para realizar este proceso necesitaremos realizar varias tareas dentro de nuestro programa necesitaremos importarnos las librerías necesarias para realizar todas las acciones, preparar los datos de entrenamiento, seleccionar un modelo o crearlo, entrenar el modelo con los datos de entrenamiento y usar el modelo para obtener las predicciones .

Hemos optado por utilizar dos modelos diferentes, uno basado en red neuronales y otro en el algoritmo para comprobar cual es la mejor opción para la clasificación de texto ya que para algunas tareas puede dar mejores resultados una opción u otra y siempre es favorable analizar cual es mas efectiva para un mejor rendimiento.

Palabras clave:

- Python
- Ciberseguridad
- Aprendizaje automático (pueden ser varias)0
- Machine learning
- Nodo
- IA
- Dataset
- Keras
- JSON
- Neurona
- ...

Índice de contenidos

1

Introducción y contexto

Machine Learning o aprendizaje automático se esta convirtiendo en una herramienta muy potente para garantizar la seguridad de un sistema, todo esto se debe a la capacidad de aprender a buscar de forma automática dentro de una gigantesca cantidad de datos los modelos de comportamiento y así poder detectar cualquier dato anómalo que sea indicativo de una amenaza para poder detectar las amenazas de forma automática, así se puede actuar inmediatamente implementando las soluciones de seguridad informática, el motor de la máquina en este caso sería el algoritmo que aprende revisando los datos y aprendiendo a predecir los futuros comportamientos, esto implica que los sistemas se van a ir mejorando de forma independiente a lo largo del tiempo que permanezca el algoritmo trabajando sin necesidad de intervención humana.

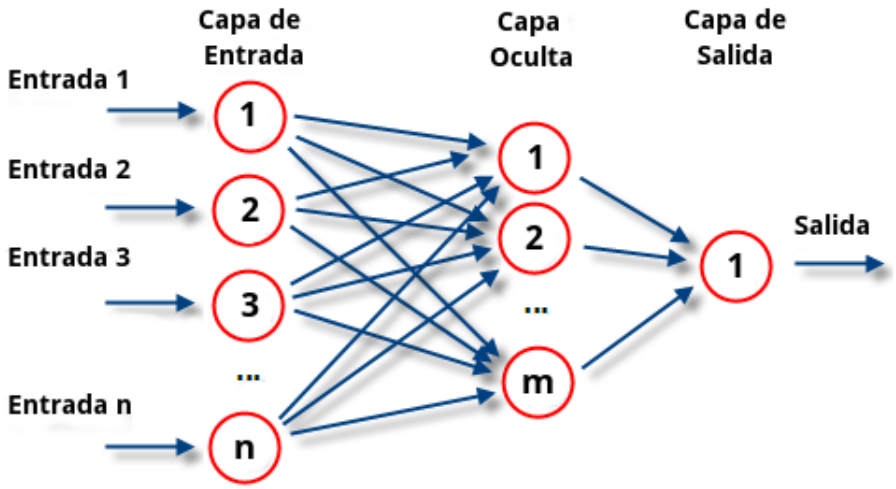
Machine Learning es un algoritmo que puede aprender entrenando con datos que ha recibido, estos algoritmos se pueden definir con redes neuronales definiendo las neuronas y las capas o bien podemos importar un modelo desde keras para facilitarlo, también es posible crear un red neuronal he importarla en formato JSON .

1.1. Que es una red neuronal

Las neuronas artificiales intentan imitar el comportamiento de una neurona cerebral, se compone de unas ramificaciones y un núcleo o nodo, existirán ramificaciones de entrada al nodo que se procesara y genera una información de salida que se trasmite por las ramificaciones de salida hacia otras neuronas.

Para este proceso vamos a utilizar una red neuronal que vaya a adquirir conocimientos mientras recibe información para relacionarla entre sí. De esta manera, puede aprender igual que un cerebro humano.

Con este proceso se crean patrones de comportamiento que se obtienen a partir del análisis de un conjunto de datos para que las maquinas aprendan solas a base de los anteriores resultados, por lo que a mayor volumen de datos disponibles nuestro algoritmo seria mas complejo.



Funcionamiento red neuronal [1]

1.2. Capas y neuronas

Las capas se pueden definir como los diferentes niveles de lo que dispone la red neuronal, a su vez estos se componen de uno o varios conjuntos de neuronas de las que sus entradas dependen de una capa anterior a menos que sea la primera capa que depende de los datos de entrada, de la primera capa las salidas serían la entrada de la próxima capa, la neurona se puede considerar la unidad funcional de los modelos de redes. En cada neurona pueden ocurrir dos tipo de operaciones:La suma ponderada de las entradas y también se aplica la función de activación.

La primera capa se conoce como la capa de entrada que deberá tener tantas neuronas como los tipos de datos que queremos analizar, en nuestro caso como queremos analizar la columna donde muestra el texto del mensaje al ser solo una columna nuestra capa de entrada solo constara de 1 neurona .

Las capas ocultas recibe los valores de la capa de entrada, que son ponderados por los pesos, las unidades se conectan con fuerzas de conexión variables (o ponderaciones), dentro de estas

capas se define el numero de neuronas que necesitamos, contra mas neuronas tenga cada capa mas tardara el modelo en entrenar pero aumenta la precisión por eso el numero de neuronas de cada capa se puede ajustar a base de prueba y error . En nuestro caso hemos optado por una neurona de entrada, dos capas ocultas, la primera de 16 neuronas y la segunda de 8 por ultimo la capa de salida tendrá 1 neurona .

1.3. Algoritmos usados en los modelos

En nuestro proyecto hemos realizado dos maquinas con dos algoritmos distintos, en este caso hemos utilizado algoritmos de redes neuronales y algoritmos bayesianos .

1. Los algoritmos bayesianos son algoritmos que utilizan el Teorema de Bayes de probabilidades para problemas de Clasificación y Regresión. Usaremos ComplementNB que asume que la probabilidad previa de características es distribución polinomial
2. Los algoritmos de redes neuronales están basados en algoritmos y estructuras que intentan imitar las funciones biológicas de las redes neuronales, principalmente se pueden utilizar para problemas de Clasificación y Regresión pero tienen una gran capacidad para poder resolver una gran cantidad de problemáticas. Suelen dar muy buenos resultados para detectar patrones.
3. Las Redes Neuronales Artificiales requieren una gran capacidad de procesamiento y memoria y con la capacidad de la tecnologia del pasado estuvieron muy limitadas hasta estos últimos años en los que han estado resurgieron con bastante hincapie dando como resultado el Aprendizaje Profundo, lo utilizaremos en nuestro modelo ya que el algoritmo de regresión logística que es un tipo de analisis estadístico que trata de modelar la predicción de una variable cualitativa binaria (dos posibles valores) en función de las variables que pueden ser independientes o predictoras.La regresión logística se usa principalmente para la creación de modelos de clasificación binaria.

Para la compilación de este modelo se ha utilizado :

1. Se ha utilizado un optimizador que implementa el algoritmo Adamax. Es una variante del algoritmo Adán basada en la norma del infinito. Los parámetros predeterminados siguen los proporcionados en el documento. Adamax es a veces superior a Adam, especialmente en modelos con incrustaciones.
2. Para las perdidas hemos utilizado 'binary crossentropy' que calcula la métrica de entropía cruzada entre las etiquetas y las predichas.

1.4. Entrenamiento del modelo

El proceso de entrenamiento de una red neuronal se realiza ajustando el valor de los pesos y bías para que así las predicciones que se vayan a generar, reduzca al maximo el menor error posible. Por este motivo, el modelo de red neuronal es capaz de identificar qué predictores tienen mayor eficacia y la forma en la que están relacionados entre ellos y con las variables de respuesta.

Alcanzar una forma de optima de implementarla ha necesitado la combinación de varios métodos matemáticos, mas concretamente, el algoritmo de retropropagación((backpropagation) y la optimización por descenso de gradiente (gradient descent).

Backpropagation es el algoritmo mas importante dentro de una red neuronal ya que permite calcular la influencia que tiene cada uno de los pesos y las bias de la red en sus predicciones. Para lograrlo, se utiliza la regla de la cadena (chain rule) para calcular el gradiente(gradient), que es el vector creado por las derivadas parciales de una función.

En el caso de las redes neuronales, la derivada parcial del error respecto a una variable ya sea peso o bias mide cuanta carga ha tenido esa variable en el error cometido. Por esto, se puede reconocer cuales son los pesos de la red hay que modificar para poder mejorar los resultados. El siguiente paso que hace falta, es decidir cuánto y cómo modificarlos (optimización).

Descenso de gradiente (gradient descent) es un algoritmo de optimización iterativo de primer orden que permite reducir una función haciendo modificaciones de sus parámetros en la dirección del valor negativo de su gradiente. En lo que respecta a las redes neuronales, el descenso de gradiente permite ir modificando los pesos y bías del modelo para reducir así reducir al máximo el error.

Dependiendo del numero iteraciones es posible que pueda llegar a ser computacionalmente costoso calcular el error de modelo en todas las observaciones de entrenamiento, para esto existe una alternativa para gradient descent que se llama gradiente estocástico (stochastic gradient descent, SGD). Este método divide el conjunto de entrenamiento en lotes (pueden ser minibatch o batch) y actualizar las variables de la red con cada uno. De esta manera, no es necesaria evaluar cada observación para modificar los parámetros, se pueden ir actualizando de forma sucesiva.

Una ronda completa de iteraciones sobre todos los batch se llama época. El número de épocas con las que se entrena una red equivale al número de veces que la red ve cada ejemplo de entrenamiento [2]

1.5. Preprocesado

Cuando se entrenan modelos basados en redes neuronales va a realizar al menos dos tipos de transformaciones de los datos que vamos a recibir en nuestro conjunto de datos o 'dataset'

1. Binarización o One hot encoding de las variables de entrada que recibimos que son denominadas variables categóricas, consisten en crear por cada valor diferente una etiqueta que exista en la pecularidad que codificamos, es decir podemos crear etiquetas para dividir nuestras variables de la forma que nos sea mas lógica.

2. Estandarización - Transformamos los valores de manera que la media de los valores sea 0 y la desviación estándar sea 1. Para realizarlo se calcula la media y la desviación estándar de los valores que hemos recibido en cada uno de los puntos de datos y después restando la media y dividirlo por la desviación estándar, normalmente este tipo de operaciones las realizaremos a través de funciones ya creadas

3. Escala de funciones- El escalado de características sería la parte final del preprocesamiento de datos en 'Machine Learning'. Es un método para estandarizar las variables independientes de un conjunto de datos dentro de un rango característico, permitiendo limitar el rango de variables para que se pueda comparar en términos comunes.

1.6. Entrenamiento y predicción

1. Primero se realizara el entrenamiento, en esta parte el programa va a aprender tendencias, comportamientos o patrones que se ajusten a los datos que hemos divido para el conjunto de prueba.Para así optimizar los parámetros de una función para que la salida que nos de sea lo mas cercano a al resultado real que queremos predecir.

2. La etapa de predicción se realiza después que se ha completado el entrenamiento con el conjunto de entrenamiento y este proceso va a evaluar datos completamente nuevos que no se han tenido en cuenta para el entrenamiento utilizando la función que se ha optimizado para obtener un resultado lo mas ajustado posible al real.

3. Dependiendo del resultado podemos dividirlo en varios tipos de problemas, al obtener una categoría sería un problema de clasificación, si el resultado es un número es un problema de regresión. Es bastante interesante poder distinguir los puntos a los que nos vamos a enfrentar para escoger la mejor estratégia posible

2

Objetivos, planificación y metodología

2.1. Objetivos

- Cargar los datos desde un archivo CSV con los datos a analizar.
- Preprocesar los datos de la forma mas optima para un mejor manejo de datos.
- Obtener información de los modelos predefinidos mas usados y que mejor nos convengan para poder realizar una comparación.
- Comparar modelos predefinidos para escoger el mejor usando datos de entrenamiento
- Crear modelo predefinido con los mejores hiperparámetros para eso buscaremos los mejores posibles para el modelo.
- Compilar y evaluar el modelo con los valores óptimos para su mejor funcionamiento.
- Definir y compilar el modelo de la red neuronal con los valores óptimos.
- Entrenar el modelo ejecución del modelo con los datos del entrenamiento.
- Evaluar el modelo para encontrar los mejores valores.
- Comparar la exactitud del modelo predefinido y de la red neuronal creada.
- Obtener un csv con el contenido de los mensajes catalogados como Spam.

2.2. Planificación

- Dividiremos las opciones de salida en spam y ham(no spam)
- Leer los datos del dataset y guardarlos en una variable llamada 'df' en la que eliminaremos los datos innecesarios y pondremos las etiquetas que necesitemos para poder categorizar los datos.

- Contaremos el volumen de registros que tenemos y los compararemos con la etiqueta que nos indica si es spam o ham y mostraremos en un grafico los datos analizados que son de cada tipo .
- Desde aquí empezaremos con la parte del algoritmo ya importado y para eso dividiremos los datos en matrices en dos conjuntos que serán de prueba y entrenamiento y crearemos una función que analice los datos, mostrando el resultado que nos indicara el nombre del modelo, la exactitud y el tiempo empleado en analizar los datos.
- Después de realizar la parte anterior utilizaremos el modelo predefinido que mejores resultados haya obtenido.
- Buscaremos los mejores hiperparámetros para optimizar nuestro modelo y crearemos nuestro modelo.
- Pasaremos los datos de entrenamiento a este modelo y mediremos la exactitud.
- Crearemos nuestra modelo de red neuronal con las capas que necesitemos.
- Compilaremos el modelo ajustando las perdidas, las métricas y el optimizador.
- Preprocesaremos los datos para este nuevo modelo de red neuronal.
- Entrenamos el modelo con los datos de entrenamiento y sacamos la exactitud.
- Comparamos cual de los dos métodos es el que tiene mas exactitud .
- Escribimos los registros de los emails que sean Spam en un archivo CSV.

2.3. Metodología

Para el desarrollo del software hemos utilizado la metodología de programación incremental.

- Incremental: en este modelo de desarrollo de software se va a desarrollar el programa con elementos del modelo en cascada agregando nuevas funcionalidades, de esta manera obtendremos rápidamente los resultados comparado con el modelo en cascada, el software se puede probar antes de finalizarlo, incluso solo una parte de programa dando bastante flexibilidad a este modelo
- Vamos a utilizar un modelo supervisado a la hora de aplicar nuestro logaritmo para Machine Learning, hablamos de modelos supervisados cuando trabajamos con algoritmos que aprenden (o se "entrenan") con datos previamente definidos con etiquetas (labels). Cuando dichos modelos tienen la suficiente cantidad de datos, pueden recibir nuevos datos sin la necesidad de clasificarlos en base a los patrones que se han venido registrando durante el entrenamiento. El algoritmo aprende a predecir unas etiquetas conocidas en función a las variables de entrada(features).

  Este modelo se caracteriza por :

  1. Es necesario que una persona humana introduzca los datos, los clasifique y los etiquete por lo que en esta parte no esta totalmente automatizado
  2. En la salida los datos que genera el algoritmo son esperados ya que los datos de entrada han sido etiquetados y clasificados previamente

  El algoritmo admite dos tipos de datos :

1. Clasificación. Clasifican los objetos en diversas categorías para así predecir los resultados.En este caso nos serviría para predecir si un correo es spam o no.
2. Regresión sirven para predecir un valor de tipo numérico comprendiendo las relaciones entre las variables numéricas que ha recibido siendo útil para predecir resultados numéricos.

Este tipo de ,Machine Learning se puede usar para:

Predicción de costes de una entidad aseguradora al dar de alta un siniestro.

Detección de spam.

Prever que sistemas operativos son mas vulnerables a un tipo de malware.

3

Estado del arte

3.1. Inteligencia artificial

En estos momentos estamos viviendo el mayor crecimiento dentro de la tecnología de inteligencia artificial, ya que actualmente se puede encontrar este tipo de tecnología en asistentes de dispositivos(Alexa), en nuestro motor de búsqueda de internet, el acceso a edificios como puede ser monitorización de la temperatura corporal por cámaras, en algoritmos de búsqueda y de optimización, etc.

La inteligencia artificial nacida en 1956 sufrío un desarrollo complicado con periodos de grandes avances seguidos por momentos de poco crecimiento.Actualmente vivimos un increible auge en el crecimiento de esta tecnología debido a que los equipos son cada vez mas potentes por lo que pueden procesar una gran cantidad de datos('Big data'), mejores GPU y el procesamiento tensorial.

Otras de las aplicaciones que nos resulta bastante interesante en el ambito de la seguridad es la Biométrica, ya que permite interacciones entre humanos y maquinas sean de una manera mas natural, en nuestro caso se puede incrementar la dificultad de acceso a datos o incluso a la propia instalación, esta opción es bastante interesante ya que sería posible bloquear un dispositivo o restringir el acceso a ellos mismos si no es la persona que tiene los permisos para ello, pudiendo reducir la probabilidad de los ciberataques.

3.2. Machine Learning

Dentro de las inteligencias artificiales nos vamos a centrar en Machine Learning o también llamado aprendizaje automático, es una disciplina de las inteligencias artificiales que desarrolla técnicas para permitir que un programa a través de la experiencia en este caso a través de analizar un documento clasificado por categorías pueda aprender. Esto lo realiza utilizando una serie de algoritmos, clasificación, análisis y recopilación de la información, las plataformas que utilizan Machine Learning tienen muchas aplicaciones dentro de varios ámbitos como puede ser la predicción de resultados, personalización de la experiencia de interacción entre un usuario y

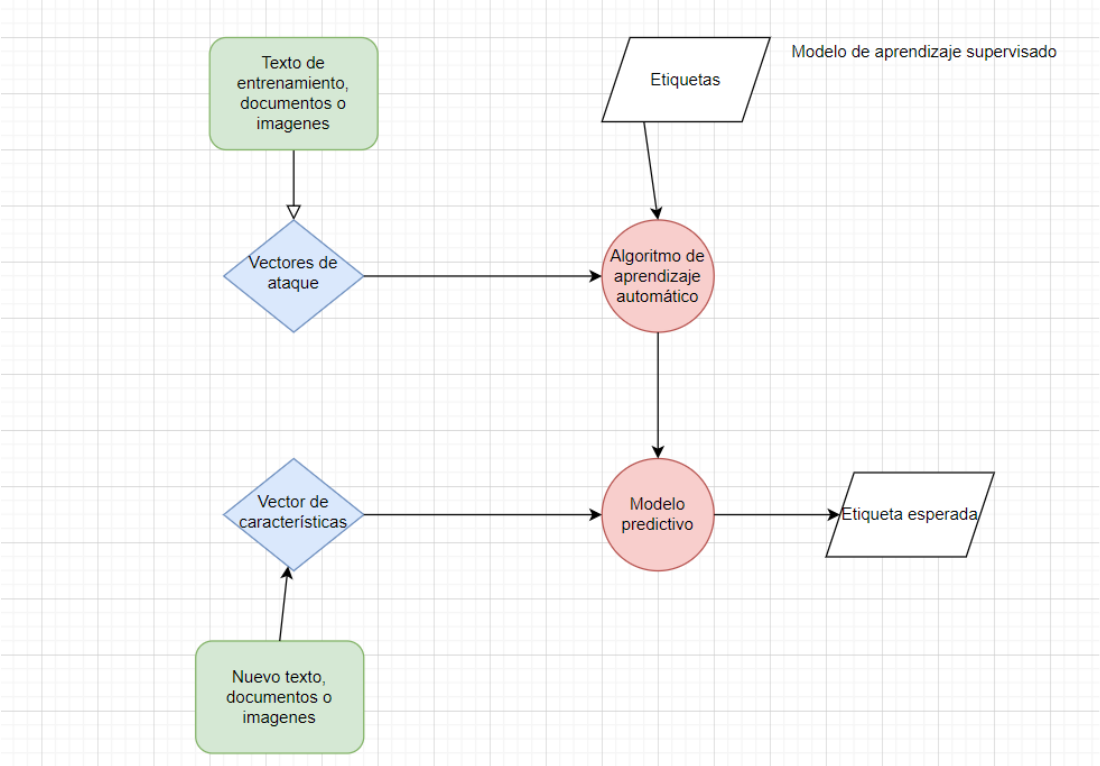
un sistema, personalización de la publicidad enviada a un usuario, etc.

El desarrollo cloud nativo es una herramienta indispensable cuando una empresa quiere estar dirigida por datos ya asi aprovechan las ventajas de la computación en la nube. El modelo tecnológico que se va implantar en sistemas de Machine Learning esta basado en una arquitectura cloud híbrida impulsada por MLOps y Kubernetes para que así la tecnología cloud puede simplificar las implementaciones de Machine Learning. Esto implica que al facilitar las implementaciones ML también esta aumentando la demanda de científicos de datos hacía áreas que necesitan sistemas de aprendizaje automático. Este avance hacia las soluciones en la nube con modelos auto ajustables de Machine Learning puede desplazar a los modelas de IA hacia la ciencia de la decisión

3.2.1. Tipos de sistemas Machine Learning

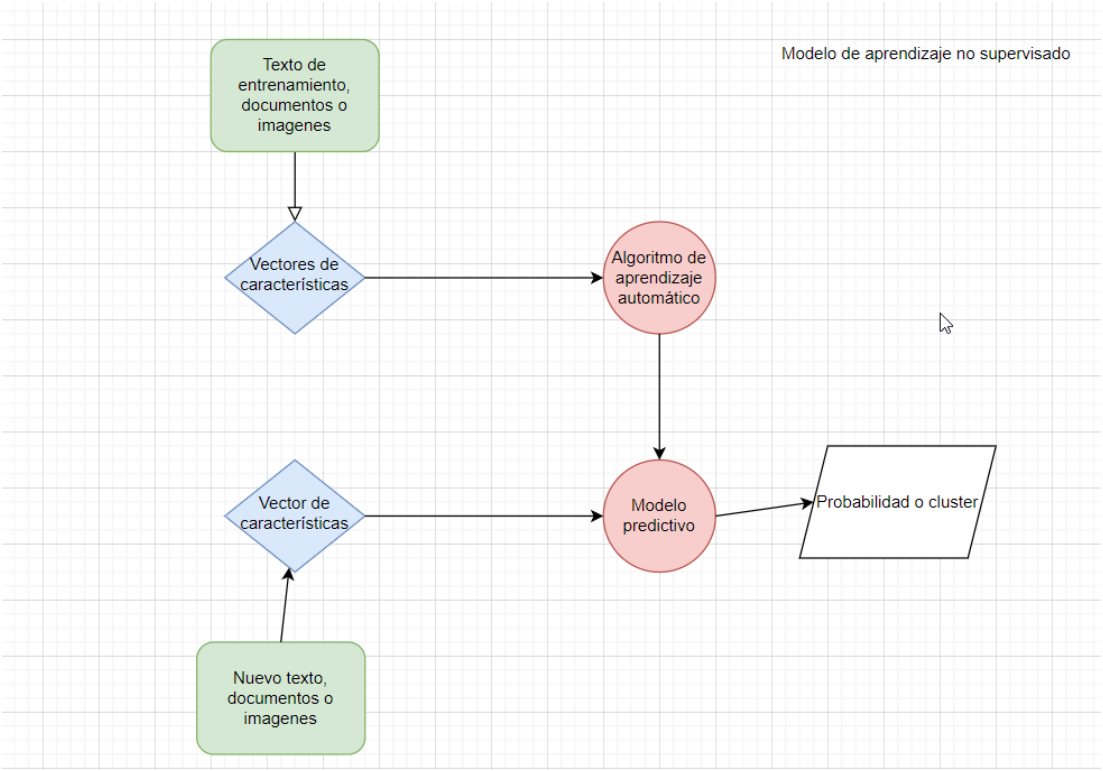
Para categorizar los diferentes tipos de machine learning podemos categorizarlos en función si requieren o no que los modelos sean entrenado por supervisión humana, si pueden o no aprender de forma incremental a medida que el sistema obtenga nuevos datos o si se trata de un aprendizaje basado en instancias o modelos.

- Aprendizaje supervisado : el objetivo es crear una función que pueda predecir el valor correspondiente a unos datos de entrada teniendo un conjunto de datos previamente clasificados y etiquetados, la salida puede resultar ser un valor número o también una etiqueta de clase.

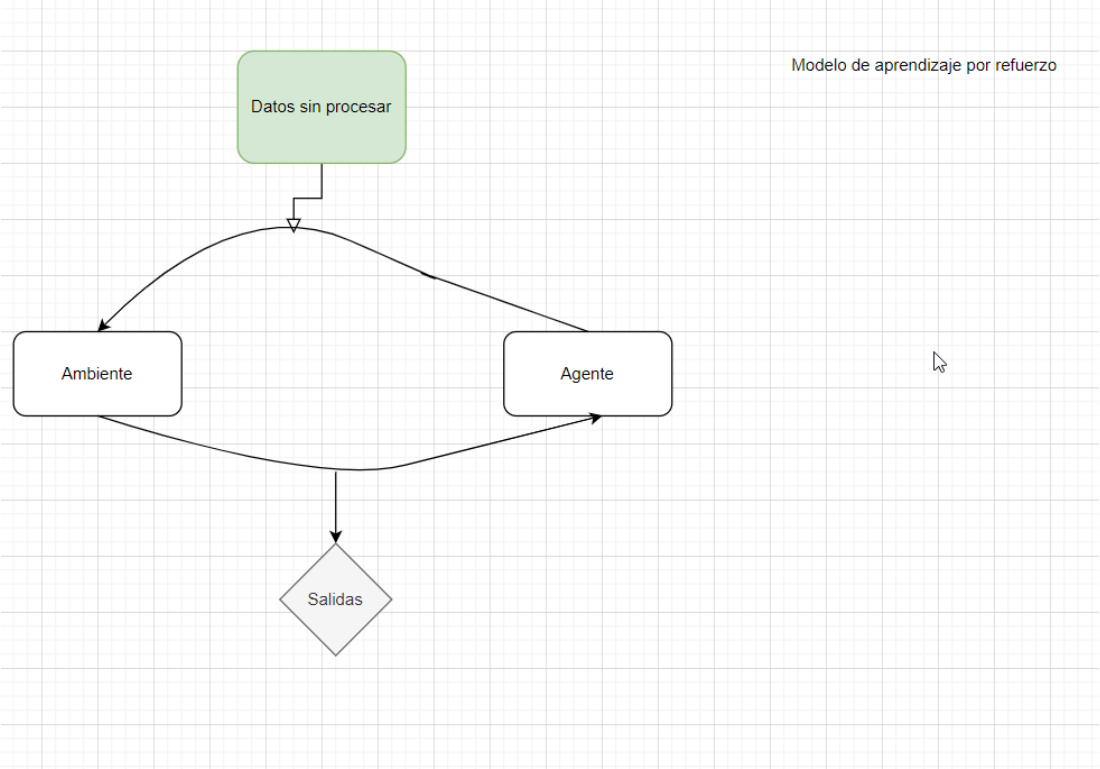


- Aprendizaje no supervisado: El sistema recibe datos de entrenamiento sin clasificar e intenta aprender, los algoritmos utilizados hacen uso de la técnica clustering, que se trata

de reunir los datos recibidos en grupos que tengan características similares entre ellos. Se utilizan en sistemas donde se quiera detectar anomalías utilizando los datos para entrenar y en cuanto se presente un dato diferente o extraño se detecta la anomalía y el sistema alerta de ello.



- Aprendizaje por refuerzo: El sistema informático va a observar el entorno y mediante una lista de acciones posibles evaluara cual es la mejor opción y tomara las decisiones, el sistema aprenderá por si solo cual es la mejor estrategia a seguir.



La inteligencia artificial es la clave para el internet de las cosas(IoT) ya que gracias a las IA nuestros dispositivos pueden comunicarse entre si para dar servicios mas personalizados a los clientes, también la inteligencia artificial se beneficiara de la computación cuántica ya que al poder obtener mas velocidad de procesamiento los ordenadores pueden trabajar mas rápido y realizar mas operaciones en menos tiempo.

4

Desarrollo/Implementación/Arquitectura

En este capitulo vamos a profundizar tanto en como hemos creado la aplicación, con que tecnologías y la arquitectura que hemos seguido.

4.1. Desarrollo

En esta sección profundizaremos sobre el desarrollo de la aplicación llevada a cabo, tanto en el lenguaje utilizado, el tipo de objeto que vamos a crear, librerías utilizadas, programas utilizados para la implementación y arquitectura.

4.1.1. Lenguaje utilizado

Vamos a desarrollarlo con python por su facilidad para la creación de machine learning, el lenguaje se hace hincapié en la transparencia y la comprensibilidad de su código.

Es un lenguaje de programación que usa varios modelos, ya que en parte soporta la orientación a objetos, también incorpora programación imperativa y en menor parte programación funcional. Es un lenguaje multiplataforma ya que puede desarrollarse en distintos entornos.

Es un lenguaje dinámico ya que las operaciones que se realizan en tiempo de compilación también se pueden realizar en tiempo de ejecución.

Es un lenguaje de programación interpretado ya que el código se ejecuta linea a linea, instrucción por instrucción.

Este lenguaje de programación tiene una gran variedad de usos y aplicaciones, en nuestro caso la usaremos por la aplicacion de algoritmos y librerías compatibles para machine learning.

4.1.2. Tipo de objetos que vamos a desarrollar

Utilizaremos el clasificador Complement Naive Bayes ya que nos sirve para corregir las suposiciones severas realizadas por el clasificador estándar Multinomial Naive Bayes. Es especialmente adecuado para conjuntos de datos desequilibrados. Hemos optado por este algoritmo ya que nos da una buena exactitud y poco tiempo de ejecución.

Utilizaremos una red neuronal con un algoritmo de regresión logística.

4.1.3. Librerías mas utilizadas

- Pandas: Esta librería nos permitirá leer y escribir ficheros en varios formatos como pueden ser CSV, Excel y bases de datos SQL, nos permitirá ordenar por índices y columnas la información extraídas del archivo.
- Numpy: Acrónimo de Numerical Python es una librería que se especializa en el análisis de datos de gran volumen ya que los archivos que se suelen recibir son de gran tamaño, permite crear vectores y matrices multidimensionales de gran tamaño, cuenta con una serie de funciones matemáticas para operar con ellas.
- Seaborn: Es una librería que nos permite crear fácilmente sofisticados gráficos. Seaborn se basa en matplotlib y proporciona una interfaz de alto nivel, en Anaconda esta librería esta instalada por defecto.
- Matplotlib: Herramienta que nos permite generar gráficos contenidos en listas de dos dimensiones a partir de los datos obtenidos, nos permite crear y personalizar los tipos de gráficos y su extensión Numpy
- Tensorflow: Es la librería por excelencia para la generación de aprendizaje profundo, es de código abierto y fue creado por Google y publicado bajo Apache 2.0, la API principalmente es usada en Python, esta librería fue diseñada para investigación y desarrollo. Tiene gran versatilidad ya que se puede usar en CPU, GPU, Dispositivos móviles y en cientos de maquinas.
- Scikit-learn: Librería de código abierto que proporciona una gran variedad de algoritmos de aprendizaje supervisado y también no supervisados, se programa en Python y como esta basado en NumPy, SciPy y matplotlib se puede reaprovechar el código que use estas librerías.
- Nltk: Contiene un conjunto de librerías y programas en Python que nos serán útiles para el procesamiento de lenguajes estadísticos.

4.2. Implementación

4.2.1. Con que aplicaciones se ha implementado

Hemos ejecutado el programa con Notebook Jupyterlab que se ejecuta a través del programa Anaconda. ¿Por qué elegir Anaconda y Jupyter Notebook?

1. Anaconda: Lo que hace que Python sea más dificil es el problema de la administración de paquetes y las diferentes versiones de Python. Anaconda, es una distribución de código abierto del lenguaje Python que puede resolver los problemas anteriores hasta cierto punto. Con Anaconda, podemos realizar más fácilmente la gestión de paquetes y la gestión del entorno para Python para desarrollar nuestro ML.
2. Jupyter Notebook: Es un cuaderno interactivo en el que puedes realizar cualquier actividad. En este cuaderno, puede escribir código, escribir documentos, tener diagramas, esquemas etc. Con otros programas, escribimos código en un editor y luego escribimos documentos en Word para presentar el proyecto. Con Jupyter Notebook, podemos implementar la programación y la escritura, y desarrollar programas de acuerdo con nuestra propia lógica.

La ejecución ha sido en local para así reducir el tiempo de ejecución del programa.

4.2.2. Implementación del proyecto en ciberseguridad

Una vez que tengamos asegurados los datos en este caso se trata de un dataset descargado de Kaggle, lo siguiente que tenemos que replantearnos es qué conocimientos o aplicaciones prácticas queremos conseguir con la implemetación del machine learning. Este aspecto es el mas importante, pues dependiendo de la necesidad específica de cada negocio se optará por alguna de las diferentes soluciones que existen dentro de los ML.

En este caso hemos optado por implementar dos soluciones :

1. Solución personalizada. Este opción suele ser la más precisa, pero requiere una gran cantidad de datos y definir los algoritmos de nuestro modelo de machine learning, en este caso lo hemos creado con nuestra red neuronal.
2. Modelos pre entrenados. son un modo de implementar machine learning de una forma rápida y bastante sencilla dentro de ML, pero puede entrañar cierta dificultad a aquellas empresas que no tienen bastante experiencia en el uso de estas herramientas o no cuentan con empleados cualificados para operar dichas herramientas.

Sin duda con estas dos opciones se trata de una buena solución para introducir a un negocio en la Inteligencia Artificial rápidamente, pero para poder obtener buenos resultados y hacerlo pronto necesitamos datos de prueba para poder entrena y afinar los logaritmos. Dentro de la ciberseguridad y nuestro proyecto se ha implementado el machine learning para recibir un archivo CSV, el siguiente paso es identificar el tipo de data que será útil para solucionar el problema propuesto, etiquetar los datos y analizarlos para poder realizar las predicciones.

4.3. Arquitectura

Dentro de las redes neuronales nos referimos a arquitectura al conjunto de:

- Número de capas neuronales
- Número de neuronas en cada capa
- Conexión entre neuronas
- Conexión entre capas
- Tipos de neuronas
- Forma de entrenar red neuronal

El tipo que utilizaremos sera Feed fordward networks, este tipo de red es una red prealimentada, este tipo de red fue una de las primeras formas de red neuronal, a su vez es una extensión de perceptrón que nos permite realizar cálculos para detectar patrones.

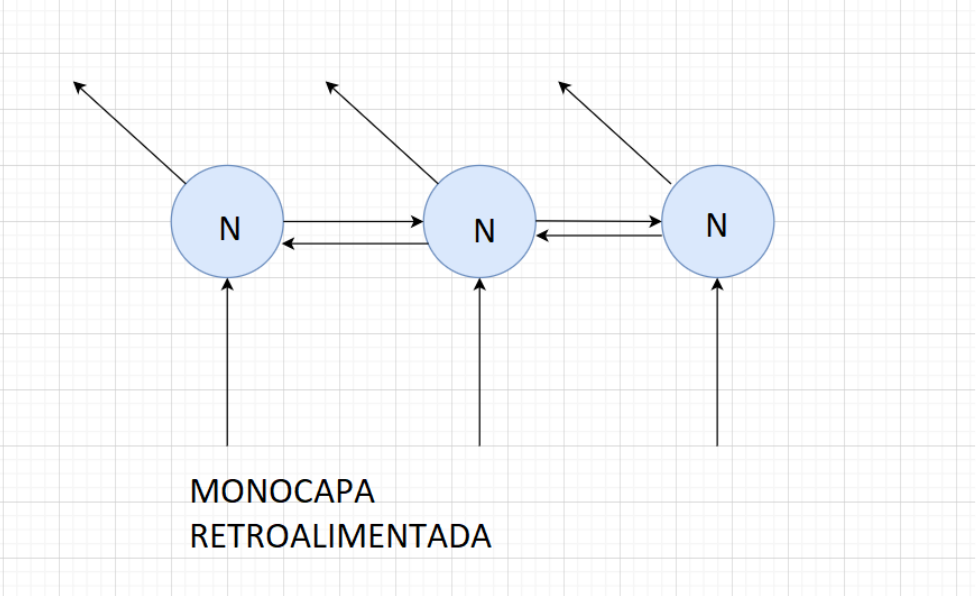
Se compone de varias capas neuronales interconectadas de manera "fully connected"(Una neurona esta conectada con todas las neuronas de las siguiente capas sucesivamente hasta la salida, existe por lo menos una capa oculta, el flujo de datos fluye de izquierda a derecha ("feed forward") y la red se entrena mediante retropropagación(back-propagation).

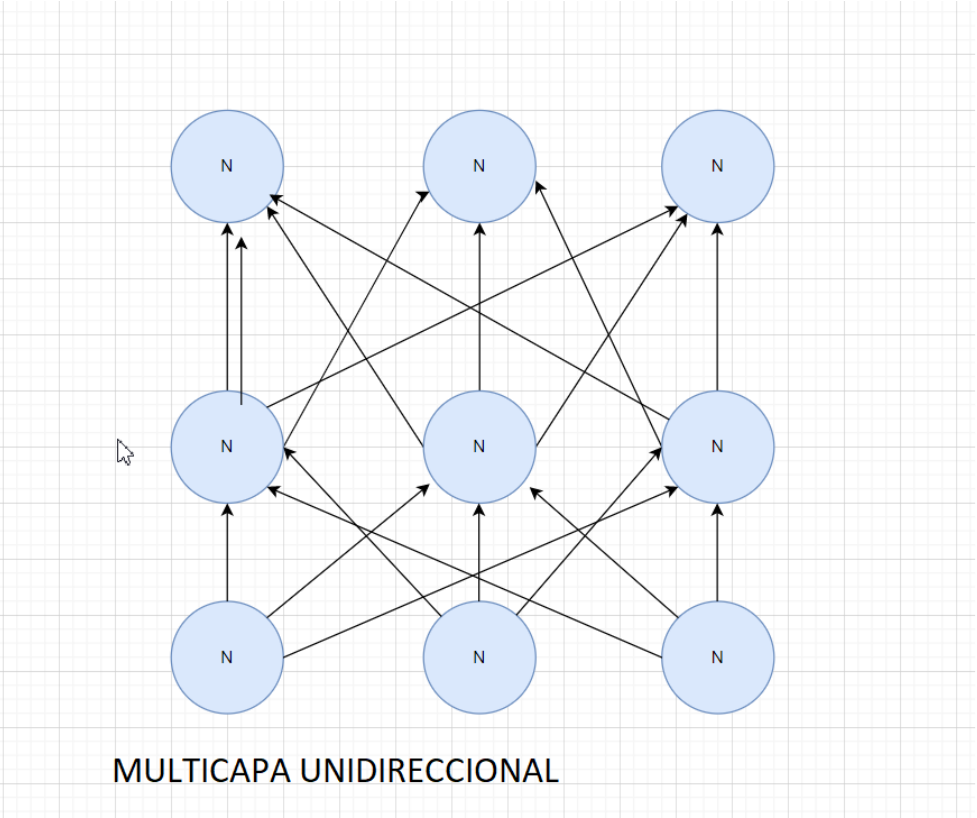
Nuestra arquitectura constara de una capa de entrada con una neurona y dos capas ocultas, la primera con un total de 16 neuronas y la segunda con un total de 8 neuronas y dispondremos de una capa de salida con 1 neurona por lo que tendría un total de 4 capas, en resumen tendremos una arquitectura unidireccional multicapa .

Retropropagación(Backpropagation) Este algoritmo se basa en que el error de las capas anteriores esta directamente relacionado con el error de las capas posteriores, por ello gracias al análisis podemos optimizar la propagación del error hacia atrás en la red

Con estos datos podemos asumir también que si tenemos el error controlado para N neurona de la m-ésima capa, entonces la configuración de parámetros para cada una de las neuronas de las m-1 capas anteriores que tienen influencia en los datos de entrada de nuestra neurona N, es relativamente poco influyente para nuestro error final.

Aquí tenemos un ejemplo de los tipos de arquitectura:


N
N
N
MONOCAPA
RETROALIMENTADA


N
N
N
N
N
N
N
N
N
MULTICAPA UNIDIRECCIONAL

5

Validación y evaluación

5.1. Pruebas prepocesado de texto

Al recibir los datos es necesario preprocesarlos para poder organizarlos y comprobar que datos nos interesan y a su ver el estado de esos datos, se puede utilizar para limpiar un texto, crear una matriz o convertir esos datos en datos que puedan ser procesados por el machine learning.

Primero validaremos los datos que hemos recibido y hemos reorganizado cambiando el nombre de las etiquetas por unas que nos resulte mas practicas y los mostramos por pantalla para ver como queda el formato de ellos y poder visualizar el numero de registros recibimos.

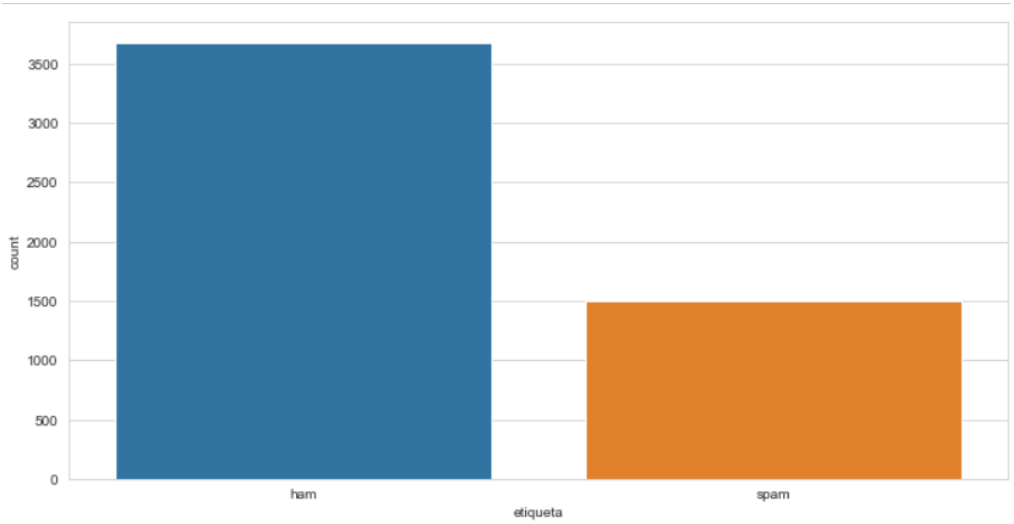
El resultado de la prueba es :

Out[3]:

	etiqueta	texto	clase
0	ham	Subject: enron methanol ; meter # : 988291\r\n...	0
1	ham	Subject: hpl nom for january 9 , 2001\r\n(see...	0
2	ham	Subject: neon retreat\r\nho ho ho , we ' re ar...	0
3	spam	Subject: photoshop , windows , office . cheap ...	1
4	ham	Subject: re : indian springs\r\nthis deal is t...	0
5	ham	Subject: ehronline web address change\r\nthis ...	0
6	ham	Subject: spring savings certificate - take 30 ...	0
7	spam	Subject: looking for medication ? we ' re the ...	1
8	ham	Subject: noms / actual flow for 2 / 26\r\nwe a...	0
9	ham	Subject: nominations for oct . 21 - 23 , 2000\...	0

Después de esta prueba utilizaremos la variable que nos muestra el resultado real y la mostraremos en forma de grafico para poder tener una mejor idea de los resultado que deberíamos dar en el futuro.

El resultado de la prueba es:



Ahora mostraremos la cabecera de nuestro texto después de limpiarlo quitándole los caracteres extraños y tokenizando el texto

Resultado:

	etiqueta	texto	clase
0	ham	subject enron methanol meter follow note gave ...	0
1	ham	subject hpl nom january see attached file hpln...	0
2	ham	subject neon retreat ho ho ho around wonderful...	0
3	spam	subject photoshop windows office cheap main tr...	1
4	ham	subject indian springs deal book teco pvr reve...	0

5.2. Pruebas selección modelo

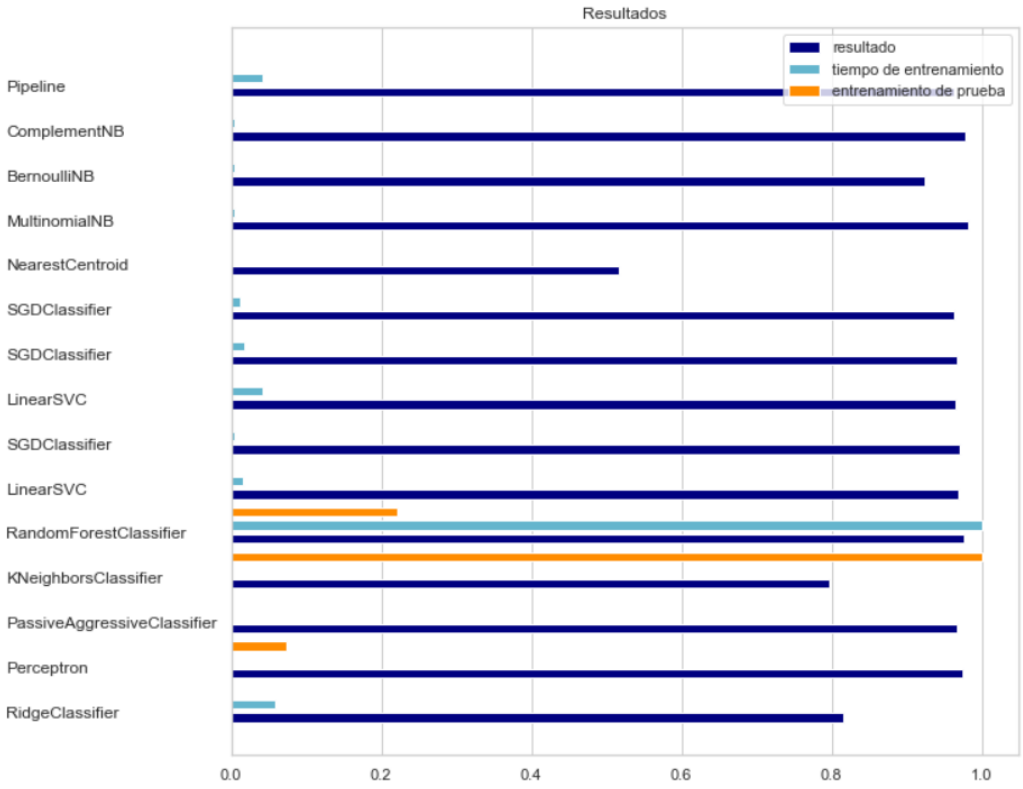
Para seleccionar el modelo previamente vamos a convertir una colección de documentos de texto en una matriz de recuentos de tokens, después vamos Dividir arreglos o matrices en subconjuntos de prueba y entrenamiento , podemos ver el numero de observaciones y el numero de tokens que consta

Resultado:

```
NÚMERO DE OBSERVACIONES: 5171
TOKENS: 45595
```

Después entrenaremos el texto con los modelos que hemos seleccionado para poder obtener el tiempo que ha tardado en entrenar con los datos facilitados y la exactitud que ha logrado cada modelo en una gráfica.

Resultado:



Necesitaremos obtener los mejores hiperparámetros para el modelo seleccionado, para eso utilizaremos RandomizedSearchCV para poder buscar los mejores parámetros y mostrarlos en un formato de lista indicando los parámetros y la exactitud que hemos conseguido.

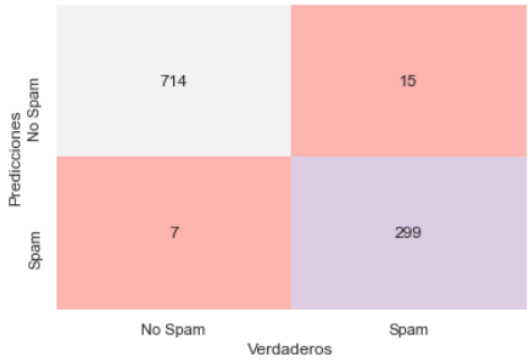
Resultado:

Out[14]:

	params	mean_test_score
1	{'alpha': 0.2, 'fit_prior': False}	0.979207
3	{'alpha': 1, 'fit_prior': False}	0.977999
5	{'alpha': 2, 'fit_prior': False}	0.977997
2	{'alpha': 1, 'fit_prior': True}	0.977273
0	{'alpha': 0.2, 'fit_prior': True}	0.976789
4	{'alpha': 2, 'fit_prior': True}	0.973887
7	{'alpha': 5, 'fit_prior': False}	0.964458
6	{'alpha': 5, 'fit_prior': True}	0.956963
9	{'alpha': 10, 'fit_prior': False}	0.945599
8	{'alpha': 10, 'fit_prior': True}	0.937619

Una vez seleccionado el modelo predefinido es hora de entrenarlo, una vez entrenado hemos conseguido una exactitud del 97,89 que es bastante buena. Mostraremos una matriz de confusión para poder ver de manera visual el desempeño que realiza el algoritmo.

Resultado:



5.3. Pruebas modelo red neuronales

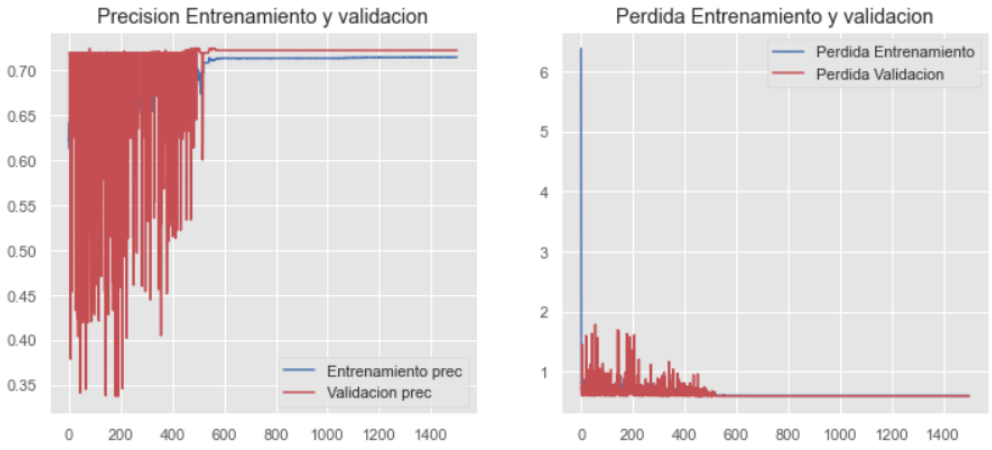
Para poder seleccionar las capas y las neuronas en nuestra red neuronal hemos entrenado la red con distintos parámetros a base de prueba y error para conseguir los parámetros óptimos, cabe mencionar que a mayor numero de neuronas se puede llegar a mejorar la exactitud pero también aumentan los tiempos que tarda en entrenar la red, a continuación vamos a mostrar el sumario de nuestra red

Resultado de la prueba:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 16)                32
_________________________________________________________________
dense_1 (Dense)              (None, 8)                 136
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 9
=================================================================
Total params: 177
Trainable params: 177
Non-trainable params: 0
_________________________________________________________________
```

Una vez probado el modelo es hora de validarlo para poder obtener la exactitud y las perdidas al evaluar los datos, obtenemos una exactitud de 71,35 .Una vez obtenidos los datos vamos a visualizarlo en un gráfico que nos mostrara la precisión en el entrenamiento y validación y la perdida en entrenamiento y validación

Resultado de la prueba:



Con esto podemos observar que con el modelo predefinido tenemos una mayor exactitud

6

Conclusiones y trabajo futuro

Las inteligencias artificiales se están desarrollando enormemente en todos los ámbitos ya que debido a su funcionalidad tiene una infinidad de usos.

Las machine learning son el futuro tanto en ciberseguridad como en todas sus aplicaciones ya que la capacidad de automatizar procesos y que puedan aprender mientras los procesan aumenta la efectividad y mejora el rendimiento.

Podríamos usarlas tanto para medir la predicción de que una maquina sea infectada por un malware pudiendo proteger esa maquina mejor, en el ámbito de seguridad de la empresa se podrían evaluar las contraseñas que se usan en su empresa

Bibliografía

[1] "Redes neurales - curso- fundamentos - luisamayateacher," https://sites.google.com/site/luisamayateacher/redes-neurales---curso, (Accessed on 11/10/2021).

[2] "Redes neuronales con python," https://www.cienciadedatos.net/documentos/py35-redes-neuronales-python.html, (Accessed on 11/24/2021).

[3] "Keras: the python deep learning api," https://keras.io/, (Accessed on 03/21/2022).

[4] "Kaggle: Your machine learning and data science community," https://www.kaggle.com/, (Accessed on 03/21/2022).

[5] "Aprende machine learning," https://www.aprendemachinelearning.com/, (Accessed on 03/21/2022).

[6] "pandas - python data analysis library," https://pandas.pydata.org/, (Accessed on 03/21/2022).

[7] "Numpy," https://numpy.org/, (Accessed on 03/21/2022).

[8] "Inicio -aprende ia," https://aprendeia.com/, (Accessed on 03/21/2022).

[3] [4] [5] [6] [7] [8]

Apéndices