UNIVERSITAT DE BARCELONA

Trabajo final de grado

GRADO EN INGENIERÍA INFORMÁTICA

Facultad de Matemáticas
Universidad de Barcelona

PREDICCIÓN DE VISITAS MEDIANTE GEOLOCALIZACIÓN A TRAVÉS DE DISPOSITIVOS MÓVILES

Autor: Andrés Vara Serrano

Director:	Pablo Almajano
Realitzado en:	Departamento de Matemáticas e Informática
Barcelona,	22 de junio de 2017

Abstract

Knowledge is information. Current technological devices are equipped with many sensors capable of gathering different kinds of data. Nowadays the use of mobile phones is widespread and it has become an easy way to obtain information as well as other kinds of help. However, these sensors obtain big amounts of data, which arises a problem when it comes to filtering it. In order to work with this set of data, Machine Learning algorithms are used to process and extract the proper information.

This information might come handy for the user in many ways. For instance, it could offer him weather details on his area, news related to his most frequented places or offer him advices to promote healthier living habits; improving in this way his day to day by providing small doses of information.

This work is focused on the development of a system capable of detecting a possible personal routine by means of a person's use of his mobile phone. The main purpose is to be able to locate this user in a specific place at a given time. For that, proper algorithms have been developed in order to process the data obtained through the gps sensor of a mobile device.

Resumen

El conocimiento es información. Los dispositivos tecnológicos actuales disponen de muchos sensores capaces de reunir información. Actualmente, está muy extendido el uso de dispositivos móviles y son una manera fácil de obtener información y proporcionar ayuda a la gente.

El problema que surge es que estos sensores proporcionan muchos datos, difíciles de procesar para sacar información. Para trabajar con este conjunto de datos se utilizan mecanismos de Machine Learning con los que se procesan y se extrae la información deseada.

Esta información puede ser muy útil para ofrecer a una persona datos de su interés como la situación meteorológica y las noticias cercanas a sus lugares más frecuentados u ofrecer asesoramiento para favorecer hábitos de vida más saludables o coordinación familiar. Pudiendo así, mejorar el día a día con pequeñas dosis de información.

Este trabajo de final de grado se ha centrado en el diseño de un sistema que consiga detectar rutinas personales a través de dispositivos móviles. El propósito es poder situar a un usuario en un lugar concreto en un momento concreto. Durante

el proyecto, se han desarrollado los algoritmos necesarios para procesar los datos obtenidos a través del sensor gps de un dispositivo móvil y tratarlos debidamente para extraer información adicional.

Resum

El coneixement és informació. Els dispositius tecnològics actuals disposen de molts sensors capaços de reunir informació. Actualment, està molt estès l'ús de dispositius mòbils i són una forma fàcil d'obtenir informació i proporcionar ajuda a la gent.

El problema que sorgeix és que aquests sensors proporcionen moltes dades, difícils de processar per treure informació. Per treballar amb aquest conjunt de dades s'utilitzen mecanismes de Machine Learning amb els que es processen i s'extreu la informació desitjada.

Aquesta informació pot ser d'utilitat per oferir a una persona dades del seu interès com la situació meteorològica i les notícies properes als seus llocs més freqüentats o oferir assessorament per afavorir una vida més saludable o coordinació familiar. D'aquesta manera, millorar el dia a dia amb petites dosis d'informació

Aquest treball de final de grau s'ha centrat en el disseny d'un sistema que aconsegueixi detectar rutines personals mitjançant dispositius mòbils. El propòsit es poder situar a un usuari en un lloc concret en un moment concret. Durant el projecte, s'han desenvolupat els algoritmes necessaris per processar les dades obtingudes a través del sensor gps d'un dispositiu mòbil i tractar-les degudament per treure informació addicional.

Índice

Índice de figuras

1. Introducción

Las tecnologías avanzan muy rápido y, con el tiempo, el uso de dispositivos móviles es más común entre las personas. El fin principal de un dispositivo electrónico es ayudar al usuario y proporcionarle información que le pueda ser de utilidad. Para ello, es necesario tener información sobre el usuario aunque, por comodidad, una persona no introduce constantemente datos sobre su rutina o sus gustos.

La motivación principal de este proyecto es poder reunir datos sobre una persona sin su supervisión. De esta manera, obtener información para conocer mejor al usuario y poder ofrecerle información que le sea de utilidad.

El objetivo se basa en el proceso de obtención de datos para conocer a una persona. Esta información se podría traspasar a otros proyectos que proporcionan información. Así, esta información se ajustaría mejor a las necesidades de cada usuario. También permitiría agrupar a los usuarios para ofrecer información que sea de utilidad a personas con un día a día semejante.

Existen diversas aplicaciones que utilizan la ubicación de los dispositivos móviles para informar al usuario de cierta información. Por ejemplo, la aplicación *El tiempo de AEMET* proporciona información oficial de la Agencia Estatal de Meteorología sobre la zona donde se encuentra el dispositivo. Una aplicación como *Google Maps*, proporciona información sobre el tráfico de las principales carreteras cercanas frecuentadas por el usuario.*Drivies* es una aplicación diseñada por *Telefónica* que aprovecha los sensores de los teléfonos móviles para analizar los hábitos de conducción y ofrecer información a los usuarios para mejorar su conducción.

Estructura de la Memoria

- Sección 2. Se da una visión general del entorno en el que se desarrolla el proyecto.
- Sección 3. Se muestra el conjunto de objetivos que se quieren alcanzar con el desarrollo del proyecto.
- Sección 4. Explicación del diseño del sistema de predicción propuesto.
- Sección 5. Se explican las implementaciones realizadas.
- Sección 6 Se explican las pruebas realizadas y los resultados obtenidos.
- Sección 7. Se explican las conclusiones sobre el proyecto y el trabajo futuro con el que se puede continuar.
- Sección 8. Se muestran los recursos bibliográficos utilizados durante el desarrollo del proyecto.
- Anexos. Se expone un manual de usuario con las instrucciones detalladas para la instalación y uso de los componentes implementados del proyecto.

2. Background

La parte más importante del proyecto se ha centrado en la implementación de algoritmos de Machine Learning [1] para el procesamiento de los datos obtenidos a través de una aplicación en un dispositivo móvil.

Machine Learning es una rama dentro del campo de la inteligencia artificial que se centra en desarrollar técnicas para que los dispositivos sean capaces de aprender y reconocer patrones para tomar decisiones mediante el análisis de datos.

Los datos recopilados suelen ser un conjunto muy grande además de difíciles de comprender a primera vista. Machine Learning ofrece herramientas para tranformar estos datos en un conjunto más sencillo y mejor estructurado, facilitando su procesamiento y la adaptación a los recursos con los que se trabaja.

Dependiendo de los datos utilizados, se pueden diferenciar dos ámbitos: Supervised o Unsupervised Learning [2]. En este proyecto se ha trabajado con datos no supervisados, es decir, los datos empleados no son etiquetados por el usuario. Se han obtenido datos de geolocalización utilizando una aplicación móvil que reunía datos a través del sensor gps del dispositivo. Se ha utilizado una herramienta de clustering para agrupar y etiquetar los datos con una similitud relevante. Inicialmente, para agrupar los datos se utilizó un algoritmo *K-Means* [3]. Se modificó, creando un proceso iterativo, para adaptar la herramienta al problema concreto del proyecto. Más adelante, se pasó a utilizar *Agglomerative Clustering* [4] ya que los recursos que proporciona se adaptan mejor al problema, proporcionando un mejor resultado y reduciendo el tiempo empleado.

Una vez etiquetados los datos, se han clasificado mediante un árbol de decisión [5]. Un árbol de decisión es un herramienta de aprendizaje inductivo donde se parte del análisis de un conjunto de datos extenso para dar un valor a un conjunto de datos concreto.

3. Objetivos generales

El objetivo de este proyecto ha sido crear una aplicación capaz de detectar los lugares visitados habitualmente por una persona y predecir cual será el siguiente al cual acudirá.

3.1. Objetivos específicos

Se puede separar en objetivos más concretos el objetivo del proyecto:

1. Diseño e implementación de una aplicación Android que detecte visitas a través del sensor gps.
    - Diseño gráfico de la aplicación
    - Detección de la geolocalización
    - Detección de visitas
2. Implementación de una base de datos.
    - Diseño de la base de datos
3. Implementación de un algoritmo de predicción de visitas.
    - Conexión con la base de datos
    - Implementación de un proceso de extracción de características
    - Implementación de un predictor de visitas
4. Testeo.
    - Testeo de la aplicación android con datos personales
    - Testeo del predictor con datos de usuario reales

4. Diseño del sistema

En este proyecto se ha propuesto el diseño de un sistema capaz obtener información de un usuario a través de su dispositivo móvil y predecir a donde acudirá próximamente.

4.1. Arquitectura

El sistema se divide en tres componentes: una aplicación móvil, un servidor y una base de datos. La aplicación móvil se encarga de coleccionar y procesar datos gps para sacar información de los lugares que frecuenta el usuario. El servidor recibe esta información y la procesa. Agrupa los lugares cercanos que el usuario frecuenta y extrae información adicional, que utiliza para clasificar los datos y poder predecir el siguiente lugar al que acudirá. Esta predicción se devuelve al dispositivo móvil, con el fin de utilizarla para ofrecer algún servicio al usuario. Durante este proceso, toda la información se guarda en una base de datos para reutilizarla cuando es necesario. Según la aplicación y el servidor van obteniendo más datos, la base de datos se actualiza con la nueva información procesada.

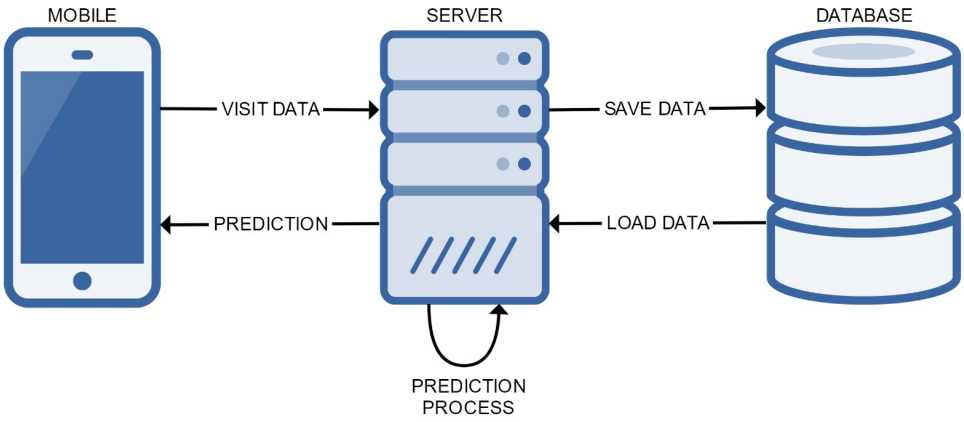

Figura 1: Arquitectura del sistema propuesto

Durante el proyecto se ha desarrollado una aplicación móvil, para sistemas operativos Android, que se encarga de rastrear datos gps para calcular los lugares frecuentados por el usuario. También se ha desarrollado un proceso de extracción de características y un modelo de clasificación, en lenguaje de programación Python, para clasificar los datos procesados y predecir la información deseada.

4.2. Casos de uso

En este apartado se describen los casos de uso de la aplicación móvil implementada.

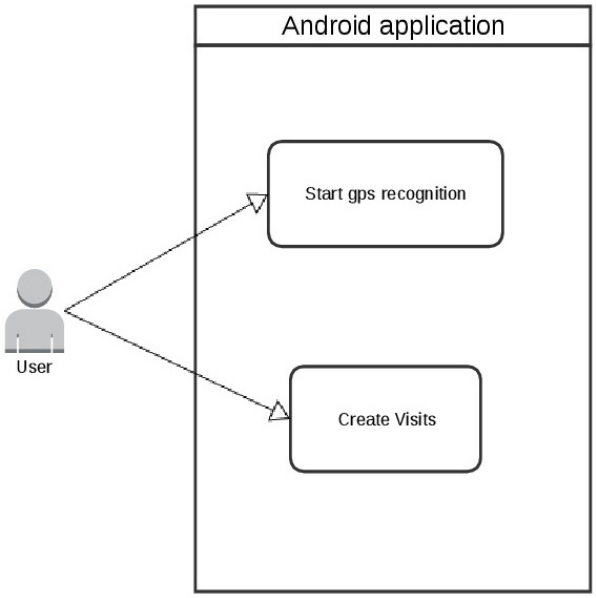

Figura 2: Diagrama de casos de uso de la aplicación Android.

Nombre:	Start gps recognition
Actor:	Usuario
Descripción:	Iniciar el rastreo de las coordenadas de la ubicación del dispositivo móvil.
Precondiciones:	Debe estar activado el sensor gps del dispositivo móvil.
Flujo normal:	1. El usuario clica el botón *Start*
Flujo alternativo:	
Postcondiciones	

Nombre:	Create Visits
Actor:	Usuario
Descripción:	Procesa la información de los datos gps para extraer la información de las visitas generadas por el usuario.
Precondiciones:	El dispositivo debe haber guardado datos gps.
Flujo normal:	1. El usuario debe clicar el botón Visits 2. Se comprueba si hay datos gps guardados en la memoria interna del dispositivo. 3. Se cargan los datos gps guardados y se precesan para extraer la información de las visitas. 4. Se guardan los datos de las visitas en la memoria interna del dispositivo móvil
Flujo alternativo:	
Postcondiciones	

5. Implementación

En esta sección analizaremos el software necesario que ha sido utilizado para desarrollar el proyecto y las diferentes partes implementadas de las que se compone.

5.1. Requisitos de software

Para llevar a cabo el proyecto ha sido necesario el uso de diferentes tecnologías. A continuación se expone el software utilizado durante el transcurso del proyecto:

Virtual Environment

Virtualenv es una herramienta de desarrollo en Python usada para crear entornos aislados para Python. En estos entornos aislados es posible instalar y utilizar cualquier librería o versión de python y evitar que interfieran con las versiones instaladas en el sistema y los problemas que pueden surgir al contener diferentes versiones de un mismo software.

Por ejemplo, si es necesario tener instaladas dos versiones diferentes de Django para dos proyectos distintos, no es posible tener las dos versiones instaladas en el directorio de bibliotecas de Python del sistema. Utilizando virtualenv, se crearían dos entornos aislados, se instalaría cada versión de Django en uno de los entornos y se trabajaría cada proyecto en el entorno adecuado.

Java

Un lenguaje de programación es un lenguaje formal con una base sintáctica y semántica diseñado para crear instrucciones capaces de ser realizadas por máquinas.

Java es un lenguaje de programación orientado a objetos. La programación orientada a objetos es una forma de programar, una manera de plantearse la programación, más cercana a expresar lo que necesitamos en el código de una manera similar a como se expresaría en la vida real.

Una de las ventajas principales de este lenguaje de programación es la posibilidad de ejecutar las aplicaciones desarrolladas en Java en cualquier sistema operativo ya que no es el sistema quien las ejecuta sino que se ejecutan en una máquina virtual, conocida como Java Virtual Machine.

Android

Android es un sistema operativo libre y multiplataforma, basado en el sistema operativo Linux, diseñado por la empresa Google para smartphones.

Consiste en un sistema operativo que ofrece la posibilidad de programar en un lenguaje de programación llamado Dalvik, una variación del lenguaje de programación Java. El sistema operativo ofrece todas las interfaces necesarias para desarrollar aplicaciones para dispositivos móviles y acceder a sus funcionalidades de una manera fácil y eficiente.

- **Android Studio**
  Un entorno de desarrollo integrado consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica para facilitarle al programador el desarrollo de software.

  Android Studio [8] es un entorno de desarrollo integrado para la plataforma Android. Está disponible para las plataformas Microsoft Windows, Linux y macOS. Ha sido diseñado para el desarrollo de aplicaciones Android.

Python

Python [9] es un lenguaje de programación interpretado, es decir, no se necesita compilar el código fuente para poder ejecutarlo y dando ventaja a poder ser utilizado en diferentes sistemas con un resultado similar. Su mayor peculiaridad es tener una sintaxis de código fácil de leer y entender.

Python es un lenguaje de programación que permite la programación orientada a objetos, programación imperativa y programación funcional.

- **Libreria scikit-learn**
  En el ámbito de la programación, una librería es un conjunto de archivos que implementan un grupo de funciones, codificadas en un lenguaje de programación concreto, preparadas para ser utilizadas de forma fácilmente accesible al programar en dicho lenguaje.

  En el transcurso del proyecto, se ha utilizado la librería scikit-learn [10], codificada en lenguaje de programación Python. Esta librería se centra en dar un abanico de funciones basadas en Machine Learning. Ofrece un conjunto de opciones eficientes y de fácil uso para la visualización y el análisis de datos. Es una librería Open Source, lo que significa que el código es accesible para todo el

mundo, para utilizar o trabajar en una mejora de este. Cabe destacar que está creado a partir de otras librerías de Python como NumPy, SciPy y matplotlib.

Django

Un framework es un conjunto de herramientas que facilitan al programador la estructura del código. Un web framework se centra en ayudar a desarrollar sitios web dinámicos, aplicaciones y servicios web.

Django [11] es un web framework gratuito y open source, codificado en Python. Este framework sigue el patrón de diseño Modelo, Vista y Controlador.

MySQL

Un modelo relacional es un modelo de organización y gestión de base de datos en tablas, formadas por filas y columnas. Cada fila representa un conjunto de valores y cada columna un tipo de dato.

Una base de datos es una entidad en la cual se pueden almacenar datos de manera ordenada y estructurada.

Un base de datos relacional es una recopilación de elementos de datos con relaciones entre ellos.

MySQL [12] es un sistema de administración de bases de datos, para bases de datos relacionales. Utiliza múltiples tablas para almacenar y organizar la información. Una de sus principales características es su interacción con lenguajes de programación como Java y su integración en distintos sistemas operativos.

Git

Un software de control de versiones es un conjunto de herramientas que gestionan cambios sobre un conjunto de códigos y permiten recuperar una versión concreta del código en un momento dado del desarrollo del proyecto. Este software facilita el uso y la administración de distintas versiones de un código.

Git es un software de control de versiones diseñado para trabajar de forma eficiente cuando se trata con un conjunto extenso de archivos de código fuente.

- **Bitbucked**
  Bitbucked es un servicio de alojamiento en la nube que utiliza un sistema de control de versiones Git. Está diseñado en Python mediante el web framework Django.

  Durante el proyecto, se ha utilizado Bitbucked para mantener controladas y guardadas las diferentes versiones del código.

5.2. Obtención de datos

La aplicación móvil se encarga de recopilar y procesar los datos gps para obtener la información de las visitas de un usuario.

La interfaz de usuario de la aplicación se compone de dos botones y un cuadro de texto donde se muestra la información obtenida cada vez que se actualizan los datos del sensor gps.

Figura 3: Interfaz gráfica de usuario de la aplicación Android.

El botón *Start* inicia la función de rastreo de coordenadas. Se activa la localización y se van obteniendo los valores de latitud y longitud de la ubicación del dispositivo, que se guardan en un archivo en la memoria interna del móvil.

El botón *Visits* activa el algoritmo que calcula visitas a partir de los datos que ha ido recogiendo la aplicación.

En el centro de la interfaz, hay un cuadro de texto donde se muestra la información obtenida cada vez que se actualiza la localización del dispositivo.

La aplicación consta de dos funciones a realizar: obtener la geolocalización del dispositivo y calcular los lugares visitados. En esta sección se explica la obtención de los datos a través del sensor gps.

Al activar la aplicación, empieza a recoger datos de geolocalización, los valores de latitud y longitud de la ubicación del dispositivo, a través del sensor gps integrado en el móvil. También se toma nota del valor de precisión del sensor gps del dispositivo en el momento de obtener los datos. Esta información se almacena en la memoria del dispositivo, junto con la fecha y hora del momento en el que se obtiene.

Al no necesitar la interacción del usuario, la aplicación sigue trabajando en segundo plano una vez activada. Las coordenadas geográficas se van recalculando cada cierto periodo de tiempo, aproximadamente cada cinco minutos.

Estos datos se almacenan en un archivo, dentro de la memoria del dispositivo móvil, para más adelante poder recuperarlos y ser utilizados.

5.3. Extracción de características

La parte del proyecto con más peso es el procesamiento de los datos obtenidos por el dispositivo móvil.

La cantidad de datos que se obtiene es grande y con un formato demasiado complicado para trabajar directamente con ellos. Para poder utilizar esta información, es necesario transformarla a un conjunto de datos comprensibles.

El primer paso es transformar los datos de geolocalización obtenidos por el sensor gps en un conjunto de visitas, reduciendo el tamaño de los datos y cambiando su estructura a una más fácil de comprender.

Una vez estructurados en visitas, se hace un proceso de extracción de características para ampliar los datos de los que se dispone y poder trabajar sobre más información.

En lo referente a la estructura de la base de datos utilizada en el proyecto, es necesario introducir un par de nociones utilizadas:

-Visit: una visita está compuesta por coordenadas geográficas, una fecha de entrada y una de salida, un valor de exactitud del sensor gps del móvil y un valor de identificación del usuario.

Las coordenadas geográficas se dividen en dos valores: latitud y longitud. Estos valores representan una ubicación concreta del planeta y su formato es en grados. Estan comprendidos entre -90º y 90º.

Las fechas recogidas se almacenan en valores de tiempo UNIX. Consiste en un sistema de medición con un valor numérico que indica el tiempo, en segundos, transcurrido desde la medianoche del 1 de enero de 1970. Es una medición muy extendida y utilizada por su simplicidad.

El valor de identificación del usuario se extrae del dispositivo móvil utilizado. Cada dispositivo móvil tiene uno o varios códigos IMEI, dependiendo del número de ranuras para tarjetas SIM que disponga. Son valores de 15 dígitos, diferentes para cada dispositivo y cada ranura SIM a la que van asociados, con lo que se obtiene un valor único para cada usuario.

-Visit Features: contiene un conjunto de características extraidas a través de la información de una visita y del resto de visitas del usuario que comparten información con dicha visita. Las características son las siguientes:

1. **Place type:** es la característica más importante, y en la que se basa el proyecto. Las visitas de los usuarios se agrupan por proximidad, creando grupos de visitas de un mismo sitio concreto. La finalidad del proyecto es conseguir predecir cual será el siguiente lugar al que acudirá el usuario, es decir, predecir el grupo al que pertenecerá la siguiente visita.

   Esta característica se divide en tres posibles valores, tomando los dos lugares más frecuentados por el usuario por separado, y el resto conjuntamente en un grupo.

2. **Day of week:** utilizando los datos temporales, extraemos el día de la semana en que se ha producido dicha visita, pudiendo tomar siete posibles valores diferentes.

3. **Working day:** una vez conocido el día de la semana, tomamos un valor booleano para representar si se trata de un día de trabajo o de fin de semana.

4. **Hour of day:** a partir del dato temporal del inicio de visita, calculamos la hora a la que empezó. Los valores varían entre 0 y 23. Hacen referencia a la hora, sin tener en cuenta la exactitud de los minutos y segundos, en que empezó.

5. **Duration:** calculamos la duración de la visita tomando la diferencia entre los valors horarios de inicio y de final de la visita. Este dato lo redondeamos para que los valores se ajusten a horas completas.

6. **Half of day:** dividimos el día en dos para asociar la visita a las primeras 12 horas del día o a las últimas 12.

7. **Third of day:** de igual forma, dividimos en tres partes de 8 horas el día, y asociamos la visita a uno de los tercios horarios.

8. **Half of month:** tomamos un valor de la visita vinculándola a las dos primeras semanas del mes o a las dos últimas.

9. **Accuracy:** otro valor que extraemos de la visita es la precisión del sensor gps en el momento de tomar los valores de latitud y longitud de la ubicación en la que se encuentra el dispositivo móvil.

10. **Distance to home:** una vez agrupadas las visitas, tomamos el lugar más frecuentado por el usuario. Para esta característica, calculamos la distancia entre los valores de latitud y longitud de la visita con la que trabajamos, y esos mismos datos de una visita del lugar más frecuentado. La distancia entre visitas se calcula en metros.

11. **Average time place visit:** para cada uno de los grupos de visitas, calculamos en tiempo medio que el usuario pasa en dicho lugar.

12. **Previous place type:** para cada visita, tomamos valores de la visita inmediatamente anterior. Uno de estos valores es el lugar visitado.

13. **Previous hour of day:** también tomamos la hora a la que se inició la visita anterior.

Esta información es la que se utiliza para predecir el lugar donde acudirá el usuario.

5.3.1. Procesamiento de los datos gps

Una vez recogidos y guardados los datos de geolocalización, la aplicación los recupera para calcular los lugares en los que ha estado el usuario.

El proceso consiste en calcular en que zonas próximas ha estado el usuario un cierto tiempo, y valorarlo como un lugar visitado.

Se consideran un par de coordenadas geográficas como punto base de la nueva visita. A continuación, se calcula la distancia entre los valores de latitud y longitud obtenidos en las siguientes coordenadas rastreadas por el sensor, hasta encontrar unas coordenadas alejadas 50 metros o más de la posición inicial. Una vez detectado que el usuario se ha desplazado, se utilizan los datos temporales en los que se han rastreado las coordenadas para determinar el tiempo que el usuario ha permanecido en el lugar.

Para valorar los datos obtenidos como una nueva visita, el usuario debe haber permanecido un mínimo de 15 minutos próximo a la ubicación inicial. Si se cumplen estos requisitos, se emplean estos datos para crear una nueva visita y las últimas coordenadas obtenidas se utilizan para el cálculo de la siguiente visita.

A continuación se detalla, en pseudocódigo, el algorimo desarrollado para detectar visitas a través de los datos obtenidos por el sensor gps.

```
first = null
last = null
while tracked.next do
    if first == null then
        first = tracked
        last = tracked
    else
        if near(first,last) then
            last = tracked
        else
            if timeinterval(first,last) then
                visit = newVisit(first,last)
                first = null
            end if
        end if
    end if
end while
```

Al detectar un nuevo lugar visitado se procede a crear una visita: se guardan en un archivo del dispositivo móvil las coordenadas gps, el valor de precisión del sensor gps al obtener las coordenadas y la fecha y la hora del momento de inicio y fin de la visita. Se almacena en un archivo junto al resto de visitas detectadas, para su posterior uso.

5.3.2. Procesamiento de los datos para la predicción

El proceso de extracción de características se encarga de procesar los datos obtenidos por la aplicación móvil, para clasificarlos y predecir el siguiente lugar al cual acudirá el usuario.

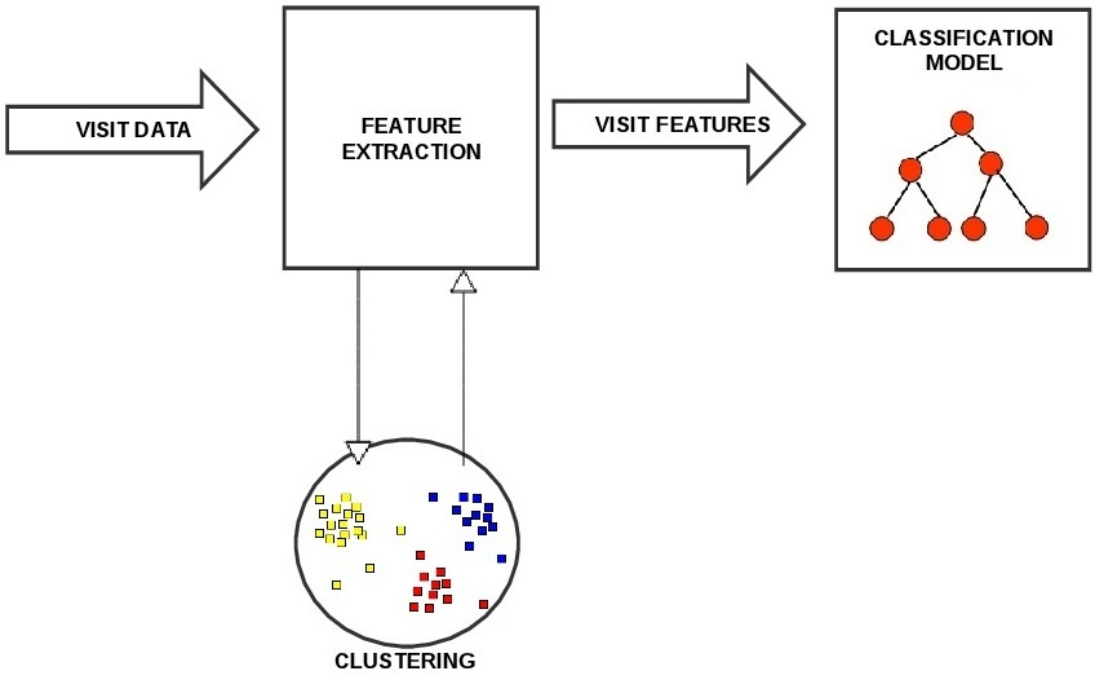

Figura 4: Proceso de extracción de características

Se divide en dos fases:

- Cargar y procesar los datos:

Los datos se importan través de un archivo csv. Este archivo contiene el listado de visitas con toda la información necesaria de cada una de ellas.

Al importar los datos, toda la información se almacena en una base de datos MySQL. En la base de datos, se dispone de dos tablas. Una para almacenar la información de las visitas y otra para la información de las características de las visitas.

Cada visita contiene las coordenadas geográficas del lugar de la visita, una fecha de inicio y una de finalización, un valor de exactitud del sensor gps del móvil al obtener los datos gps y un valor de identificación del usuario extraído del dispositivo. Al importar los datos, se comprueba que sea correcto su formato para evitar fallos en el sistema.

Después de guardar cada visita en la base de datos, se calculan las características de la *Visit Features* correspondiente.

Con los datos gps, se extrae la zona horaria en la que se ha producido la visita para poder trabajar correctamente con los datos temporales, ya que los valores horarios de las visitas están en un formato neutro y, dependiendo de la zona horaria, varia considerablemente la hora, e incluso puede variar el día, en que se han obtenido los datos. Con el valor temporal de inicio de la visita, se calcula el día y la hora a la que se ha producido y con ellos se extraen los valores de las características *day of week*, *working day*, *hour of day*, *half of day*, *third of day* y *half of month*.

Para la duración de la visita, se calcula la diferencia horaria entre los valores temporales de inicio y fin de la visita.

La característica *accuray* se obtiene directamente de la información de la visita.

El resto de características, *place type*, *average time place visit*, *previous place type* y *previous hour of day* se guardan en la base de datos con un valor por defecto. Estos valores se deben calcular cuando se ha importado toda la información del usuario con el que se trabaja. Es decir, es necesaria la información de todas las visitas del usuario para poder determinar estos valores para cada una de las *Visit Features* vinculadas con el usuario.

- Agrupar los datos:

Una vez introducida en la base de datos la información importada del archivo csv, se procede a agrupar los datos para extraer los valores de las características que faltan por calcular. Para ello, se recuperan de la base de datos la información de cada usuario para tratarla por separado.

El primer valor que se calcula es la característica *place type*. Esta caracterísitica representa el lugar en que se ha producido la visita y se utiliza para poder relacionar las visitas que se producen en un mismo lugar. Se emplea un clustering sobre los valores de latitud y longitud de las visitas para agruparlas por esta característica. Para relacionar dos visitas, debe haber entre sus coordenadas una distancia inferior a 50 metros. La distancia entre visitas se calcula con la *vicenty distance*, que calcula la distancia entre dos pares de coordenadas de latitud y longitud, teniendo en cuenta que el planeta no es una esfera perfecta.

Para el desarrollo del proyecto, se agruparon las visitas en tres clusters: los dos primeros con los conjuntos de visitas cercanas con mayor afluencia, y un tercer cluster con el resto de visitas.

Inicialmente se utilizó el algoritmo *K-Means* para agrupar las visitas en tres clusters. Este algoritmo intenta minimizar la varianza entre las distancias de las visitas agrupadas. Se iteraba el algoritmo sobre cada cluster en el que había una distancia superior a 50 metros entre algún par de visitas que lo formaban. Recursivamente se iban creando grupos de visitas cercanas hasta obtener un conjunto de clusters adecuado. Para obtener los tres clusters necesarios, se seleccionaban los dos con mayor número de visitas y se juntaban el resto de visitas en un tercer grupo.

Al avanzar en el proyecto se cambio el algoritmo utilizado para el clustering de los datos. Se pasó a utilizar un algoritmo de *Agglomerative Clustering*, que permite un agrupamiento jerárquico. Con este nuevo método, se agrupan las visitas en busca de encontrar una división de clusters adecuada, con el mínimo número de clusters posibles. Es decir, encontrar el mínimo número de clusters en los cuales las visitas que los forman sean visitas cercanas, que se encuentren a menos de 50 metros. Una vez obtenido este conjunto de clusters, se procede de la misma forma que anteriormente: se seleccionan los dos clusters con mayor afluencia y se unen en un tercer grupo todas las visitas restantes.

Una vez actualizado el valor de la característica place type en cada *Visit Features*, se procede a calcular la característica *average time place visit*. Se calcula la media de tiempo de las visitas en cada lugar, sumando la duración y dividiendo por el número de visitas en dicho lugar.

Finalmente se obtienen los valores de las características *previous place type* y *previous hour of day*. Al tener datos temporales, se pueden ordenar las visitas por la hora en que se ha producido, obteniendo una linealidad en el tiempo de las visitas de un usuario. Estos valores son los datos de las características *place type* y *hour of day* de la visita inmediatamente anteior.

Una vez calculados todos los datos de una *Visit Features*, estos se actualizan en la base de datos.

5.4. Modelo de clasificación

Una vez se tiene toda la información calculada a partir de los datos de las visitas, se procede a clasificarla.

El objetivo es predecir cual será el siguiente lugar al que acudirá el usuario, es decir, predecir cual será el valor de la característica *place type* de la siguiente visita. Esta característica puede tomar tres valores, dependiendo de si se refiere a uno de los dos lugares más frecuentados por el usuario o al grupo del resto de visitas..

Para determinar este valor, definimos dos variables binarias. Cada una de ella representa si la visita pertenece, o no, a uno de los grupos de lugares más frecuentados. Les asignamos el valor 1 cuando se refiere al lugar correspondiente y 0 si es a uno de los otros dos grupos posibles.

Para decidir cual es el valor que queremos asociar a la siguiente visita, definimos dos árboles de decisión. Utilizamos cada árbol para predecir por separado el valor de la variable binaria de asociamos a cada posible *place type* de los dos lugares más frecuentados.

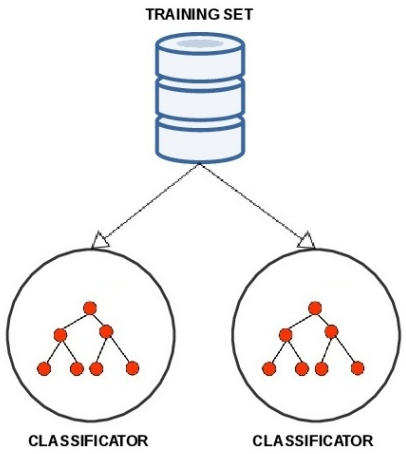

Figura 5: Modelo de clasificación en dos árboles de decisión

Para entrenar los árboles de decisión, creamos una matiz con el conjunto de valores de las *Visit Features* y de las variables binarias. Es decir, cada fila de la matriz contiene la información de una visita. Esta matriz la utilizaremos para ambos árboles. Además de la matriz, es necesario crear un vector para cada árbol de decisión en el que asignaremos, a cada visita de la matriz, el valor de la característica *place type* de la siguiente visita a la que se refiere la fila correspondiente a la posición del vector. En cada vector se asignan los valores binarios de la característica *place type* de las visitas correspondientes.

Según los datos con los que entrenamos, cada árbol dará el valor con mayor probabilidad, con los que se obtienen tres posibles resultados:

- Ambos valores son 0: en este caso, suponemos que la siguiente visita no será a ninguno de los lugares más frecuentados, por lo que asociamos el *place type* al tercer grupo de visitas.

- Obtenemos un 1 y un 0: uno de los árboles, el que da el valor 1, predice que la siguiente visita se realizará en el lugar al que se asocia dicho árbol. El valor predecido por el otro árbol significa que no asocia la siguiente visita ese grupo de visitas, sino al grupo del resto de visitas del usuario o al grupo del otro lugar más frecuentado.

- Ambos valores son 1: ambos árboles de decisión toman como mayor probabilidad la opción de que la siguiente visita sea en el *place type* asignado a su árbol. Al no tener más datos, optamos por no decidir por ninguna opción concreta y no asignamos la siguiente visita a ninguno de los tres grupos.

Una vez clasificados los datos, comparamos los valores de ambos árboles de decisión. El valor resultante es nuestra predicción sobre el lugar que visitará el usuario.

6. Testeo y resultados

El testeo del proyecto se ha dividido de dos fases. Por una parte se ha testeado personalmente la aplicación móvil. El modelo de clasificación se ha testeado utilizando un conjunto de datos de un usuario anónimo.

6.1. Testeo de la aplicación

Una de las partes más importantes en el desarrollo de las aplicaciones móviles es el testing. Que las aplicaciones funcionen como se espera que hagan, de forma eficiente y efectiva, nos indica un buen rendimiento de nuestra aplicación.

El testeo de la aplicación lo he realizado personalmente. Durante varios dias consecutivos he ido recogiendo datos de geolocalización por Barcelona haciendo uso de la aplicación.

Estos datos recogidos se iban almacenando en la memoria del dispositivo para utilizarlos más adelante en el cálculo de los lugares frecuentados. Dependiendo de la ruta diaria, se encontraban lugares comunes y lugares nuevos.

En general, el sensor gps obtenia correctamente los datos de la ubicación del dispositivo móvil. En zonas cerradas, como el metro o los sótanos de la universidad, el sensor no recibía bien la señal y devolvía los valores de la última posición rastreada.

6.2. Testeo del modelo de clasificación con datos de usuarios reales

El testing del sistema de predicción se ha producido durante todo el desarrollo del mismo, para evaluar las variaciones introducidas continuamente y decidir cual es el mejor camino para conseguir unos resultados óptimos.

Disponíamos de un conjunto de datos de visitas de un usuario, obtenidas durante un período aproximado de 8 meses. Los datos reflejaban la información de 1000 visitas, aproximadamente 4 visitas detectadas diariamente.

El primer paso fue procesar los datos y descartar los datos que no estuvieran completos. En algunas visitas, los datos temporales no eran correctos y tenian un valor por defecto.

Una vez procesados los datos de las visitas, se procedió a un estudio sobre las características que se podían extraer de la información de las visitas. Se calcularon características que más adelante se despreciarían por su poca importancia en el

proceso de aprendizaje del algoritmo. Alguna de estas fueron el día o el mes en que se producia la visita. Otras en cambio, como la precisión del sensor gps, se iban añadiendo con el proceso al analizar los resultados obtenidos.

El avance con mayor utilidad fue separar en un conjunto de valores binarios la característica *place type*. Se pasó de utilizar un árbol de decisión, para predecir uno de los tres posibles valores de la característica, a utilizar dos árboles de decisión para calcular la posibilidad de que la siguiente visita se produzca o no en un lugar concreto, y comparando los resultados de ambos árboles, decidir qué valor se ajusta más a la nueva visita.

En el último paso se trabajó en la ventana temporal escogida para predecir los valores deseaos, es decir, el número de visitas con las que entrenar los árboles de decisión para predecir el lugar correcto con una mayor precisión. Después de unas pruebas exhaustivas, variando la ventana temporal y las características utilizadas, se encontró que se obtenían los mejores resultados utilizando entre 280 y 300 visitas para entrenar los árboles de decisión.

6.3. Resultados

Los resultados del proyecto se basan en el clasificador y los datos utilizados para testear el sistema de predicción.

Los resultados obtenidos se dividen en dos términos: precisión y exhaustividad [14]. Ambos datos se calculan para cada grupo de visitas que el clasificador intenta predecir, es decir, para los dos lugares más frecuentados por el usuario.

La precisión es el porcentaje de acierto de los valores positivos que se calculan, es decir, las veces que el clasificador ha predecido correctamente el tipo de lugar al que acudirá el usuario, dividido por el número de veces que ha predecido dicho lugar.

La exhaustividad es el porcentaje de acierto de las veces que ha predecido correctamente un lugar, dividido por el número de veces que debería haber predecido dicho lugar.

Tomando los datos con los que se ha testeado el sistema, se ha alcanzado un acierto del 70 % de precisión y del 68 % de exhaustividad aproximadamente.

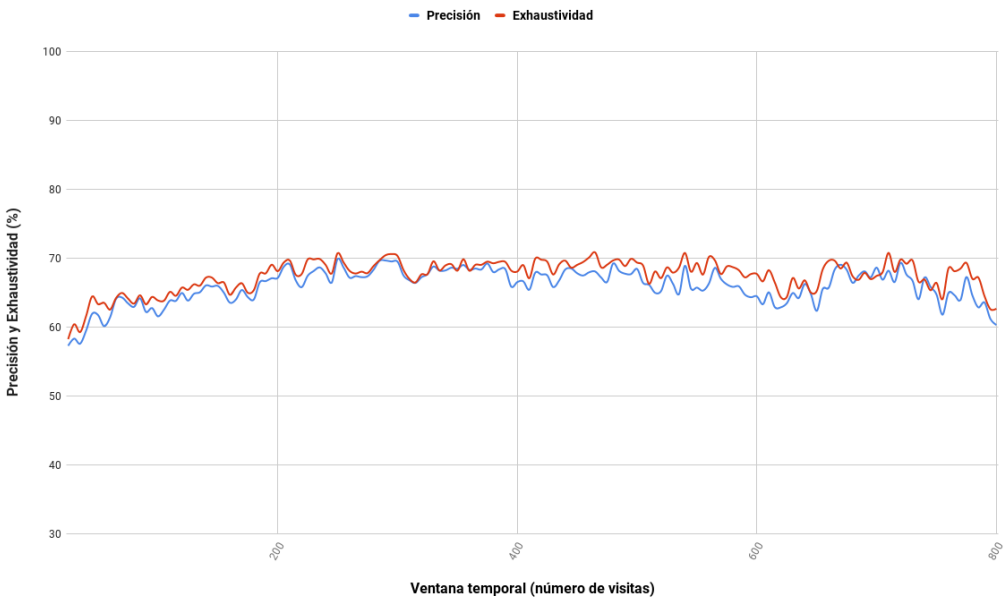

Figura 6: Gráfica de resultados

Estos valores son los que dan pie a escoger una ventana temporal adecuada para predecir con un mayor acierto para futuros usos. La ventana temporal escogida es de dos meses y medio, aproximadamente 300 visitas.

7. Conclusiones

7.1. Conclusiones sobre el proyecto

En este proyecto se ha desarrollado un sistema capaz de reunir y procesar datos, obteniendo información útil sobre usuarios, con un buen resultado.

Por una parte, se ha desarrollado una aplicación móvil capaz de obtener datos gps y procesarlos para calcular la información necesaria sobre las visitas que realiza un usuario. Para trabajar con esta información, se ha implementado un proceso de extracción de características, con el que se extrae la información necesaria para utilizar en un modelo de predicción desarrollado para predecir el siguiente lugar al que acudirá un usuario.

Personalmente ha sido una etapa en la que he podido entender la dificultad y los problemas que surgen durante el desarrollo de un proyecto. La necesidad de aprender durante el transcurso del proyecto y el uso de los conocimientos obtenidos durante la carrera.

Asignaturas como *Projecte Integrat de Software* o *Factors Humans* me han ayudado al desarrollo de la aplicación Android. Otras como *Tallers de Nous Usos de la Informàtica* a la implementación del sistema de predicción y *Enginyeria del Software* a la planificación del proyecto.

En general, creo que ha sido una buena experiencia que me ayudará más adelante cuando tenga que afrontar otros proyectos y nuevos retos.

7.2. Ampliaciones y trabajo futuro

El proyecto consta de dos partes diferenciadas, la aplicación android, para recoger datos del usuario, y el algoritmo de clasificación. Ambas partes trabajan para un objetivo común, conseguir datos de los usuarios en tiempo real y clasificarlos para predecir información.

Ambas partes del proyecto han quedado no conectadas, trabajando por separado con su función particular cada una de ellas. El siguiente paso sería conectar las dos partes mediante un servidor, al cual enviar los datos recogidos con la aplicación móvil en tiempo real. En el servidor se deberían clasificar convenientemente para poder utilizar la información en cualquier momento que fuese necesario, pudiendo enviar dicha información de nuevo al dispositivo móvil.

Un cambio considerable para la aplicación android y los datos recogidos consistiría en determinar, de una manera más precisa, la categoría de estos datos, procedimiento que deberá hacer manualmente el usuario. En otras palabras, obtener datos etiquetados por el usuario. Esta mejora evitaría tener que agrupar las nuevas visitas obtenidas, evitando así un posible error al utilizarlas para la clasificación.

Otra dirección en la que avanzar en el proyecto es utilizar diferentes algoritmos de clasificación y comparar los resultados para mejorar el sistema de predicción, pudiéndose basar en los datos recogidos por el usuario.

Las mejoras propuestas podrán ser utilizadas en una aplicación para la asistencia familiar (Proyecto de Investigación Industrial Torres Quevedo PTQ- 14-07020)

8. Bibliografia

Referencias

[1] James, G., Witten, D., Hastie, T. and Tibshirani, R. (2015). *An Introduction to Statistical Learning*. USA.

[2] Supervised and Unsupervised Learning: *http://scikit-learn.org/stable/supervised learning.html* Last access June 2017

[3] K-Means documentation: *http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html* Last access June 2017

[4] Agglomerative Clustering documentation: *http://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html* Last access June 2017

[5] Decision Tree documentation: *http://scikit-learn.org/stable/modules/tree.html* Last access June 2017

[6] Java documentation: *https://docs.oracle.com/javase/7/docs/api/* Last access June 2017

[7] Android documentation: *https://developer.android.com/reference/android/package-summary.html* Last access June 2017

[8] Android Studio documentation: *https://developer.android.com/studio/index.html* Last access June 2017

[9] Python documentation: *https://docs.python.org/2/* Last access June 2017

[10] Scikit-learn library: *http://scikit-learn.org/stable/index.html* Last access June 2017

[11] Django documentation: *https://docs.djangoproject.com/* Last access June 2017

[12] MySQL documentation: *https://dev.mysql.com/doc/* Last access June 2017

[13] Location Android documentation: *https://developer.android.com/reference/android/location/LocationManager.html* Last access June 2017

[14] Precision and recall: *https://en.wikipedia.org/wiki/Precision and recall* Last access June 2017

9. Anexos

9.1. Manual técnico

En esta sección se explican los pasos a seguir para testear la aplicación android y el proceso de predicción.

9.1.1. Manual para el testeo de la aplicación android

La aplicación móvil se ha desarrollado para sistema operativos Android. Se ha diseñado utilizando el entorno de desarrollo Android Studio, en sistema operativo Linux.

Para testear el proyecto de Android Studio adjuntado, es necesario:

1. Instalar Android Studio, versión 2.3.3 o posterior.
2. Abrir Android Studio e importar el proyecto Visits Tracker adjuntado junto a la memoria.
3. Conectar un dispositivo móvil android al ordenador, con una versión del sistema operativo Android 7.0 o superior. Se deben permitir el acceso al dispositivo desde el ordenador para que Android Studio lo reconozca como dispositivo disponible para ejecutar el proyecto. También es posible crear en Android Studio un emulador de dispositivo Android para ejecutar la aplicación. No es recomendable ya que se deben entrar manualmente los cambios de localización y no se ciñe al uso principal de la aplicación.
4. Una vez conectado el dispositivo, se debe ejecutar el proyecto. Se instalará la aplicación en el dispositivo móvil y se abrirá la aplicación automáticamente.
5. Al abrir por primera vez la aplicación, en el dispositivo se pedirán los permisos necesarios. Se deben permitir todos. Son permisos para utilizar el sensor gps del dispositivo y escribir y leer en archivos de la memoria interna.
6. El diseño de la aplicación consta de dos botones
   - **Start:** inicia el reconocimiento gps. A partir del sensor gps va obteniendo los valores de latitud y longitud de la posición del dispositivo. Muestra por pantalla últimos valores captados y la hora en la que se han obtenido. Aproximadamente cada cinco minutos, se vuelven a calcular estos datos y se actualiza la información mostrada por pantalla. Cada vez que se reciben nuevos datos, se guardan en un archivo rawgps.txt en la memoria interna del dispositivo móvil, en el directorio *Documents/VisitsTracker/data/* El directorio es creado la primera vez que se quiere guardar información desde la aplicación en la memoria del móvil. La aplicación sigue funcionando en segundo plano, lo que permite seguir capturando y guardando los datos gps mientras el usuario sigue utilizando el móvil para otras tareas.

- **Visits:** al activarlo, el dispositivo recupera al información guardada en la memoria interna en el archivo rawgps.txt y recupera los datos gps obtenidos por el sensor durante todo el uso de la aplicación. Una vez cargados, los datos pasan por el algoritmo de detección de visitas y guarda la información de las visitas detectadas en el archivo visitsdata.txt, en el mismo directorio. Este archivo contiene la información de las visitas en el mismo estilo necesario para importar en el sistema de predicción.

9.1.2. Manual para el testeo del proceso python de predicción

El modelo de clasificación se ha desarrollado en lenguaje de programación Python.

Para la instalación de todos los componentes necesarios, se recomienda utilizar un sistema operativo Linux y crear un entorno virtual para trabajar en un entorno similar al utilizado durante el desarrollo del proyecto.

Se ha adjuntado un archivo requirements.txt donde se lista el conjunto de softwares necesarios. Para instalarlos, abrimos una Terminal y accedemos al directorio del archivo requirements.txt y utilizamos la instrucción *pip install -r requirements.txt* para instalar todo el software necesario.

Una vez instalado todo, se debe registrar la información de la base de datos a utilizar. Para ello, hay que acceder al archivo *tracks/tracks/settings.py* y poner la información en el apartado DATABASES. En el código entregado están como ejemplo los valores utilizados durante el desarrollo.

Una vez especificada la información de la base de datos, se deben crear las tablas donde guardaremos los datos. Para ello, ir al directorio *tracks/* y ejecutar los comandos *python manage.py migrate* y *python manage.py makemigrations*.

Para probar el modelo de clasificación, se deben tener los datos en un archivo csv con el formato siguiente: la primera linea es un header, sin importancia. A partir de la segunda linea, se deben tener los valores de las visitas en elorden siguiente: *id,latitud,longitud,accuracy,fecha de inicio de visita,fecha de fin de visita,fecha de obtencion de datos,id de usuario, fecha de recepción de visita*. Los datos *id,fecha de obtencion de datos* y *fecha de recepción de visita* no se utilizan en el proceso pero el modelo está implementado para recibir estos datos.

Para cargar los datos, se debe ir al archivo *tracks/test.py* y cambiar el valor de la variable *path* del método *loadData* por la del archivo csv.

Finalmente, para ejecutar el script de python, desde consola se debe acceder al directorio *tracks/* y ejercutar el comando *./manage.py shell* para entrar en la consola de django. Una vez en este entorno, ejecutar el comando *import test*. Esto ejecutará el script del archivo *test.py*. El script mostrara las gráficas de los clusters de cada usuario y finalmente, guardará en el archivo *ResultsAllUsers.txt* los resultados obtenidos.