CARLOS LUZZATTI VÁZQUEZ

TRABAJO FIN DE MÁSTER

IA/Machine Learning aplicada a ciberseguridad

- **Indice.**

1. Resumen.

1.1 ¿Cómo la IA y las aplicaciones de machine learning pueden ser útiles ?

Las aplicaciones de la inteligencia artificial en la actualidad han abierto la puerta a la integración de varias soluciones en distintas áreas de la sociedad, por lo cual la ciberseguridad no es una excepción ya que las amenazas de ciberseguridad informáticas ocasiones millones en daños.

Pues podemos llevar a cabo tareas de forma "inteligente", sin la necesidad de que sean programadas para ello, de la siguiente manera, los algoritmos de machine learning o aprendizaje automático un subconjunto o aplicación de la IA forman un modelo matemático en base a datos de entrenamiento, aprovechando este modelo para realizar predicciones cuando se entregan nuevos datos.

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

1.2 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.

2. Introducción y contexto.

2.1.**Como funciona el machine learning en ciberseguridad**

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 gran cantidad de datos los patrones de comportamiento en los datos y detectar anomalías para así detectar cualquier tipo de amenaza automáticamente.

De esta forma, las soluciones de seguridad informática pueden actuar de manera inminente, bloqueando cualquier tipo de amenaza que quiera penetrar en una infraestructura IT.

Machine learning es una disciplina científica del ámbito de la Inteligencia Artificial que crea sistemas que aprenden automáticamente. Aprender en este contexto quiere decir identificar patrones complejos en millones de datos. La máquina que realmente aprende es un algoritmo que revisa los datos y es capaz de predecir comportamientos futuros. Automáticamente, también en este contexto, implica que estos sistemas se mejoran de forma autónoma con el tiempo, sin intervención humana.

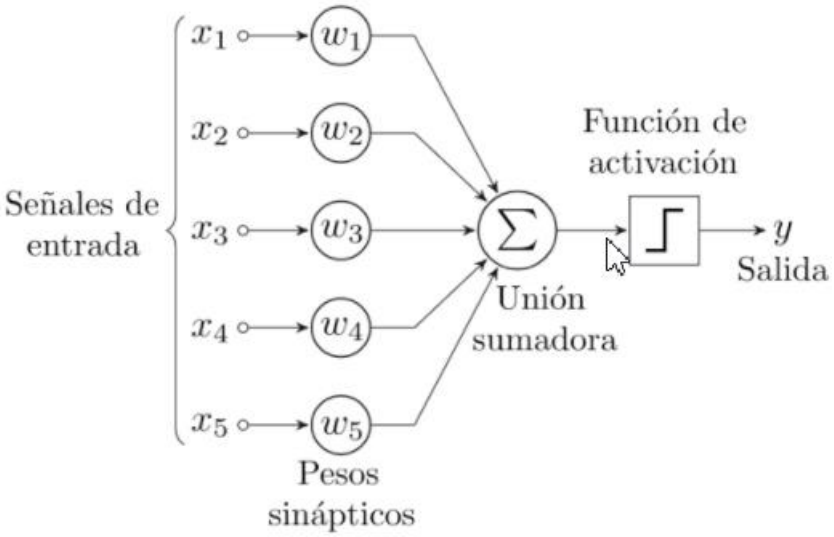
Machine learning es un algoritmo que puede aprender entrenando con datos que ha recibido, estos algoritmos se pueden definir con redes neuronales definiendo las neuronas y las capas o bien podemos importar un modelo desde keras para facilitarlo, también es posible crear un red neuronal he importarla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

2.2**¿Que es una red neuronal?**

Las neuronas artificiales imitan 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 permite adquirir conocimientos mediante la captación de información para relacionarla entre sí. De esta manera, aprender tal y como lo haría nuestro cerebro.

imagen con la explicación de cómo funciona una red neuronal artificial.



Con este proceso se crean patrones de comportamiento a partir del análisis de millones de datos para que las maquinas aprendan solas a base de los anteriores resultados, por lo que a mayor datos disponibles nuestro algoritmo sera mas completo.

2.3.**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 un conjunto de neuronas cuyas entradas provienen de una capa anterior o de los datos de entrada es decir la primera capa y cuyas salidas son la entrada de una capa posterior,la neurona es la unidad funcional de los modelos de redes. Dentro de cada neurona, ocurren simplemente dos operaciones: la suma ponderada de sus entradas y la aplicación de una función de activación.

La primera capa se conoce como la capa de entrada que deberá tener tantas neuronas como los tipos de datos que se van a utilizar, 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 .

Capas ocultas recibe los valores de la capa de entrada, 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, 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 .

2.4.**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 .

Algoritmo *Bayesiano* son algoritmos que utilizan explícitamente el Teorema de Bayes de probabilidad para problemas de Clasificación y Regresión. Usaremos *ComplementNB* que asume que la probabilidad previa de características es distribución polinomial

Algoritmo de redes neuronales son algoritmos y estructuras inspirados en las funciones biológicas de las redes neuronales, se suelen utilizar para problemas de Clasificación y Regresión pero realmente tienen un gran potencial para resolver multitud de problemáticas. Son muy buenas para detectar patrones.

Las Redes Neuronales Artificiales requieren mucha capacidad de procesamiento y memoria y estuvieron muy limitadas por la tecnología del pasado hasta estos últimos años en los que resurgieron con mucha fuerza dando lugar al Aprendizaje Profundo, utilizaremos en nuestro modelo el algoritmo de regresión logística que es un método estadístico que trata de modelar la probabilidad de una variable cualitativa binaria (dos posibles valores) en función de una o más variables independientes. La principal aplicación de la regresión logística es la creación de modelos de clasificación binaria.

Para la compilación de este modelo se ha utilizado :

- Se ha utilizado un optimizador que implementa el algoritmo Adamax.
  Es una variante de 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.

- Para las perdidas hemos utilizado '*binary crossentropy*' que calcula la métrica de entropía cruzada entre las etiquetas y las predichas.

2.5.Entrenamiento del modelo

El proceso de entrenamiento de una red neuronal consiste en ajustar el valor de los pesos y bías de tal forma que, las predicciones que se generen, tengan el menor error posible. Gracias a esto, el modelo es capaz de identificar qué predictores tienen mayor influencia y de qué forma están relacionados entre ellos y con la variable respuesta.

Si bien la idea parece sencilla, alcanzar una forma de implementarla ha requerido la combinación de múltiples métodos matemáticos, en concreto, el algoritmo de retro propagación (*backpropagation*) y la optimización por descenso de gradiente (*gradient descent*).

Backpropagation es el algoritmo que permite cuantificar la influencia que tiene cada peso y bias de la red en sus predicciones. Para conseguirlo, hace uso de la regla de la cadena (*chain rule*) para calcular el gradiente, que no es más que es el vector formado por las derivadas parciales de una función.

En el caso de las redes, la derivada parcial del error respecto a un parámetro (peso o bias) mide cuanta "responsabilidad" ha tenido ese parámetro en el error cometido. Gracias a esto, se puede identificar qué pesos de la red hay que modificar para mejorarla. El siguiente paso necesario, es determinar cuánto y cómo modificarlos (optimización).

Descenso de gradiente (*gradient descent*) es un algoritmo de optimización que permite minimizar una función haciendo actualizaciones de sus parámetros en la dirección del valor negativo de su gradiente. Aplicado a las redes neuronales, el descenso de gradiente permite ir actualizando los pesos y bías del modelo para reducir su error.

Dado que, calcular el error del modelo para todas las observaciones de entrenamiento, en cada iteración, puede ser computacionalmente muy costoso, existe una alternativa al método de descenso de gradiente llamada gradiente estocástico (*stochastic gradient descent, SGD)*. Este método consiste en dividir el conjunto de entrenamiento en lotes (*minibatch o batch)* y actualizar los parámetros de la red con cada uno. De esta forma, en lugar de esperar a evaluar todas las observaciones para actualizar los parámetros, se pueden ir actualizando de forma progresiva. 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.

2.6.**Preprocesado**

A la hora de entrenar modelos basados en redes neuronales, es necesario realizar, al menos, dos tipos de transformaciones de los datos.

1. Binarización (*One hot encoding*) de las variables categóricas-Consiste en crear nuevas variables dummy con cada uno de los niveles de las variables cualitativas. Por ejemplo, una variable llamada color que contenga los niveles rojo, verde y azul, se convertirá en tres nuevas variables (color_rojo, color_verde, color_azul), todas con el valor 0 excepto la que coincide con la observación, que toma el valor 1.

2. Estandarización y escalado de variables numéricas-Cuando los predictores son numéricos, la escala en la que se miden, así como la magnitud de su varianza pueden influir en gran medida en el modelo. Si no se igualan de alguna forma los predictores, aquellos que se midan en una escala mayor o que tengan más varianza dominarán el modelo aunque no sean los que más relación tienen con la variable respuesta. Existen principalmente 2 estrategias para evitarlo:

    - Centrado: consiste en restarle a cada valor la media del predictor al que pertenece. Si los datos están almacenados en un dataframe, el centrado se consigue restándole a cada valor la media de la columna en la que se encuentra. Como resultado de esta transformación, todos los predictores pasan a tener una media de cero, es decir, los valores se centran en torno al origen.

    - Normalización (estandarización): consiste en transformar los datos de forma que todos los predictores estén aproximadamente en la misma escala.

2.7.**Entrenamiento y predicción**

1. Primero se realizara el entrenamiento, en esta fase el sistema trata de aprender tendencias, comportamientos y/o patrones que se ajusten a los datos que forman el conjunto de entrenamiento. Desde un punto de vista matemático, el entrenamiento de un sistema inteligente se corresponde con un proceso de optimización de los parámetros de una función para que su salida sea lo más parecida posible al resultado que queremos obtener.

2. En la etapa de predicción se realiza una vez se ha completado el entrenamiento y consiste en evaluar datos nuevos que no se han tenido en cuenta para el entrenamiento utilizando la función optimizada para obtener un resultado.

3. Si el resultado que se obtiene es una categoría, estamos ante un problema de clasificación. Si, por el contrario, el valor que devuelve la función es un número, estamos ante un problema de regresión. Llegados a este punto, resulta muy interesante ser capaces de distinguir en una empresa aquellos puntos dentro de un proyecto en los que machine learning pueden marcar la diferencia a la hora de obtener resultados.

3. Objetivos, planificación y metodología.

3.1. 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.

3.2.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.

3.3.Metodología

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

- Incremental: en esta metodología de desarrollo de software se va construyendo el producto final de manera progresiva. En cada etapa incremental se agrega una nueva funcionalidad, lo que permite ver resultados de una forma más rápida en comparación con el modelo en cascada. El software se puede empezar a utilizar incluso antes de que se complete totalmente y, en general, es mucho más flexible que las demás metodologías.
- 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 introducirse nuevos datos sin necesidad de etiquetas, en base a patrones distintos que se han venido registrando durante el entrenamiento. El

algoritmo aprende a predecir unas etiquetas conocidas en función de las variables de entrada (*features*).

Este modelo se caracteriza por :

- Necesita la intervención humana para etiquetar, clasificar e introducir los datos en el algoritmo.
- El algoritmo genera datos de salida esperados, ya que en la entrada han sido etiquetados y clasificados por alguien.

Existen dos tipos de datos que pueden ser introducidos en el algoritmo:

- Clasificación: clasifican un objeto dentro de diversas clases. Por ejemplo, para determinar si un paciente está enfermo o si un correo electrónico es *spam*.
- Regresión: predicen un valor numérico. Sería el caso de los precios de una casa al escoger diferentes opciones o la demanda de ocupación de un hotel.

Algunas aplicaciones prácticas de este tipo de machine learning:
La predicción de coste de un siniestro en el caso de las compañías de seguros.
La detección de fraude bancario por parte de entidades financieras.
La previsión de avería en la maquinaria de una compañía.

4. Estado del arte.

4.1.Inteligencia Artificial

Vivimos en la actualidad uno de los momentos de mayor crecimiento de la Inteligencia Artificial, en estos momentos la Inteligencia Artificial es más común de lo que creemos. Es posible hallarla en los asistentes de nuestros dispositivos móviles, en las búsquedas en Internet, en la optimización de sistemas, en los semáforos de las ciudades, los accesos a edificios, los algoritmos de búsqueda, etc.

La Inteligencia Artificial desde su comienzo, hace ya sesenta años, no ha tenido un desarrollo fácil, con muchos periodos "*hype*" seguidos por momentos muy bajos. Sin embargo, en la actualidad estamos viviendo uno de los momentos de mayor crecimiento. Este auge se produce sobre todo gracias a la disponibilidad de gran cantidad de datos (*big data*), el procesamiento gráfico (*GPU*) y el procesamiento tensorial.

Otra aplicación de la AI a la que debemos observar con detalle por ser bastante interesante es la Biométrica. La biométrica permite que las interacciones entre humanos y máquinas sean más naturales. Se ocupa de la identificación, medición y análisis de aspectos físicos del cuerpo y su forma. Es una de las tecnologías más tangibles dentro de la inteligencia artificial. Asimismo, tiene potencial como herramienta para diagnóstico médico y como un nivel de seguridad para el acceso a información en dispositivos móviles. Dentro de la ciberseguridad 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.

4.2.Machine learning

Dentro de las aplicaciones de la IA vamos a profundizar en el aprendizaje automático o aprendizaje de máquinas (machine learning); que es una disciplina de la inteligencia artificial que se dedica a desarrollar técnicas que permitan a un programa aprender sobre la experiencia. Lo hace a través de algoritmos, mejora de herramientas, recopilación y análisis de información y más, las plataformas de machine learning tienen muchas aplicaciones otras aplicaciones como, por ejemplo; el refinamiento y personalización de la interacción entre máquina y usuario y hacer predicciones de comportamiento y clasificación en marketing como la optimización la entrega de publicidad pagada y contenido a las audiencias seleccionadas.

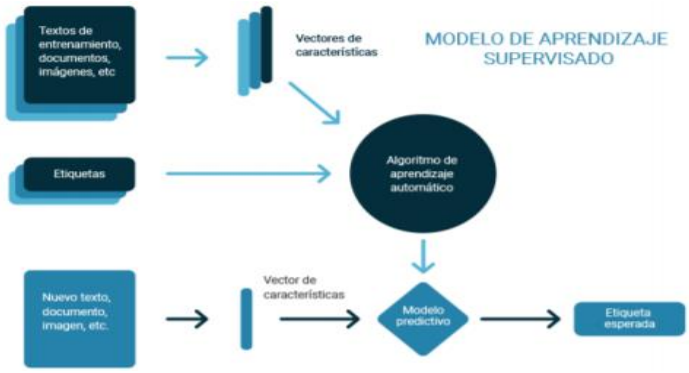
El desarrollo cloud nativo se convierte en herramienta imprescindible cuando una empresa quiere estar dirigida por datos. De esta forma, el modelo tecnológico a implantar se basa en una arquitectura cloud híbrida impulsada por *MLOps* y *Kubernetes*, pudiendo escalar de manera rentable. Así, las tecnologías en la nube facilitarán las implementaciones sencillas de *ML*. Esto implicará una mayor demanda de científicos de datos hacia áreas más complejas que requieren los sistemas de aprendizaje automático.

Este avance a medio plazo hacia soluciones cloud con modelos de machine learning que se auto ajustan, puede hacer que la IA se convierta en una "*commodity*", desplazándose hacia la ciencia de la decisión.

4.3.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.

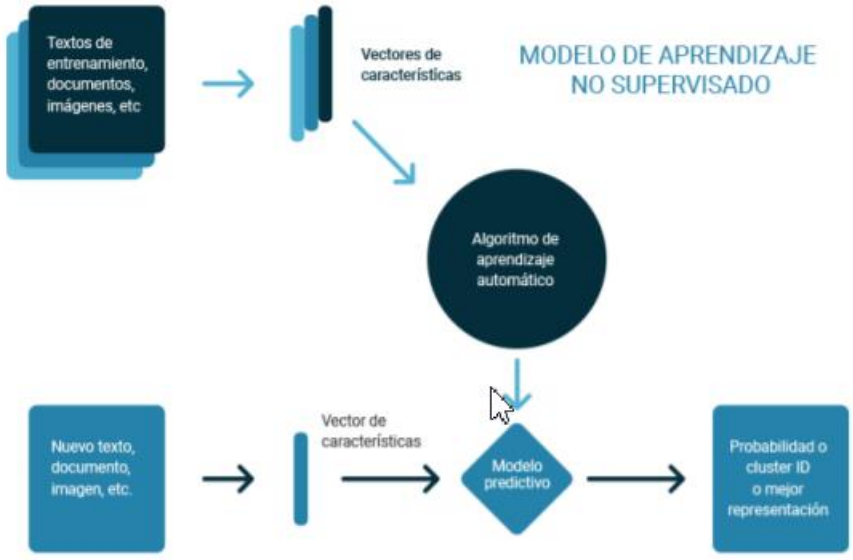
- Aprendizaje supervisado : el objetivo es crear una función que pueda predecir el valor correspondiente a unos datos de entrada teniendo un conjunto de datos previamente clasificados y etiquetados, la salida de la función puede ser un valor numérico o una etiqueta de clase.



Modelo de aprendizaje supervisado

- 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 basa en reunir los datos en grupos que posean características similares entre si. 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.

Modelo de aprendizaje no supervisado



- Aprendizaje por refuerzo: El sistema puede observar el entorno y mediante una selección de acciones posibles tomara las decisiones, el sistema debe aprender cual es la estrategia a seguir

Modelo de aprendizaje por refuerzo

En cierre, la Inteligencia Artificial es la llave del mundo hiperconectado que se ha bautizado como Internet de las Cosas (*IoT*).

La inteligencia artificial también se podrá beneficiar en su desarrollo de los avances en la computación cuántica, que permitirá desarrollar ordenadores extremadamente rápidos, capaces de realizar cálculos, simulaciones y análisis que con la computación actual no son viables.

5. Desarrollo/Implementación/Arquitectura

5.Desarrollo

5.1.Lenguaje utilizado

Vamos a desarrollarlo con *python* por su facilidad para la creación de machine learning, este es un lenguaje de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código.

Se trata de un lenguaje de programación multiparadigma, ya que soporta parcialmente la orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, dinámico y multiplataforma.

Se utiliza para la aplicación de algoritmos, los cuales pueden ser de *machine learning*, *deep learning* o *NLP (Natural Language Processing)*

5.2.Tipo de objetos que vamos a desarrollar

Utilizaremos el clasificador *Complement Naive Bayes* que fue diseñado para corregir las "suposiciones severas" hechas 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.

5.3.Librerías mas utilizadas

- *Pandas*:Es una librería especializada en el manejo y análisis de estructuras de datos. ... Permite leer y escribir fácilmente ficheros en formato CSV, Excel y bases de datos SQL. Permite acceder a los datos mediante índices o nombres para filas y columnas.

- *Numpy*: Acrónimo de Numerical Python es una librería especializada en el cálculo numérico y el análisis de datos, especialmente para un gran volumen de datos, da soporte para crear vectores y matrices grandes multidimensionales, junto con una gran colección de funciones matemáticas de alto nivel para operar con ellas.

- *Seaborn*: Es una librería que permite generar fácilmente elegantes gráficos. Seaborn esta basada en *matplotlib* y proporciona una interfaz de alto nivel que es realmente sencilla de aprender. Dada su gran popularidad se encuentra instalada por defecto en la distribución Anaconda.

- *Matplotlib*: Sirve para la generación de gráficos a partir de datos contenidos en listas o arrays de dos dimensiones, permite crear y personalizar los tipos de gráficos mas comunes y su extensión matemática *NumPy*.

- *Tensorflow*:Es una biblioteca de código abierto para la computación numérica rápida. Fue creado y es mantenido por Google y publicado bajo la licencia de código abierto Apache 2.0. La API es nominalmente para el lenguaje de programación *Python*, aunque hay acceso al lenguaje subyacente *C++*. A diferencia de otras librerías numéricas destinadas a ser usadas en el Aprendizaje Profundo como *Theano*, *TensorFlow* fue diseñado para su uso tanto en investigación y desarrollo, como por ejemplo en sistemas de producción, como *RankBrain* en la búsqueda de Google y el divertido proyecto *DeepDream2*. Puede funcionar en una sola CPUs, GPUs, así como dispositivos móviles y sistemas distribuidos a gran escala de cientos de máquinas.

- *Scikit-learn*: Es probablemente la librería más útil para machine learning en *Python*, es de código abierto y es reutilizable en varios contextos, fomentando el uso académico y comercial. Proporciona una gama de algoritmos de aprendizaje supervisados y no supervisados en *Python*. Está basada en *NumPy*, *SciPy* y *matplotlib*, de forma que es fácil reaprovechar el código que use estas librerías.

- *Nltk*: es un conjunto de bibliotecas y programas para el procesamiento del lenguaje natural (*PLN*) simbólico y estadísticos para el lenguaje de programación *Python*. *NLTK* incluye demostraciones gráficas y datos de muestra.

5.Implementación

5.4.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 problemático es el problema de la administración de paquetes y las diferentes versiones de *Python*. Anaconda, como una distribución de código abierto del lenguaje *Python*, 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*.

2. *Jupyter Notebook*: Es un cuaderno interactivo. En este cuaderno, puede escribir código, puede escribir documentos, puede tener diagramas. Anteriormente, escribimos código en un editor y luego escribimos documentos en Word para ilustrar el proyecto. Con Jupyter Notebook, podemos integrar la programación y la escritura, y desarrollar programas de acuerdo con nuestra propia lógica de pensamiento.

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

5.5.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 plantear es qué conocimientos o aplicaciones prácticas queremos conseguir con la adopción del machine learning. Este aspecto es crucial, pues dependiendo de la necesidad específica del negocio se optará por alguna de las diferentes soluciones que existen en el mercado.

En este caso hemos optado por implementar dos soluciones :

- Solución personalizada. Este opción suele ser la más precisa, pero requiere una gran cantidad de datos y definir la algoritmia de nuestro modelo de machine learning, en este caso lo hemos creado con nuestra red neuronal.

- Modelos pre entrenados. son un modo de implementar machine learning de una forma rápida, pero puede entrañar dificultad a aquellas empresas que no tienen suficiente experiencia en el uso de estas herramientas.

Sin duda se trata de una buena solución para introducir a un negocio en la Inteligencia Artificial, pero para poder obtener buenos resultados y hacerlo pronto.
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 y analizarla para poder realizar las predicciones.

5.6.Arquitectura

El concepto de arquitectura referida a redes neuronales hace mención no solo al número de capas neuronales o al número de neuronas en cada una de ellas, sino a la conexión entre neuronas o capas, al tipo de neuronas presentes e incluso a la forma en la que son entrenadas.

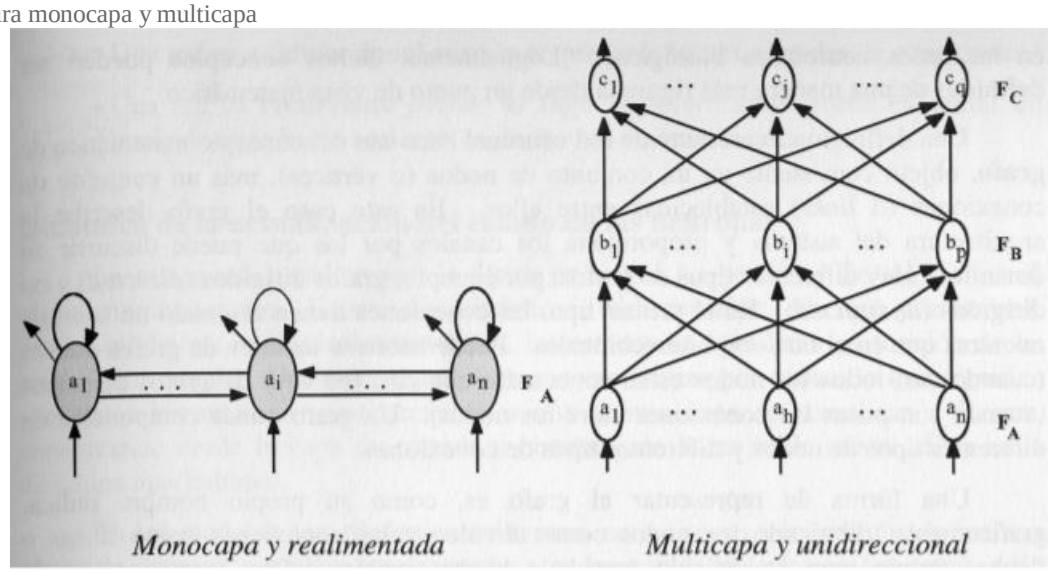
- El tipo que utilizaremos sera *Feed fordward networks* este tipo de redes son una extensión del perceptrón. Constan de varias capas de neuronas, "*fully connected*" (es decir, una celda está conectada con todas las celdas de la siguiente capa), hay al menos una capa oculta, la información circula de izquierda a derecha ("*feed forward*") y el entrenamiento de la red se realiza mediante *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 una capa de salida con 1 neurona por lo que tendría un total de 4 capas, en resumen tendremos una arquitectura unidireccional multicapa .

- *Backpropagation*-Es un algoritmo descrito en 1986 en "*Learning representations by back-propagating errors*".. En una red neuronal el error de las capas anteriores depende directamente del error de las capas posteriores, es por ello que podemos ir analizando y optimizando la propagación (*propagation*) del error hacia atrás (*back*) en la red. De esta forma podemos asumir también que si tenemos el error controlado para una neurona N 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 parámetros de entrada de nuestra neurona N, también es poco influyente para nuestro error final.

Aquí tenemos un ejemplo de los tipos de arquitectura:

Arquitectura monocapa y multicapa



6. Validación y evaluación.

6.1.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

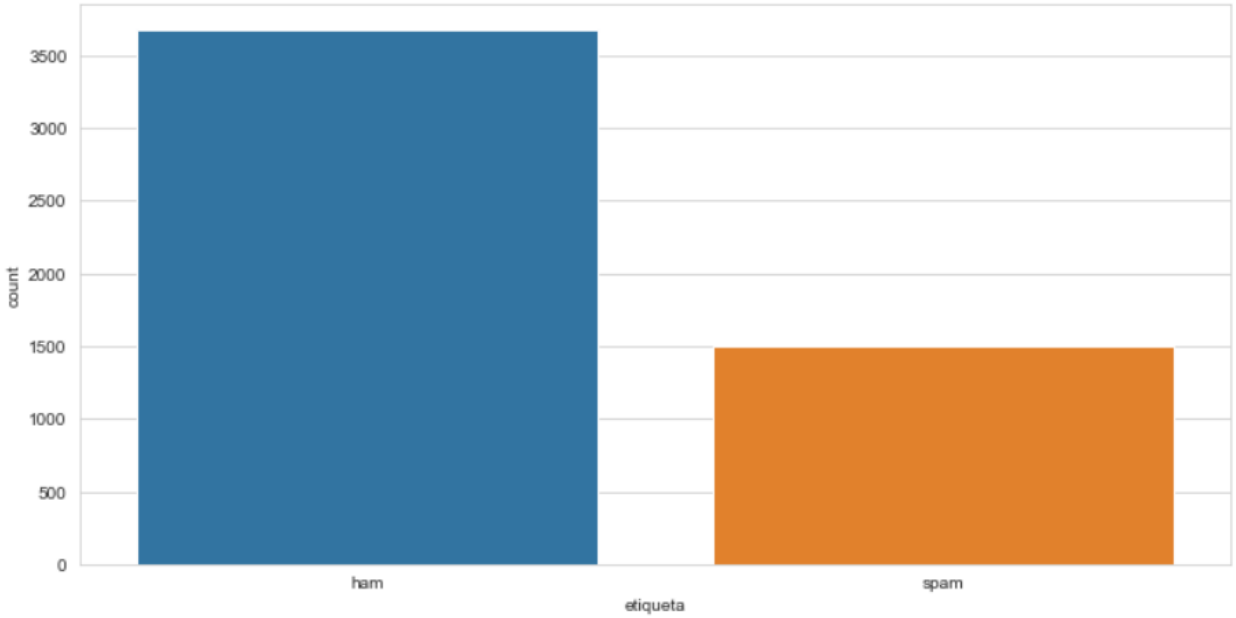
Primero validaremos los datos que hemos recibido y hemos reorganizado cambiando el nombre de las etiquetas por unas que nos resulte mas practicas y los mostramos por pantalla para ver como queda el formato de ellos y poder visualizar el numero de registros recibimos .

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

6.2.Pruebas selección modelo

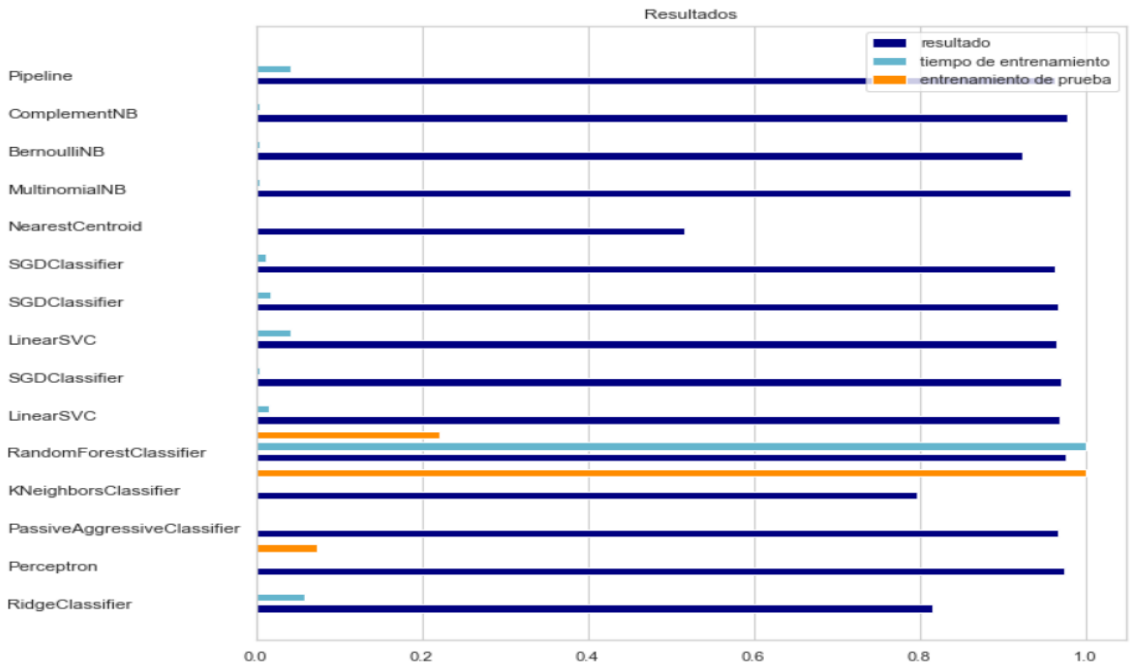
Para seleccionar el modelo previamente vamos a convertir una colección de documentos de texto en una matriz de recuentos de tokens, después vamos Dividir arreglos o matrices en subconjuntos de prueba y entrenamiento , podemos ver el numero de observaciones y el numero de tokens que consta

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 grafica.

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.

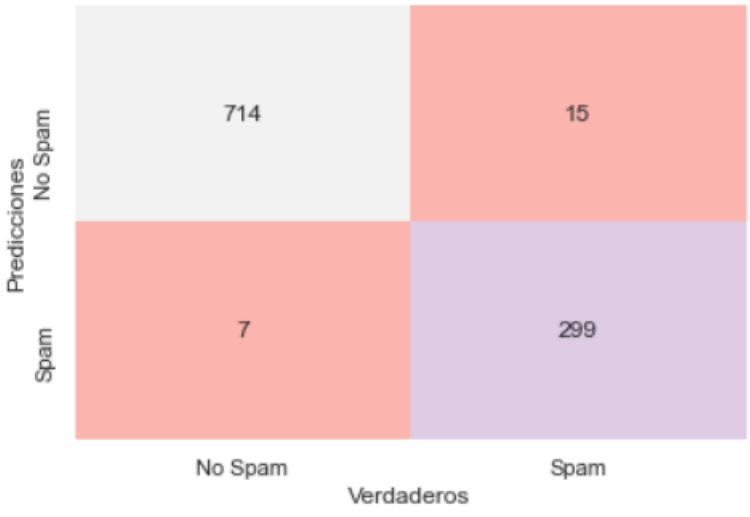
Resultado:

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:



6.3.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

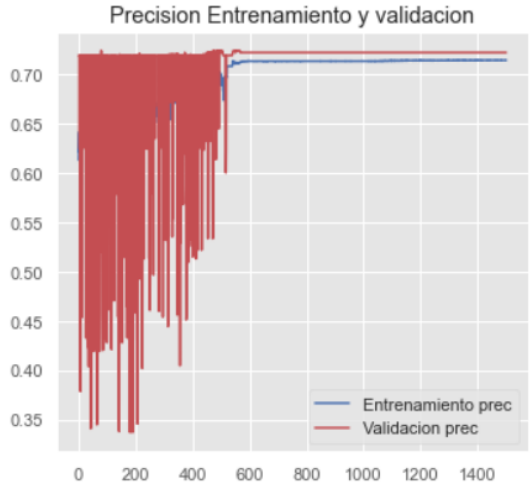
Resultado de la prueba:

```
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 grafico 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



7. 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, también para ayudar a los profesionales de la seguridad a tratar la cada vez mayor complejidad de los sistemas modernos de IT, IoT, así como la ingente cantidad de datos creados por ellos, e intentar estar por delante de los ciberatacantes. La ciberseguridad se enfrenta a múltiples retos, como son la detección de intrusiones, la protección de la privacidad, la defensa proactiva, la identificación de comportamientos anómalos o la detección de amenazas sofisticadas, pero, sobre todo, a las cambiantes amenazas que aparecen continuamente.

Realmente, la IA puede utilizarse en todas las etapas de una seguridad integral inteligente: identificación, protección, detección, respuesta y recuperación ante incidentes.

En la actual situación de pandemia debida al COVID-19, lo que se ha observado es la gran capacidad que han mostrado los cibercriminales para adaptarse rápidamente al nuevo contexto vulnerable de teletrabajo aprovechando las conexiones a internet de los hogares para acceder a datos y sistemas de la empresa. Los cibercriminales han personalizado los vectores de ataque con métodos avanzados de robo de credenciales, ataques de phishing muy orientados, sofisticados ataques de ingeniería social y técnicas avanzadas de ocultación de malware, entre otros. En la medida en que estas técnicas se vayan combinando cada vez más con la IA, los ataques serán más difíciles de detectar y tendrán un mayor éxito, según el citado informe de ENISA.

Por lo tanto los atacantes pueden utilizar sistemas de IA no sólo para tomar sus decisiones, sino también para manipular las decisiones que toman otros. De forma simplificada, un sistema de IA no deja de ser un sistema de software en el que se utilizan datos, modelos y algoritmos de procesamiento. En este sentido, un sistema de IA que no haya sido desarrollado con ciberseguridad desde el diseño puede ser vulnerable a ciberataques que tengan como objetivo los datos, el modelo o el algoritmo de la IA y provocar resultados no deseados o decisiones equivocadas en el sistema, con lo cual en el futuro no solo se va a utilizar las IA y las machine learning si no que va a ser necesario desarrollarlas con ciberseguridad desde el diseño.

8. Bibliografía.

https://keras.io/
https://www.kaggle.com/
https://scikit-learn.org/
https://www.tensorflow.org/
https://www.aprendemachinelearning.com/
https://pandas.pydata.org/
https://www.cienciadedatos.net/
https://aprendeia.com/
https://numpy.org/
Python Machine Learning 3rd Edition by Sebastian Raschka
Hands-On Machine Learning with Scikit-Learn and TensorFlow