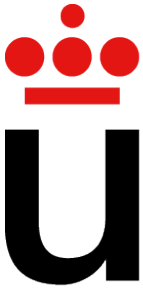

Universidad
Rey Juan Carlos

GRADO EN INGENIERIA EN SISTEMAS AUDIOVISUALES Y MULTIMEDIA

Curso Académico 2020/2021

Trabajo Fin de Grado

IMPLEMENTACIÓN DE FUNCIONALIDADES EN LEARNINGML

Autor : Isaac Merchán Blanco

Tutor : Dr. Gregorio Robles

Trabajo Fin de Grado

IMPLEMENTACIÓN DE FUNCIONALIDADES EN LEARNINGML

Autor : Isaac Merchán Blanco

Tutor : Dr. Gregorio Robles

La defensa del presente Proyecto Fin de Carrera se realizó el día de de 2021, siendo calificada por el siguiente tribunal:

Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente calificación:

Calificación:

Fuenlabrada, a de de 2021

Dedicado a

mi familia y amigos

II

Agradecimientos

Quiero agradecer a mis padres quienes han sido capaces de aguantar mis peores momentos durante esta carrera. A mi madre que desde pequeño siempre ha estado ayudandome con todo lo que he necesitado y confiando en mi. Y a mi padre que aunque no pase tanto tiempo con él me ha inculcado unos valores y una forma de ver la vida que me han llevado a ser quien soy a día de hoy.

Quería agradecer también a mis amigos por estar siempre dispuestos a sacar un rato para vernos y hablar, esos momentos de descanso también forman parte de esto y sin ellos nada sería lo mismo.

Gracias también a mis companeros con los que he pasado estos últimos años de mi vida y una etapa muy importante, sin ellos esto hubiera sido aún más difícil. Han hecho que las clases, las tardes de estudio, las prácticas y, en general, la carrera sea más divertida.

Por último, agredecer a Gregorio su manera de explicar y su disposición para ayudarme siempre que he necesitado algo y a Juanda por dejarme colaborar en su página.

Resumen

Aquí viene un resumen del proyecto. Ha de constar de tres o cuatro párrafos, donde se presente de manera clara y concisa de qué va el proyecto. Han de quedar respondidas las siguientes preguntas:

- ¿De qué va este proyecto? ¿Cuál es su objetivo principal?
- ¿Cómo se ha realizado? ¿Qué tecnologías están involucradas?
- ¿En qué contexto se ha realizado el proyecto? ¿Es un proyecto dentro de un marco general?

Lo mejor es escribir el resumen al final.

Summary

Here comes a translation of the “Resumen” into English. Please, double check it for correct grammar and spelling. As it is the translation of the “Resumen”, which is supposed to be written at the end, this as well should be filled out just before submitting.

Índice general

Índice de figuras

Capítulo 1

Introducción

Hoy en día vivimos en un mundo digital en el cual hay una cantidad inmensa de datos. La obtención de estos crece a un ritmo exponencial, solo en 2018 se generaron 33 zettabytes (un zettabyte equivale a 1.000 millones de terabytes) lo que equivale a 16.5 veces más que en 2009 [1]. A pesar de esto, solo somos capaces de procesar el 0.5 % [13] y ese porcentaje cada vez va a menos ya que no se incrementa de la misma forma la capacidad de obtención que de procesado. El principal problema de esto está en la complejidad y cantidad de los datos pues los humanos no somos capaces muchas veces de extraer información útil de esos datos. Esto ha hecho que se fomente el estudio de distintos campos, uno de ellos es el *machine learning*. Estas limitaciones mencionadas anteriormente desaparecen en el momento en el cual el *machine learning* entra en juego, ya que es capaz de procesar grandes cantidades de datos complejos y transformarlos en información legible y fácil de analizar por los los humanos.

El *machine learning* (o aprendizaje automático) es la rama de la inteligencia artificial que trata de implementar el aprendizaje automático de máquinas a través de un entrenamiento. A pesar de ser un concepto relativamente nuevo, se basa en campos de estudio anteriores como la modelación estadística y o el reconocimiento de patrones.

1.1. Presentación de la aplicación

Este Trabajo Fin de Grado tiene como principal objetivo la integración de los algortimos de aprendizaje automático de Naive Bayes y K-NN en la web de LearningML que actualmente solo utiliza el método de redes neuronales para clasficar imágenes o texto.

LearnignML es una web creada por Juan David Rodríguez García utilizando Angular y esta construida de tal manera que cualquier persona sin grandes conocimientos sobre el aprendizaje autmático pueda crear un modelo, entrenarlo y después probarlo. La web además permite descargar el modelo creado en un archivo JSON para después poder cargarlo sin necesidad de crearlo de nuevo, esto es muy útil ya que de esta manera no se tienen que volver a introducir todas las entradas para el aprendizaje, si no que directamente se puede importar un modelo creado previamente e introducir una entrada para ver su funcionamiento. También tiene la opcion de registrarse y poder guardar los modelos en tu cuenta o incluso tener proyectos compartidos entre varias personas. Por último, permite crear un modelo en Scratch y utilizarlo en la web aunque esta parte no se verá durante este trabajo.

1.2. Estructura de la memoria

La memoria empieza con un resumen, en el que se explica brevemente en que consiste el proyecto, así como las tecnologias que se han usado para llevarlo a cabo. A continuación, se sitúan los agradecimiento y los índices, tanto el general como el de figuras. Finalmente, van los 6 capítulos en lo que se divide la memoria.

- **Capítulo 1: Introducción.** En el primer capítulo se hace una breve explicación de en que consiste la web de LearningML y la estructura que va a seguir el trabajo.
- **Capítulo 2: Objetivos.** En este capítulo se describen los objetivos principales y específicos del trabajo y la planificación temporal de este.
- **Capítulo 3: Estado del arte.** Se trata de una breve explicación sobre la mayoría de las teconologías utilizadas en el desarrollo de la aplicación web.
- **Capítulo 4: Diseño e implementación.** Se desccribe de manera detallada tanto los cambios previos en la estructura como la arquitectura final de la web así como la implementación de los dos modelos de aprendizaje.
- **Capítulo 5: Experimentos y validación.** Se comprueba si funcionan correctamente las funcionalidades implementadas en la web de LearningML.

- **Capítulo 6: Conclusiones.** Por último, se repasa todo lo aprendido durante el desarrollo del proyecto y se comentan algunas mejoras o funciones que se podrían implementar en un futuro.

Capítulo 2

Objetivos

2.1. Objetivo general

Mi trabajo fin de grado consiste en añadir los algortimos de aprendizaje automático de Bayes y K-NN de manera intuitiva e independientemente de si se quieren clasificar textos o imágenes a la web de LearningML.

2.2. Objetivos específicos

Para la realización del objetivo general se han planteado los siguientes objetivos específicos:

- Reorganizar la manera de trabajar de LearningML para separar la obtención de datos del algorítmo utilizado.
- Implementar el algorítmo de Bayes y K-NN para que funcionen como dos clasificadores de tal forma que primero entrenen con las entradas de aprendizaje y luego evaluen otra entrada proporcionada por el usuario y la asigne correctamente a uno de los grupos creados previamente.
- Añadir en la interfaz una manera de poder elegir el tipo de algorítmo que se quiere utilizar de forma independiente a si los elementos a clasificar son texto o imágenes.
- Dejar que el usuario pueda tocar algunos de los parámetros de los clasificadores.

2.3. Planificación temporal

La primera idea que tuve de empezar con el Trabajo Fin de Grado (TFG) fue en Enero de 2021 y quería hacerlo a la vez que las prácticas entonces me puse en contacto con Gregorio y me comentó esto. Me llamo la atención la web de LearningML ya que recuerdo haber utilizado Scratch en el colegio y me parecía interesante aportar algo aunque sea de manera indirecta a que los niños se interesen por la programación. En principio este proyecto iba a ser tanto prácticas como TFG y le iba a poder dedicar bastante horas todos los dias, sin embargo, me aceptaron en unas prácticas y no tuve tanto tiempo ya que estas eran de jornada completa. Además de esto, aun tenía una asignatura pendiente asi que era difícil establecer un horario concreto para dedicarle tiempo al proyecto. Cuando no tenía un examen cerca, al salir de trabajar me ponía con ello un rato y luego los fines de semana por la mañana también le dedicaba algo de tiempo. Recuerdo que primero tuve que aprender como funcionaba Angular y Typescript con un curso que tiene Juanda. Practicando y familiarizandome con la forma de trabajar en Angular. Después tuve varias reuniones con Juanda (creador de LearningML) para ver como funcionaba la aplicacion de LearningML y donde había que añadir los cambios. También tuve que buscar que librerias ya construidas de Bayes y K-NN podiamos adaptar para usarlas en la web y finalmente ponerme a ello. Para K-NN si que conseguí encontrar una librería que me sirvió aunque por como se obtienen los datos, había que procesarlos de nuevo antes de usarlos. Para Bayes no encontré nada y tuve que crear el clasificador desde cero lo que me llevo algo más de tiempo que en el otro caso. Por último, quedaba la tarea de trasladar todo eso al papel. Me puse en contacto con mi tutor y enseguida me paso una plantilla con los apartados a rellenar y comencé a hacerlo hasta que finalmente, con no mucho tiempo de margen para entregar el trabajo, terminé.

Capítulo 3

Estado del arte

En este apartado veremos tanto las tecnologias utilizadas como una breve explicación sobre lo que es el modelo de Bayes y de K-NN.

3.1. HTML5

HTML [7] [8] (HyperText Markup Language) es un lenguaje de marcado que se utiliza para describir la estructura de las páginas web. HTML5 es la quinta versión de este lenguaje. A lo largo de todas las versiones se han incorporado y eliminado diferentes características, con el fin de hacerlo más eficiente y facilitar el desarrollo web compatibles con distintos navegadores y plataformas, y manteniéndose también compatible con las versiones anteriores. A pesar de ser un estándar a cargo del Consorcio WWW, fue la asociación WHATWG, formada por Apple, Opera y Mozilla, la que publicó el primer borrador de HTML5 cuando W3C decidió dejar de evolucionar HTML. Más tarde W3C y WHATWG trabajaron juntos durante varios años en el desarrollo de HTML5, hasta que se separaron debido a sus diferentes objetivos, W3C quería publicar una versión terminada mientras que WHATWG quería seguir trabajando, manteniendo HTML5 en constante evolución. Algunas de las principales características de HTML5 son:

- Incorpora nuevas etiquetas que especifican la semántica del contenido, ayudando a interpretar mejor la página. Antes solo teníamos para definir cualquier sección, ahora podemos utilizar <header>, <nav>, <section>, <footer>, entre otros.

- Ofrece también mejoras en los formularios como validaciones, nuevos tipos de campos

(email, date, range, etc) y nuevos atributos, por ejemplo, multiple, placeholder o form.

- Cuenta con numerosas APIs. Algunas de ellas son:

  - Audio y video: permite incrustar elementos de audio y vídeo y reproducirlo desde el propio navegador.
  - Web Storage: permite almacenar datos del lado del cliente, cuando la sesión está activa.
  - Canvas: que permite hacer dibujos, juegos, animaciones, etc.
  - Geolocalización: permite mostrar e interactuar con un mapa de Google Maps.

3.2. TypeScript

TypeScript [9] un lenguaje de programación, de código abierto y desarrollado por Microsoft. Nació como una necesidad de mejorar algunos problemas de JavaScript, entre ellos el hecho de que es bastante complicado crear aplicaciones a gran escala con JavaScript. Está pensado para el desarrollo de aplicaciones robustas y escalables, y se puede utilizar tanto en el lado del cliente como en el del servidor junto a Node.js.

TypeScript es un superconjunto de JavaScript, es decir, se trata de un lenguaje creador a partir de JavaScript. Los programas de JavaScript son programas válidos de TypeScript. Esto permite integrar JavaScript en proyectos ya existentes, sin tener que rehacer todo el proyecto en TypeScript, ya que todo el código TypeScript se puede compilar en JavaScript nativo. Las características más importantes de TypeScript son:

- Tenemos a nuestra disposición las herramientas de JavaScript ES6 (Ecmascript 6). De esta forma se mantiene actualizado con las últimas mejoras de JavaScript.
- Cuenta con un tipado estático, por tanto, al crear variables podemos añadir el tipo de dato, así podemos evitar errores en tiempo de ejecución.
- Objetos basados en clases, lo cual hace más sencilla la programación orientada a objetos y añade más funcionalidad.

3.3. Node.js

Node [6] es un entorno de ejecución de JavaScript controlado por eventos asíncronos, diseñado para construir aplicaciones de red escalables. Fue creado por Ryan Dahl en 2009 y está influenciado por algunos sistemas como Event Machine de Ruby ó Twisted de Python.

Cuando se utiliza JavaScript en el lado del cliente, es el navegador quien, a través de un programa interno (un motor), interpreta el código JavaScript. De este mismo modo, Node.js proporciona un entorno de ejecución JavaScript del lado del servidor, haciendo uso del motor V8 de Google, que se encarga de interpretar el código y ejecutarlo. El motor V8 es el que Google usa en su navegador Chrome y es posible descargarlo e introducirlo en cualquier aplicación.

Node.js está pensado para soportar grandes cantidades de solicitudes muy altas gracias a su naturaleza asíncrona. El modelo de concurrencia más común está basado en la utilización de hilos del sistema operativo. Se genera un nuevo hilo para cada conexión, de modo que a mayor cantidad de personas, mayor cantidad de recursos consumidos del servidor y mayor número de servidores son necesarios. Node.js emplea un único hilo y un bucle de eventos asíncrono. Las nuevas peticiones son tratadas como eventos en este bucle,de esta forma, no se produce ningún tipo de bloqueo en el flujo de trabajo.

3.4. Archivos JSON

JSON (JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato de intercambio de datos [5]. Es sencillo tanto de escribir como de leer, lo que facilita que tanto los humanos como las máquinas sean capaces de utilizarlos y generarlos. Aunque esta basdo en un subconjunto del lenguaje de programación Javascript, es un formato de texto completamente independiente del lenguaje.

En el siguiente texto en formato JSON, podemos ver las dos estructuras que nos ofrece, una colección de pares de nombre/valor y una lista de valores. En el ejemplo, vemos que el tipo de clasificador será de texto en este caso y que dentro de los datos tendremos dos etiquetas, una de vocales con una lista de las entradas que pertenecen a esta etiqueta y otra de consonantes.

```
{
    "type": "text",
    "data": {
        "vocales": [
```

```
            "aaa",
            "eee",
            "iii"
        ],
        "consonantes": [
            "sss",
            "ddd",
            "fff"
        ]
    }
}
```

3.5. Angular

Angular [3] es un *framework* de código abierto diseñado por Google. Permite crear aplicaciones web de una sola página, lo que se denomina SPA (Single Page Application). Utiliza TypeScript y HTML para el desarrollo de las aplicaciones web.

Angular permite separar el *frontend* del *backend* y sigue un modelo MVC (Modelo-Vista-Controlador) el cual se verá como funciona en el siguiente apartado. De esta forma, las modificaciones y las actualizaciones de las aplicaciones son rápidas y sencillas.

Una de las principales ventajas de este *framework* y de las páginas SPA es la velocidad de carga entre las diferentes vistas de la aplicación. Cuando se cambia de vista no se recarga la página si no que éstas se cargan de manera dinámica, rápida y reactiva.

Angular dispone de varias versiones, aportando todas ellas mejoras en dicho *framework*. La primera versión de Angular, se conoce como AngularJS, y es la más distinta a las demás. El resto son actualizaciones de Angular 2. Cuando se definió la segunda versión se decidió modificar el nombre del *framework* y dejarlo como Angular. La última versión estable es la versión 10, utilizada por LearningML.

La arquitectura de una aplicación Angular se basa en clases de 4 tipos distintos (módulos, componentes, servicios y directivas). Estas clases se identifican a través de decoradores, los cuales permiten cargar los metadatos necesarios que determinan a Angular como debe usar dichas clases.

Módulos.

Los módulos suministran el contexto de compilación de los componentes, es decir, es aquí donde se definen los diferentes componentes que van a formar la aplicación, las dependencias, las clases que actúan como servicios en la aplicación y las rutas de navegación que establecen las vistas de la aplicación.

Toda aplicación angular tiene un módulo principal o raíz. Este módulo suele recibir el nombre de *AppModule* y es el que inicia el sistema de arranque de la aplicación.

Al igual que en JavaScript, los módulos de Angular permiten importar y exportar funcionalidades proporcionadas por otros módulos, por lo que una aplicación puede tener más de un módulo, siendo cada uno de ellos independiente. La manera en que se organizan estos módulos ayuda a la creción de aplicaciones más complejas y a la reutilización de código. Además, permite que la carga inicial sea mínima, ya que cada módulo se carga a petición, es decir, cuando se va a utilizar.

Componentes.

Las aplicaciones Angular tienen un componente que es nexo de unión entre los diferentes componentes que forman la aplicación y el modelo de objeto del documento de la página (DOM). Los componentes son los que tienen la lógica y los datos de la aplicación. Son los que controlan las diferentes plantillas HTML que se cargan cuando se modifican las URLs. Estas plantillas HTML son las vistas que forman la interfaz de usuario.

De la misma forma que las aplicaciones diseñadas con Angular pueden tener más de un módulo, también pueden tener subcomponentes, que se pueden relacionar entre sí mediante dos tipos de vinculación de datos, eventos y propiedades. A través de los eventos, la aplicación responde a las entradas del usuario actualizando los datos. A través de las propiedades, se envían valores calculados de los datos de la aplicación a los HTMLs que forman las vistas. Esta vinculación de datos se puede producir en ambas direcciones, es decir, igual que los datos de la aplicación pueden modificar los HTMLs, los cambios en el DOM pueden modificar los datos de la aplicación.

El decorador @Component es el que determina que una clase actúe como un componente.

Servicios

Los servicios son clases en las que se definen datos o funcionalidades generales que no están asociadas a una determinada vista. Son usados para compartir datos y operaciones entre componentes. También es donde se suele realizar toda la operativa de las peticiones a la API

de las aplicaciones. Deben llevar el decorador @Injectable, pues es el que permite obtener los metadatos necesarios para que otras clases puedan inyectar sus dependencias.

Directivas

Las directivas son clases en las que se definen los términos claves que se usan en las plantillas. Cuando se carga una vista, Angular procesa estos términos clave que modifican el HTML y el DOM de la aplicación. Por tanto, una plantilla, además de utilizar HTML para definir la vista, usa marcas propias de Angular que son estas directivas.

Existen dos clases de directivas: (i) las de atributo, que modifican el comportamiento del componente, y (ii) las estructurales, que únicamente modifican la apariencia.

Angular dispone de un enrutador, módulo que contiene un servicio para definir las rutas de navegación de la aplicación. Este módulo se ajusta a las convenciones de los navegadores. Es decir, tanto si se pone la URL en la barra de direcciones, como si se pincha un enlace en la aplicación, como si se hace *click* en el botón de retroceso o avance, se carga a la página correspondiente. El enrutador es el que muestra u oculta una vista. Cada vez que se modifica la URL, el enrutador se encarga de añadir la ruta en el historial del navegador, es esto lo que permiete que los botones de retroceso y avance funcionen. Cada ruta de navegación está asociado a un componente.

3.5.1. MVC

Las aplicaciones web se suelen desarrollar siguiendo una serie de pautas. El patrón que sigue Angular es el Modelo Vista Controlador (MVC), cuyos elementos son [2] :

- **Modelo:** es la representacion de los datos de toda la aplicación y, por tanto, maneja las consultas y todas las acciones relacionadas con base de datos en caso de que las hubiera. También tratará los datos para cumplir con las funcionalidades de la aplicacióon.

- **Vista:** recibe los datos y los sirve al usuario final en la interfaz. Ni el controlador ni el modelo se preocupan por el diseño ni la apariencia final, de todo el apartado visual eso se encargarán las vistas.

- **Controlador:** es el elemento principal, encargado de responder a las acciones de entrada y lanzar una serie de funciones asignadas previamente. Es decir, recibe un evento de

entrada, se comunica con el modelo en caso de que sea necesario, consultan las vistas y responden al evento con una salida.

Las principales ventajas del MVC son las siguientes:

- Mejora la escalabilidad, puesto que es un modelo sencillo en el que cada tipo de lógica esta separado.
- Facilita mantenimiento.
- Ofrece la posibilidad de reutilizar los componentes.

3.5.2. Angular CLI

Angular CLI (Command Line Interface) [10] es una herramienta de línea de instrucciones creada por el equipo de Angular. Es muy util a la hora de iniciar una aplicación web diseña con Angular, ya que con una sola instrucción, se genera el esqueleto de carpetas y archivos necesarios de la aplicación. Además, contiene herramientas predefinidas que ayudan al desarrollo y mantenimiento de este tipo de aplicaciones. Al crear una aplicación web con Angular CLI, dentro de la estructura de archivos, se crea un archivo de configuración, en el cual se añaden las dependencias necesarias para que la aplicación web compile y ejecute. Este archivo se va modificando conforme se van creando los componentes, servicios o directivas. Para ello, Angular CLI tiene instrucciones que permiten crear estas clases de manera sencilla, creando los distintos archivos que conforman un componente, un servicio o una directiva. Entre las herramientas predefinidas destaca el compilador, el sistema de testing y el servidor web.

También existe la posibilidad de poner notas al pie de página, por ejemplo, una para indicarte que visite la página del GSyC[1].

3.6. TensorFlow

TensorFlow [4] es un sistema de aprendizaje automático de segunda generación creado por Google Brain y es la herramienta más utilizada en el mundo del Deep Learning. La decisión de liberar TensorFlow se tomó en noviembre de 2015, por lo que hoy es posible acceder a

[1] `http://gsyc.es`

esta herramienta libremente y editarla en función de las necesidades. Esta biblitoeca, es una gran plataforma para construir y entrenar redes neuronales, que permiten detectar y descifrar patrones y correlaciones, similares al aprendizaje o razonamiento que usamos los humanos.

3.6.1. MobileNets

MobileNets [11] es, además de una librería de TensorFlow la cual vamos a utilizar, una red neuronal convolucional más eficiente ya que se basa en una capa convolucional normal, de la dimensión deseada, pero con un solo filtro. De esta manera, al pasar información por una capa se genera una sola convolución, que esta a su vez sirve como entrada a otra capa de convolución de tamaño fijo de filtro 1x1, con el número de filtros deseados en la convolución a sustituir.

Aunque todo esto suena bastante complejo, lo que a nosotros nos interesa es que vamos a usar esta librería para transformar las imágenes en vectores que serán las entradas del clasificador sin importar el tipo. Lo que haremos será quitar la última capa de la red neuronal que genera MobileNets y quedarnos con ese vector.

3.7. Modelo de Naive Bayes

El modelo de Naive Bayes [10] consiste en un clasificador probabilístico bastante simple y que supone que las distintas características que se evalúan son independientes. Aunque esto no siempre es así, cuando evaluamos una gran cantidad de características, tiende a cumplirse. Esto es lo que simplifica en gran medida el clasificador y por lo que es muy utilizado cuando se necesitan procesar grandes dimensiones de datos y no se requiere demasiada exactitud. Este clasficador requiere unos datos de entrada de entrenamiento y cuantos más reciba, mejor funcionará. Además, el modelo de Naive Bayes es un modelo paramétrico, es decir, usa el conjunto de entrenamiento para aprender el modelo explícitamente. Lo que hace Naive Bayes es aprender de los datos de entrenamiento y luego predice la clase de la entrada de prueba con la mayor probabilidad a posteriori.

La fórmula de la probabilidad de una clase C dadas unas características desde F1 a Fn es la figura 3.1.

En la práctica [13] solo importa el numerador, puesto que el denominador no depende de C y los valores de Fi son datos, por lo que el denominador es constante en la práctica. En el

$$p(C|F_1,\ldots,F_n) = \frac{p(C)\,p(F_1,\ldots,F_n|C)}{p(F_1,\ldots,F_n)}.$$

Figura 3.1: Fórmula de probabilidad de Naive Bayes.

$$p(C|F_1,\ldots,F_n) = \frac{1}{Z}p(C)\prod_{i=1}^{n}p(F_i|C)$$

Figura 3.2: Fórmula de final de Naive Bayes.

numerador es donde se asume que cada Fi es independiente de cualquier otra Fj (siempre que j es distinta de i) cuando estas dependen de C. Esto significa que haciendo estos supuestos, la probabilidad de C teniendo en cuenta las variables clasificatorias puede expresarse como muestra la imágen 3.2 donde Z es un factor que depende unicamente de F1 ...Fn, es decir, constante si los valores de Fi son conocidos.

Lo que hace el clasificador finalmente es calcular la probabilidad de cada posible clase con las características de entrada y asgina la entrada a esa clase.

3.8. Modelo K-NN o K vecinos más cercanos

El modelo de K-NN (*k-nearest neighbors*) o *k* vecinos más cercanos [12] [14] es un método estadístico de reconocimiento de patrones supervisado. Es un modelo no paramétrico ya que usa directamente los datos de entrenamiento para inferir cada vez la clasificación de una nueva entrada pero sin construir explicitamente un modelo. Lo primero que hace el algorítmo de K-NN es colocar las entradas de entrenamiento en una dimensión de tamaño i, donde i es el número de características. Luego coloca la entrada que queremos clasificar en esta misma dimensión. Después de esto, calcula la distancia a los K puntos más cercanos y en función de la clase de la que tenga más vecinos cerca calsificara la entrada en esa clase. Lo más habitual es seleccionar valores de K pequeños e impares, para evitar los empates.

En la imagen 3.3 podemos ver como en este caso, las características serían dos, una en cada eje y por otro lado tendríamos tendríamos dos clases. En el caso de utilizar una K con valor tres, asignaría la entrada que estamos evaluando a la clase B, sin embargo, si tomamos el valor

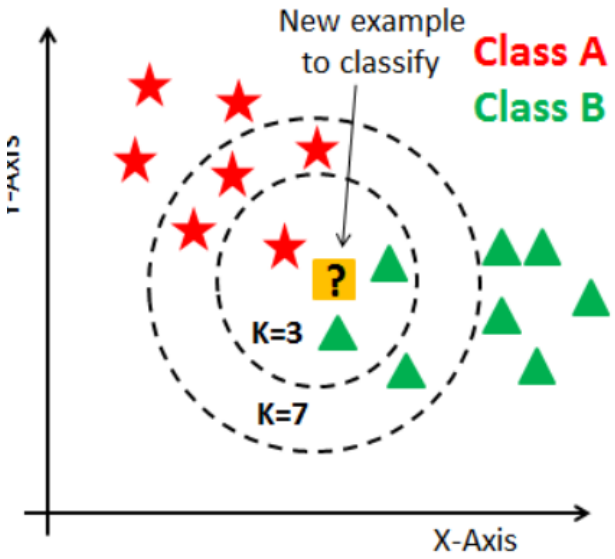

Figura 3.3: Clasificador K-NN

siete para la K, la clasificaría como clase A. Con este ejemplo, nos damos cuenta de que es importante que el valor de K sea pequeño. Este parámetro será uno de los que pueda modificar el usuario en la web de LearningML.

Capítulo 4

Diseño e implementación

Aquí viene todo lo que has hecho tú (tecnológicamente). Puedes entrar hasta el detalle. Es la parte más importante de la memoria, porque describe lo que has hecho tú. Eso sí, normalmente aconsejo no poner código, sino diagramas.

4.1. Arquitectura antigua de LearningML

Lo primero que hay que tener claro a la hora de hacer algún cambio en una aplicación web, es su estructura, por lo que vamos a ver como funcionaba la de LearningML antes de realizar el cambio en su forma de trabajar. Hay que tener en cuenta que LearningML está construido por Angular, por lo que esta compuesto por modulos, componentes y servicios. Es importante recordar que antes solo se usaba un algorítmo (redes neuronales) de ahí esta estructura.
Según esta construido LearningML, en el módulo principal `AppModule` se carga el componente `ml-model` y el componente `ml-test-model` uno para construir el modelo y otro para probarlo respectivamente. Estos dos componentes tienen importados a su vez los servicios `image-classifier` y `classifier`, uno para crear un modelo de imágenes y otro de texto. El servicio para clasificar texto se llama unicamente `classfier` porque fue el primero que se incorporó cuando Juanda creo la web y no sabía si se añadirían más en un futuro.
De manera transversal a todo esto tenemos el servicio `labeled-data-manager`, que es una especie de çajón de sastre"donde se almacenan algunos datos que necesitamos en los otros servicios y componentes. Para la parte de test hay un solo componente llamado `ml-test-model` que llama a la función `classify` de `image-classifier` o a la función `run` de `classifier`,

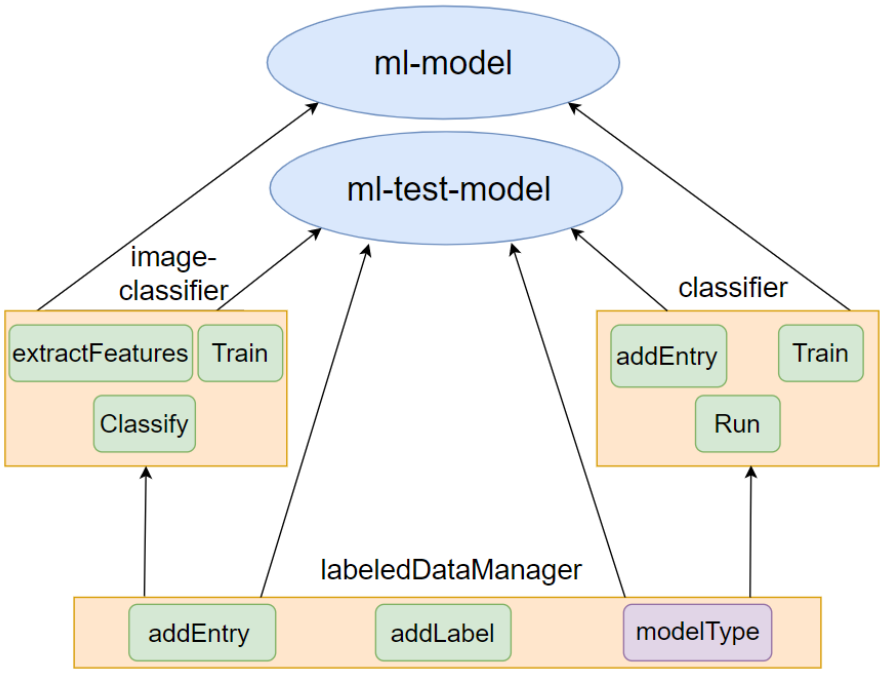

Figura 4.1: Estructura antigua de LearningML. En azul los componentes, en naranja los servicios, en verde las funciones y en morado las variables.

dependiendo de si se estan clasificando imágenes o textos respectivamente. Vamos a fijarnos en la figura 4.1 para ver que contiene cada uno de los servicios nombrados anteriormente.

- Labeled-Data-Manager: Es un servicio que se encarga de guardar los datos en bruto, ya sean imágenes o textos con la función `addEntry`. También almacena las etiquetas, es decir, las distintas posibles clases a las que puede pertenecer una entrada utilizando la función `addLabel`. Por último, guarda el tipo de modelo que se quiere construir directamente en la variable `modelType`. En caso de que carguemos los datos de un JSON, tambíen es aquí donde se realiza tanto la carga como la exportación.

- ImageClassifier: Dentro de este servicio encontramos tres funciones principales, `extractFeature` que se encarga de transformar las imágenes de entrada en tensores, `train` que construye el modelo y lo entrena y por último `classify` que evalua la imagen de entrada de test y la asgina a una de las posibles opciones (etiquetas). Dentro de esta última función tenemos que volver a llamar a `extractFeature` para que transforme la imágen a tensor y poder analizarla.

- Classifier: Este servicio lo que hace es transformar las entradas de texto en bruto en vectores, en esta ocasión se utiliza la librería BrainText, que es una adaptación que hizo Juanda de la librería Brain.js [1]. De nuevo, contiene la función `train` encargada de crear el modelo y entrenarlo y la función `run`, que sería el equivalente a la función `classify` del punto anterior, es decir, lee el texto introducido, lo trata y lo evalúa y por último determina a cuál de los grupos o etiquetas pertenece.

4.2. Arquitectura actual de LearningML

Como hemos visto en la sección anterior, tenemos tanto la extracción de características (el proceso de convertir los datos en bruto en un vector que pasar como entrada al constructor del modelo) como la construcción del modelo en el mismo servicio. La idea que tenemos en mente es que por un lado se extraigan las características y por otro lado se genere el modelo independientemente de si estamos clasificando imágenes o texto, ya que al constructor del modelo le entrara un vector sin importar si se ha construido de una forma u otra.
Ahora el componente `ml-model` importa los servicios de `feature-extraction` y `ml-algorithm`. Dentro de estos dos servicios, se importan a su vez en el primero `feature-extractor-image` y `feature-extractor-text` y en el segundo `ml-algorithm-image` y `ml-algorithm-text`.
En el componente `ml-model` existe la función `train`, la cual existia también antes pero llamaba directamente a uno de los dos antiguos servicios, ahora llama a `feature-extraction` y es este quién elige que servicio usar para extraer el vector dependiendo del tipo de dato. Después llama a `ml-algorithm` y este servicio decide que algorítmo utilizar para construir el modelo.
En la parte de prueba se ha dividido el componente `ml-test-model` en dos `ml-test-image` y `ml-test-text` para imágenes y texto respectivamente. El componente `ml-test-image` contiene la función `takeSnapshot` (en caso de que la imagen de prueba sea una foto a través de la cámara) y la función `onLoaded` (en caso de que la imagen sea un archivo ya existente), en ambos casos, se llama primero a `feature-extraction` y después a `ml-algorithm`.
En el caso del texto funciona de la misma manera pero solo podemos introducir texto por teclado.

[1] `https://brain.js.org/`

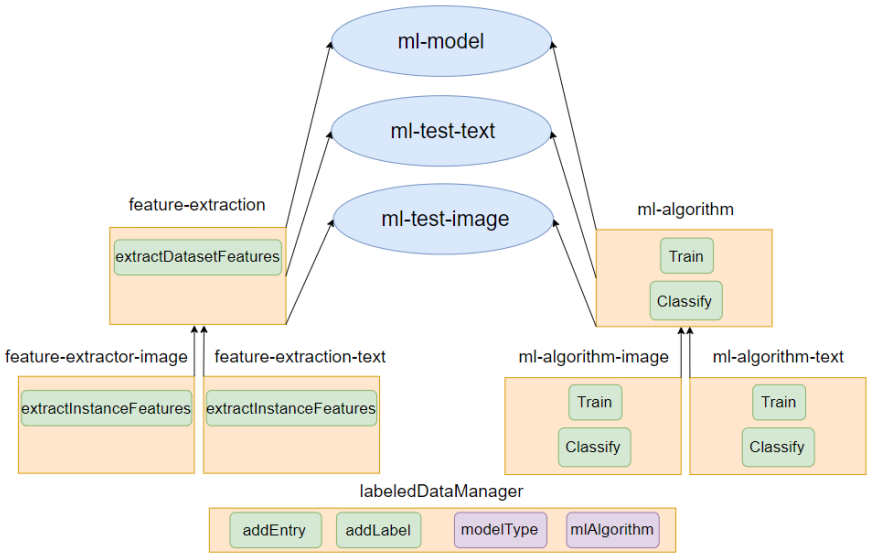

Figura 4.2: Estructura nueva de LearningML. En azul los componentes, en naranja los servicios, en verde las funciones y en morado las variables.

El servicio de `labeled-data-manager` sigue siendo transversal a todos los servicios y componentes, con el único añadido de que ahora guardamos el tipo de algorítmo que vamos a utilizar en la variable `mlAlgorithm`. Esta estructura simplifica mucho las cosas y hace que el código sea más fácil de entender y de modificar en caso de que algún día haya que añadir nuevos tipos de datos o nuevos tipos de algorítmos. Hace que las funciones y en general el código sean similares para distintos tipo de datos, además, separa los dos problemas por completo haciendolos totalmente independientes uno de otro. De nuevo, vamos a ver la figura 4.2 para entenderlo mejor.

- Labeled-Data-Manager: Este servicio como ya se ha comentado sigue siendo utilizado en todo el resto de servicios y componentes, con la diferencia de que ahora, además de los datos de etiquetas, datos en bruto de las entradas y tipo de modelo que ya guardaba antes, guarda también el tipo de algortimo en la variable `mlAlgorithm`.

- Feature-Extraction: En este servicio encontramos la función de `extractDatasetFeatures` que es la encargada de llamar a los otros dos servicios dependiendo de si se quieren extrar características de una imagen o de un texto.

    - Feature-Extractor-Image: Se encarga de extraer las características de las imágenes,

es decir, de convertir la imagen en bruto a un tensor. La función que realmente hace esto de obtener el tensor dada una imagen es `extractInstanceFeatures`. Aunque tiene más funciones para por ejemplo construir un mapa con la etiqueta que corresponde a cada imagen.

- Feature-Extractor-Text: Se encarga de extraer las características de los textos, por tanto, con un texto como entrada obtenemos un tensor. De nuevo, para que la arquitectura tenga una estructura coherente, la función recibe el nombre de `extractInstanceFeatu`

- ml-Algorithm: Este servicio cuenta con dos funciones principales. `train` encargada de elegir a que sub-servicio llamar dependiendo del algorítmo que se quiere utilizar. Y la función `classify` que de nuevo, elige que sub-servicio utilizar en funcion del algorítmo.

  - ml-Algorithm-knn: Cuenta con dos funciones que son las que llama el servicio que esta por encima para entrenar (`train`) o evaluar (`classify`) una entrada en el caso de que el algorítmo elegido por el usuario sea K-NN.

  - ml-Algorithm-bayes: Cuenta con las mismas funciones que K-NN pero en este caso construye o evalua un clasificador utilizando el algorítmo de Naive Bayes.

Finalmente, recalcalcar que esta estructura si se utiliza correctamente permite añadir un nuevo tipo de dato o de clasificador únicamente creando los nuevos servicios. Debemos tener en cuenta que estos nuevos servicios para que encajen con lo que ya hay deben tener al menos las mismas funciones que los ya existentes, ya que son a las que llaman los servicios superiores. De esta manera, sin apenas tocar los servicios superiores (solo habría que añadir en el punto donde se elije el tipo de dato a extraer o el tipo de modelo, la opción de elegir lo nuevo que hayamos creado) podemos implementar nuevas funcionalidades en un futuro de manera más limpia y sencilla.

4.3. Arquitectura general

Si tu proyecto es un software, siempre es bueno poner la arquitectura (que es cómo se estructura tu programa a “vista de pájaro”).

4.4. Crear el clasificador de Naive Bayes

4.5. Crear el clasificador de K-NN

Capítulo 5

Experimentos y validación

Este capítulo se introdujo como requisito en 2019. Describe los experimentos y casos de test que tuviste que implementar para validar tus resultados. Incluye también los resultados de validación que permiten afirmar que tus resultados son correctos.

5.1. Prueba del clasificador de Naive Bayes

5.2. Prueba del clasificador de K-NN

Capítulo 6

Conclusiones

6.1. Consecución de objetivos

Esta sección es la sección espejo de las dos primeras del capítulo de objetivos, donde se planteaba el objetivo general y se elaboraban los específicos.

Es aquí donde hay que debatir qué se ha conseguido y qué no. Cuando algo no se ha conseguido, se ha de justificar, en términos de qué problemas se han encontrado y qué medidas se han tomado para mitigar esos problemas.

Y si has llegado hasta aquí, siempre es bueno pasarle el corrector ortográfico, que las erratas quedan fatal en la memoria final. Para eso, en Linux tenemos aspell, que se ejecuta de la siguiente manera desde la línea de *shell*:

```
aspell --lang=es_ES -c memoria.tex
```

6.2. Aplicación de lo aprendido

Aquí viene lo que has aprendido durante el Grado/Máster y que has aplicado en el TFG/TFM. Una buena idea es poner las asignaturas más relacionadas y comentar en un párrafo los conocimientos y habilidades puestos en práctica.

1. a

2. b

6.3. Lecciones aprendidas

Aquí viene lo que has aprendido en el Trabajo Fin de Grado.

1. dd
2. Aquí viene otro.

6.4. Trabajos futuros

Ningún proyecto ni software se termina, así que aquí vienen ideas y funcionalidades que estaría bien tener implementadas en el futuro.

Es un apartado que sirve para dar ideas de cara a futuros TFGs/TFMs.

Apéndice A

Manual de usuario

Esto es un apéndice. Si has creado una aplicación, siempre viene bien tener un manual de usuario. Pues ponlo aquí.

Bibliografía

[1] Artículo sobre cantidad de datos.
https://es.statista.com/grafico/17734/cantidad-real-y-prevista-de-datos-generados-en-todo-el-mundo/.

[2] Características y ventajas de la arquitectura mvc.
https://marketiweb.com/empresa/blog/item/114-que-es-la-arquitectura-mvc-y-cuales-son-sus-ventajas.

[3] Página de Angular.
https://angular.io.

[4] Página de TensorFloe.
https://www.tensorflow.org/.

[5] Página oficial de JSON.
http://www.json.org/json-es.html.

[6] Página oficial de Node.js.
https://nodejs.org/es/.

[7] Página sobre HTML.
https://www.w3schools.com/html/html5_intro.asp.

[8] Página sobre HTML.
https://html.spec.whatwg.org/multipage/.

[9] Páina oficial de Typescript.
https://www.typescriptlang.org/.

[10] Máquinas de Soporte Vectorial, Clasificador Naive Bayes y Algoritmos Genéticos para la Predicción de Riesgos Psicosociales en Docentes de Colegios Públicos Colombianos. *Información Tecnológica*, 29:153 – 162, 12 2018.

[11] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[12] M. J. Islam, Q. J. Wu, M. Ahmadi, and M. A. Sid-Ahmed. Investigating the performance of naive-bayes classifiers and k-nearest neighbor classifiers. In *2007 International Conference on Convergence Information Technology (ICCIT 2007)*, pages 1541–1546. IEEE, 2007.

[13] S. Marsland. *Machine Learning: An Algorithmic Perspective*. CRC Press, 2015.

[14] P. Sinha and P. Sinha. Comparative study of chronic kidney disease prediction using knn and svm. *International Journal of Engineering Research and Technology*, 4(12):608–12, 2015.