MyShopOpener

Eric Jaén Marco

Resumen— La apertura y cierre de establecimientos es una de las tareas más comunes y repetitivas con que se encuentran los comerciantes. El tiempo empleado, el esfuerzo y la falta de automatización de este proceso hacen que sea un aspecto del trabajo en el que se requiere más experiencia. MyShopOpener es una aplicación Android con el objetivo de facilitar el trabajo y la eficacia en la apertura y cierre de negocios. Para hacerlo posible la aplicación trabaja conjuntamente con una placa programable Arduino, que se encarga de controlar los pulsadores de las persianas. Además emplea un algoritmo de *machine learning* para adaptarse al usuario experto, mejorando así el tiempo empleado en la apertura y cierre del local y permitiendo a usuarios no expertos en estos procedimientos un aprendizaje y adaptación progresivos. MyShopOpener demuestra que una aplicación de este tipo de algoritmos puede significar un cambio en las interacciones que hacemos actualmente con cualquier tipo de dispositivo, ya que, al adaptarse al usuario, puede abrir campos que hasta ahora no han sido explorados.

Palabras clave—Android, Random Forest, Naive Bayes, Arduino, Somfy, IHM (Interfaz Hombre-maquina), Machine Learning, Weka, inteligencia artificial, Android Studio.

Abstract— Opening and closing of establishments is one of the most common and repetitive task that the owners could have. The time spent, effort and lack of automation of this process makes it an aspect of the job that requires more experience. MyShopOpener is an Android application which facilitates the opening and closing of business security shutters. To make this possible the application works in conjunction with an Arduino programmable motherboard which is responsible for operating the shutter control buttons. Also, the application uses an algorithm for *machine learning* which adapts to the expert user, thereby reducing the time spent on the opening and closing of the business and allows all required personnel to use the system without prior experience. Finally MyShopOpener aims to demonstrate that the application of such algorithms can change the way we interact with different devices, because the user can adapt to open fields that until now have not been explored.

Index Terms— Android, Random Forest, Naive Bayes, Arduino, Somfy, HMI (Human-machine Interface), Machine Learning, Weka, artificial intelligence, Android Studio.

1 INTRODUCCIÓN

Antonio es un autónomo que tiene un pequeño comercio en Sabadell. Se trata de una joyería situada en una esquina la cual tiene cuatro persianas que ha de abrir (y cerrar) dos veces al día.

Por su sistema de candados (los cierres de seguridad de la persiana) para abrir la persiana tiene que descolocar cada candado (2 en 3 persianas y 3 en una más grande) individualmente y entrar en el negocio para subir un poco las mismas para después (finalmente) volver a subir las persianas hasta el final de su recorrido.

Para lo opuesto (cerrarlas) sólo hay una persiana en la que tiene que prestar atención una vez ha pulsado el botón de bajar, y es que esta se ha de centrar con el pivote del candado manualmente.

En total el tiempo empleado para la apertura total del negocio es de 8 minutos para el caso que nos ocupa, ya que como hemos comentado se tiene que entrar y salir del local en varias ocasiones por el hecho de que los pulsadores que controlan las mismas están dentro del local.

1.1. Problema

Como se ha comentado el procedimiento es muy tedioso, hace que para abrir una simple persiana se tenga que descolocar el candado, entrar en el local y subir la persiana unos centímetros (subir y pararla en un periodo corto), volver a salir para recolocar el candado (esta vez ya separado del pivote) y entrar nuevamente para que (¡por fin!) se suba la persiana hasta el final de su recorrido.

Esto supone un gran esfuerzo físico y una pérdida de tiempo a la hora de abrir las persianas y por ende el local. Además como hemos visto no es un procedimiento que cualquier nuevo trabajador del negocio sepa cómo hacerlo de primeras de la manera correcta sin obtener una formación previa.

También es un problema que se puede extrapolar a muchos otros pequeños comercios, incluso fábricas que posean una gran cantidad de persianas por lo que su aplicación podría ser muy extensa.

1.2. Objetivos

Los objetivos a conseguir para la solución de este problema pasarían por la reducción de tiempo empleado en todo el proceso así como la mejora del confort del usuario a la hora de la realización del mismo.

También se debería poder reducir la necesidad de alguien experto (con experiencia) para la realización del proceso y así que con un proceso definido, cualquiera pueda hacerse cargo del proceso y realizarlo de la misma manera que el la persona con más conocimiento sobre el mismo.

Todo esto conllevaría una mayor facilitación del proceso de apertura y cierre del local ya que el tiempo empleado sería mucho menor y no siempre lo tendría que hacer la misma persona, ya que según la necesidad podrían combinarse el proceso.

El coste de la solución tiene que ser lo suficientemente competitivo para que su amortización sea rápida, ya que esre proyecto no mejora rendimientos económicos sino de tiempo, por lo que el coste tiene que ser reducido para así suscitar mayor interés en la implantación y poder accer a un nicho de mercado mucho mayor.

1.3. Requerimientos del proyecto (HW/SW)

1.3.1. Hardware

Como placa programable disponemos de una Arduino Mega 2560 la cual tiene un coste de mercado de 30€, en conjunto con ella tendremos una placa Shield Ethernet que ayudara a realizar la conexión con la aplicación Android a través de Internet.

Para el dispositivo móvil disponemos de un LG G4 que tendrá el sistema operativo Android 4.4.1.

1.3.2. Software

Para trabajar con la placa (Arduino, 2016) se utilizara el software propio de nuestra placa que es compatible con PC y Mac.

Conjuntamente para el desarrollo de la aplicación móvil se hará uso del programa (Android Studio , 2016), ya que es la herramienta oficial de Android y es gratuita.

1.4. Estructura del documento

El resto del documento se organiza de la siguiente manera, comenzaremos con el estado del arte para la apertura y cierre de persianas y/o ventanas.

A continuación se hablará de la prueba de concepto, la cual es la base del proyecto en el que se puede observar si el proyecto es posible o no.

Seguidamente se explicará el funcionamiento de la aplicación y todos los mecanismos utilizados para llevar a cabo el proyecto así como el porqué la elección del algoritmo de aprendizaje.

Finalmente se harán las conclusiones del proyecto incluyendo las posibles líneas futuras del mismo.

- *E-mail de contacto: ejmarco@gmail.com*
- *Mención realitzada: Computación Trabajo Tutorizado por: Lluís Ribas Xirgo*
- *Curso 2015/16*

2 ESTADO DEL ARTE

Para el estado del arte comprovaremos 2 maneras distintas de llegar a la consecución del proyecto, una es comprar una solución ya establecida en el mercado y la otra será desarrollar nuestra propia solución, además de ver las ventajas y desventajas de cada una de ellas.

2.1 Comprar una solución

Para la resolución de estos problemas y la consecución de los objetivos se podría aplicar algún tipo de programa o solución ya realizado por otras empresas y ver que ventajas y desventajas nos conllevaría.

Tras realizar una búsqueda por internet la mejor solución para el problema que nos concierne es Somfy. Ésta es una empresa que se dedica a la instalación de persianas, puertas de garaje, toldos eléctricos etc. que pueden ser controlados desde un mando a distancia.

2.1.1 Pros

Es un sistema que ya está creado y que tiene una gran empresa con muchos años de experiencia detrás del mismo, con lo que se obtiene un sistema fiable, duradero y de calidad.

Además tienen una gran variedad de aplicaciones para diferentes tipos de dispositivos (móviles, tabletas, ordenadores, etc.).

Según los cálculos realizados con el siguiente documento (Somfy, 2012), el coste total para el negocio con el que trabajamos sería superior a los 800€.

2.1.2 Contras

El que haya una gran empresa detrás también implica un coste mucho más elevado, por lo que la amortización del producto sería más costosa y haría replantearse la implantación del mismo en pequeños comercios, ya que el presupuesto de este tipo de locales suele ser limitado.

Además la solución no es específica para persianas de aluminio/metal que son las que comúnmente se usan en los establecimientos, con el añadido de que esta solución no mejoraría el problema de que hace falta una persona experta para realizar la apertura/cierre del local.

2.2 Desarrollar una solución

Otra posible resolución para el problema que nos ocupa sería el de desarrollar nuestra propia solucion, que aparte

de solucionar un problema específico podría servir como solución para problemáticas similares.

El objetivo de MyShopOpener es proporcionar una solución que se adapte y solucione el problema que nos ocupa, y aprovechar los esfuerzos para que pueda servir de prueba de concepto para otro producto que pueda tener un mercado mucho mayor del que inicialmente está previsto.

2.2.1 Pros

Las ventajas que podríamos obtener de desarrollar una solución propia es que ésta se adapte perfectamente a nuestros requisitos, por lo que el problema se puede resolver de la manera que creamos más conveniente.

Esto hace que no estemos limitados por requisitos del fabricante o productos que no cumplan todos nuestros requisitos.

Además éste producto podría adaptarse a otros problemas similares y que pueda ser de ayuda a gente que esté interesada en nuestra solución.

Por otra parte, el mercado nos ofrece una gran cantidad de dispositivos programables que tienen un coste muy reducido y que se adaptan a cualquier tipo de producto, ya que tienen una gran comunidad que ayuda al desarrollo de aplicaciones para las mismas. Esto es por ejemplo las placas programables Arduino, Raspberry Pi, Picaxe para controlar la electrónica de las persianas, y Android como Sistema operativo de móvil para controlar éstas.

Finalmente el coste de ésta sería mucho menor que el comprar una ya desarrollada.

2.2.2 Contras

Por contrapartida tenemos la inversión de tiempo que se ha de aplicar para crear una solución sin fallos y totalmente funcional, ya que como se ha comentado anteriormente comprar una solución ya creada hace que no se haya de invertir tiempo en el desarrollo del mismo.

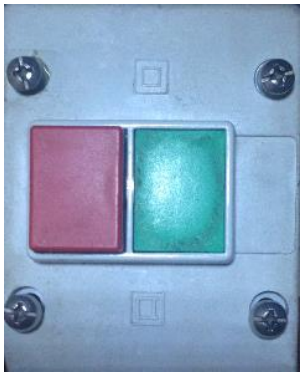
Por otra parte tampoco tendríamos el soporte necesario para que resuelva cualquier incidencia con rapidez y eficacia.

3 PRUEBA DE CONCEPTO

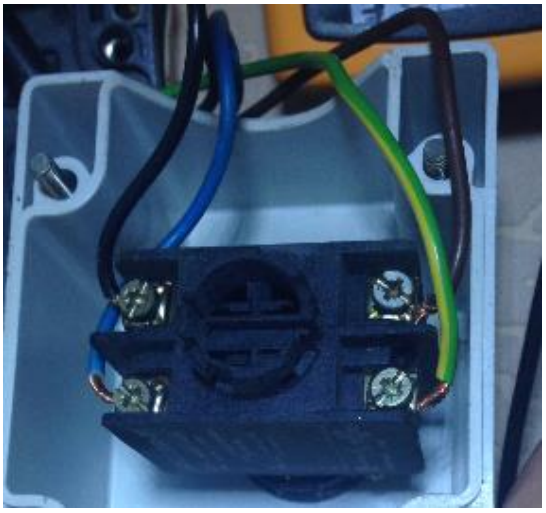
Antes de empezar el proyecto primero hemos hecho una prueba de que este sistema puede ser viable y funcional, por lo que se ha comprobado en primera estancia es que se puede hacer la pulsación sin necesidad de pulsar el botón específico para esa tarea.

3.1 Pulsador

A continuación mostramos el pulsador asociado a una persiana en el que se han realizado las pruebas. Éste controlador de la persiana tiene 2 botones, el verde que es el encargado de subir o bajar la persiana y el rojo que es el encargado de parar la persiana en cualquier punto en el que se encuentre la misma.



Seguidamente mostraremos el conexionado interior explicando la función de cada uno de ellos dentro del esquema de conexionado.



Como vemos en la imagen superior hay 4 cables, verde/amarillo, azul, marron y negro. Con la ayuda de un multímetro averiguamos que el cable verde/amarillo es tierra, mientras que si uníamos el cable negro con el azul era la pulsación del botón verde, y si uníamos el negro con el marron hacíamos el rojo.

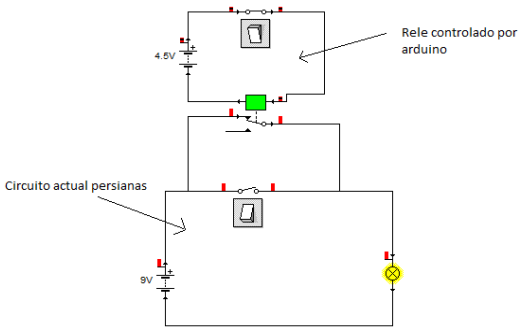
Con esta información se instalaron dos cables más, uno en la ranura de conexión del cable negro y otra del azul y se comprobó que al unirlos se hacía la misma acción que si se hubiese pulsado el botón verde.

3.2 Prueba desde Arduino

Una vez comprobado que se podía suplantar la pulsación del botón simplemente al unir dos cables que cerraran el circuito del cable negro y azul se realizó una prueba intermedia desde Arduino.

Esta prueba consistía en un relé, que es un dispositivo electromagnético que funciona como si fuese un interruptor controlado por un circuito electrio, que al recibir un corriente permite abrir o cerrar otros circuitos eléctricos independientes.

A continuación mostramos el conexionado que se realizó para hacer la prueba con el relé y que simplemente activando el pin que estaba conectado al relé desde la placa se consiguió hacer la pulsación desde Arduino.



En la parte superior vemos un rele en otro circuito aparte que lo que hace es "puentear" el pulsador, éste circuito estara controlado por arduino que activara o desactivara el circuito para que podamos obtener el mismo resultado pero desde el telefono.

Como vemos cuando al rele se le aplica un voltaje abre un circuito que este conectara el motor de la persiana (en el dibujo la bombilla) con el voltaje haciendo así un circuito cerrado.

3.3 Aplicación Android

Por último y para realizar la prueba definitiva se desarrolló una pequeña aplicación Android que envíaba una petición HTTP del tipo GET a la placa Arduino.

Cuando ésta recibia la petición activará o desactivará el pin que controla el relé, al hacer esto se provoca el cierre del circuito que junto lo que se ha explicado anteriormente hace que, se pueda suplantar la pulsación desde el teléfono móvil.

4 MyShopOpener

4.1 INTRODUCCIÓN

En el siguiente apartado se explica cómo se pretende llevar a cabo el proyecto final. Éste se hará como se ha explicado anteriormente empleando una placa programable para el control de los pulsadores de las persianas y una aplicación móvil que controle las mismas.

El principal reclamo de nuestro planteamiento de nuestro proyecto y función principal de la aplicación es la adaptabilidad del sistema al usuario, esto es que el sistema aprenda el uso que cada usuario hace de la aplicación para:

1- Facilitar el uso adaptando la interfaz a la acción que espera realizar el usuario.
2- Permitir que un usuario no experto pueda realizar las acciones de igual manera (y sin conocimiento previo) en la que lo haría uno con conocimientos.
3- Demostrar que los algoritmos de *machine learning* pueden ser perfectamente aplicables a tareas que realizamos día a día con cualquier tipo de dispositivo.

3.2 Placa Arduino

El funcionamiento principal que se espera de la placa es que trabaje como servidor, para que reciba las peticiones de tipo GET para poder controlar los pines de la misma desde la aplicación Android.

En la petición se recibirá el numero de pin que se tendrá que activar, que a su vez abrirá/cerrará el circircuito del pulsador para suplantar el movimiento de presionar el botón.

3.3 Aplicación Android

Como hemos comentado anteriormente el principal objetivo de la aplicación es que se adapte al usuario. Por lo que la dividiremos en módulos para así explicar mejor el funcionamiento esperado de la aplicación.

3.3.1 Interfaz

Para la aplicación se espera una interfaz que sea intuitiva y que se adapte al uso que tendrá, que es el de substituir a los pulsadores físicos por botones en la aplicación. Para esto se esperan tener 2 pestañas:

- La primera pestaña será en la que se mostrará un histórico de todos los dispositivos introducidos (con acceso a los pulsadores), en el que vea cada

local que tenga a su disposición y todas las persianas del mismo.
- La segunda pestaña será la que proporcionara la adaptabilidad al usuario, esto es que mostrará una lista de todos los botones y ésta se ira modificando dependiendo del momento, localización, botones anteriores pulsados, para que quien utilice la aplicación tenga en primera instancia el botón que quiere pulsar.

3.3.2 Trabajando con el servidor

Como hemos dicho, esta aplicación mandara peticiones al servidor Arduino para que active/desactive pines de la placa, esto es para simular el movimiento de pulsar un botón.

Para llevarlo a cabo se implementara una clase en la que envíe peticiones GET (que el servidor escuchará) en las que cada botón (de persiana) pulsado sobre la aplicación, implique un pin sobre la placa.

Ésta deberá ser una clase Asíncrona para que en caso de error no implique un mal funcionamiento de la aplicación principal.

Para ello dispondremos de la ayuda del sitio web de (Arduino, s.f.).

3.4 Aprendizaje de la Aplicación

Esta parte es el pilar de la aplicación con lo cual es la que más atención se le deberá prestar. Para ello primero se realizará un estudio de distintos tipos de algoritmos de aprendizaje para implantarlos en la aplicación y que éstos ayuden a predecir los movimientos futuros que el usuario realizará.

Para el estudio se trabaja sobre dos grandes tipos de algoritmos como son (Random Forest, 2016) y (Naive Bayes, 2016), éstos son algoritmos de aprendizaje supervisado.

Que sean supervisados implican que primero se debe entrenar a los algoritmos con unos casos de ejemplos ficticios para que una vez entrenados puedan trabajar con normalidad y que su porcentaje de aciertos sea el más elevado posible.

3.4.1 Random Forest

Random forest es un algoritmo que usa una combinación de árboles de decisión en el que cada árbol depende de unos datos probados independientemente.

Explicado de otra manera, una vez tenemos un tamaño de datos observados suficientemente grandes, se dividen estos datos en pequeños grupos aleatorios y con la misma distribución para cada uno de éstos.

Lo que obtenemos por cada subgrupo del Dataset (número total de casos observados) es un árbol que respecto a unos datos de entrada nos predecirá una clase observada que nosotros indiquemos.

Random Forest es la unión de todos estos árboles, si nuestro algoritmo ha creado 11 árboles independientes y 5 dicen que respecto a un valor de entrada la clase predicha es A y los 6 restantes dicen que es B, la clase predicha por el algoritmo será B.

3.4.2 Naive Bayes

Éste es un clasificador probabilístico fundamentado en el teorema de Bayes que asume que la presencia o ausencia de una característica particular no está relacionada con la presencia o ausencia de cualquier otra característica dada una clase variable (por la que se clasificara).

En otras palabras es un clasificador que funciona por probabilidades dado un Dataset en el que contra más frecuente sea una variable más probabilidades tiene de serlo sin tener en cuenta sus características.

Como ejemplo para ayudar a entenderlo podríamos decir que una fruta puede ser considerada como una manzana si es roja, redonda y de alrededor de 7 cm de diámetro. Para este caso Naive Bayes considera que cada una de estas características contribuye de manera independiente a la probabilidad de que esta fruta sea una manzana, de manera independiente a la presencia o ausencia de otras características.

Por lo que si tenemos en nuestro Dataset 7 casos con diámetro 7 y 5 son manzanas y 2 son peras, el resultado será manzanas.

3.4.3 Elección de método

En nuestro caso hemos decidido escoger Random Forest, ya que nos proporcionará un resultado mucho más preciso al no tener en cuenta solo el número de resultados (entendemos como resultado el número de instancias clase por las que vamos a clasificar en nuestro dataset) sinó que también tiene en cuenta los atributos o características del mismo.

Ya que por ejemplo con Naive Bayes un comportamiento del usuario aleatorio y anormal puede reducir considera-

blemente la precisión de clasificación del algoritmo, mientras que con Random Forest este comportamiento anómalo no tendría apenas consecuencias a no ser que fuera algo común (con lo cual se adaptaría a ese comportamiento perfectamente).

Es por esto por lo que obtenemos mejores resultados con Random Forest, ya que todas las acciones realizas en nuestra aplicación se hacen dadas unas características (features) especificas que se irán repitiendo a lo largo del tiempo, por lo que será mas fácil encontrar un patrón consiguiendo así una alta probabilidad de acierto en las predicciones.

3.4.4 Trabajando con Random Forest

Para nuestra aplicación cada instancia o valor del Dataset tendrá una serie de características que nos permitirá clasificar y predecir mejor el siguiente movimiento del usuario.

Cada vez que el usuario pulse un botón se guardara una instancia con el día, hora, mes, año, día de la semana, los últimos botones pulsados, etc. y como atributo de clasificación el botón pulsado. Con esto lo que conseguimos es saber y predecir horarios de verano, horarios de apertura y cierre distintos (abrir y cerrar por la mañana y por la tarde) etc.

3.4.5 Random Forest con Weka

Una vez decidido el algoritmo de aprendizaje y planteado la manera en la que se trabajará sobre el mismo hemos de buscar la manera de implementarlo en nuestro sistema.

Como se ha comentado anteriormente la utilización de esta clase de algoritmos y su entrenamiento requieren una gran capacidad de cómputo y de información. Dado que estamos trabajando con un dispositivo móvil sabemos que los recursos son limitados y se han de buscar otras alternativas.

Una alternativa que se ha encontrado es (Weka, s.f.), una API centrada en algoritmos de aprendizaje para ordenadores, ofreciendo una versión muy completa con diferentes tipos de algoritmos y funciones para saber el rendimiento de estos.

Dado que Weka es una API centrada en muchos algoritmos de gran potencia hace que para nuestro proyecto no sea viable, ya que el peso de la versión java supera los 2 GB, por suerte tienen una versión reducida con las mismas funcionalidades adaptadas para dispositivos con Android 4.0 o superior por lo que hace que sea la solución ideal para nuestras necesidades.

Dentro de Weka encontramos clases especificas para cada tipo de algoritmo, nosotros trabajaremos con la clase especifica RandomForest que es la encargada de generar, mantener, personalizar y observar el rendimiento de este tipo de algoritmo además de proporcionarnos la clase Instance que nos permitirá la creación de eventos con sus resultados, características y observaciones que se usarán para entrenar al algoritmo para que mas tarde se puedan obtener el objetivo que buscamos, la predicción de pulsación respecto a cada tipo de usuario.

5 RESULTADOS

A continuación se detallarán los resultados obtenidos con el algoritmo de Random forest implementado usando WEKA.

5.1 Requisitos

Antes de comencar a testear el algoritmo y su comportamiento en la aplicación descubrimos que este algoritmo tiene unos requisitos para poder comenzar a utilizarlo, estos hacen que tengamos una seríe de valores que respetar para el correcto funcionamiento del mismo.

El número de “hojas” del árbol tiene que ser siempre menor o igual al número de instancias observadas de nuestro dataset. Esto obliga a que para obtener cualquier tipo de clasificación necesitamos hacer como mínimo 1 instancia o pulsación si lo relacionamos con nuestro proyecto.

Continuando con lo anterior puede parecer que con solo una instancia ya se puede obtener una clasificación ya que el número de instancias sería igual al número de hojas, pero weka tiene otro requisito, y es que el número de hojas ha de ser mayor que uno, por lo que esto implica que para obtener cualquier tipo de clasificación hemos de tener por lo menos 2 instancias para poder cumplir el requisito explicado en el párrafo anterior.

5.2 Rendimiento

En las pruebas realizadas con la api de Weka y más concretamente con el algoritmo Random Forest de Android se ha estudiado el rendimiento al entrenar y clasificar instancias con un dataset de mas de 1000 instancias.

Los resultados obtenidos respecto al número de instancias comentado anteriormente hacen que a partir de 30 árboles el rendimiento decaiga progresivamente. Los valores probados implican que a partir de 70 se vean esperas que podrían llegar a ser molestas, a partir de 120 hacen realmente molesta la navegación y superior a 400 el tiempo

de espera es mayor a 4 segundos por lo que es algo inviable si queremos cumplir con los requisitos.

Como conclusión tenemos que un número mayor a 50 haría inviable en términos de experiencia de usuario.

Respecto al número de hojas se ha observado que a partir de 200 el rendimiento decaía lo suficiente como para suponer un problema en la experiencia de usuario sin aumentar que esta aumente la probabilidad de acierto al clasificar una instancia concreta.

5.3 Valores de configuración Random Forest

A continuación se explicarán los valores de configuración óptimos en base a las observaciones de las estadísticas del algoritmo, modificando el número de instancias, número de hojas y número de árboles. Ya que para este proyecto es igual de importante la calidad de las predicciones del algoritmo como la experiencia de usuario al trabajar con éste.

Número de árboles

La base principal del algoritmo como se ha explicado anteriormente són el número de árboles, ya que cada uno estos realizará una predicción de un botón a pulsar. Es por esto que el número de árboles será un valor impar con el fin de evitar en la medida de lo posible el empate en las decisiones del algoritmo.

Dado lo anterior y observadas las predicciones con diferentes tipos de configuraciones se ha decidido que el número de árboles óptimo serían 23. Éste valor viene dado a que respecto a las pruebas con 2 locales de 4 persianas cada uno las predicciones se mantenían al aumentar el numero por encima de 23, por lo que mantenemos un numero reducido que tendrá un rendimiento adecuado a los requisitos esperados en la experiencia de usuario.

Número de hojas

Para las hojas y respecto a los mismos requistos anteriores se ha decido poner que el numero de hojas sea igual al número de instancias hasta 150, ya que se ha visto que es un valor en el que se obtiene un pronostico adecuado a los requisitos de predicción y no tiene un impacto notorio en el rendimiento de la aplicación al entrenar el algoritmo.

Instancias

Como se ha comentado anteriormente una instancia es un valor observado que tiene unas ciertas características, por ejemplo al pulsar el botón de abrir de la persiana 1 de un local concreto observamos distintas características, como el día de la semana, la hora, los minutos etc.

Dado lo anterior y con los requisitos comentandos anteriormente no se empezará a clasificar hasta que no se obtengan por lo menos 2 instancias, que es el mínimo establecido para poder crear un algoritmo de Random Forest.

Se ha establecido un límite de 1000 instancias, ya que es el número suficiente como para que no se vea comprometido el rendimiento, por lo que cada vez que se llegue a 1000 instancias el el árbol será guardado y se eliminarán las 800 primeras, para así no comprometer al algoritmo aplicándome un número mucho mayor de instancias.

5.4 Estadísticas

Después de todas las pruebas realizadas y los valores de configuración anteriormente comentados, conseguimos un porcentaje de acierto de un 90 % a las 3 iteraciones completas (mismos movimientos para todo un día).

Haciendo así un valor lo suficientemente viable para el usuario, que si el uso sigue siendo el mismo el porcentaje se ira incrementando conforme se incrementen las instancias e iteraciones.

Con ese porcentaje de acierto, la reordenación de la lista de botones respecto a lo que random forest predice hace de uso algo muy útil e intuitivo dado su alto nivel de acierto, demostrando así que un algoritmo de estas características en aplicaciones de índole similar puede suponer un gran cambio en la interacción que el usuario hace con su dispositivo en conjunto con las aplicaciones móviles.

6 CONCLUSIONES

En este trabajo de final de grado se ha explicado que es, en que consiste y como se aplica una aplicación inteligente para el control de persianas.

Se han estudiado diversas opciones para resolver una problematica concreta con vistas a solucionar casos múcho mayores, estas opciónes han sido estudiadas y observadas con sus pros y contras para sacar la conclusión de que la realización de una aplicación propia era la mejor manera de solucionar la problemática propuesta.

Como se ha comentado se ha desarrollado una aplicación en Android ya que es la opción que nos parecía más viable, más económica y más funcional haciendo así que en conjunto con una placa progamable Arduino y un algoritmo de *machine learning* se consigue no solo una aplicación que hace de controlador de los pulsadores, sinó que además adapta la interfaz al usuario para así facilitar el tiempo y el traspaso de conocimientos a gente no experta.

Por otra parte a mitad del desarrollo se observo la gran cantidad de aplicaciones que podrían tener este tipo de algoritmos para la interaccion con los usuarios, abriendo así un campo mucho más extenso y sirviendo este proyec-

to como muestra de que es posible aplicarlos eficazmente.

Es por todo lo comentado a lo largo del artículo que se ha demostrado que el aprender del usuario es algo posible y muy útil abriendo así un campo que todavía esta por explotar. Además de lo anterior es un sistema fácilmente exportable gracias a la gran cantidad de APIs externas de código libre que hay en internet haciendo su aplicación mucho más sencilla y con una variedad mucho mayor dadas las diferentes opciones de las que se disponen.

6.1 Conclusiones extraídas del proyecto

Se podría decir que este ha sido un proyecto que a primera vista puede no parecer una gran revolución como aplicación en sí, pero qué con la implantación de los algoritmos que hemos visto anteriormente puede suponer una revolución en la manera en la que interactuamos con los dispositivos.

El que una aplicación sepa lo que quieres hacer puede suponer ahorro de tiempo y calidad de trabajo para aplicaciones como venta de entradas, las PDAs de los camareros en restaurantes, información en aeropuetros, etc.

Todo en base al conocimiento previamente observado con el uso que cada usuario le de a la aplicación, consiguiendo así que una aplicación igual, al cabo de dos meses sea totalmente diferente entre usuarios.

A nivel personal los conocimientos obtenidos sobre algoritmos de Inteligencia Artificial, programación en Android y Arduino han incrementado notablemente respecto a la base con la que partía. Espero poder aprovecharles y seguir aumentándolos con la experiencia obtenida y conseguir aplicarlos en alguna aplicación real y funcional que pueda llegar a venderse.

6.2. Revisión de planificación y problemas encontrados

Respecto a la planificación inicial los objetivos se han cumplido totalmente, ya que se ha desarrollado en su totalidad el servidor, la aplicación, y el algoritmo del aprendizaje.

Por lo que la planificación inicial ha resultado ser la óptima ya que como comentaremos mas adelante ha habido varios errores, que de no ser por de la planificación inicial teniendo en cuenta estos imprevistos el proyecto se hubiese retrasado o no se hubiesen conseguido los objetivos iniciales del proyecto.

Como se ha comentado anteriormente son varios los problemas que se han encontrado, a nivel de interfaz con un funionamiento ánomalo, cierres inesperados de la aplicación, problemas con los datos guardados etcétera, sin duda el más importante y que más retraso nos ha ocasionado ha sido el algoritmo de aprendizaje, éste ha supuesto un reto por las dificultades que comentaremos a continuación.

Planteamiento: El principal desafio al que se puede enfrentar es la aplicación del algoritmo una vez elegido el tipo, las variables, las características o features etc. Por lo que es vital el realizar una buena planificación y un buen estudio de opciones de las que se disponen para evitar retrasos no previstoes en la realización de su propio proyecto con algoritmos de *machine learning*.

Como hemos hablado a lo largo del artículo, Weka es una API muy interesante que puede servir para infinidad de proyectos que necesiten algoritmos de aprendizaje, es muy versátil y fácil de usar por lo que es una de las mejores opciones que se han encontrado.

Es importante saber que la implantación de esta puede resultar un prolema si no se tienen todas las bases sobre la aplicación o sobre algoritmo de clasificación.

6.3. Líneas futuras

Pese a que la planificación ha sido la correcta y ha evitado retrasos que hubiesen supuesto la no consecución del proyecto, se podría haber potenciado aún más la interaccion con el algoritmo de IA, permitiendo así no sólo la reordenación de botones en una pestaña sinó de toda la aplicación.
Consiguiendo que al abrir la aplicación pudieras acceder a donde realmente querias, tener avisos y alarmas respecto a lo que la aplicación cree que quieres hacer.

Todo eso supondría un salto cualitativo muy elevado que mejoraría la experiencia de uso. El tiempo se podría ver aún más reducido haciendo, por ejemplo, que al conectarse a la red de un local que tenemos agregado y según la hora que saliera un mensaje para pulsar el botón que el algoritmo ha predicho que querríamos.

A parte también nos hubiese gustado el conseguir un registro por cuenta, haciendo así no solo el traspaso de conocimiento al dejar el móvil, sino pudiendo compartir los movimientos de apertura/cierre con un link, añadiendo más cuentas sobre la aplicación etc.

Todo esto no haría más que reafirmar nuestra opinión sobre el potencial de este tipo de algoritmos en aplicaciones móviles consiguiendo por ejemplo una aplicación de viajes en las que el planning se haga automáticamente respecto a gustos, gente parecida, lugares más visitados etc.

Agradecimientos

Ante todo dar las gracias a mi tutor Lluís Ribas, pese a que el proyecto no era del todo de su campo siempre ha intentado plantear puntos de vista distintos, exigiendo siempre el máximo y consiguiendo así un trabajo de mayor calidad del que se había considerado en un principio, modificando creencias sobre domótica y funcionamientos de algoritmos que antes tenía y que han ayudado múcho en la consecución de los objetivos.

Después quisiera mencionar a mi familia que siempre me ha apoyado y animado a tomar este camino de la universidad y a mi pareja, que aún habiendo tenido momentos de mucha duda respecto a éste camino en el que me embarqué hace 4 años, ha conseguido que llegue hasta el final sacando lo mejor de mi mismo y apoyándome en todo momento.

BIBLIOGRAFIA

Android Studio . (2016). *Developers Android*. Obtenido de http://developer.android.com/sdk/index.html

Arduino. (2016). *Arduino CC*. Obtenido de https://www.arduino.cc/

Arduino. (s.f.). *Forum Arduino*. Obtenido de https://forum.arduino.cc/

Naive Bayes. (2016). *Wikipedia*. Obtenido de https://en.wikipedia.org/wiki/Naive_Bayes_classifier

Random Forest. (2016). *Wikipedia*. Obtenido de https://en.wikipedia.org/wiki/Random_forest

Somfy. (Enero de 2012). *Somfy-profesional.* Obtenido de https://www.somfy-profesional.es/documents/1933827/0/tarifa_somfy_2012_es.pdf

Weka. (s.f.). *Weka Source Forge*. Obtenido de http://weka.sourceforge.net/doc.stable/

APENDICE

A1. CÓDIGO ARDUINO

```
#include <Ethernet.h>
#include <SPI.h>
boolean reading = false;
//////////////////////////////////////////////////////////////////////
//Valores de configuracion
//////////////////////////////////////////////////////////////////////
 //byte ip[] = { 192, 168, 1, 199 };   //Manual setup only
 //byte gateway[] = { 192, 168, 0, 1 }; //Manual setup only
 //byte subnet[] = { 255, 255, 255, 0 }; //Manual setup only
 // if need to change the MAC address (Very Rare)
  byte mac[] = {
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED
};
IPAddress ip(192, 168, 1, 177);
 EthernetServer server = EthernetServer(80); //port 80
 boolean state[8] = {false,false,false,false,false,false,false,false};
//////////////////////////////////////////////////////////////////////
void setup(){
  Serial.begin(9600);
  //Pins 10,11,12 & 13 are used by the ethernet shield
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  //Ethernet.begin(mac);
  Ethernet.begin(mac, ip);
  //Ethernet.begin(mac, ip, gateway, subnet); //for manual setup
  server.begin();
  Serial.print("server is at ");
  Serial.println(Ethernet.localIP());
}
void loop(){
  // listen for incoming clients, and process qequest.
  checkForClient();
}
```

```
void checkForClient(){
  EthernetClient client = server.available();
  if (client) {
    // an http request ends with a blank line
    boolean currentLineIsBlank = true;
    boolean sentHeader = false;
    while (client.connected()) {
      if (client.available()) {
        if(!sentHeader){ //Lectura de la petición HTLP del tipo GET
          // send a standard http response header
          client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: text/html");
          client.println();
          sentHeader = true;
        }
        char c = client.read();
        if(reading && c == ' ') reading = false;
        if(c == '?') reading = true; //found the ?, begin reading the info

        if(reading){
          Serial.print(c);

           switch (c) {
            case '2':
              //add code here to trigger on 2
              triggerPin(2, client);
              break;
            case '3':
            //add code here to trigger on 3
              triggerPin(3, client);
              break;
            case '4':
            //add code here to trigger on 4
              triggerPin(4, client);
              break;
            case '5':
            //add code here to trigger on 5
              triggerPin(5, client);
              break;
            case '6':
            //add code here to trigger on 6
              triggerPin(6, client);
              break;
            case '7':
            //add code here to trigger on 7
              triggerPin(7, client);
              break;
            case '8':
            //add code here to trigger on 8
              triggerPin(8, client);
              break;
            case '9':
            //add code here to trigger on 9
              triggerPin(9, client);
              break;
          }

        }
        if (c == '\n' && currentLineIsBlank)  break;

        if (c == '\n') {
          currentLineIsBlank = true;
        }else if (c != '\r') {
          currentLineIsBlank = false;
        }
      }
    }

    delay(1); // give the web browser time to receive the data
    client.stop(); // close the connection:
  }

}
void triggerPin(int pin, EthernetClient client){
  if(state[pin-2] == false){ //Si el led estaba apagado lo encendemos
    client.print("ON");
    state[pin-2] = true;
    digitalWrite(pin, HIGH);
  }
  else{
    client.print("OFF"); //Si el led esta encendido lo apagamos
     state[pin-2] = false;
    digitalWrite(pin, LOW);
  }
  delay(25);
}
```