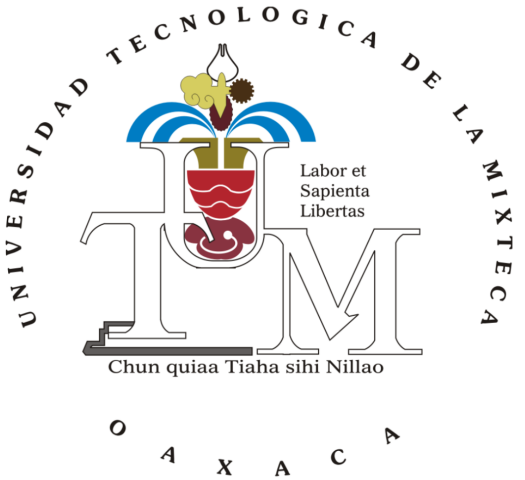


Universidad Tecnológica de la Mixteca

Reconocimiento distribuido de patrones de actividades físicas

TESIS

PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMPUTACIÓN

PRESENTA:

ANTONIO JAVIER MÉNDEZ JUÁREZ

DIRECTOR DE TESIS:
DR. RAÚL CRUZ BARBOSA

CO-DIRECTOR DE TESIS:
DR. ARTURO TÉLLEZ VELÁZQUEZ

Huajuapan de León, Oaxaca, México, Junio de 2018

A mis padres,
Javier y Alejandra por su confianza y paciencia,
es gracias a ustedes que es posible el presente trabajo.

A mis hermanos,
por su cariño y sus palabras de apoyo.

Agradecimientos

Quiero manifestar mi agradecimiento a mis asesores de tesis, Dr. Raúl Cruz Barbosa y Dr. Arturo Téllez Velázquez, por su apoyo en la dirección de esta tesis. Por los conocimientos transmitidos durante la realización de este trabajo, la paciencia, los comentarios y observaciones que sin duda me ayudaron a mejorar profesionalmente.

A mis padres, Javier Méndez y Alejandra Juárez, no sólo por su apoyo económico sino también por su comprensión, consejo y paciencia (mucha paciencia). Por enseñarme con su ejemplo la importancia del trabajo.

A mis hermanos, por estar siempre ahí y ser una parte importante en mi vida.

A Tomás, por ser un gran amigo, por acompañarme a pesar de la distancia, motivarme a ser mejor cada día y contagiarme de su pasión por la resolución de problemas.

A Christopher, Luis Ángel y Diana por el apoyo, la paciencia y las enseñanzas que me otorgaron para mejorar como persona. Gracias por sus consejos, sin ustedes seguramente 'habría terminado este trabajo "mucho antes".

A mis compañeros de departamento, Daniel, Murgas, Aldahir, Edgar, Iván, por las aventuras, el tiempo que pasamos durante los años de carrera, los momentos que vivimos juntos y las enseñanzas.

A mis compañeros de los concursos de programación, Manuel, Ulises, José Carlos, Rafael, Irving, Kevin, Salvador por ayudarme, inspirarme, motivarme y enseñarme a trabajar en equipo para ser mejor cada día.

A mis sinodales, por el tiempo dedicado a este trabajo de tesis, por los comentarios y observaciones realizadas con la finalidad de mejorar este trabajo.

Al profesor Moisés Emmanuel Ramírez Guzmán por ser un excelente docente y amigo, por su apoyo y confianza brindada durante toda mi estancia en la universidad.

Finalmente, quiero mencionar que este trabajo de tesis fue apoyado parcialmente por el Consejo Nacional de Ciencia y Tecnología (CONACYT) en el marco del proyecto Cátedras-CONACYT número 1170.

Resumen

La identificación de actividades físicas humanas es un área de investigación que ha cobrado gran notoriedad por los diferentes tipos de problemas que intenta resolver, entre los que se encuentra el cuidado de la salud. Este problema ha sido investigado también por el campo de estudio de reconocimiento de patrones, para lo cual es necesario adquirir datos que ayuden con la identificación. Para la adquisición de información se han utilizado diversos enfoques, los cuales van desde el análisis de vídeo hasta el uso de distintos tipos de sensores. Uno de esos enfoques propone el uso de los teléfonos inteligentes, los cuales han mostrado ser una buena alternativa para la recolección de información.

Para mejorar el cuidado de la salud es necesario (entre otras cosas) mantener un monitoreo constante de las actividades diarias de las personas, lo cual requiere de la generación de grandes cantidades de información. El procesamiento de grandes cantidades de datos no es factible al ser tratado con enfoques tradicionales como el cómputo secuencial.

Dada esta problemática, en este trabajo de tesis se plantea realizar el reconocimiento de patrones de actividades físicas humanas que integre el seguimiento de éstas. Para lograr este propósito se utilizan los sensores que vienen integrados en un teléfono móvil para la adquisición de información. Asimismo, se utiliza la herramienta de cómputo distribuido Apache Spark, la cual está diseñada para realizar procesamiento de grandes cantidades de información que son enviadas por los teléfonos móviles.

Para lograr el reconocimiento de las actividades físicas, se propone una aplicación móvil que recolecte información del acelerómetro y del giroscopio de un teléfono inteligente. La información recolectada posee datos de los usuarios relacionados con las actividades: caminar, estar acostado, estar de pie, estar sentado, bajar y subir escaleras. Esta información es enviada a un servidor, donde las actividades son caracterizadas y posteriormente son clasificadas por un reconocedor de patrones de actividades físicas humanas instalado en un clúster computacional.

En cuanto al seguimiento de las actividades, éste se realiza en una aplicación

Web, en la cual los usuarios observan el registro de las actividades realizadas durante un periodo de tiempo.

Para la selección del algoritmo de clasificación, se plantearon diversos experimentos. Primero se generó un conjunto de datos a partir de la participación de voluntarios, obteniendo así un conjunto con 22,315 muestras. Dicho conjunto fue utilizado para los experimentos con seis algoritmos de aprendizaje supervisado, los cuales se encuentran implementados en la herramienta Apache Spark.

A partir de estos algoritmos, el que mejor rendimiento mostró durante los experimentos fue el de bosques aleatorios con una exactitud del 90.4 %, por lo cual, éste es el algoritmo incrustado en el reconocedor de actividades.

Para analizar el rendimiento de ejecución de los bosques aleatorios se realizaron una serie de experimentos de cómputo secuencial y distribuido, observando así que el algoritmo de bosques aleatorios logró acelerar 11.8 veces la etapa de entrenamiento y 13 veces la etapa de predicción.

Finalmente, este clasificador fue integrado en la aplicación Web que permite hacer el seguimiento de las actividades de los usuarios, con la finalidad de sensibilizar a los usuarios para que realizacen actividades que ayuden a mejorar su salud.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En los últimos años, las enfermedades relacionadas a la insuficiente actividad física se han incrementado. La Organización Mundial de la Salud (WHO, del inglés World Health Organization) menciona que aproximadamente 3.2 millones de muertes cada año se le atribuyen a la inactividad física [WHO, 2014]. Estudios [Lee et al., 2012] muestran evidencia de que la inactividad física incrementa el riesgo para la salud, incluyendo enfermedades cardiovasculares, diabetes tipo 2, cánceres de mama y de colon, obesidad, ansiedad, osteoporosis, y es un factor para la disminución de la esperanza de vida. Se ha reportado que entre el 6 % y 10 % de todas las muertes por estas enfermedades en todo el mundo pueden ser atribuidas a esta causa [Lee et al., 2012]. Este porcentaje es aún mayor para enfermedades específicas, por ejemplo 30 % para la cardiopatía isquémica. Estos estudios evidencian que la inactividad física se ha convertido en un problema de salud pública global. Algunas investigaciones [Kohl et al., 2012] muestran que la inactividad física se ha convertido en la cuarta causa de muerte en el mundo.

Existen diversas causas por las que la inactividad física se ha vuelto un problema global. La razón que más destaca es el aumento en el comportamiento sedentario, esto debido a que los ambientes en los que las personas se desenvuelven requieren que realicen cada vez menor esfuerzo físico.

Para enfrentar este problema el gobierno mexicano ha implementado programas de salud, donde se recomienda a las personas realizar actividades como caminatas en períodos cortos, cambiar de hábitos de alimentación, usar transporte alternativo que promueva la actividad física y evitar pasar grandes períodos de tiempo sentados o acostados [IMSS, 2015].

La tecnología puede ser empleada para monitorear las actividads físicas. Actualmente, esto se ha popularizado con el uso de pulseras inteligentes que miden la cantidad de pasos aproximados recorridos por día y el tiempo que se han realizado. El inconveniente con este tipo de dispositivos es que no toman en cuenta el tiempo de inactividad física de la persona. Para poder registrar los tiempos realizados

de cada actividad es necesario distinguirlas, lo cual es un problema complejo y estudiado por el área de reconocimiento de patrones [Casale et al., 2011, Eunju et al., 2010].

El reconocimiento de actividades tiene como objetivo identificar actividades físicas humanas en ambientes cotidianos. Para lograrlo, es necesario afrontar diversos retos, debido a que las actividades físicas humanas son complejas y muy diversas. La importancia del reconocimiento de actividades radica en que ésta puede ayudar a mejorar la calidad de vida de las personas, contribuyendo al desarrollo de diversas áreas de estudio, como la inteligencia ambiental y la vida cotidiana asistida por el entorno para personas dependientes.

Usualmente, el reconocimiento de actividades implica la adquisisión de información relacionada con dichas actividades. Tal adquisición involucra el uso de sensores vestibles [Casale et al., 2011]. Otros enfoques hacen uso de información como el movimiento [Iglesias et al., 2011], ubicación [Choujaa and Dulay, 2008], temperatura [Parkka et al., 2006], entre otras. Sin embargo, el uso de este tipo de sensores trae consigo diversos inconvenientes que degradan la portabilidad, la independencia y la comodidad. Tales inconvenientes surgen cuando se depende de otros sistemas para la captura de datos y cuando se requieren de elementos externos que les suministren energía para su funcionamiento. Esto puede provocar que los usuarios alteren la forma en la que realizan sus actividades cotidianas.

Gracias a los avances en el hardware de los dispositivos móviles y al incremento de su capacidad computacional y la portabilidad de sus sensores, ha sido posible la integración de estos en gran parte de la vida diaria de las personas. Entre los dispositivos móviles más utilizados en la población mexicana se encuentran los *teléfonos inteligentes*. En México se estima que al año 2016, los usuarios de teléfonos celulares representan el 73.6 % de la población con edades a partir de los seis años en adelante, y tres de cada cuatro usuarios cuentan con un teléfono inteligente (Smartphone), conforme a la información publicada por el Instituto Nacional de Estadística y Geografía (INEGI) [INEGI, 2017].

El uso de teléfonos inteligentes para la identificación de actividades humanas se ha popularizado debido a que cada vez son más accesibles y más personas cuentan con uno. Entre los trabajos que hacen uso de estos dispositivos se puede encontrar el publicado por [Lee and Cho, 2011], en el cual hacen uso de sensores integrados en los dispositivos de la línea *Desire*, de la marca HTC®. En el trabajo mencionado se clasifican accciones diarias, tales como estar de pie, caminar, etc. En otro trabajo [Su et al., 2014] hacen uso del acelerómetro y giroscopio para clasificar actividades diarias, como son descansar, caminar, correr, subir y bajar escaleras.

Como puede verse, el uso de los sensores de un teléfono inteligente es una manera plausible para monitorear la actividad física de millones de personas en México y en todo el mundo al mismo tiempo, ya que éste no interfiere en la manera

de realizar las actividades diarias. Esta situación tendría como consecuencia la generación de datos que pueden alcanzar volúmenes muy altos y por consecuencia requerir de procesamiento a gran escala.

Dado que el cómputo de grandes cantidades de datos hoy en día es un desafío, la tendencia actual se inclina hacia el uso del *cómputo distribuido* [McAfee and Brynjolfsson, 2012]. Actualmente, en el mercado se encuentran herramientas muy poderosas para manejar y procesar grandes volúmenes de información y que además permiten la escalabilidad en las aplicaciones que las utilizan [Apache, 2015, Apache, 2014, Apache, 2011a].

La presente tesis tiene como objetivo desarrollar una aplicación basada en aprendizaje computacional para el reconocimiento distribuido de patrones de actividades físicas y su seguimiento. Para esto, se hace uso de dispositivos portátiles con sensores integrados (teléfonos inteligentes), de tal forma que estos dispositivos no generen incomodidad en las personas y evite modificar la forma de realizar sus actividades cotidianas de manera natural.

Para lograr el objetivo planteado, se extrae información de los sensores del dispositivo con el fin de caracterizar cada una de las actividades básicas diarias realizadas por las personas (acostarse, caminar, sentarse, pararse, bajar y subir escaleras). Lo anterior se realiza haciendo un análisis de las señales producidas por el acelerómetro y el giroscopio de un teléfono inteligente.

Posteriormente, mediante el uso de algoritmos de aprendizaje computacional, se creará un modelo para el reconocimiento de los patrones de actividades físicas propuestas.

Debido a que la cantidad de información recabada puede crecer en proporción al número de usuarios y al tiempo, realizar un manejo secuencial de información masiva no representa una opción viable para la clasificación y toma de decisiones oportunas. Para mejorar los grandes tiempos de ejecución, se hace uso de un entorno de cómputo distribuido, con el cual se pretende reducir los tiempos de ejecución de los algoritmos utilizados, sin comprometer su exactitud.

1.1. Planteamiento del problema

La identificación de actividades humanas es un tema que en la actualidad ha cobrado una importancia notoria. La relevancia radica en que puede aplicarse a diversos problemas de la vida real centrados en la mejora de la calidad de vida de las personas, como son el cuidado de personas mayores y el cuidado de la salud.

En México, por ejemplo, según cifras publicadas por el INEGI [INEGI, 2016], más de la mitad de la población realiza una cantidad insuficiente de actividades físicas, lo cual puede traer problemas serios de salud a la población.

En primera instancia, la selección de la tecnología que permita capturar las actividades físicas humanas cotidianas es primordial para caracterizar el perfil de actividad de los usuarios. Cabe aclarar que el uso del adjetivo "cotidiano" implica que el dispositivo idealmente acompañe a los usuarios a lo largo de toda su jornada de forma inadvertida.

En este sentido, se ha enfrentado el problema de diversas maneras usando la tecnología actual, como es la tecnología vestible, que ha ayudado a llevar un control y monitoreo de las actividades físicas realizadas a lo largo del día. Estos dispositivos normalmente requieren de un esfuerzo mayor de los usuarios, pues tienen conectividad y batería limitadas, y normalmente requieren de dispositivos adicionales para llevar el control de lo censado. Además, en ocasiones los dispositivos inalámbricos pueden provocar interferencia entre éstos [Phunchongharn et al., 2010].

Otro enfoque distinto al antes mencionado es hacer uso de dispositivos que ya se encuentren en las manos de la mayoría de los usuarios, específicamente los teléfonos inteligentes. El inconveniente de este tipo de enfoque es cuando se pretende procesar la información en los modelos de clasificación dentro del mismo teléfono, ya que los cálculos pueden utilizar gran parte de los recursos disponibles, ocasionando que las otras tareas realizadas por el dispositivo se ralenticen.

En segunda instancia, el problema de la generación de grandes cantidades de datos, no es factible para procesamiento en un solo servidor. Es por ello que se han implementado soluciones basadas en el cómputo distribuido, las cuales direccionan el problema de dos formas. Por un lado, la clasificación masiva de actividades físicas humanas cotidianas no ha sido fácil cuando se procesa de manera secuencial, debido a las limitaciones de tiempo y capacidad computacional [Azzi et al., 2014].

Por otro lado, la clasificación distribuida de las actividades físicas humanas cotidianas ha permitido particionar los datos y por consecuencia acelerar el procesamiento de los algoritmos de aprendizaje computacional. Algunos trabajos han hecho uso de entornos de cómputo paralelo [Paniagua et al., 2012], utilizando algoritmos paralelos basados en el enfoque *MapReduce* [Dean and Ghemawat, 2004], el cual es utilizado en otros trabajos [Srirama et al., 2011]. Estas herramientas permiten escalar fácilmente el procesamiento masivo de actividades físicas humanas, mejorando el desempeño en velocidad de ejecución y conservando el buen rendimiento del algoritmo de clasificación en cuestión.

Siguiendo el enfoque de cómputo distribuido, en [Alsheikh et al., 2016] utilizaron la herramienta Apache Spark para reconocer actividades cotidianas utilizando muestras obtenidas solamente del acelerómetro de los teléfonos. Sus experimentos muestran una buena aceleración de la ejecución y buenos resultados en el reconocimiento al utilizar este enfoque. Sin embargo, su propuesta central no incluye la adquisición de los datos desde algún dispositivo en tiempo real, utilizando únicamente un repositorio de muestras previamente recolectado.

Por lo anterior, en este trabajo de tesis se pretende implementar una aplicación para el reconocimiento y seguimiento de actividades físicas humanas, a través de técnicas de aprendizaje computacional y un entorno de cómputo distribuido que ayude a mejorar el rendimiento en tiempo de ejecución de estas tareas.

Para realizar lo mencionado, primero se adquiere información del acelerómetro y del giroscopio de un teléfono móvil. Dicha información es procesada y caracterizada para formar un conjunto de datos. A continuación, este conjunto de datos es utilizado para realizar experimentos con distintos algoritmos de clasificación que se encuentren incluidos en una herramienta de cómputo distribuido previamente seleccionada, eligiendo al clasificador que muestre la mejor exactitud. Con el algoritmo seleccionado, se realizan pruebas de manera secuencial y distribuida. Además, dicho algoritmo es empotrado en una aplicación distribuida que permite el seguimiento de dichas actividades a través de una aplicación web y una aplicación en el teléfono celular.

1.2. Justificación

Debido a la creciente problemática causada por la inactividad física en el mundo, es importante identificar y monitorear la actividad física realizada por las personas, con el fin de ayudar a prevenir enfermedades asociadas a la falta de ésta.

Aún cuando el problema de reconocimiento masivo de patrones de actividades físicas humanas cotidianas ha sido tratado con anterioridad, ya sea con diversas tecnologías para la adquisición de información de los usuarios o mediante el uso de cómputo secuencial o distribuido, este trabajo pretende utilizar tecnología actual para mejorar el rendimiento en velocidad de las tareas de reconocimiento.

Asimismo, para afrontar la problemática antes mencionada, por un lado, se utilizan los sensores integrados en los dispositivos comúnmente utilizados por la población mexicana, tales como los teléfonos inteligentes. De esta manera, se monitorean las actividades físicas de los usuarios en cuestión.

Por otro lado, promoviendo los beneficios del cómputo distribuido se seleccionará una herramienta sofisticada de ésta área. La selección de esta herramienta permitirá mejorar el rendimiento en velocidad del reconocimiento masivo de los patrones de actividades físicas humanas cotidianas. Cabe destacar que los datos de entrada provendrán de los teléfonos inteligentes de distintos usuarios beneficiarios.

En este proyecto de tesis se plantea implementar un clasificador de actividades humanas usando cómputo distribuido, recolectando información de los sensores de los teléfonos inteligentes para clasificar 6 distintas actividades físicas cotidianas, las cuales son: subir escaleras, bajar escaleras, caminar, sentarse, recostarse y

mantenerse parado.

Para hacer esto posible, la adquisición y la transmisión de la información que ayude a caracterizar las actividades de los usuarios será realizada mediante la configuración y la instalación de una aplicación para teléfonos inteligentes. Por otro lado, la recepción de la información y las tareas de reconocimiento serán realizadas en un clúster computacional administrado con alguna herramienta vanguardista de cómputo distribuido, capaz de mejorar el rendimiento en velocidad drásticamente, comparado con el rendimiento obtenido con su versión secuencial.

1.3. Hipótesis

Con el uso del cómputo distribuido es posible mejorar el rendimiento de ejecución de métodos para el reconocimiento de patrones de actividades físicas, utilizando grandes volúmenes de información de entrada.

1.4. Objetivos

1.4.1. Objetivo general

Implementar una aplicación para el reconocimiento y seguimiento de actividades físicas humanas, a través de un entorno de cómputo distribuido que ayude a mejorar el rendimiento en tiempo de ejecución correspondiente.

1.4.2. Objetivos particulares

- Revisar el estado del arte sobre métodos de reconocimiento de patrones de actividades físicas y de entornos de cómputo distribuido que integren herramientas de aprendizaje computacional.
- Implementar una aplicación para el teléfono inteligente que permita extraer información de los sensores integrados en éstos, para la caracterización de las actividades físicas de un conjunto de personas, como son subir escaleras, bajar escaleras, caminar, recostarse, sentarse y mantenerse de pie.
- Usar una herramienta actual de cómputo distribuido que integre algoritmos de aprendizaje computacional de tipo supervisado para procesamiento ma-

sivo, que permita la construcción de un clúster de computadoras para tal fin.

- Seleccionar un algoritmo de aprendizaje computacional supervisado integrado en la herramientda de cómputo distribuido y que sea el más apropiado para el problema de reconocimiento de patrones de actividades físicas humanas.
- Realizar dos implementaciones, una secuencial y otra distribuida, de una aplicación de reconocimiento de patrones de actividades físicas usando el algoritmo de clasificación seleccionado.
- Comparar el desempeño de velocidad de ejecución entre las implementaciones secuencial y distribuida del algoritmo de clasificación seleccionado.
- Desarrollar una aplicación que reciba la información adquirida a través de los teléfonos inteligentes de los usuarios y haga uso del algoritmo y el entorno de cómputo distribuido seleccionados, con el fin de mantener un control estadístico y darle seguimiento a las actividades realizadas por los usuarios.

1.5. Metas

1. Elaboración de un reporte sobre métodos de reconocimiento de patrones de actividades físicas y de entornos de cómputo distribuido que integren herramientas de aprendizaje computacional.
2. Implementación de una aplicación para un télefono inteligente que extraiga información de los sensores para la caracterización de las actividades físicas.
3. Implementación de una aplicación secuencial y otra distribuida para el reconocimiento de patrones de actividades físicas usando un algoritmo de clasificación integrado en una herramienta de cómputo distribuido.
4. Elaboración de un reporte comparativo del desempeño de las implementaciones secuencial y distribuida de los algoritmos de aprendizaje computacional.
5. Implementación de una aplicación para un clúster que clasifique masivamente las actividades físicas humanas con datos adquiridos a través de teléfonos inteligentes.

1.6. Trabajos relacionados

Debido a la problemática de salud pública global que se ha generado a partir del sedentarismo, a lo largo del mundo se han promovido diversas campañas para contrarrestar este problema. En México, se han implementado programas de salud como Chécate, Mídete, Muévete [IMSS, 2015], donde se promueve un estilo de vida saludable, fomenta la activación física, una alimentación saludable y el seguimiento del estado de salud por parte de las personas. Una campaña similar es Muévete y Métete en Cintura [SEDESA, 2008], la cual promueve estilos de vida saludables, mediante la orientación alimentaria y el fomento a la actividad física. De manera similar, la estrategia nacional de activación física [CONADE, 2017] fomenta el desarrollo de la *cultura física*, por medio de la activación física a través de distintas modalidades, además de contribuir a disminuir el sedentarismo, la obesidad y las adicciones a través de la masificación de la actividad física y del correcto aprovechamiento de espacios públicos.

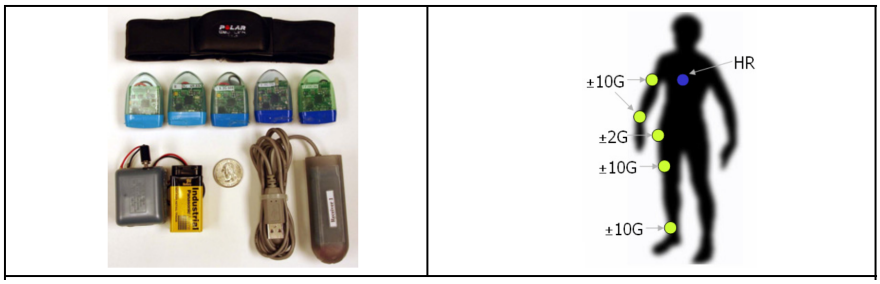
Este tipo de soluciones suelen ser muy buenas cuando hay disposición por parte del usuario. Sin embargo, no existe un seguimiento continuo de las actividades a lo largo del día que permita al usuario conocer el tiempo dedicado a una actividad como subir escaleras o caminar.

Para mejorar el control y el seguimiento de las actividades físicas realizadas por los usuarios, existen soluciones que utilizan la tecnología y el reconocimiento de patrones para realizar un seguimiento continuo a lo largo de toda la jornada. Para lograr esto, es necesario que las actividades puedan diferenciarse una de otra. En el trabajo de [Eunju et al., 2010] se proponen diversas formas de realizar la clasificación de actividades físicas. Entre los enfoques que se mencionan, se encuentra la extracción de información a través del análisis de vídeos. Las actividades a reconocer son representadas construyendo gramáticas probabilistas libres de contexto, donde las poses de las personas son el alfabeto de la gramática.

Este no ha sido el único tipo de solución propuesta que haga uso de la tecnología. También se han utilizado podómetros electrónicos para contabilizar actividades físicas [Bravata et al., 2007]. Uno de los principales usos de estos dispositivos es el cálculo de las distancias recorridas. Aunque, este enfoque deja de lado la contabilización de los tiempos de inactividad de sus usuarios.

Por otro lado, en [Casale et al., 2011] proponen el uso de dispositivos vestibles que cuenten con un acelerómetro. En dicho trabajo se analizan las señales capturadas del acelerómetro y son clasificadas mediante el uso del algoritmo de aprendizaje computacional supervisado, conocido como bosques aleatorios. Otra investigación [Iglesias et al., 2011] propone el uso de dos acelerómetros, uno en cada pierna, junto con un GPS para estimar el movimiento de los usuarios; en la Figura 1.1a se observan los dispositivos mencionados y la manera en la que

(a) Dispositivos y configuración utilizada en la investigación de [Casale et al., 2011]



(b) Dispositivos y diagrama de ubicación propuesta por [Tapia et al., 2007]

Figura 1.1: Ejemplos de dispositivos utilizados en distintos trabajos de investigación.

son utilizados por los usuarios. De manera similar, en [Tapia et al., 2007] utilizan cinco acelerómetros triaxiales inalámbricos, un monitor inalámbrico de frecuencia cardíaca y una computadora con un receptor inalámbrico. Estos dispositivos y la ubicación en la cual deben colocarse son mostrados en la Figura 1.1b. Entre las actividades reconocidas se encuentran: estar acostado, estar de pié, estar sentado, caminar, correr, andar en bicicleta y usar escaleras. Sin embargo, los resultados de los trabajos antes mencionados son difíciles de llevar a la práctica. Como se muestra en la Figura 1.1, los dispositivos vestibles pueden presentar inconvenientes con su portabilidad e independencia.

Teniendo en cuenta los problemas del uso de la tecnología vestible, se ha popularizado el uso de teléfonos inteligentes. Como muestra el trabajo de [Lee and Cho, 2011], en el cual se hace uso de un teléfono HTC® de la línea *Desire*, donde se analizan los datos producidos por el acelerómetro del teléfono y se reconocen las actividades utilizando modelos ocultos de Márkov. Entre las acciones clasificadas se encuentran estar de pié, caminar, correr, subir y bajar escaleras; además, se reconocen otras actividades más complejas, tales como comprar, tomar el autobús

y moverse (caminar rápido).

Dado que los teléfonos inteligentes cada vez incluyen más sensores, distintos investigadores han aprovechado esta característica. En la investigación de [Ortiz, 2014], se utiliza el acelerómetro y el giroscopio incluidos en el teléfono inteligente Samsung® *Galaxy S II*. A partir del uso de estos sensores obtiene información de las señales producidas, las cuales son procesadas y analizadas con distintos algoritmos de aprendizaje computacional supervisado, donde destacan las máquinas de soporte vectorial para clasificar seis actividades distintas (subir escaleras, bajar escaleras, estar acostado, estar de pié, estar sentado y caminar).

Ahora bien, el uso de los teléfonos inteligentes, los avances tecnológicos que éstos han tenido y la mejora en los servicios dio como resultado la penetración de los teléfonos en la vida cotidiana de las personas. Sin embargo, el procesamiento de la información que los teléfonos generan está limitado por sus recursos, ya que generalmente procesar datos generados por sus sensores integrados requiere de muchos recursos. Es por esto que se ha apostado por el procesamiento en la nube para tratar grandes cantidades de datos provenientes de estos dispositivos. Un ejemplo de este enfoque es la investigación presentada por [Srirama et al., 2011], donde se obtiene información a través del acelerómetro de un teléfono móvil y con el uso de la herramienta Hadoop se analizan los datos de manera distribuida. Dicho análisis se realiza con el algoritmo de *MapReduce*, el cual produce dos medidas estadísticas con las cuales se determina si el usuario se encuentra en movimiento o no. En un trabajo semejante, [Paniagua et al., 2012] también hacen uso del enfoque del cómputo distribuido para clasificar actividades físicas humanas. Para reconocerlas recurre a un acelerómetro y a tres algoritmos de clasificación (árboles de desición ID3, Bayes ingenuo y K-vecinos más cercanos), los cuales fueron implementados para funcionar en conjunto con el algoritmo *MapReduce* para realizar el procesamiento distribuido.

Por otro lado, en [Alsheikh et al., 2016] utilizan la herramienta Apache Spark para analizar la información de un conjunto de datos alojado en un repositorio especializado. Este conjunto de datos mencionado fue generado con el acelerómetro de teléfonos móviles y la información que contiene describe seis actividades distintas (caminar, caminar rápido, usar escaleras, sentarse, estar de pié y estar acostado). En el trabajo mencionado, se utilizan distintos algoritmos de aprendizaje computacional, destacando el *aprendizaje profundo* con entrenamiento greedy layer-wise, con el que logra una exactitud de predicción del 86.6 % y se obtiene una ganancia de aceleración de 4.1x al utilizar 64 núcleos en el clúster. Sin embargo, esta investigación no propone un modelo para la adquisición de la información que utilizan los algoritmos, ni describe una solución para darle seguimiento a dichas actividades.

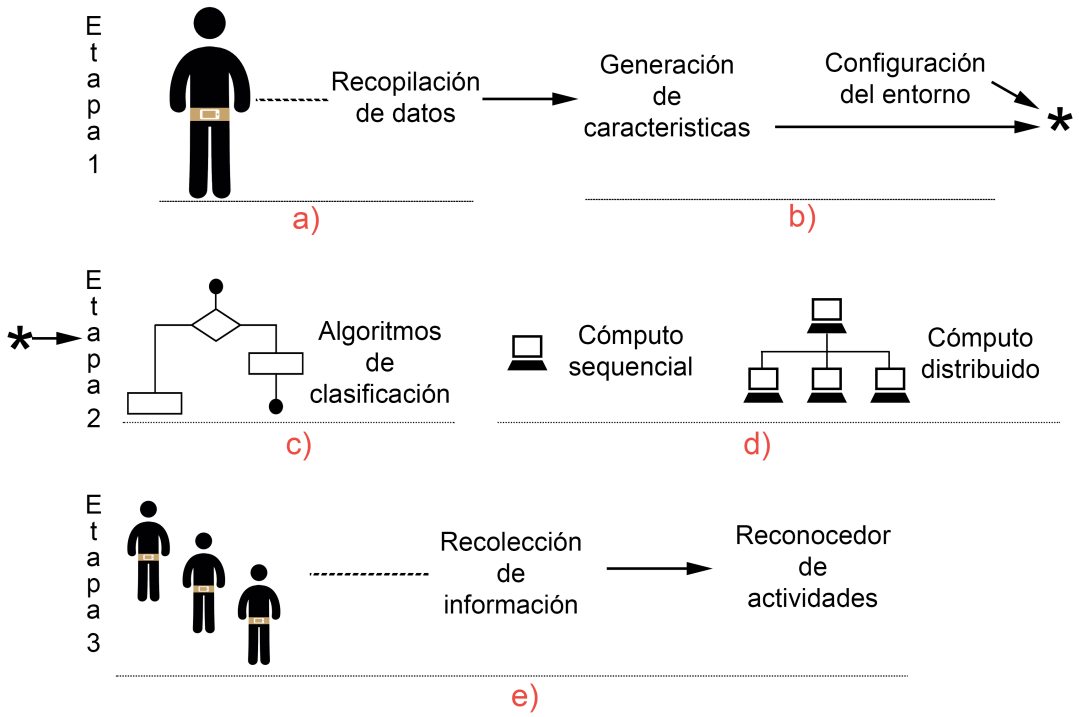


Figura 1.2: Etapas de la metodología. a) Adquisición de datos a través de un teléfono inteligente; b) generación de características con los datos adquiridos del teléfono; c) pruebas con algoritmos de clasificación; d) pruebas con versiones secuenciales y distribuidas del algoritmo seleccionado; e) aplicación que clasifica y monitorea actividades

1.7. Metodología

El desarrollo del proyecto de tesis consta de diversas tareas, el cual es mostrado en la Figura 1.2 y se describe a continuación.

(1) **Adquisición de los datos**. Para esto, se hace uso de un teléfono inteligente que cuente con acelerómetro y giroscopio (ver Figura 1.2a). Por otro lado, se solicitará de la colaboración de voluntarios para la adquisición de los datos de entrenamiento y de prueba, realizando las 6 actividades cotidianas básicas planteadas en los objetivos de este trabajo de tesis, como son: caminar, sentarse, pararse, acostarse, bajar escaleras y subir escaleras. Cada actividad se realiza en un

período aproximado de 5 minutos. El teléfono inteligente cuenta con una aplicación instalada (la aplicación de prueba) que se encarga de realizar la adquisición de la información en intervalos de 50Hz para su posterior tratamiento (ver Sección 3.4).

(2) **Preprocesamiento del conjunto de datos**. En esta parte, como se representa en la Figura 1.2b, se realiza la generación de 561 características a partir de las señales producidas por el acelerómetro y el giroscopio del teléfono inteligente. Para esto se sigue el proceso descrito en la Sección 3.4.

(3) **Configuración del ambiente de trabajo**: Se realiza la configuración del entorno de cómputo distribuido a utilizar, en este caso Apache Spark, con el cual se lleva a cabo la experimentación. Esta configuración es preámbulo de la etapa 2 mostrada en la Figura 1.2c.

(4) **Pruebas con diferentes algoritmos de clasificación**. Se hace uso del ambiente de trabajo mencionado en el paso anterior y los datos obtenidos en el paso 2. De esta manera, se realizan los experimentos con distintos algoritmos de clasificación incluidos en la herramienta de cómputo distribuido usada y se selecciona el que presente el mejor rendimiento, tal como se muestra en la Figura 1.2c. Finalmente, con el algoritmo seleccionado se realizan las pruebas de manera secuencial y distribuida, representadas en la Figura 1.2d, considerando como principal factor, el tiempo de ejecución.

(5) **Comparación de resultados**. Se realiza una comparativa minuciosa de los experimentos del paso 4, haciendo hincapié en el tiempo de ejecución y la exactitud mostrada por el algoritmo seleccionado.

(6) **Desarrollo y pruebas de aplicación de muestra**. Para mostrar el uso práctico de esta metodología y de las bondades del cómputo distribuido, se desarrolla una aplicación, usando los lenguajes de programación Python y Scala. Esta aplicación monitorea múltiples usuarios conectados al mismo tiempo y clasifica periódicamente las actividades planteadas en este proyecto de tesis, tal como se muestra en la Figura 1.2e. Una vez reconocida la actividad del usuario, la aplicación primero registra dicha actividad y después realiza un postprocesamiento, habilitando el envío de mensajes a la aplicación web para estimular e incentivar a los usuarios a realizar actividades no sedentarias [WHO, 2010, Strath et al., 2013].

Capítulo 2

Marco teórico

Hoy en día, la presencia de la tecnología es imprescindible en la vida cotidiana de las personas. Ésto se traduce en que cada minuto se genera una gran cantidad de datos. Un ejemplo de esto es la plataforma de video YouTube®, a la cual se suben alrededor de 400 horas de video por minuto[1]. Muchos de los datos subidos a Internet día con día parecen no poseer información importante, pero algunas empresas como Netflix®[Netflix, 2012], hacen uso de esta gran cantidad de información para proveer un servicio de calidad, y todo esto gracias al uso del aprendizaje computacional.

En esta sección se explica qué es el aprendizaje computacional, y una descripción de los tipos de aprendizaje computacional más empleados en la actualidad. Asimismo, se presenta una descripción de los métodos más comunes para la evaluación de estos algoritmos. Por otro lado, también se explican algunos conceptos básicos y recursos del cómputo distribuido desde el punto de vista de la herramienta Apache Spark, ya que ésta posee algoritmos de aprendizaje computacional integrados que son utilizados en este trabajo de investigación.

2.1. Aprendizaje computacional

El aprendizaje computacional es un conjunto de métodos que son capaces de encontrar y clasificar patrones automáticamente a partir de los datos, y usarlos para predecir datos futuros [Murphy, 2012]. Con base en estas predicciones, es posible tomar decisiones importantes. Éste es un campo altamente interdisciplinario que se basa en la teoría estadística, la informática, la ingeniería, la ciencia cognitiva, la

[1]Fuente: https://www.blog.google/topics/google-europe/improving-our-brand-safety-controls/

teoría de la optimización y muchas otras disciplinas de la ciencia y la matemática [Ghahramani, 2004].

2.1.1. Tipos de aprendizaje computacional

El aprendizaje computacional se divide generalmente en dos categorías principales:

- Aprendizaje supervisado
- Aprendizaje no-supervisado

Algunos autores e investigaciones recientes incluyen otras categorías como [Bian et al., 2009, Carlson et al., 2010]:

- Aprendizaje semi-supervisado
- Aprendizaje por refuerzo

Aprendizaje supervisado. En este tipo de aprendizaje la meta es aprender a mapear un conjunto de entradas x a un conjunto de salidas y, dado un conjunto de pares de entrada-salida $D = \{(x_i, y_i)\}_{i=1}^{N}$, donde D es llamado el conjunto de datos de entrenamiento y N es el número de ejemplos de entrenamiento. [Murphy, 2012].

En una configuración simple, cada entrada de entrenamiento x_i es un vector de dimensionalidad d, que representa variables como pueden ser la altura o el peso de una persona. Estas son llamadas **características** o **atributos**.

La forma de la variable de salida o de respuesta puede ser en principio cualquier cosa, pero la mayoría de los métodos asumen que y_i es una variable categórica o nominal de algún conjunto finito, $y_i \in \{1 \dots C\}$, o que y_i es un valor escalar real.

Aprendizaje no-supervisado. En este tipo de aprendizaje solamente se tienen las entradas $D = \{x_i\}_{i=1}^{N}$, y la meta es encontrar "regularidades" en los datos. Este es un problema menos definido, ya que se desconocen los patrones que se buscan, y no hay una métrica de error obvia que pueda ser usada para comparar los resultados encontrados. En otras palabras, no tenemos una salida deseada para cada entrada. [Murphy, 2012].

Aprendizaje semi-supervisado. Este tipo de aprendizaje se encuentra en un punto intermedio entre lo que se realiza en el aprendizaje supervisado y el no supervisado. La idea es utilizar algunos datos etiquetados para guiar la clasificación e ir aprendiendo junto con los datos no etiquetados. Formalmente, en este aprendizaje, se tienen datos de entrenamiento etiquetados $\{(x_i, y_i)\}_{i=1}^{N}$ y no etiquetados $\{x_i\}_{i=N+1}^{N+u}$, y un predictor $f : x \to y, f \in F$, donde F es un

espacio de hipótesis. El objetivo es enseñar al predictor f a que prediga la clase correspondiente de los patrones no etiquetados de mejor manera en comparación si solo hubiera aprendido de los datos de entrenamiento etiquetados [Zhu and Goldberg, 2009].

Aprendizaje por refuerzo. Este tipo de aprendizaje es útil para aprender a actuar o comportarse cuando se le dan señales ocasionales al sistema de recompensa o de castigo. La máquina interactúa con el ambiente produciendo acciones $a_1, a_2, \ldots, a_n$. Estas acciones afectan el estado del ambiente, lo que da como resultado que la máquina reciba algunas recompensas escalares (o castigos) $r_1, r_2, \ldots, r_m$. El objetivo de la máquina es aprender a actuar de una manera que maximice las futuras recompensas que recibe (o minimice los castigos) a lo largo de su vida. Este tipo de aprendizadje está estrechamente ligado a los campos de teoría de la decisión (en estadística y ciencia de la administración) [Sutton and Barto, 1998].

2.1.2. Algoritmos utilizados en el aprendizaje supervisado

Diversos modelos del aprendizaje supervisado han sido desarrollados para la resolución de tareas, tales como la clasificación y regresión [Bishop, 2006]. Entre estos enfoques se encuentran los modelos deterministas que tratan de encontrar relaciones entre eventos. Otros, son probabilísticos y asumen que los datos están distribuidos usando una función de probabilidad. A continuación se describen algunos de los algoritmos más utilizados.

- **Árboles de decisión (Decision Tree):** Es un modelo predictivo que hace elecciones a partir de un conjunto de reglas jerárquicas relacionadas con los datos de entrada. Se han propuesto diferentes versiones tales como ID3 y C5.4 [Quinlan, 1986]. Es un enfoque común para la clasificación, debido a que los modelos resultantes son fácilmente interpretables por los seres humanos (debido a su estructura arbórea intrínseca).

- **Bosque aleatorio (Random Forest):** es un meta-clasificador del aprendizaje computacional que se construye usando un conjunto de árboles de decisión. La clase predicha es elegida como la más frecuente entre la salida de cada árbol de decisión [Breiman, 2001].

- **Bayes ingenuo (Naïve Bayes):** es un clasificador probabilístico basado en el teorema de Bayes que predice la clase de una muestra dada, asumiendo un modelo de probabilidad subyacente de los datos y haciendo suposiciones de independencia entre sus características. Aunque su formulación es bastante simple, ha demostrado un buen desempeño en diversas aplicaciones [Bishop,

2006]. Por ejemplo, cuando se supone que los datos tienen una distribución Gaussiana, es posible entrenar el modelo calculando únicamente la media y la varianza de los datos de entrada.

- **Red neuronal artificial (Artificial Neural Network):** Es un enfoque de aprendizaje computacional con inspiración biológica. Simula cómo el cerebro y su sistema nervioso (compuesto de neuronas interconectadas) es capaz de aprender de la experiencia y capturar la estructura subyacente de los datos. Las neuronas se organizan en una estructura de capas y tienen pesos asociados que son capaces de adaptar su salida con base en los datos de entrenamiento a través de una función de costos. Este enfoque ha demostrado un buen desempeño en muchas aplicaciones [LeCun et al., 1989], incluyendo problemas no lineales. Su principal desventaja radica en la necesidad de un gran conjunto de datos para su etapa de entrenamiento. El perceptrón Multicapa (MLP, del inglés Multilayer perceptron) es un modelo popular de Red Neuronal Artificial que mapea un conjunto de características hacia una categoría ó clase. Éste ha sido implementado en la mayoría de las herramientas convencionales de aprendizaje computacional, tales como MATLAB® ó Weka.

- **Máquinas de soporte vectorial (Support Vector Machine):** Es un clasificador no lineal sofisticado. Separa los datos de entrada asignándolos a un espacio de características de alta dimensión, donde se construye un hiperplano. Este hiperplano crea una superficie de decisión que tiene una distancia máxima a los puntos más cercanos en el espacio de características [Cortes and Vapnik, 1995].

- **Árbol de potenciación del gradiente (Gradient Boosted tree):** Es una técnica usada para construir modelos basados en árboles predictivos (árboles de decisión). Esta técnica construye clasificadores débiles que tienen buen rendimiento en regiones locales pero su desempeño general no es bueno. Estos modelos actualizan los pesos del conjunto de entrenamiento basados en los clasificadores previamente construidos para mejorar la importancia de los datos mal clasificados. Finalmente, agrupan todos los clasificadores débiles en uno más fuerte para formar un modelo final que generaliza a todas las regiones [Friedman, 2002].

2.1.3. Evaluación de rendimiento

Una parte importante de los modelos de aprendizaje computacional es su validación. La manera predominante de realizar esto es mediante el análisis

Cuadro 2.1: Matriz de confusión para la clasificación binaria

	Predicción del clasificador		
Clase real	Positivo	Negativo	
Positivo	Positivo verdadero (tp)	Negativo falso (fn)	p
Negativo	Positivo falso (fp)	Negativo verdadero (tn)	n
	p'	n'	

estadístico usando los datos experimentales disponibles.

Resultados de clasificación

Los algoritmos de aprendizaje computacional pueden etiquetar a los patrones de diversas formas. Cuando un patron $x = \{x_1 \dots x_n\}$ se clasifica, éste puede ser asignado a alguna de las clases $\{C_1 \dots C_m\}$, por lo que se pueden tener distintos casos [Sokolova and Lapalme, 2009]:

- **Binaria:** El patrón puede ser clasificado en una clase C_1 o en C_2. Donde ambas clases (C_1, C_2) son mutuamente excluyentes.
- **Multiclase:** El patrón puede ser clasificado en una sola clase, de las m clases que son mutuamente excluyentes.
- **Multi-etiquetado:** El patrón de entrada puede ser clasificado en varias de las m clases con distintos grados de pertenencia a cada una.

Matriz de confusión

La matriz de confusión $A = [A(i, j)]$ está definida de tal manera que su elemento $A(i, j)$ es el número de elementos cuya clase verdadera es i pero el clasificador estima que son de la clase j.

En el Cuadro 2.1 se muestra la forma de representar una clasificación binaria mediante la utilización de una matriz de confusión.

Para generalizar a más de 2 clases, suponiendo que hay m clases disponibles, una matriz de confusión típica consiste en una matriz cuadrada de tamaño $m \times m$, donde las clasificaciones erróneas son visibles fuera de la diagonal. Esto se puede ver en el Cuadro 2.2. Por ejemplo, si consideramos un conjunto de datos compuesto de n patrones, donde cada uno corresponde a un par ordenado de características extraídas y su etiqueta de actividad correspondiente $(x_i, y_i) \forall i \in \{1 \dots n\}, x_i \in \mathbb{R}^d$, y $y_i \in \{1 \dots m\}$, podemos obtener C, cuando se proporcionan las etiquetas predichas $f_c(x)$.

Cuadro 2.2: Matriz de confusión para la clasificación multiclase con m distintas clases

	Predicción del clasificador			
Clase real	Clase 1	Clase 2	...	Clase m
Clase 1	$C_{1,1}$	$C_{1,2}$	...	$C_{1,m}$
Clase 2	$C_{2,1}$	$C_{2,2}$	...	$C_{2,m}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\ddots$
Clase m	$C_{m,1}$	$C_{m,2}$	...	$C_{m,m}$

Las filas representan las clases reales y las columnas las clases predichas. Por lo tanto, cada celda $C_{i,j}$ de la matriz muestra el número de instancias de clase i, que se predijo como clase j. Esto es, los valores en la diagonal de la matriz $C_{i,i} = tp_i$ son predicciones correctas y los que se encuentran fuera de dicha diagonal son errores en la clasificación. Las siguientes medidas pueden ser obtenidas a partir de una matriz de confusión para problemas de clasificación binaria o multi-clase.

Exactitud (Accuracy): Se obtiene de la proporción de resultados verdaderos (positivos verdaderos y negativos verdaderos) entre el número total de ejemplos examinados.
Para la clasificación binaria es posible calcular la exactitud de clasificación usando los datos de la matriz de confusión (Véase Cuadro 2.1):

$$Exactitud(Accuracy) = \frac{tp+tn}{tp+fn+fp+tn}$$

y para clasificación multiclase a partir de la matriz de confusión muticlase (Véase Cuadro 2.2):

$$Exactitud(Accuracy) = \frac{\sum_{i=1}^{m} C_{i,i}}{\sum_{i=1}^{m}\sum_{j=1}^{m} C_{i,j}}$$

Precisión: El número de ejemplos positivos clasificados correctamente dividido por el número de ejemplos etiquetados como positivos.
Para clasificación binaria:

$$Precisión = \frac{tp}{tp+fp}$$

Y para clasificación multiclase:

$$Precisión_i = \frac{C_{i,i}}{\sum_{j=1}^{m} C_{j,i}}$$

Cobertura (Recall): El número de ejemplos positivos clasificados correctamente dividido por la suma el número de ejemplos verdaderos positivos más los negativos falsos en los datos.
Para clasificación binaria:

$$Cobertura(Recall) = \frac{tp}{tp+fn}$$

Y para clasificación multiclase:

$$Cobertura(Recall)_i = \frac{C_{i,i}}{\sum_{j=1}^{m} C_{i,j}}$$

F1-score: La media armónica de la precisión y el recall. Por lo cual, toma en cuenta los positivos falsos y los negativos falsos.

$$F_1 = 2 * \frac{Precisión \times Recall}{Precisión + Recall}$$

2.2. Cómputo distribuido

Debido a la gran cantidad de información que los sistemas actuales tienen que procesar, no es extraño el creciente interés por el computo distribuido. El presente trabajo pretende hacer uso de este enfoque, por lo que en este capítulo se introduce el *middleware* conocido como Apache Spark.

El campo de cómputo distribuido cubre todos los aspectos del cómputo y acceso a la información a través de múltiples elementos de procesamiento conectados por alguna forma de red de comunicación. Un sistema distribuido es una colección de entidades de cómputo independientes que cooperan para resolver un problema, que de resolverlo individualmente tendría un costo muy elevado.

Un sistema distribuido según [Kshemkalyani and Singhal, 2011], puede ser caracterizado como una colección de procesadores autónomos que se comunican a través de una red y tienen las siguientes características:

- *No comparten un reloj físico.*
- *No comparten memoria.*
- *Tienen una separación geográfica.*
- *Tienen autonomía y pueden ser heterogéneos.*

2.2.1. Apache Spark

Apache Spark es una plataforma de computación en clúster diseñada para ser rápida y de propósito general [Karau et al., 2015].

Fue desarrollada en la Universidad de Berkeley en el año 2009 y actualmente es mantenida por la Apache Software Fundation. La plataforma se encarga de programar, distribuir y monitorear aplicaciones que consisten en muchas tareas realizadas en diversas máquinas de trabajo.

Apache Spark es un proyecto de código abierto, y en su lanzamiento se consideró como el primer software de programación distribuida con este tipo de licencia. Una de las ventajas de Spark es que facilita el manejo de grandes cantidades de información gracias a su arquitectura, la cual está inspirada en el modelo de *MapReduce* de Apache Hadoop [Malak and East, 2016].

Spark extiende el popular modelo MapReduce para soportar eficientemente más tipos de operaciones, incluyendo consultas interactivas. Una de las principales características que Spark ofrece es la mejora en velocidad con respecto a su antecesor Hadoop, debido a su capacidad al ejecutar los cálculos en la memoria, además de ser más eficiente que MapReduce para las aplicaciones complejas que se ejecutan en el disco duro.

Spark está diseñado para ser altamente accesible, ofreciendo API's en Python, Java y Scala, y diversas bibliotecas integradas, así como la integración con otras herramientas de *Big Data*. En particular, Spark puede ejecutarse en clústeres Hadoop y acceder a cualquier fuente de datos de otros sistemas de archivos como el *Sistema de Archivos Distribuidos de Hadoop* (por sus siglas en inglés, HDFS, Hadoop Distributed File System).

Spark está compuesto de varios componentes, los principales se pueden ver en la figura 2.1, mismos que se presentan a continuación:

Spark Core contiene la funcionalidad básica de Spark, incluyendo componentes para la programación de tareas, administración de memoria, recuperación de fallas, interacción con sistemas de almacenamiento, etc.

Spark SQL es el paquete de Spark para trabajar con datos estructurados. Permite consultar datos a través de SQL y soporta muchas fuentes de datos.

Spark Streaming es un componente Spark que permite el procesamiento de flujos de datos en vivo.

MLlib es una biblioteca que proporciona múltiples tipos de algoritmos de aprendizaje computacional, incluyendo clasificación, regresión, agrupación y filtrado

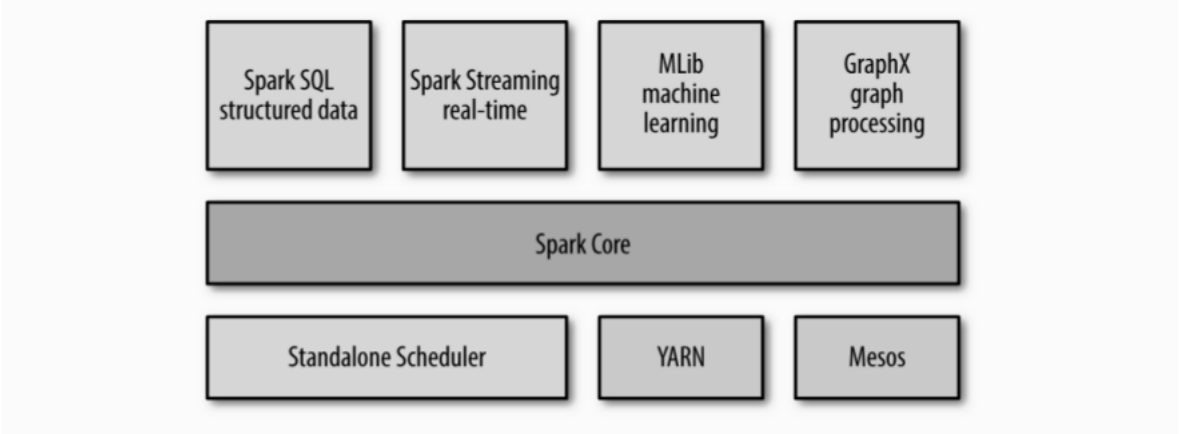


Figura 2.1: Componentes de Spark.

colaborativo, así como funciones de soporte como la evaluación de modelos y la importación de datos. Todos estos métodos están diseñados para escalar el procesamiento a través de un clúster.

Cluster Managers es el encargado de lograr escalar eficientemente los nodos y maximizar la flexibilidad. Además, permite la integración con una variedad de gestores de clúster, incluyendo Hadoop YARN, Apache Mesos y un planificador de clúster simple incluido en Spark llamado *Standalone Scheduler* (Planificador Independiente).

Funcionamiento

Spark es responsable de convertir un programa en unidades de ejecución física llamadas *tareas*. Spark crea un *Grafo Acíclico Dirigido* (DAG), que es utilizado para planificar la ejecución de una aplicación mediante el uso del modelo de evaluación *Lazy* [Malak and East, 2016]. Esto es característico de la programación funcional, de tal forma que éstas se ejecutan en un clúster en particular con cierto grado de paralelismo.

Cuando Spark transforma el grafo en un plan de ejecución, también realiza varias optimizaciones como "segmentar" transformaciones para después fusionarlas y convertir el plan de ejecución en un conjunto de *etapas*. Cada etapa, a su vez, consiste en múltiples tareas [Karau et al., 2015].

Ventajas y desventajas de Spark

Entre las ventajas que Spark ofrece se encuentran:

- La flexibilidad de la herramienta que integra soporte de forma nativa para tres lenguajes de programación, que son: a) Python, b) Java, y c) Scala (spark-shell y spark-submit).
- La integración con Hadoop, Mesos y otras herramientas.
- La velocidad en el rendimiento, además de la eficiencia cuando se escribe en disco.
- Las bibliotecas de aprendizaje computacional y consultas SQL, por mencionar algunas.

Entre las desventajas del uso de Spark se encuentran:

- La memoria, la capacidad de ésta se puede convertir en un cuello de botella y una limitación muy importante, ya que con Spark los datos se procesan lo más rápido posible y mantenerlos en la memoria conlleva un alto costo.
- No se tiene procesamiento en tiempo real, lo más cercano al procesamiento de datos generados en vivo que se realiza a través de Spark Streaming.
- Otra desventaja es que el tiempo de ejecución sigue dependiendo del tráfico y las características de la red, al igual que sucede con otros administradores de clústeres.

2.2.2. La aceleración como métrica de comparación

Para conocer el grado en que un sistema de ejecución paralelo mejora con respecto a un sistema de ejecución secuencial, se hace uso de algunas leyes. Particularmente para sistemas paralelos y con arquitecturas multinúcleo se encuentra la Ley de Amdahl. A continuación se presenta una breve descripción de dicha ley.

Ley de Amdahl

La ley de Amdahl [Amdahl, 1967] muestra la aceleración potencial de un programa usando múltiples procesadores comparado con un solo procesador.

$$Aceleración_{Amdahl} = \frac{1}{(1-f) + \frac{f}{m}} \tag{2.1}$$

donde f es la porción de la tarea que puede ser paralelizada y m es el número de procesadores.

Esta ley puede ser generalizada para evaluar cualquier proporción de código que puede mejorarse en tiempo de ejecución en un sistema computacional [Stallings, 2009]. Dicha mejora conlleva a un intercambio de aceleración de acuerdo a la relación:

$$\begin{aligned} Aceleración(P) &= \frac{Rendimiento\ antes\ de\ la\ mejora}{Rendimiento\ después\ de\ la\ mejora} \\ &= \frac{Tiempo\ antes\ de\ la\ mejora}{Tiempo\ después\ de\ la\ mejora} \end{aligned} \tag{2.2}$$

En el caso de las aplicaciones distribuidas [Zuberek, 2011], el total de la carga de trabajo es dividida entre los procesadores del sistema con una expectativa de ejecución simultánea de las tareas distribuidas. Una de las principales características de rendimiento de una aplicación distribuida es su aceleración, la cual, usualmente se define como la proporción del tiempo de ejecución de la aplicación en un solo procesador $T(1)$ contra el tiempo de ejecución de la misma carga de trabajo en un sistema conformado por P procesadores, $T(P)$:

$$Aceleración(P) = \frac{T(1)}{T(P)} \tag{2.3}$$

Esta función de aceleración se deriva de la ley de Amdahl y depende de una serie de factores, los cuales incluyen el número de procesadores y su rendimiento, los tiempos de las conexiones entre los procesadores, el algoritmo usado para la distribución de la carga de trabajo, etc. Algunos de estos factores pueden ser difíciles de tener en cuenta para estimar la aceleración mediante la Ley de Amdahl de una aplicación distribuida. Por lo cual, en muchos casos, se usa un análisis simplificado. Dicho análisis se basa en diversas suposiciones con la distribución uniforme de la carga de trabajo entre los procesadores, tiempos de comunicación constante, etc.

2.3. Procesamiento de señales de actividades físicas humanas

El uso del acelerómetro y del giroscopio se ha popularizado para la obtención de información de actividades físicas humanas. Una manera de tratar las señales producidas por estos sensores es descrita a continuación.

Para esto son aplicados una serie de filtros para el acondicionamiento de las señales obtenidas directamente del acelerómetro $a_r(t)$ y del giroscopio $\omega_r(t)$ en

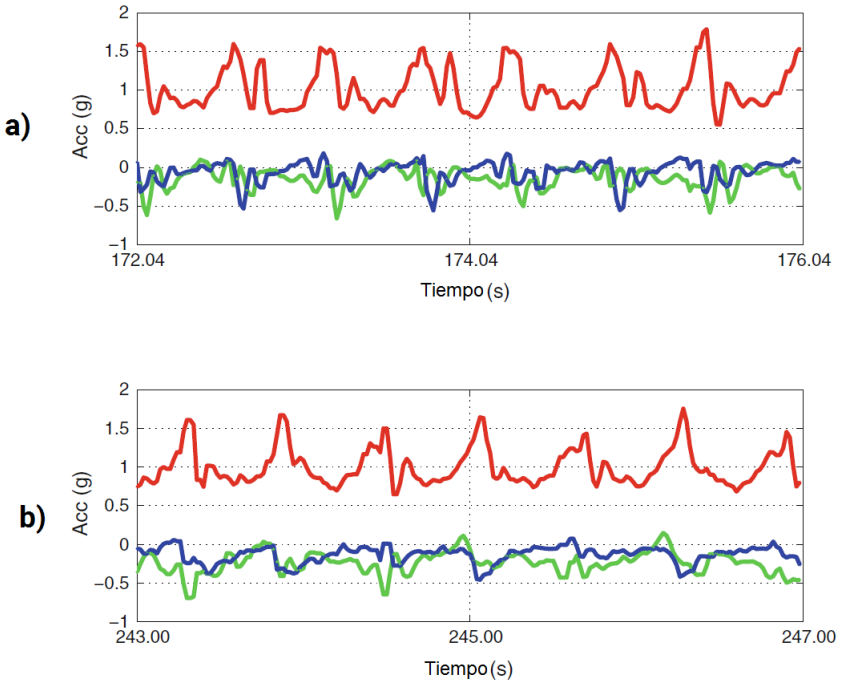


Figura 2.2: Ejemplo de señales de inercia del acelerómetro al realizar dos actividades: a) caminar, b)subir escaleras.

el tiempo t. En la Figura 2.2 se observa un ejemplo de señales de inercia del acelerómetro al realizar dos actividades distintas. En principio, lo que se hace es reducir el ruido de las señales, aplicando un filtro de mediana de tercer orden y posteriormente un filtro Butterworth de pasa bajas de tercer orden, cuya frecuencia de corte es 20 Hz. Esta frecuencia se selecciona a partir del trabajo presentado en [Karantonis et al., 2006] en el que se establece que el espectro de energía del movimiento del cuerpo humano se encuentra dentro del intervalo de 0 a 15 Hz. Luego de aplicar los filtros mencionados, se obtuvieron señales limpias de aceleración total triaxial $a_\tau(t)$ y velocidad angular $\omega_\tau(t)$. Esta serie de filtros aplicados es representado por la función H_1.

Para obtener sólo información de la aceleración del cuerpo, es necesario separar la fuerza de gravedad de la señal de aceleración. Si se asume que la componente gravitatoria sólo afecta a las frecuencias más bajas, entonces es posible obtener la aceleración corporal $a(t)$ aplicando un filtro pasa altas, con una frecuencia de corte de 0.3Hz [Karantonis et al., 2006]. Esta serie de filtros es representada por la función H_2.

Finalmente, la señal de gravedad $g(t)$ se puede encontrar usando la aceleración

Cuadro 2.3: Principales operaciones de procesamiento de señales aplicadas a las señales inerciales de los sensores de los teléfono inteligentes.

Nombre	Símbolo	Fórmula
Aceleración total	$a_\tau(t)$	$H_1(a_r(t))$
Aceleración del cuerpo	$a(t)$	$H_2(a_\tau(t))$
Gravedad	$g(t)$	$a_\tau(t) - a(t)$
Jerk del cuerpo	$a'(t)$	$d(a(t))/dt$
Magnitud de la aceleración del cuerpo	$a_{mag}(t)$	$a(t)$
Velocidad angular	$\omega(t)$	$H_2(H_1(\omega_r(t)))$
Aceleración angular	$\omega'(t)$	$d(\omega(t))/dt$
Magnitud de la velocidad angular	$\omega_{mag}(t)$	$\omega(t)$

total y la aceleración del cuerpo de la siguiente forma:

$$g(t) = a_\tau(t) - a(t) \tag{2.4}$$

Diversos trabajos han separado la aceleración corporal y la gravedad de una manera similar, como en [Bruno et al., 2012].

Además, a la señal $\omega_\tau(t)$ también se aplica un filtro pasa altas para eliminar cualquier polarización que afecte al giroscopio, ya que es uno de los posibles errores de calibración que se pueden encontrar en estos sensores, obteniendo así la señal $\omega(t)$.

El resultado después del filtrado de ruido y la segmentación de la señal consiste en tres señales: $a(t)$, $g(t)$ y $\omega(t)$. Estos proveen información suficiente sobre el movimiento corporal del usuario, la orientación de la persona (por ejemplo, esta información suele ser útil para la distinción entre las actividades estar acostado y estar de pie) y los patrones de movimiento que las personas tienen al realizar algunas actividades (por ejemplo, para reconocer actividades no sedentarias). También, se realiza una transformación adicional a $a(t)$ y $\omega(t)$, derivándolas con respecto al tiempo, mismas que han mostrado ser relevantes y han sido utilizadas exitosamente en algunas aplicaciones, como en la detección de los estados Encendido (energéticos y con facilidad para moverse)/Apagado (rígidos, lentos y con dificultad para moverse) en pacientes con enfermedad de Parkinson [Samà et al., 2011]. Por último, la magnitud (norma euclidiana) también se aplica a las señales inerciales triaxiales para obtener $a_{mag}(t)$ y $\omega_{mag}(t)$. Una compilación de las transformaciones de las señales se presenta en el Cuadro 2.3.

Capítulo 3

Desarrollo del proyecto

En este capítulo se presenta el desarrollo de este proyecto de tesis. En las primeras tres secciones, se presenta una descripción del ambiente de trabajo y los módulos de software desarrollados con el propósito de verificar la veracidad de la hipótesis. Posteriormente, en la cuarta sección se detalla la arquitectura y configuración de la aplicación de muestra para el seguimiento de actividades humanas.

3.1. Especificaciones de hardware y software

En esta sección se presentan el hardware y el software que se utilizaron para el desarrollo de este proyecto. Además, se presentan las aplicaciones y bibliotecas desarrolladas especialmente para la resolución del problema.

El desarrollo y los experimentos del presente proyecto se realizaron utilizando cuatro servidores, los cuales conforman el clúster computacional. El servidor maestro del clúster es el servidor cuyas características están descritas en el Cuadro 3.1. Mientras que los nodos del clúster son 3 servidores con las características presentadas en el Cuadro 3.2.

Para la implementación de las aplicaciones de aprendizaje computacional y los sistemas distribuidos se utilizó el lenguaje de programación Scala versión 2.11, la herramienta SBT [1] versión 1.0.2, Apache Spark versión 2.2.0 y Apache Hadoop versión 2.8.1.

Para la adquisición de datos se usó un teléfono LG® G3[2]. Las características del télefono son mostradas en el Cuadro 3.3. Debido a que el software que utiliza es Android, las aplicaciones móviles utilizadas durante el desarrollo de los

[1]Fuente: https://www.scala-sbt.org/

[2]Fuente: http://www.lg.com/mx/celulares/lg-D855P

Cuadro 3.1: Ficha técnica del servidor maestro

Modelo	Dell® Precision T7600
Memoria	16 GB RDDR3
Disco duro	Disco de estado sólido SATA III de 120 GB
Procesador	Intel® Xeon® E5-2620 de 2.00GHz x12

Cuadro 3.2: Ficha técnica de los servidores nodos

Modelo	Torre Dell® PowerEdge T630
Memoria	16 GB RDDR4
Disco duro	Disco de estado sólido SATA III de 120 GB
Procesador	Dual Intel® Xeon® E5-2630 v3 de 2.40GHz x16

experimentos se elaboran usando el software Android Studio. Estas aplicaciones están escritas en el lenguaje de programación Java y se hace uso del framework de Android con el que se desarrollaron las interfaces gráficas de usuario.

En el caso de software a utilizar en el proyecto, la lista es la siguiente:

- Apache Spark versión 2.2.0
- Apache Hadoop versión 2.8.1
- Apache Kafka versión 1.0.0
- Apache Cassandra versión 5.0.1
- PostgreSQL versión 9.6.8
- IntelliJ IDEA Ultimate 2017.3
- Android Studio 3.0.1
- Java Runtime Enviroment OpenJDK versión 1.8.0
- Scala versión 2.11.8
- Python versión 2.7.14
- Python PIP versión 9.0.1

Cuadro 3.3: Ficha técnica del télefono LG G3

Dimensiones físicas	146.3 x 74.6 x 93 mm, 153.7 gramos
Pantalla	5.5 pulgadas
Resolución	1440x2560 (538 ppp)
Procesador	Qualcomm Snapdragon 801, quad-core a 2.46 GHz
RAM	2 GB
Memoria interna	16 GB
Versión software	Android 4.4.2 (KitKat)
Conectividad	LTE, NFC, WiFi 802.11a/b/g/n/ac, BT 4.0 LE, USB 2.0
Acelerómetro	InvenSense LGE Accelerometer con potencia de 0.45mA
Giroscopio	InvenSense LGE Gyroscope con potencia de 3.2mA

La instalación y configuración de estos programas está descrita en la Sección 3.2.

3.2. Configuración del ambiente de trabajo

El entorno de trabajo fue configurado sobre el sistema operativo Ubuntu 16.04.3 LTS Server (Xenial Xerus) versión de 64 bits. A continuación se presenta una lista del software distribuido necesario, una breve descripción y la liga con los detalles para su instalación:

- Apache Spark versión 2.2.0 [1]: Herramienta de cómputo distribuido, que además integra algoritmos de aprendizaje computacional.
- Apache Hadoop versión 2.8.1 [2]: Herramienta que incluye un sistema de archivos distribuidos.
- Apache Kafka versión 2.11.1 [3]: Herramienta de cola distribuida de mensajes.

[1]Fuente: https://spark.apache.org/docs/2.2.0/spark-standalone.html

[2]Fuente: https://hadoop.apache.org/docs/r2.8.0/hadoop-project-dist/ hadoop-common/SingleCluster.html

[3]Fuente: https://kafka.apache.org/quickstart

Los Ambientes de Desarrollo Integrado (IDE, del inglés Integrated Development Environment) utilizados son IntelliJ IDEA[1] e IDE Android Studio[2], ambos productos de la empresa JetBrains®. El primero, es usado para el desarrollo del clasificador, mientras que, el segundo es utilizado para implementar la aplicación para el teléfono móvil, que se encarga de obtener información de los sensores. Para la instalación de la herramienta IntelliJ IDEA se utilizaron las instrucciones descritas en [JetBrains, 2017] y para Android Studio se siguió el procedimiento descrito en [Google, 2013].

Por otro lado, el lenguaje de programación Python es empleado para la implementación de la aplicación Web. El ambiente de ejecución de Java (Java Runtime Enviroment) es utilizado por Spark para su ejecución; el gestor de base de datos SQL Postgres se emplea para llevar el control de los usuarios registrados en la aplicación Web y el gestor de base de datos NoSQL Apache Cassandra lleva el registro de las actividades diarias realizadas por los usuarios del sistema. Estas herramientas se instalan a través del gestor de paquetes incluido en Ubuntu, Advanced Packaging Tool (APT)[3]. Por otro lado, las bibliotecas para el lenguaje Python son instaladas a través del administrador de paquetes PIP[4].

Para el lenguaje Scala, la instalación y administración es realizada por la herramienta IntelliJ IDEA. Mientras que, la versión de Java para el desarrollo de la aplicación para el teléfono móvil es administrada e instalada por la herramienta Android Studio, ambas herramientas mencionadas con anterioridad.

3.3. Módulos del proyecto

El proyecto consta de tres etapas para lograr los objetivos planteados. Dichas etapas se mencionan en la Sección 1.7 y se muestran en la Figura 1.2. La primera etapa se centra en la obtención de información y su procesamiento para la generación de un conjunto de datos. La segunda etapa se divide en dos fases, la primer fase se encarga de la obtención del mejor clasificador junto con sus parámetros, mientras que, la segunda se enfoca en la realización de pruebas utilizando cómputo secuencial y cómputo distribuido. Finalmente, la tercera etapa es la aplicación de muestra, la cual es detallada en la Sección 3.5.

El proceso utilizado para realizar la etapa 1 es mostrado en la Figura 3.1. Para realizar este proceso se requieren de dos módulos, una aplicación móvil para la captura de información de los sensores y un generador de características.

[1]Fuente: https://www.jetbrains.com/idea/

[2]Fuente: https://developer.android.com/studio/index.html

[3]Fuente: https://help.ubuntu.com/lts/serverguide/apt.html

[4]Fuente: https://pypi.python.org/pypi/pip

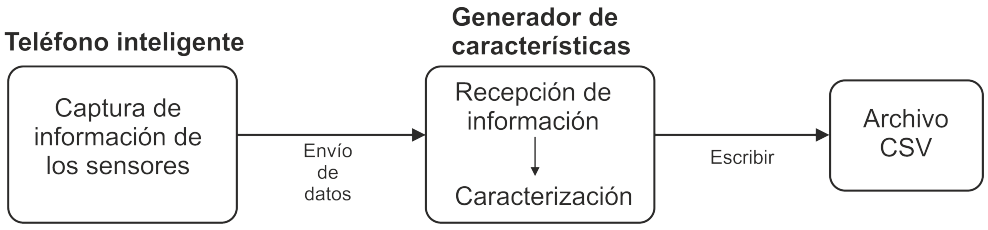


Figura 3.1: Diagrama del proceso utilizado en la etapa 1 para la recopilación de datos y su caracterización.

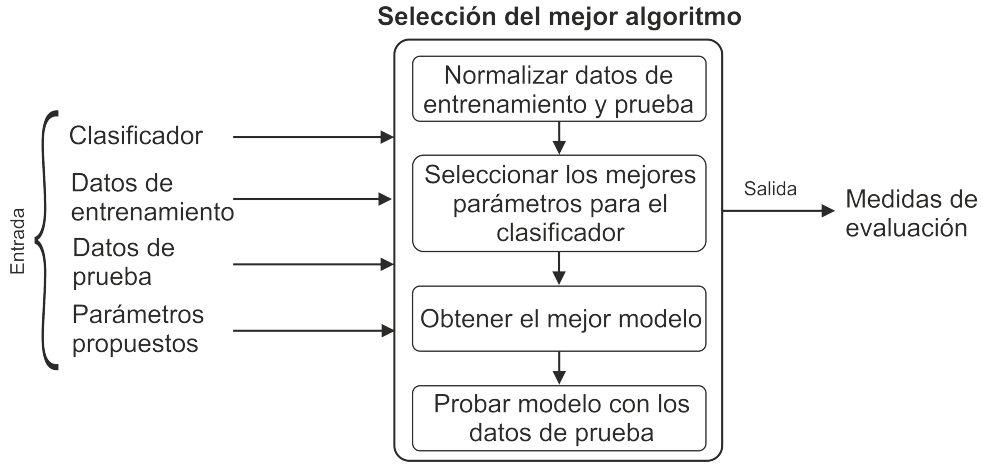
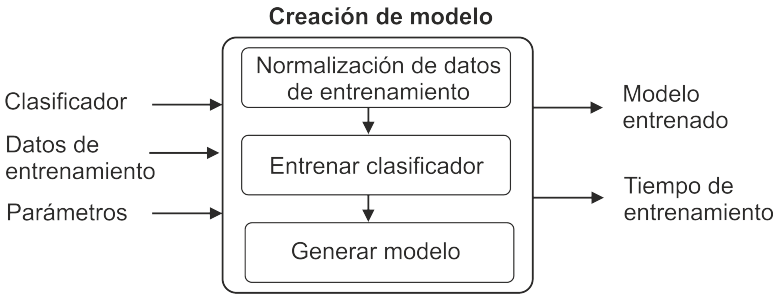


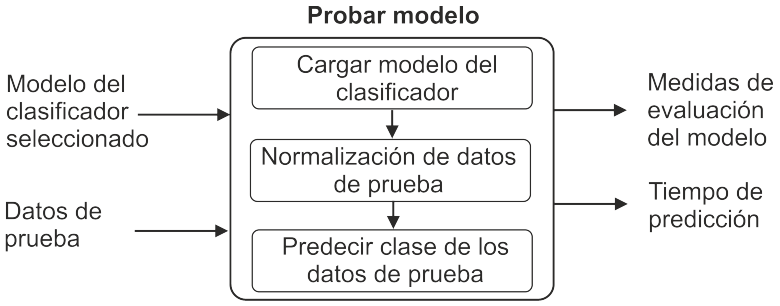
Figura 3.2: Diagrama del proceso utilizado en la etapa 2 para la selección del mejor algoritmo.

La aplicación móvil se encarga de adquirir información del acelerómetro y del giroscopio de un teléfono inteligente. Esta información es enviada al módulo "generador de características" para su procesamiento y caracterización. Al término del procesamiento, se generan 561 características por cada muestra, las cuales son almacenadas en un archivo CSV para su uso futuro.

La segunda etapa está dividida en dos fases. La primera tiene por objetivo la selección del algoritmo de aprendizaje computacional que mejor clasifique las muestras. Para esto, se desarrolla el módulo mostrado en la Figura 3.2. En dicho módulo, se entrenan diversos algoritmos de clasificación, se obtienen los parámetros que mejoran el rendimiento de cada clasificador, así como diversas medidas de validación que son comparadas para elegir el mejor clasificador. En la segunda fase, se realizan experimentos con cómputo secuencial y distribuido utilizando el algoritmo elegido en la fase anterior. Los experimentos consisten en entrenar un clasificador y probarlo mientras se utilizan distintas configuraciones



(a) Diagrama del proceso para la creación de un modelo de un clasificador.



(b) Diagrama del proceso para probar el clasificador generado.

Figura 3.3: Flujo de trabajo para los experimentos usando cómputo secuencial y distribuido. Ambos pertenecientes a la etapa 2 de la metodología.

en el clúster computacional. Estos experimentos proveen de información acerca del impacto que tiene el uso del cómputo distribuido en los tiempos de ejecución de los algoritmos de aprendizaje computacional. El flujo de trabajo de dichos experimentos es presentado en la Figura 3.3.

3.4. Generación de características

El tratamiento de las señales adquiridas con el teléfono inteligente se realiza siguiendo el proceso descrito en la Sección 2.3. Además, se presenta el proceso para la caracterización, el cual fue propuesto en el trabajo de [Ortiz, 2014].

Siguiendo lo pasos mencionados, las señales son segmentadas en ventanas deslizantes en tiempo. El muestreo se realiza dentro de una ventana de actividad de 2.56 segundos debido a las siguientes razones:

- La cadencia de una persona promedio caminando está dentro del intervalo

Cuadro 3.4: Señales utilizadas para la generación de características y el dominio en el que son analizadas.

Nombre	Tiempo	Frecuencia
Aceleración del cuerpo-XYZ	1	1
Aceleración gravitacional-XYZ	1	0
Jerk de la aceleración del cuerpo-XYZ	1	1
Velocidad angular del cuerpo-XYZ	1	1
Jerk de la velocidad angular del cuerpo-XYZ	1	0
Magnitud de la aceleración del cuerpo	1	1
Magnitud de la aceleración gravitacional	1	0
Magnitud del Jerk la de aceleración del cuerpo	1	1
Magnitud de la velocidad angular del cuerpo	1	1
Magnitud de la aceleración angular del cuerpo	1	1

[90, 130] pasos/min [BenAbdelkader et al., 2002], es decir, un mínimo de 1.5 pasos/segundo.

- Es preferible tener al menos un ciclo completo de pasos, es decir, un mínimo de 2 pasos.
- Para considerar a las personas con una cadencia más lenta se utiliza una velocidad igual al 50 % de la cadencia humana promedio (2 pasos en un periodo de 2.56 segundos).
- Las señales también se analizan en el dominio de frecuencia a través de la *Transformada Rápida de Fourier* (FFT), que está optimizada para vectores con longitudes N de potencias de dos, por lo cual se usa un valor de 128 ($2.56 \quad segundos \times 50Hz = 128 \quad ciclos$).

Una representación reducida compuesta de características con información relevante de las actividades se obtiene a través de las ventanas de las actividades en el dominio del tiempo. Estas ventanas también se analizan en dominio de frecuencia usando la FFT [Duhamel and Vetterli, 1990, Ho, 2004].

En este trabajo, se incluyen medidas estándar que ya se han propuesto para el reconocimiento de actividades físicas en varios trabajos [Yang et al., 2008], como son la media, la correlación entre pares de señales, el Área de Magnitud de

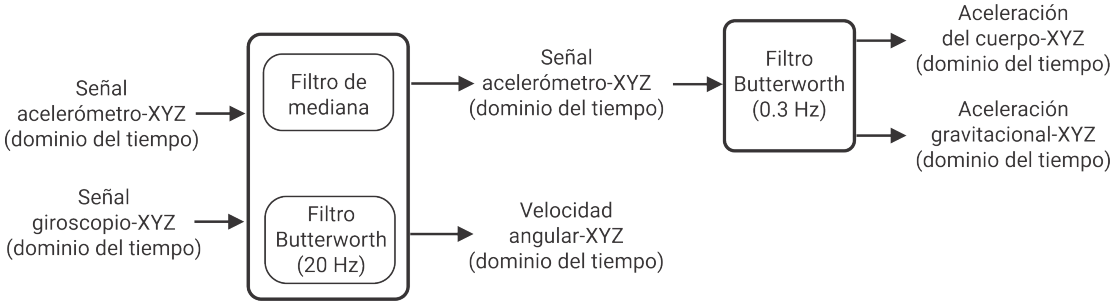


Figura 3.4: Diagrama del proceso para la eliminación del ruido y obtención de las señales base en el dominio del tiempo.

Señal (SMA), los coeficientes de autoregresión [Khan et al., 2010] y la energía de diferentes bandas de frecuencia [Samà, 2013]. También se incluyen medidas originales como la asimetría, la curtosis del espectro de frecuencia y los ángulos entre las señales triaxiales. Estas medidas se aplican tanto a las señales del acelerómetro procesadas como a las del giroscopio (Cuadro 2.3). Por este motivo y teniendo en cuenta la cantidad de señales involucradas, el número generado de características puede aumentar en gran medida. El Cuadro 3.4 muestra las señales seleccionadas e indica los dominios de los que se extrajeron las características.

Asimismo, en el Cuadro 3.5 se muestran las medidas aplicadas a las señales para generar los conjuntos de datos. Algunas de estas medidas son aplicadas a cada uno de los tres ejes posibles X, Y o Z, mientras que las señales tridimensionales son denotadas con el acrónimo "-XYZ". De esta manera, se extrae un total de 561 características para describir cada ventana de actividad.

El proceso implementado en el módulo generador de características se describe a continuación. Primero, se aplican una serie de filtros a las señales capturadas del acelerómetro y del giroscopio en los 3 ejes, ésto para reducir el ruido. Además, se separa la aceleración gravitacional de la aceleración del cuerpo, ésto es mostrado en la Figura 3.4. El resultado de esta fase son tres grupos de señales, éstas son la aceleración del cuerpo-XYZ, la aceleración gravitacional-XYZ y la velocidad angular-XYZ del cuerpo.

A continuación, la aceleración del cuerpo-XYZ y la velocidad angular-XYZ son derivadas en el tiempo para obtener las señales Jerk-XYZ, tal como se muestra en la Figura 3.5. De esta manera y agregando 2 grupos más de señales de información, se obtiene la magnitud de las señales tridimensionales mediante el uso de la norma euclidiana, lo cual es ejemplificado en la Figura 3.6. A continuación se aplica una FFT a las señales marcadas en la columna Frecuencia del Cuadro 3.4, lo que produce señales en el dominio de la frecuencia.

Finalmente, con cada una de estas treinta y tres señales se calculan las medidas

Cuadro 3.5: Lista de medidas utilizadas para generar el vector de características.

Función	Descripción
Mean (s)	Media aritmética
Std (s)	Desviación estándar
Mad (s)	Desviación media absoluta
Max (s)	El valor más grande en el arreglo
Min (s)	El valor más pequeño en el arreglo
Skewness (s)	Skewness de señal de frecuencia
Kurtosis (s)	Curtosis de la señal de frecuencia
MaxfreqInd (s)	Componente más grande de la frecuencia
Energy (s)	Promedio de la suma de los cuadrados
Sma (s_1, s_2, s_3)	Área de magnitud de la señal
Entropy (s)	Entropía de la señal
Iqr (s)	Rango intercuartil
Autoregression (s)	Coeficientes de autorregresión Burg de 4º orden
Correlation (s_1, s_2)	Coeficiente de correlación de Pearson
MeanFreq (s)	Promedio ponderado de una señal de frecuencia
EnergyBand (s, a, b)	Energía espectral de una banda de frecuencia [a, b]
Angle (s_1, s_2, s_3, v)	Angulo entre la media de la señal triaxial y el vector

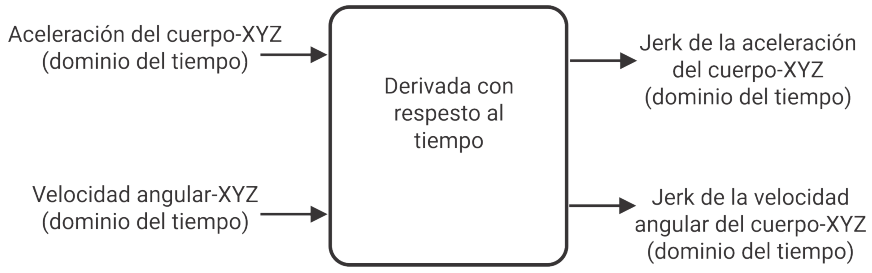


Figura 3.5: Diagrama del proceso para la obtención de las señales en el dominio del tiempo derivadas a partir de las básicas.

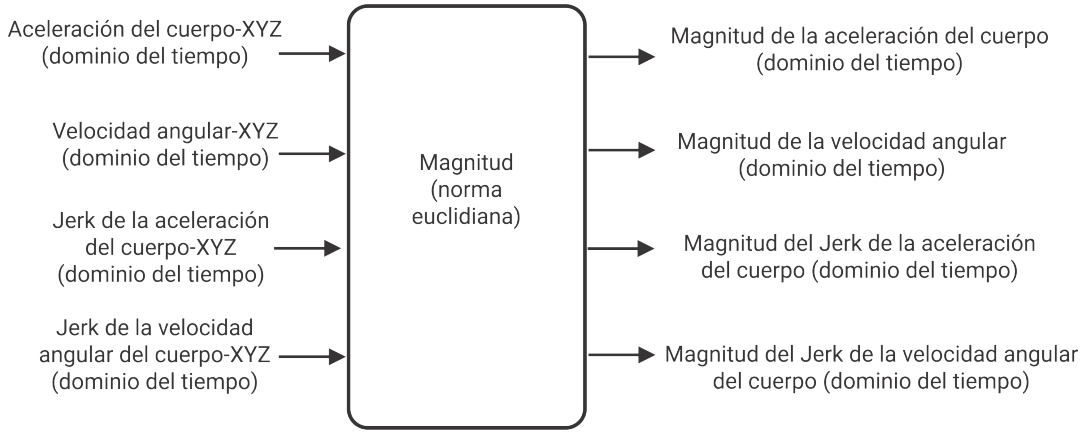


Figura 3.6: Diagrama del proceso para la obtención las medidas escalares básicas en el dominio del tiempo.

descritas en el Cuadro 3.5, con excepción de los ángulos. Ésto produce un subtotal de 554 características. Las 7 características restantes se obtienen del cálculo de los ángulos de la media gravitacional con los tres ejes, la media de la aceleración del cuerpo con la gravedad, la media de los Jerk de la aceleración del cuerpo con la gravedad, la media de la velocidad angular con la media gravitacional y la media de los Jerk de la velocidad angular con la media de la gravitacional, todas éstas en el dominio del tiempo. Como resultado, se obtienen 561 características por cada muestra caracterizada.

3.5. Aplicación de muestra

Para realizar la clasificación de actividades humanas cotidianas usando datos reales generados por usuarios, es necesario contar con una infraestrucura y una aplicación de muestra. Para esto se requiere de una aplicación instalable en el teléfono inteligente del usuario para la adquisición de datos, la cual debe comunicarse con la infraestructura de software-hardware que se describe a continuación.

El funcionamiento de la aplicación de muestra se divide en 3 partes. La primera corresponde al cliente, la segunda al servidor y la tercera al clúster computacional. En el lado del cliente se encuentran 2 aplicaciones: la aplicación móvil para la recolección de la información de los sensores del teléfono y la aplicación Web para el seguimiento de las actividades. En el servidor se ejecutan los módulos que se encargan de llevar el registro de las actividades realizadas por los usuarios. Mientras que, en el clúster computacional se lleva a cabo la clasificación de las

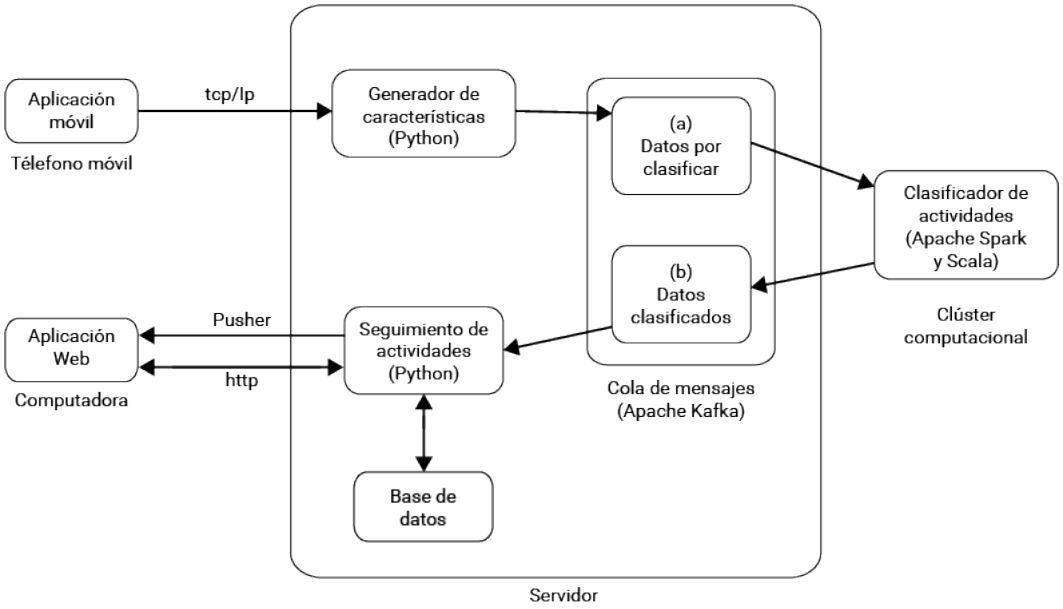


Figura 3.7: Arquitectura de la aplicación desarrollada.

actividades físicas realizadas por los usuarios del sistema. La arquitectura del sistema es mostrada en la Figura 3.7.

El flujo de la aplicación es el siguiente. Primero se obtiene información del teléfono móvil a través de la aplicación desarrollada y los datos obtenidos a través de ésta son enviados al módulo generador de características. Este módulo caracteriza los datos y los envía a la cola de mensajes de entrada, como puede observarse en la Figura 3.7(a), para su clasificación. A continuación, la aplicación montada en el clúster lee los datos de la cola de mensajes y los clasifica utilizando un modelo previamente entrenado; el resultado del clasificador es enviado a la cola de mensajes de salida (ver Figura 3.7(b)). Posteriormente, el módulo de seguimiento de actividades lee los resultados que se encuentran en dicha cola de mensajes y guarda los resultados en la base de datos. Además, analiza los resultados y en caso de ser pertinente envía una notificación a la aplicación web a través de un *pusher* (*Pusher* es un sistema que mantiene una conexión en tiempo real entre el servidor y sus clientes a través de WebSockets) para incentivar al usuario a realizar actividades no sedentarias. Asimismo, se cuenta con una aplicación Web, con la cual el usuario puede observar las estadísticas detalladas de sus actividades.

Aplicación móvil. La aplicación móvil se encarga de la recolección de información extraída del acelerómetro y del giroscopio del teléfono móvil de cada

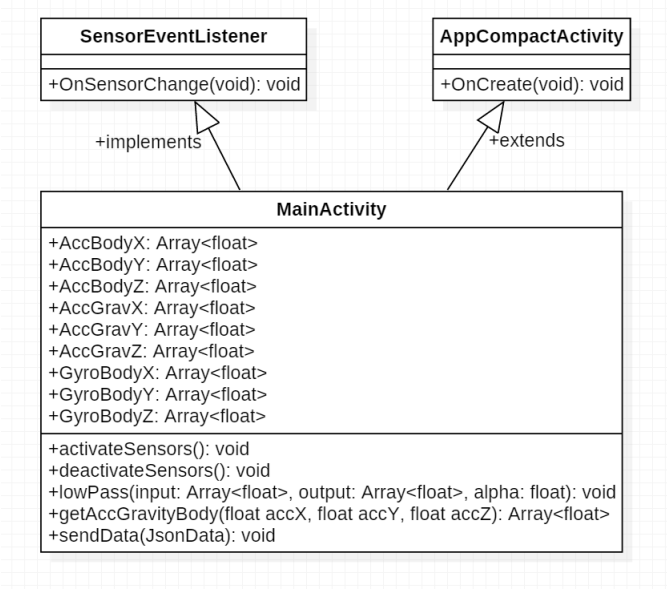


Figura 3.8: Diagrama de clase de la aplicación móvil.

usuario. Esta información es enviada al módulo generador de características para la caracterización de la muestra. Esta aplicación se desarrolla en el lenguaje de programación Java usando la herramienta de desarrollo Android Studio. La clase principal de esta aplicación se muestra en la Figura 3.8. En el Cuadro 3.6 se muestra una breve descripción de los atributos y métodos asociados a la clase mencionada. Observe que el método que se utiliza para extraer la información del giroscopio y del acelerómetro es *OnSensorChange()*, con el que es posible obtener el vector de 9 elementos de punto flotante (ver Atributos del Cuadro 3.6) con el que se constituye un registro del archivo CSV que se envía al servidor.

En cuanto al diseño de la interfaz de usuario, ésta se basa en la normativa de diseño *Material Design* [Google, 2014]. Basándose en este tipo de diseño se eligieron colores que favorecen el contraste y la visibilidad al usarlos a la luz del día. De esta forma, es posible mejorar la experiencia del usuario al hacer uso de la aplicación, ya que se busca que ésta sea intuitiva y fácil de usar. En el caso de la aplicación desarrollada, se eligió como color base el azul y como secundario el rojo.

Aplicación Web. Al mismo tiempo que la aplicación móvil se encuentra en operación en el teléfono celular, la aplicación Web ayuda al usuario a visualizar un análisis estadístico de las actividades que ha realizado a lo largo del día. Esta aplicación puede ser accedida desde cualquier dispositivo con conexión a Internet y un navegador Web instalado. El acceso a la aplicación es realizado a través de

Cuadro 3.6: Clase MainActivity con sus atributos y métodos asociados

Clase MainActivity	
Atributos	AccBodyX: Señal del acelerómetro con respecto al cuerpo en el eje X. AccBodyY: Señal del acelerómetro con respecto al cuerpo en el eje Y. AccBodyZ: Señal del acelerómetro con respecto al cuerpo en el eje Z. AccGravX: Señal del acelerómetro con respecto a la gravedad de la Tierra en el eje X. AccGravY: Señal del acelerómetro con respecto a la gravedad de la Tierra en el eje Y. AccGravZ: Señal del acelerómetro con respecto a la gravedad de la Tierra en el eje Z. GyroBodyX: Señal del giroscopio en el eje X. GyroBodyY: Señal del giroscopio en el eje Y. GyroBodyZ: Señal del giroscopio en el eje Z.
Métodos	OnCreate(): Constructor. OnSensorChange(): Lectura de los sensores. activateSensors(): Activar los sensores (acelerómetro giroscopio). lowPass(...): Aplicar un filtro de pasa baja. getAccGravityBody(...): Obtener la aceleración con respecto al cuerpo y a la tierra. sendData(...): Enviar los datos capturados al servidor.

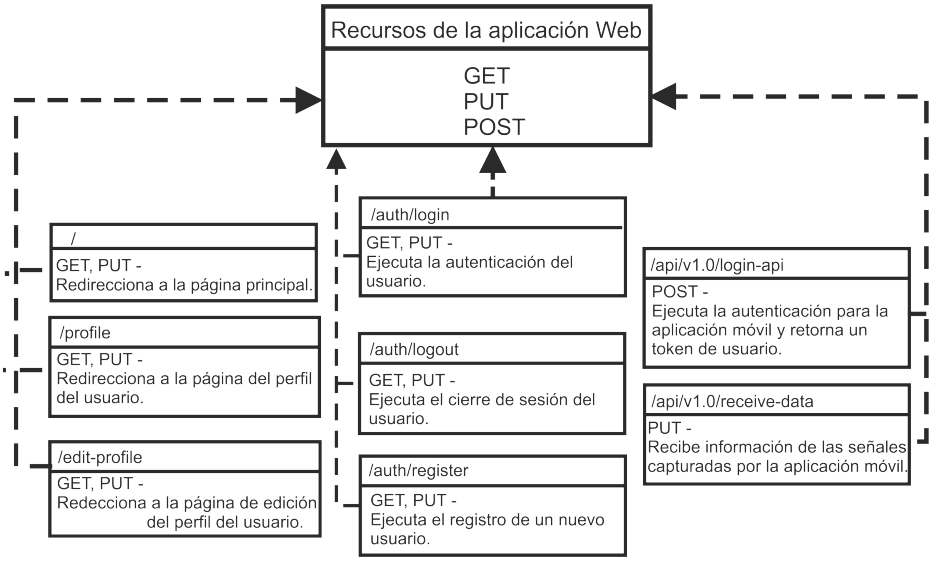


Figura 3.9: Diagrama de end-points de la aplicación Web.

un nombre de usuario y una contraseña. En la aplicación es posible observar un resumen de las actividades realizadas a lo largo del último día, semana o mes. Además, se muestran notificaciones para estimular e incentivar a los usuarios a realizar distintas actividades físicas en caso de pasar mucho tiempo inactivo. El comportamiento de esta aplicación se desarrolla en el lenguaje de programación Python con ayuda del framework *Flask*. En la Figura 3.9 se pueden observar los principales end-points (rutas de conexión que responden a una petición), mientras que en el Cuadro 3.7 se describe brevemente su comportamiento.

Generador de características. Este módulo del servidor se encarga de recibir la información generada en el dispositivo móvil y caracterizar dicha información siguiendo el procedimiento descrito en la Sección 3.4. Una vez generadas las 561 características, éstas son enviadas a la cola de mensajes (véase Figura 3.7a) para su futuro procesamiento. Para esto, se utiliza el módulo desarrollado en el lenguaje de programación Python, mismo que es descrito en el Anexo A. En la Figura 3.10 se muestra la función implementada para la generación de características.

Seguimiento de actividades. Con este módulo se analizan los datos clasificados por el clúster computacional, los cuales se encuentran alojados en una cola de mensajes (véase Figura 3.7b). Además, este módulo se encarga de escribir los resultados en la base de datos y realizar recomendaciones en caso de ser necesarias. El análisis de los datos está basado en [WHO, 2010, Strath et al., 2013]. Las notificaciones se envían a la aplicación Web mediante el uso de un *pusher* [Pusher,

Cuadro 3.7: End-points más importantes de la aplicación Web.

End-points de la aplicación web	
/	Muestra la página principal de la aplicación.
/profile	Muestra el perfil con la información del usuario.
/edit-profile	Muestra una interfaz que permite modificar la información del usuario.
/auth/login	Muestra la pantalla para iniciar sesión.
/auth/logout	Cierra la sesión del usuario y redirige hacia la pantalla para iniciar sesión.
/auth/register	Muestra una interfaz que permite crear una nueva cuenta en el sistema.
/api/v1.0/login-api	Autentifica un usuario en el sistema y en caso de éxito devuelve una código único con el cual se interactúa con el sistema.
/api/v1.0/receive-data	Recibe datos en formato JSON, los valida y en caso de provenir de usuarios autorizados los coloca en una cola de mensajes para su procesamiento.

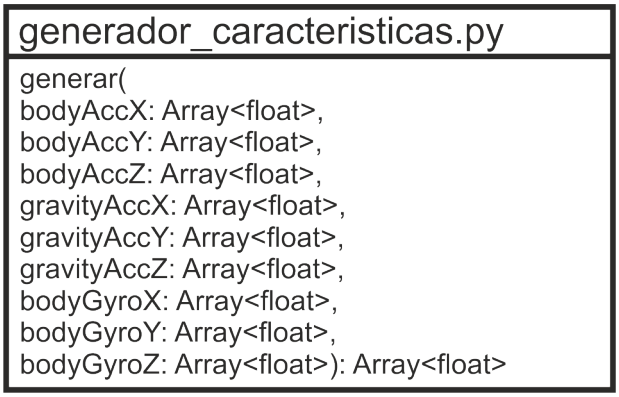


Figura 3.10: Diagrama de la función del generador de características.

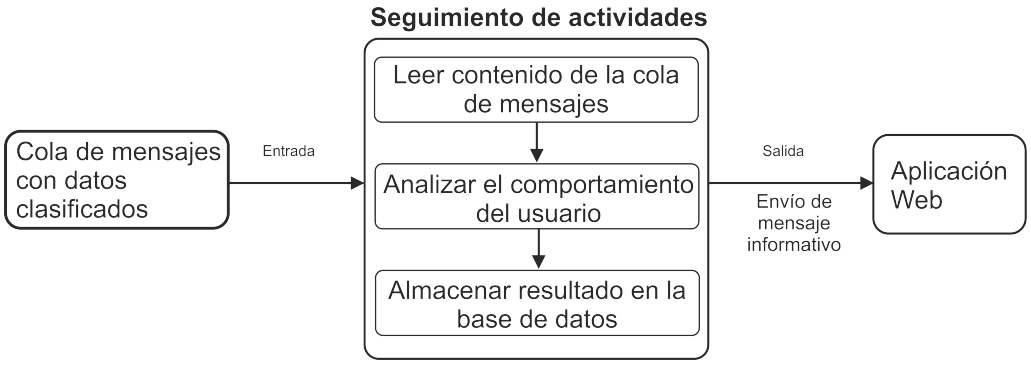


Figura 3.11: Diagrama de flujo del módulo para dar seguimiento a las actividades.

Datos a clasificar	
user_id	number
created_at	string
data	Array[number]

Datos clasificados	
user_id	number
created_at	string
prediction	number

Figura 3.12: Esquema de los datos en formato JSON almacenados en las colas de mensajes.

2010]. Este módulo de seguimiento de actividades se desarrolla en el lenguaje de programación Python. El procedimiento interno es el mostrado en la Figura 3.11.

Colas de mensajes. Las colas de mensajes permiten a diferentes partes de un sistema comunicarse y procesar las operaciones de forma asíncrona. Una cola de mensajes ofrece un búfer ligero que almacena temporalmente los mensajes, y puntos de enlace que permiten a los componentes de software conectarse a la cola para enviar y recibir mensajes. En nuestro caso, los componentes que interactúan son el generador de características (productor) y el clasificador de actividades (consumidor). De manera similar, el clasificador de actividades (productor) interactúa con el sistema de seguimiento de actividades (consumidor). Ésto puede observarse con mayor detalle en la Figura 3.7a y 3.7b.

Por esta razón, la aplicación de muestra hace uso de 2 colas de mensajes. En la Figura 3.12 se presenta el esquema que mantiene los datos enviados a través de la cola de mensajes. Cabe mencionar que dichos esquemas tienen formato JSON. En el caso de la aplicación, se utiliza la cola mensajes Apache Kafka [Apache, 2011b] versión 2.11.1.

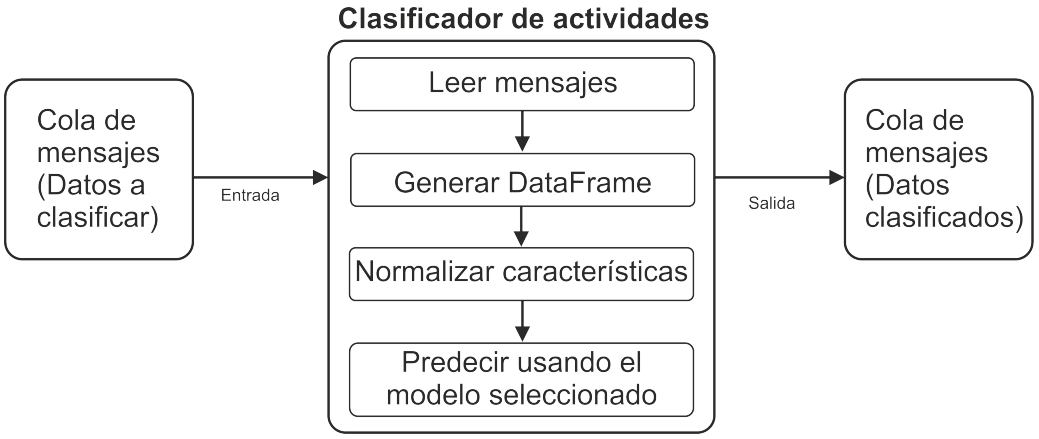
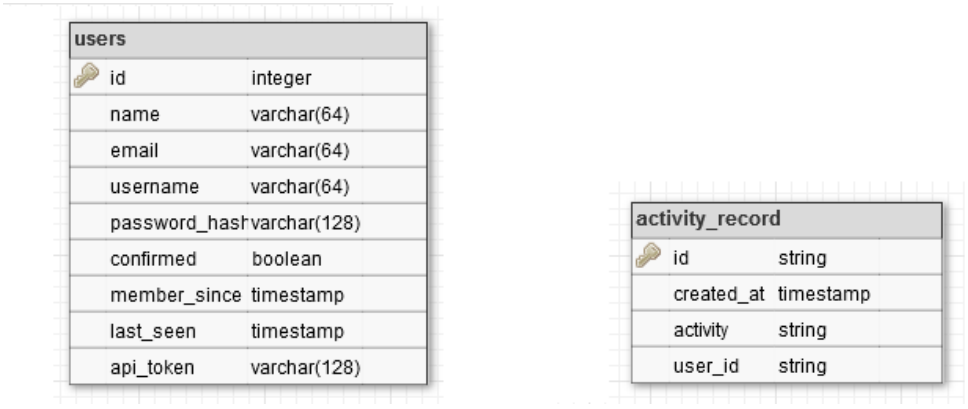


Figura 3.13: Diagrama de flujo del módulo para clasificar actividades.

Clasificador de actividades. El clasificador de actividades se encuentra instalado en un clúster computacional administrado por Apache Spark [Apache, 2014]. Su función es consumir los mensajes alojados en la cola de mensajes de datos para clasificarlos mediante un modelo creado con los datos obtenidos en la Sección 4.1 y colocar el resultado en una nueva cola de mensajes de datos clasificados. En la Figura 3.13 se observa la secuencia que sigue el clasificador cuando se encuentra en ejecución. El lenguaje de programación elegido para este módulo es Scala, debido a que éste es el lenguaje nativo de la herramienta Spark.

Base de datos. Para el almacenamiento de los datos se utiliza una base de datos SQL, en este caso PostgreSQL versión 9.6.8, la cual mantiene el registro de los usuarios que hacen uso de la aplicación. Por otro lado, los resultados de la clasificación de las actividades son almacenados en una base de datos NoSQL, Apache Cassandra versión 5.0.1 [Apache, 2010]. Esta última, es una base de datos masiva, escalable, no-relacional y distribuida diseñada para almacenar grandes cantidades de información. En la Figura 3.14a se observa la tabla utilizada en PostreSQL para llevar la gestión de los usuarios registrados en la aplicación. Mientras que, en la Figura 3.14b se muestra el esquema utilizado en Cassandra para llevar el registro de las actividades realizadas por los usuarios a lo largo del día.

Para más detalles de la aplicación de muestra consulte el Anexo C.



(a) **(b)**

Figura 3.14: Esquemas base de las bases de datos utilizados para almacenar información. (a) Tabla utilizada para la gestión de usuarios; b) KeySpace de Cassandra utilizado para almacenar los resultados de la clasificación.

Capítulo 4

Resultados

En este capítulo se presentan los resultados obtenidos con los que es posible cumplir con los objetivos planteados. Como se menciona en el Capítulo 3, los experimentos se enfocan en la obtención de un algoritmo de aprendizaje computacional adecuado para el problema de reconocimiento de actividades físicas humanas haciendo uso de la información extraída de los sensores integrados (giroscopio y acelerómetro) en los teléfonos celulares inteligentes. De esta manera, el algoritmo seleccionado forma parte de un sistema de seguimiento de actividades, que también ha sido propuesto en este trabajo de tesis, de manera que una gran cantidad de usuarios puedan beneficiarse del uso del cómputo distribuido que ofrece Apache Spark para el seguimiento de sus propias actividades. Las siguientes secciones de este capítulo, presentan detalladamente la manera de cómo se realizaron los experimentos y los resultados correspondientes.

4.1. Generación de datos

Para la obtención de datos se realizan distintos experimentos. Para esto, distintas personas realizan las actividades: caminar, estar de pie, estar acostado, estar sentado, subir y bajar escaleras. Dichas actividades son realizadas por el usuario mientras usa el teléfono móvil montado en un cinturón (Véase Figura 4.1). Cada persona realiza seis experimentos, uno por cada actividad. Así pues, un solo experimento dura aproximadamente cinco minutos, por lo que la duración de todos los experimentos por persona es de aproximadamente 30 minutos. Como resultado de cada experimento se obtiene la información de las señales producidas por el acelerómetro y el giroscopio.

Durante estos experimentos se utiliza la aplicación desarrollada específicamente para este propósito.

Cuadro 4.1: Descripción del conjunto de datos

Actividad	Número de muestras
Bajar escaleras	3,800
Estar acostado	3,800
Estar sentado	3,800
Estar de pie	3,800
Subir escaleras	3,800
Caminar	3,315
Total	**22,315**

Figura 4.1: Cinturón utilizado para la captura de datos.

Posteriormente, se procesa la información adquirida de los sensores y se generan las características siguiendo los pasos descritos en el trabajo de [Ortiz, 2014].

En los experimentos descritos participaron un total de 20 voluntarios en un intervalo de edad que va desde los 16 años hasta los 30. Como se mencionó, todos los voluntarios realizaron cada una de las actividades por un periodo aproximado de cinco minutos. Es así como se obtuvo un conjunto de datos de 22,315 registros, cada uno conformado por 561 características. La distribución de las clases a partir de las muestras obtenidas está descrita en el Cuadro 4.1. Para más detalles del módulo desarrollado, véase el Anexo A.

En la Figura 4.2 se pueden observar imágenes tomadas de los voluntarios realizando algunas actividades, en las que se obtuvieron las señales para la generación de los datos. En estas imágenes también se puede observar el uso del teléfono inteligente y el cinturón descritos con anterioridad (ver Figura 4.1).

(a) Bajando escaleras. **(b)** Subiendo escaleras. **(c)** Estar sentado.

Figura 4.2: Usuarios realizando distintas actividades durante la obtención de las señales para la creación del conjunto de datos.

4.2. Experimentación y análisis de resultados

En esta sección se describen las pruebas realizadas y sus resultados. Dado que el problema de reconocimiento de actividades físicas humanas ha sido tratado desde el punto de vista del aprendizaje supervisado, se someten a prueba seis modelos de aprendizaje supervisado integrados en Spark. Dichas pruebas realizadas con estos algoritmos se dividen en dos etapas.

La primera etapa tiene como objetivo seleccionar el mejor modelo de aprendizaje supervisado para este problema (Véase la Sección 4.2.1). La segunda etapa tiene como objetivo comparar el rendimiento en tiempo de ejecución entre la versión secuencial y la versión distribuida del clasificador seleccionado en la primera etapa (Véase la Sección 4.2.2).

El clasificador seleccionado al final de estos experimentos, forma parte de la aplicación de prueba planteada en el Capítulo 3.

En las pruebas se usan las muestras obtenidas en la etapa de recolección y generación de datos (Véase la Sección 4.1). Estos datos son normalizados utilizando un escalador *MinMax*. Para la validación de los resultados de los clasificadores se utiliza la validación cruzada con 10 iteraciones. Ambos procesos están integrados en Spark [Apache, 2017].

4.2.1. Etapa 1: Selección del mejor algoritmo

La primera etapa tiene como objetivo seleccionar el clasificador que obtenga la mejor exactitud posible en la clasificación, dado que este modelo es el que clasifica las actividades que realizan todos los usuarios de la aplicación final. Esta etapa debe realizarse en dos fases como se explica a continuación. La primera fase tiene como objetivo reducir el número de clasificadores y obtener solamente tres modelos con los mejores rendimientos utilizando una muestra aleatoria estratificada. La segunda fase se enfoca en encontrar el clasificador (de entre los tres seleccionados en la fase anterior) con el mejor rendimiento en exactitud usando una mayor cantidad de datos para el entrenamiento.

Fase 1: Reducción de los modelos de clasificación

La primera fase se enfoca en obtener los tres clasificadores con los mejores rendimientos de clasificación, seleccionando mediante una búsqueda los parámetros óptimos. Para esto, se realizaron pruebas con seis modelos de aprendizaje supervisado incluidos en la biblioteca MLlib de Apache Spark [Apache, 2017]. El conjunto de datos utilizado es extraido del conjunto completo con 22,315 muestras y se conforma de 1,500 muestras de cada una de las seis actividades a clasificar (9,000 muestras en total). De este conjunto se utiliza el 70 % para el entrenamiento, mientras que para la prueba se utiliza el 30 % restante. Este particionamiento es para obtener una mejor generalización y medidas de validación más precisas.

Durante la experimentación, se utilizaron los siguientes clasificadores incluidos en Apache Spark [Apache, 2014]:

- Árboles de decisión (*Decision tree*)
- Bosques aleatorios (*Random forest*)
- Árbol de potenciación del gradiente (*Gradient-boosted tree*)
- Perceptrón multicapa (*Multilayer perceptron*)
- Máquinas de soporte vectorial lineal (*Linear Support Vector Machine*)
- Bayes ingenuo (*Naïve Bayes*)

El procedimiento planteado para las pruebas es el siguiente. Primero, se obtienen los parámetros que maximizan la exactitud de los clasificadores; para esto, se usan las herramientas integradas de Spark que permiten optimizar los parámetros de los algoritmos [Apache, 2017]. Para cada uno de los clasificadores utilizados se propone un conjunto de parámetros, los cuales son mezclados para

construir una tabla de combinaciones sobre los cuales se busca la permutación que genera el mejor clasificador. Para validar cuál es la mejor combinación de parámetros, se hace uso de la validación cruzada con 10 iteraciones.

La búsqueda de parámetros se realiza usando dos refinamientos: uno grueso y uno fino. El refinamiento grueso es una búsqueda en el espacio de combinaciones de parámetros, donde se selecciona la combinación que produce el mejor resultado con el clasificador. En este refinamiento, el incremento de los valores propuestos para cada parámetro suele tener gran tamaño. El refinamiento fino es una búsqueda alrededor de los parámetros obtenidos en el refinamiento grueso, con la diferencia de que el incremento entre los valores propuestos para cada parámetro es de menor tamaño. Este último refinamiento permite obtener un incremento adicional en el rendimiento conseguido con el refinamiento grueso. En resumen, la búsqueda de parámetros con los refinamientos respectivos depende de cada clasificador y es descrita a continuación.

Árboles de decisión. La búsqueda se realiza sobre el parámetro de la máxima profundidad de los árboles. Ya que la implementación de Spark soporta profundidades máximas menores o iguales a 30, se tomó este valor como la cota superior. Por lo cual, en el refinamiento grueso se usaron valores en incrementos de diez ($10, 20$ y 30). Este refinamiento mostró que la mejor profundidad es de 20; a continuación, se realiza una segunda búsqueda (refinamiento fino) alrededor de este valor con un incremento de ± 3. El refinamiento fino fue sobre los valores $17, 18, 19, 20, 21, 22$ y 23. Esta búsqueda arrojó que la profundidad que maximiza la clasificación es de **17**.

Bosques aleatorios. En los bosques aleatorios la búsqueda se realiza sobre dos parámetros, la profundidad máxima de los árboles y el número de árboles a entrenar. En el refinamiento grueso se usan los siguientes valores para el número de árboles: $50, 100, 150$ y 200; mientras que, con la máxima profundidad se utilizaron: $15, 16, 17, 18$ y 19. Los resultados muestran que los mejores parámetros fueron **200** para el número de árboles, y **17** para la profundidad máxima. Por lo que, en el refinamiento fino se busca alrededor del mejor valor del número de árboles con un incremento de ± 20, con valores espaciados en incrementos de diez ($180, 190, 200, 210$ y 220) y se utilizan los mismos valores de la máxima profundidad que en el refinamiento grueso. El mejor modelo a partir de estos valores es obtenido utilizando nuevamente los parámetros **200** para el número de árboles y **17** para la profundidad máxima.

Bayes ingenuo (Naïve Bayes). En caso del Bayes ingenuo se utilizó un suavizado Laplaciano para evitar probabilidades de cero y evitar problemas en

el procesamiento. El valor del Laplaciano utilizado es 1.0. Dado que el modelo Bayes ingenuo no tiene parámetros de ajuste, la búsqueda no fue necesaria.

Perceptrón multicapa. La implementación del perceptrón multicapa nos permite variar el máximo número de iteraciones y la configuración de las capas de neuronas. La elección del número de capas a utilizar fue basado en el trabajo [Ronao and Cho, 2016]. Por esta razón se utilizan 3 capas, 1 capa de entrada, 1 capa oculta y 1 capa de salida. La capa de entrada y la de salida usaron un número fijo de nodos: 561 nodos de entrada y 7 de salida, los cuales corresponden al número de características de las muestras y al número de clases más uno (ésto debido a que el algoritmo reconoce la clase cero, es decir una clase que no corresponde a ninguna de las otras), respectivamente. Adicionalmente, se utilizó una tolerancia de 10^{-3}. En la etapa del refinamiento grueso se usan los valores: $100, 200, 300, 400$ y 500 para el máximo número de iteraciones y para el número de nodos en la capa oculta se usaron los valores $100, 200, 300, 400$ y 500. Esta prueba mostró que los parámetros que maximizan la exactitud del clasificador es de **300** para el número de iteraciones y de **500** para el número de nodos en la capa oculta. En el refinamiento fino la búsqueda se realiza alrededor del valor del máximo número de iteraciones con un incremento de ± 10, con valores espaciados en incrementos de cinco ($290, 295, 300$ y 305); mientras que para el número de nodos en la capa oculta se usa un incremento de ± 15 con valores espaciados en incrementos de cinco ($485, 490, 495, 500, 505, 510$ y 515). Esta última prueba exhibió que los mejores parámetros obtenidos fueron nuevamente **300** para el número de iteraciones y **500** para el número de nodos en la capa oculta.

Máquinas de soporte vectorial. La versión de Spark utilizada en este proyecto de tesis sólo incluye las máquinas de soporte vectorial con kernel lineal. Este clasificador cuenta con un solo parámetro de pérdida C que puede modificarse, con el que se controla la flexibilidad del margen de separación de la máquina. Este parámetro permite que existan pérdidas de clasificación intencionalmente para mejorar la generalización de la misma. Además, esta implementación sólo soporta la clasificación binaria, por lo que para extenderlo a la clasificación multiclase se utiliza la estrategia *One-vs-Rest* [Bishop, 2006].

Tomando como base el trabajo [Hsu et al., 2003], se utilizaron los valores $C = 2^{-5}, 2^{-3}, \ldots, 2^{15}$ para el refinamiento grueso. Este refinamiento mostró que el valor de $C = 2^{-5}$ maximizaba la exactitud de clasificación. Por lo cual, en el refinamiento fino se buscó con los valores para C de $2^{-5}, 2^{-4}, 2^{-3}$ y 2^{-2}. Al igual que en el refinamiento grueso, el valor de C que maximizó la exactitud del clasificador fue de 2^{-5}.

Cuadro 4.2: Resultados del promedio de distintas medidas de rendimiento.

Clasificador	Refinamiento 1				Refinamiento 2			
	Exactitud	Precisión	Recall	F1	Exactitud	Precisión	Recall	F1
Árboles de decisión	0.77374	0.94113	0.94113	0.94113	0.77262	0.94075	0.94075	0.94075
Bosques aleatorios	0.88215	0.97303	0.97303	0.97303	0.88351	0.97341	0.97341	0.97341
Árbol de potenciación del gradiente	0.87320	0.97075	0.97075	0.97075	0.87747	0.97189	0.97189	0.97189
Perceptrón multicapa	0.87155	0.97037	0.97037	0.97037	0.87580	0.97151	0.97151	0.97151
Máquinas de soporte vectorial	0.80655	0.95176	0.95176	0.95176	0.80780	0.95214	0.95214	0.95214
Bayes ingenuo	0.56980	0.84656	0.84656	0.84656	-	-	-	-

Árbol de potenciación del gradiente Para este modelo, se trabajó con el máximo número de iteraciones. De esta manera, para el refinamiento grueso se utilizaron los valores $100, 200, 300, 400$ y 500. Este refinamiento mostró que el clasificador maximizaba su exactitud con un máximo número de iteraciones de **200**; por otro lado, para el refinamiento fino se realizó una búsqueda alrededor este valor con un incremento de ± 10 con valores espaciados en incrementos de cinco. Los valores utilizados fueron de $190, 195, 200, 205$ y 210. La exactitud del clasificador se maximizó cuando el número de iteraciones fue **205**.

El Cuadro 4.2 muestra los clasificadores junto con las medidas obtenidas al utilizar los parámetros que maximizan la exactitud y otras medidas de clasificación adicionales.

De acuerdo con estos resultados, los tres clasificadores que muestran el mejor desempeño en exactitud de clasificación son: los árboles aleatorios, los arboles de potenciación del gradiente y el perceptrón multicapa.

Fase 2: Selección de modelo de clasificación

En esta fase, se hacen pruebas con los tres mejores clasificadores obtenidos de la fase anterior de esta sección. En esta etapa se utiliza el conjunto con todos los datos obtenidos, el cual consta de 22,315 muestras, con la distribución descrita en el Cuadro 4.1. Para el entrenamiento de los modelos de clasificación también se utiliza el 70 % de las muestras totales y el 30 % restante forman el conjunto de los datos de prueba.

Debido al incremento de muestras, se realiza una nueva búsqueda de parámetros. Esta nueva búsqueda se realiza alrededor de los parámetros obtenidos en la Fase

Cuadro 4.3: Resultados promedio de distintas medidas de rendimiento utilizando el conjunto completo de datos.

Clasificador	Rendimiento			
	Exactitud	Precisión	Recall	F1
Bosques aleatorios	0.90437	0.97877	0.97877	0.97877
Árbol de potenciación del gradiente	0.90369	0.97862	0.97862	0.97862
Perceptrón multicapa	0.90304	0.97846	0.97846	0.97846

1. De esta forma, se obtiene el mejor modelo de cada clasificador con el cual se maximiza la exactitud. Los resultados obtenidos a partir de estos tres clasificadores son comparados entre sí, con lo cual es posible seleccionar el clasificador con el mejor rendimiento. Los parámetros encontrados con el refinamiento fino para cada clasificador son presentados a continuación.

Bosques aleatorios. Para este clasificador se buscó alrededor del mejor valor del número de árboles 200 ± 10 en incrementos de cinco. Los valores sobre los que se buscó el número de árboles fueron: $190, 195, 200, 205$ y 210. Mientras que, para la profundidad máxima se utilizaron los valores $15, 16, 17, 18, 19$ y 20. El mejor modelo obtenido fue utilizando los parámetros **195** para el número de árboles y **19** para la profundidad máxima.

Perceptrón multicapa. Para el caso del perceptrón multicapa, se utilizó la misma topología que en la fase anterior.

Dado que el mejor valor para el número de iteraciones fue **300** y para el número de nodos en la capa oculta fue **500**, se buscó alrededor de estos valores. Para el máximo número de iteraciones se usaron los valores: $290, 295, 300, 305$ y 310; mientras que para el número de nodos en la capa oculta se usaron los valores: $490, 495, 500, 505$ y 510. El mejor modelo obtenido hace uso nuevamente de los parámetros **300** para el número de iteraciones y **500** para el número de nodos en la capa oculta.

Árbol de potenciación del gradiente Para este modelo, la búsqueda se realizó alrededor del máximo número de iteraciones de 200, por lo que los valores utilizados fueron de $190, 195, 200, 205$ y 210. La exactitud del clasificador se maximizó cuando el número de iteraciones fue de **205**.

En el Cuadro 4.3 se muestran los resultados promedio de distintas medidas para cada uno de los clasificadores utilizando los parámetros que maximizaron

la exactitud. Para más detalles de los algoritmos utilizados en esta fase véase el Anexo B.

Con el fin de analizar la contribución de cada clase en la clasificación general (Cuadro 4.3), se muestran las matrices de confusión de cada clasificador en los Cuadros 4.4, 4.5 y 4.6. En éstos se observa de manera general que los 3 clasificadores utilizados en esta etapa muestran un buen rendimiento al diferenciar las actividades sedentarias (estar sentado, estar de pie y estar acostado), esto debido a que las posturas en cada actividad sedentaria son distintas, particularmente en la actividad "estar acostado''. Sin embargo, los algoritmos de clasificación muestran un problema de rendimiento al tratar con las actividades no sedentarias (caminar, bajar y subir escaleras).

Después de analizar el Cuadro 4.4, aunque los bosques aleatorios hacen una buena discriminación entre las actividades sedentarias y las no sedentarias, la principal confusión de este clasificador se presenta entre las clases subir y bajar escaleras. En contraste, las clases que mejor se pudieron discriminar son las de estar acostado y estar sentado. El perceptrón multicapa obtiene resultados similares a los bosques aleatorios, como se muestra en el Cuadro 4.5; la principal desventaja se observa en la clase "bajar escaleras", ya que aumenta la confusión con las otras clases no sedentarias, específicamente con la clase "caminar". Además, la clase estacionaria "estar de pie" es confundida aún más con la clase no sedentaria "bajar escaleras". Asimismo, el modelo del árbol de potenciación del gradiente mostró resultados similares a los bosque aleatorios, teniendo la desventaja de aumentar la confusión de la clase "subir escaleras" con otras clases no sedentarias (ver Cuadro 4.6).

Todas estas confusiones que presentan estos clasificadores tienen diversas causas. Una razón es que ninguno de los sensores empleados mide la altitud, lo cual permitiría discriminar de mejor manera estas clases. Asimismo, la captura de las muestras de las actividades caminar, bajar y subir escaleras, fueron realizadas por los voluntarios intentando mantener una cadencia muy similar, disminuyendo las vibraciones propias de este tipo de actividades. Por consecuencia, las señales generadas por los sensores suelen ser muy parecidas entre estas tres últimas actividades, por lo que es difícil diferenciarlas con los clasificadores empleados.

Cabe destacar que la diferencia de rendimiento de clasificación es mínima entre los tres clasificadores en cuestión. Sin embargo, el modelo de bosques aleatorios presenta mejor exactitud de clasificación y la complejidad del mismo suele ser menor en comparación del modelo del *árbol de potenciación del gradiente*. Por lo tanto, el modelo seleccionado que formará parte del sistema de clasificación de patrones de actividades cotidianas en la aplicación de prueba es el de **bosques aleatorios**.

En la práctica, utilizando la aplicación desarrollada y el algoritmo de bosques aleatorios, se constató que el error al clasificar actividades sedentarias es mínimo.

Cuadro 4.4: Matriz de confusión de los Bosques aleatorios en la Fase 2.

		Clase predicha					
		Bajar escaleras	Estar acostado	Estar sentado	Estar de pie	Subir escaleras	Caminar
Clase real	Bajar escaleras	938	0	0	8	37	11
	Estar acostado	0	1140	0	0	0	0
	Estar sentado	0	1	1138	0	1	0
	Estar de pie	11	0	3	1121	1	4
	Subir escaleras	27	0	0	10	1093	10
	Caminar	5	0	0	1	9	980

Mientras que, al clasificar actividades no sedentarias, la clase caminar mantuvo un buen rendimiento; no obstante, las clases bajar y subir escaleras mostraron una confusión mayor. Estas observaciones, son congruentes con los resultados mostrados en las matrices de confusión de los tres algoritmos utilizados en esta fase (Véase Cuadro 4.4, Cuadro 4.5 y Cuadro 4.6).

4.2.2. Etapa 2: Comparación de rendimiento de ejecución secuencial y distribuido

La etapa 2 permite realizar una comparación entre la versión secuencial y la versión distribuida del clasificador elegido en la etapa anterior. Los experimentos hacen hincapié en los tiempos obtenidos en la ejecución del entrenamiento y de la predicción.

Para poder conocer las ventajas del uso de Spark en el problema que se plantea en este trabajo de tesis y comprobar la veracidad de la hipótesis planteada, se pusieron en marcha diversos experimentos de ejecución, en las cuales se realizó una variación del número de núcleos disponibles en el clúster computacional. Para la versión secuencial se usa un solo núcleo del *nodo* 0, mientras que para la versión distribuida se usan los tres *nodos*, variando el número de núcleos a utilizar. La

Cuadro 4.5: Matriz de confusión del Perceptrón multicapa en la Fase 2.

		Clase predicha					
		Bajar escaleras	Estar acostado	Estar sentado	Estar de pie	Subir escaleras	Caminar
Clase real	Bajar escaleras	938	0	1	10	34	11
	Estar acostado	0	1140	0	0	0	0
	Estar sentado	0	0	1138	1	1	0
	Estar de pie	17	0	1	1116	3	3
	Subir escaleras	32	0	2	8	1088	10
	Caminar	2	0	1	0	4	988

Cuadro 4.6: Matriz de confusión de los *Gradient-boosted-tree* en la Fase 2.

		Clase predicha					
		Bajar escaleras	Estar acostado	Estar sentado	Estar de pie	Subir escaleras	Caminar
Clase real	Bajar escaleras	942	0	0	9	33	10
	Estar acostado	0	1139	0	0	0	1
	Estar sentado	0	0	1139	1	0	0
	Estar de pie	8	0	3	1123	1	5
	Subir escaleras	29	0	1	11	1086	13
	Caminar	6	0	0	0	9	980

Cuadro 4.7: Núcleos del clúster utilizados en las pruebas de la etapa 2.

Prueba	Nodo 0	Nodo 1	Nodo 2	Número total de núcleos
1	1	0	0	1
2	1	1	1	3
3	2	2	2	6
4	4	4	4	12
5	8	8	8	24
6	16	16	16	48
7	32	32	32	96

variación de los núcleos en los *nodos* se realiza como se muestra en el Cuadro 4.7.

En esta etapa, se usa el conjunto completo de datos obtenidos, el cual consta de 22,315 muestras. El conjunto de entrenamiento está formado por el 70 % de las muestras y el conjunto de prueba está conformado por el 30 % restante. Dicho conjunto de pruebas es replicado 115 veces para formar un conjunto nuevo de pruebas que contenga aproximadamente 1 millón de registros. El propósito de estos experimentos es simular la captura y procesamiento masivo de 1 millón de datos de distintos usuarios potenciales de la aplicación de muestra. Ésto con la finalidad de observar una mejora sustancial en el tiempo de ejecución con el uso del cómputo distribuido que ofrece Spark, durante el entrenamiento y la predicción. El clasificador utilizado es el de *bosques aleatorios*, el cual fue seleccionado en la etapa anterior (Sección 4.2.1).

En total se realizan siete experimentos, para el entrenamiento y la prueba del clasificador mencionado. Para este fin, se usa el conjunto de datos replicados y la configuración de uso de núcleos del clúster, tal como se muestra en el Cuadro 4.7. La idea principal de estos experimentos es observar el comportamiento del sistema cuando se maximiza el uso de los núcleos de cómputo disponibles en el clúster computacional.

Los resultados de los experimentos al variar los núcleos son mostrados en el Cuadro 4.8. El primer renglón corresponde a la versión secuencial del algoritmo. El incremento de los núcleos destinados al procesamiento se va incrementado en potencias de 2 hasta alcanzar los 32 en cada *nodo*, que es el total de núcleos disponibles. Se observa que los tiempos, tanto de entrenamiento como de predicción, mejoran a medida que se incrementa el número de núcleos utilizados para el procesamiento.

Esta información permite realizar una comparación entre la ejecución de

Cuadro 4.8: Resumen general de los resultados de los experimentos al realizar variaciones en el número de núcleos, usando el clasificador de bosques aleatorios.

Configuración de núcleos			Número total de núcleos	Rendimiento						
Nodo 1	Nodo 2	Nodo 3		Tiempo de entrenamiento	Aceleración en el entrenamiento	Tiempo de predicción	Aceleración en la predicción	Exactitud	Precisión	Recall	F1
1	0	0	1	70.86 minutos	1.00x	115.28 minutos	1.00x	0.89941	0.97755	0.97755	0.97755
1	1	1	3	27.74 minutos	2.55x	36.50 minutos	3.15x	0.90380	0.97862	0.97862	0.97862
2	2	2	6	15.05 minutos	4.70x	19.33 minutos	5.96x	0.90130	0.97801	0.97801	0.97801
4	4	4	12	8.54 minutos	8.29x	12.57 minutos	9.17x	0.90189	0.97816	0.97816	0.97816
8	8	8	24	6.03 minutos	11.75x	8.87 minutos	12.99x	0.90379	0.97862	0.97862	0.97862
16	16	16	48	8.32 minutos	8.51x	9.33 minutos	12.35x	0.90008	0.97770	0.97770	0.97770
32	32	32	96	32.68 minutos	2.16x	13.01 minutos	8.86x	0.90443	0.97877	0.97877	0.97877

la versión secuencial y la versión distribuida de este clasificador. Ésto se logra haciendo uso de la Ecuación 2.3, la cual es un derivado de la Ley de Amdahl aplicada al análisis de sistemas distribuidos, mencionada en la Sección 2.2.2 y es utilizada para calcular la aceleración de una aplicación distribuida, comparada con su versión secuencial.

Como se puede observar en los resultados de los experimentos, el uso de un mayor número de núcleos destinados al procesamiento de la información afecta los tiempos de ejecución de los algoritmos de aprendizaje computacional, cuando se realiza el entrenamiento y la predicción. El gráfico presentado en la Figura 4.3 nos muestra que las etapas se aceleraron hasta *11.8x* en el entrenamiento y hasta *13.0x* en la predicción al compararlo con la versión secuencial. Un punto clave es que estas variaciones de tiempo no afectan la exactitud arrojada por el clasificador en la etapa anterior, obteniendo valores de rendimiento muy similares en todos los experimentos.

La mejor aceleración se logró cuando se usaron una cuarta parte de los núcleos disponibles. Esta configuración utilizó únicamente 8 núcleos en cada *nodo*, dando un total de 24 núcleos en el clúster. Cabe destacar que, después de este pico la aceleración comenzó a decaer. Esta situación puede observarse en la Figura 4.3 y se debe a diversos factores, entre los que destacan el particionamiento de los datos, la recuperación de la información después de realizar las operaciones de manera distribuida (*shuffle*), el tráfico y las limitaciones propias de la red.

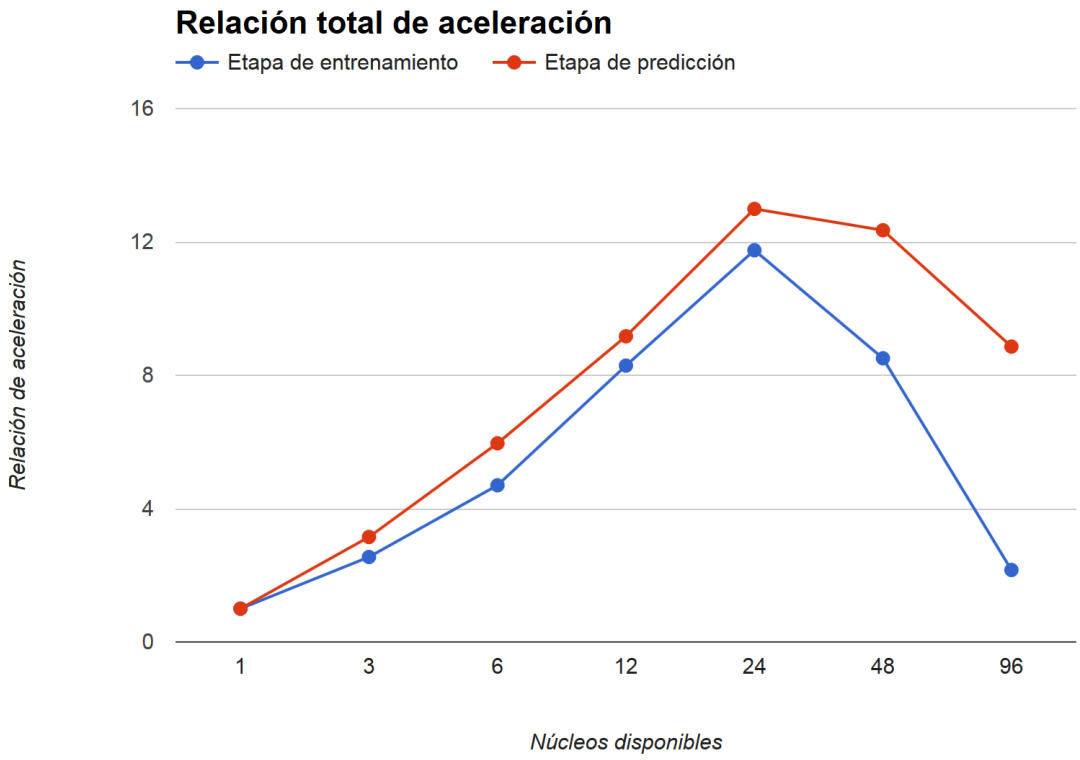


Figura 4.3: Relación total de aceleración de la etapa de entrenamiento y de predicción.

Capítulo 5

Conclusiones y trabajo a futuro

En este trabajo de tesis se analizó el desempeño de diversos algoritmos de aprendizaje computacional supervisado aplicados al reconocimiento distribuido de patrones de actividades físicas humanas. Dichos algoritmos se encuentran implementados en Spark, la cual es una herramienta que se enfoca en el cómputo distribuido a través de clústeres computacionales. Actualmente, dicha herramienta está ganando popularidad debido a su flexibilidad, los módulos de aprendizaje computacional que tiene integrados y la facilidad para manejar grandes volúmenes de información.

Los algoritmos utilizados fueron entrenados y probados con datos recolectados por usuarios con un teléfono inteligente a través de una aplicación móvil desarrollada para este propósito. Dicha aplicación móvil hace uso del acelerómetro y del giroscopio integrados en el teléfono inteligente para la obtención de los datos.

Se exhibió que mediante el uso de la aplicación de muestra (sistema cliente-servidor) es posible llevar el control de las actividades diarias de diversos usuarios simultáneamente. Además, con el uso de la aplicación desarrollada y el algoritmo de bosques aleatorios, se constató que se presenta un porcentaje de error de clasificación muy pequeño al reconocer las actividades sedentarias. Mientras que, las actividades no sedentarias, tienden a ser confundas entre ellas en mayor proporción. Por otra parte, también se observó que la confusión entre las actividades sedentarias y las actividades no sedentarias es mínima.

Por otro lado, durante los experimentos de ejecución secuencial y distribuida, se pudo observar que la implementación distribuida mejoró los tiempos ejecución en comparación con los tiempos de la ejecución secuencial. Además, se observó que el rendimiento en tiempo de ejecución de la implementación distribuida depende en gran parte del número de núcleos disponibles para el procesamiento.

Los resultados de la implementación distribuida del clasificador mostraron que, sin importar el número de recursos disponibles (número de nodos y número de núcleos de procesamiento), la exactitud del clasificador se mantuvo estable

(aproximadamente 90 % de exactitud).

Al realizar el análisis de los tiempos de ejecución (secuencial y distribuido) mediante el uso de un derivado de la ley de Amdahl, se observó que la versión distribuida mejoró el tiempo de ejecución en la etapa de entrenamiento aproximadamente 11.8x. Mientras que, la etapa de predicción se mejoró aproximadamente de 13.0x.

Por último, cabe señalar que después de alcanzar una máxima aceleración, a medida que se utilizaba una mayor cantidad de núcleos la rapidez empezó a decaer. Dicho comportamiento concuerda con lo mencionado en la ley de Amdahl, la cual indica que tras alcanzar un punto máximo en la aceleración, ésta deja de crecer a pesar de que se tengan más recursos disponibles. Dos causas de este comportamiento son la sincronización de los nodos y el tráfico de red generado inherentemente de las operaciones usadas por el clasificador.

De esta forma, se muestra que a través de un entorno de cómputo distribuido se mejora el rendimiento de ejecución de los algoritmos de clasificación al aplicarlos al reconocimiento de patrones de actividades físicas humanas, mostrando así la veracidad de la hipótesis de este trabajo de tesis.

Debido a que la exactitud mostrada por los clasificadores aún puede mejorarse, se deja para un trabajo futuro la utilización de métodos distintos para la caracterización de las actividades y la selección de características. Esto resultaría particularmente interesante porque podría impactar directamente en la exactitud y potencialmente en el tiempo de ejecución de los clasificadores.

Por otro lado, se tiene la posibilidad de agregar más recursos al clúster computacional para mantener una mayor replicación de los datos en los sistemas distribuidos y mejorar el procesamiento de estos. De esta forma el sistema lograría manejar una cantidad significativamente mayor de usuarios simultáneos utilizando esta aplicación.

Bibliografía

[Alsheikh et al., 2016] Alsheikh, M. A., Niyato, D., Lin, S., p. Tan, H., and Han, Z. (2016). Mobile big data analytics using deep learning and apache spark. *IEEE Network*, 30(3):22–29.

[Amazon, 2010] Amazon, W. S. (2010). Deploying a Flask application to AWS Elastic Beanstalk. https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create-deploy-python-flask.html. [Accedido el 13-03-2018].

[Amdahl, 1967] Amdahl, G. M. (1967). Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of Spring Joint ACM Computer Conference*, AFIPS '67, pages 483–485, NY, USA.

[Apache, 2010] Apache, S. F. (2010). Apache Cassandra. http://cassandra.apache.org/. [Accedido el 15-01-2018].

[Apache, 2011a] Apache, S. F. (2011a). Apache Hadoop. http://hadoop.apache.org/. [Accedido el 21-09-2017].

[Apache, 2011b] Apache, S. F. (2011b). Apache Kafka. https://kafka.apache.org/. [Accedido el 10-01-2018].

[Apache, 2014] Apache, S. F. (2014). Apache Spark. http://spark.apache.org/. [Accedido el 15-05-2017].

[Apache, 2015] Apache, S. F. (2015). Apache Flink. http://flink.apache.org/. [Accedido el 21-12-2017].

[Apache, 2017] Apache, S. F. (2017). Machine learning library (MLlib). https://spark.apache.org/docs/2.2.0/ml-guide.html. [Accedido el 31-10-2017].

[Azzi et al., 2014] Azzi, S., Bouzouane, A., Giroux, S., Dallaire, C., and Bouchard, B. (2014). Human activity recognition in big data smart home context. In *Proceedings of 2014 IEEE International Conference on Big Data (Big Data)*, pages 1–8, DC, USA.

[BenAbdelkader et al., 2002] BenAbdelkader, C., Cutler, R., and Davis, L. (2002). Stride and cadence as a biometric in automatic person identification and verification. In *Proceedings of Fifth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 372–377, DC, USA.

[Bian et al., 2009] Bian, J., Liu, Y., Zhou, D., Agichtein, E., and Zha, H. (2009). Learning to recognize reliable users and content in social media with coupled mutual reinforcement. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 51–60, Madrid, Spain.

[Bishop, 2006] Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer, New York, USA.

[Bravata et al., 2007] Bravata, D. M., Smith-Spangler, C., Sundaram, V., Gienger, A. L., Lin, N., Lewis, R., Stave, C. D., Olkin, I., and Sirard, J. R. (2007). Using pedometers to increase physical activity and improve health: a systematic review. *Journal of the American Medical Association*, 298(19):2296–2304.

[Breiman, 2001] Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

[Bruno et al., 2012] Bruno, B., Mastrogiovanni, F., Sgorbissa, A., Vernazza, T., and Zaccaria, R. (2012). Human motion modelling and recognition: a computational approach. In *Proceedings of 2012 IEEE International Conference on Automation Science and Engineering (CASE)*, pages 156–161, Seoul, South Korea.

[Carlson et al., 2010] Carlson, A., Betteridge, J., Wang, R. C., Hruschka, Jr., E. R., and Mitchell, T. M. (2010). Coupled semi-supervised learning for information extraction. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 101–110, NY, USA.

[Casale et al., 2011] Casale, P., Pujol, O., and Radeva, P. (2011). Human activity recognition from accelerometer data using a wearable device. In *Proceedings of the 5th Iberian Conference on Pattern Recognition and Image Analysis*, IbPRIA'11, pages 289–296, Las Palmas de Gran Canaria, Spain.

[Choujaa and Dulay, 2008] Choujaa, D. and Dulay, N. (2008). TRAcME: temporal activity recognition using mobile phone data. In *Proceedings of 2008 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing - Volume 01*, volume 1 of *EUC '08*, pages 119–126, DC, USA.

[CONADE, 2017] CONADE (2017). Estrategia nacional de activación física. https://www.gob.mx/conade/acciones-y-programas/muevete-en-30-30m. [Accedido el 12-02-2018].

[Cortes and Vapnik, 1995] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.

[Dean and Ghemawat, 2004] Dean, J. and Ghemawat, S. (2004). MapReduce: simplified data processing on large clusters. In *Proceedings of the 6th Conference*

on Symposium on Opearting Systems Design & Implementation - Volume 6, OSDI'04, pages 10–10, CA, USA.

[Duhamel and Vetterli, 1990] Duhamel, P. and Vetterli, M. (1990). Fast fourier transforms: a tutorial review and a state of the art. *Signal processing*, 19(4):259–299.

[Eunju et al., 2010] Eunju, K., Sumi, H., and Diane, C. (2010). Human activity recognition and pattern discovery. *IEEE Pervasive Computing*, 9(1):48–53.

[Friedman, 2002] Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378.

[Ghahramani, 2004] Ghahramani, Z. (2004). Unsupervised learning. In *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia*, volume 3176, pages 72–112. Springer-Verlag.

[Google, 2013] Google, I. (2013). Instalación de Android Studio. https://developer.android.com/studio/install.html. [Accedido el 2-11-2017].

[Google, 2014] Google, I. (2014). Material design. https://material.io/. [Accedido el 13-12-2017].

[Ho, 2004] Ho, J. J. C. (2004). *Interruptions: using activity transitions to trigger proactive messages*. PhD thesis, Massachusetts Institute of Technology.

[Hsu et al., 2003] Hsu, C.-W., Chang, C.-C., and Lin, C.-J. (2003). A practical guide to support vector classification. https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf. [Accedido el 15-01-2018].

[Iglesias et al., 2011] Iglesias, J., Cano, J., Bernardos, A. M., and Casar, J. R. (2011). A ubiquitous activity-monitor to prevent sedentariness. In *Procee-*

dings of 2011 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), pages 319–321, WA, USA.

[IMSS, 2015] IMSS (2015). Chécate, mídete, muévete. http://checatemidetemuevete.gob.mx/. [Accedido el 01-10-2017].

[INEGI, 2016] INEGI (2016). Módulo de práctica deportiva y ejercicio físico (MOPRADEF). http://www.beta.inegi.org.mx/proyectos/enchogares/modulos/mopradef/default.html. [Accedido el 21-09-2017].

[INEGI, 2017] INEGI (2017). Estadísticas a propósito del día mundial de Internet (17 de mayo). http://www.inegi.org.mx/saladeprensa/aproposito/2017/internet2017_Nal.pdf. [Accedido el 28-06-2017].

[JetBrains, 2017] JetBrains, s. (2017). Install and set up IntelliJ IDEA - Help. https://www.jetbrains.com/help/idea/install-and-set-up-intellij-idea.html. [Accedido el 2-11-2017].

[Karantonis et al., 2006] Karantonis, D. M., Narayanan, M. R., Mathie, M., Lovell, N. H., and Celler, B. G. (2006). Implementation of a real-time human movement classifier using a triaxial accelerometer for ambulatory monitoring. *IEEE transactions on information technology in biomedicine*, 10(1):156–167.

[Karau et al., 2015] Karau, H., Konwinski, A., Wendell, P., and Zaharia, M. (2015). *Learning Spark*. O' Reilly Media, Inc., Sebastopol, CA, USA.

[Khan et al., 2010] Khan, A. M., Lee, Y.-K., Lee, S., and Kim, T.-S. (2010). Human activity recognition via an accelerometer-enabled-smartphone using kernel discriminant analysis. In *Proceedings of 2010 5th IEEE International Conference on Future Information Technology (FutureTech)*, pages 1–6, Busan, South Korea.

[Kohl et al., 2012] Kohl, H. W., Craig, C. L., Lambert, E. V., Inoue, S., Alkandari, J. R., Leetongin, G., and Kahlmeier, S. (2012). The pandemic of physical inactivity: global action for public health. *The Lancet*, 380(9838):294 – 305.

[Kshemkalyani and Singhal, 2011] Kshemkalyani, A. D. and Singhal, M. (2011). *Distributed computing: principles, algorithms, and systems*. Cambridge University Press., NY, USA.

[LeCun et al., 1989] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

[Lee et al., 2012] Lee, I.-M., Shiroma, E. J., Lobelo, F., Puska, P., Blair, S. N., and Katzmarzyk, P. T. (2012). Effect of physical inactivity on major non-communicable diseases worldwide: an analysis of burden of disease and life expectancy. *The Lancet*, 380(9838):219 – 229.

[Lee and Cho, 2011] Lee, Y.-S. and Cho, S.-B. (2011). Activity recognition using hierarchical hidden Markov models on a smartphone with 3d accelerometer. In *Proceedings of 6th International Conference on Hybrid Artificial Intelligent Systems, Part I, HAIS 2011*, pages 460–467, Wroclaw, Poland. Springer-Verlag.

[Malak and East, 2016] Malak, M. and East, R. (2016). *Spark GraphX in action*. Manning Publications Company, Shelter Island, NY, USA.

[McAfee and Brynjolfsson, 2012] McAfee, A. and Brynjolfsson, E. (2012). Big data: the management revolution. *Harvard Business Review*, 90(10):60–68.

[Murphy, 2012] Murphy, K. P. (2012). *Machine learning: A probabilistic perspective*. MIT Press.

[Netflix, 2012] Netflix, T. B. (2012). Netflix recommendations: Beyond the 5 stars. https://medium.com/netflix-techblog/netflix-recommendations-beyond-the-5-stars-part-1-55838468f429. [Accedido el 27-10-2017].

[Ortiz, 2014] Ortiz, J. L. R. (2014). *Smartphone-based human activity recognition*. PhD thesis, Universidad Politécnica de Cataluña, Barcelona, España.

[Paniagua et al., 2012] Paniagua, C., Flores, H., and Srirama, S. N. (2012). Mobile sensor data classication for human activity recognition using MapReduce on cloud. *Procedia Computer Science*, 10(Supplement C):585 – 592.

[Parkka et al., 2006] Parkka, J., Ermes, M., Korpipaa, P., Mantyjarvi, J., Peltola, J., and Korhonen, I. (2006). Activity classification using realistic data from wearable sensors. *IEEE Transactions on Information Technology in Biomedicine*, 10(1):119–128.

[Phunchongharn et al., 2010] Phunchongharn, P., Hossain, E., Niyato, D., and Camorlinga, S. (2010). A cognitive radio system for e-health applications in a hospital environment. *IEEE Wireless Communications*, 17(1):20–28.

[Pusher, 2010] Pusher, L. (2010). Pusher. https://pusher.com/. [Accedido el 15-01-2018].

[Python, 2016] Python, S. F. (2016). Python packaging authority. https://www.pypa.io/en/latest/. [Accedido el 17-09-2017].

[Quinlan, 1986] Quinlan, J. R. (1986). Induction of Decision Trees. *Machine Learning*, 1(1):81–106.

[Ronacher, 2017] Ronacher, A. (2017). mod_wsgi (Apache). http://flask.pocoo.org/docs/0.12/deploying/mod_wsgi/. [Accedido el 13-03-2018].

[Ronao and Cho, 2016] Ronao, C. A. and Cho, S.-B. (2016). Human activity recognition with smartphone sensors using deep learning neural networks. *Expert Systems with Applications*, 59:235 – 244.

[Samà, 2013] Samà, A. (2013). *Human movement analysis by means of accelerometers: Application to human gait and motor symptoms of Parkinsons Disease*. PhD thesis, Universitat Politécnica de Catalunya, Barcelona, España.

[Samà et al., 2011] Samà, A., Angulo, C., Pardo, D., Català, A., and Cabestany, J. (2011). Analyzing human gait and posture by combining feature selection and kernel methods. *Neurocomputing*, 74(16):2665–2674.

[SEDESA, 2008] SEDESA (2008). Campaña muévete y métete en cintura. http://www.noalaobesidad.df.gob.mx/index.php?option=com_content&view=article&id=70&Itemid=55. [Accedido el 12-02-2018].

[Sokolova and Lapalme, 2009] Sokolova, M. and Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437.

[Srirama et al., 2011] Srirama, S. N., Flores, H., and Paniagua, C. (2011). Zompopo: mobile calendar prediction based on human activities recognition using the accelerometer and cloud services. In *Proceedings 2011 Fifth International Conference on Next Generation Mobile Applications, Services and Technologies, NGMAST'11*, pages 63–69, Cardiff, UK.

[Stallings, 2009] Stallings, W. (2009). *Computer organization and architecture: designing for performance*. Prentice Hall Press, Upper Saddle River, NJ, USA, 8th edition.

[Strath et al., 2013] Strath, S. J., Kaminsky, L. A., Ainsworth, B. E., Ekelund, U., Freedson, P. S., Gary, R. A., Richardson, C. R., Smith, D. T., and Swartz,

A. M. (2013). Guide to the assessment of physical activity: clinical and research applications. *Circulation*, 128(20):2259–2279.

[Su et al., 2014] Su, X., Tong, H., and Ji, P. (2014). Activity recognition with smartphone sensors. *Tsinghua Science and Technology*, 19(3):235–249.

[Sutton and Barto, 1998] Sutton, R. S. and Barto, A. G. (1998). Reinforcement learning: an introduction. *Neural Networks*, 13(1):360.

[Tapia et al., 2007] Tapia, E. M., Intille, S. S., Haskell, W., Larson, K., Wright, J., King, A., and Friedman, R. (2007). Real-time recognition of physical activities and their intensities using wireless accelerometers and a heart rate monitor. In *Proceedings of 11th IEEE International Symposium on Wearable Computers 2007*, pages 37–40, Boston, USA.

[WHO, 2010] WHO (2010). Global recommendations on physical activity for health. http://www.who.int/dietphysicalactivity/publications/9789241599979/en/. [Accedido el 17-01-2018].

[WHO, 2014] WHO (2014). Physical inactivity: a global public health problem. http://www.who.int/dietphysicalactivity/factsheet_inactivity/en/. [Accedido el 27-09-2017].

[Yang et al., 2008] Yang, J.-Y., Wang, J.-S., and Chen, Y.-P. (2008). Using acceleration measurements for activity recognition: An effective learning algorithm for constructing neural classifiers. *Pattern recognition letters*, 29(16):2213–2220.

[Zhu and Goldberg, 2009] Zhu, X. and Goldberg, A. (2009). Introduction to semi-supervised learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 3(1):1–130.

[Zuberek, 2011] Zuberek, W. (2011). Analysis of the Speedup of Distributed Applications. http://web.cs.mun.ca/~wlodek/pdf/12-KIRY.pdf. [Accedido el 07-02-2018].

Anexo A

Biblioteca para la generación de características

Como parte de la metodología descrita en la Sección 1.7, Etapa 1 (Figura 1.2b), se requiere de la recolección de datos a través del acelerómetro y giroscopio de los teléfonos inteligentes y la caracterización de dichos datos. Con el propósito que el lector pueda replicar el presente proyecto de tesis, es necesario instalar el software necesario para extraer dicha información de los sensores del teléfono, procesarla y posteriormente, generar un vector de 561 características que describa las actividades sensadas. En el presente anexo, se describe como configurar y utilizar el módulo generador de características. Dicho módulo se encarga de crear un vector con las características que describen una actividad física a partir de las señales producidas por el teléfono inteligente.

Para poder usar la biblioteca desarrollada (generador_caracteristicas.py) es necesario instalar una serie de bibliotecas adicionales. La manera más simple de realizarlo es haciendo uso de PIP Python [Python, 2016], el cual es un sistema de

```
1  sudo apt-get update && sudo apt-get -y upgrade
2  sudo apt-get install python-pip
3  sudo apt-get install python-tk
```

Figura A.1: Script para la instalación de Python PIP.

```
1  pip install -r requerimientos/generador.txt
```

Figura A.2: Script para instalar las dependencias del generador de características.

gestión de paquetes utilizado para instalar y administrar paquetes de software escritos en el lenguaje de programación Python. Para la instalación en un sistema Ubuntu se utiliza el script mostrado en la Figura A.1.

Para terminar la instalación es necesario instalar las dependencias, lo cual se puede hacer utilizando el script de la Figura A.2. El archivo *requerimientos/generador.txt* tiene listadas las bibliotecas a utilizar con sus respectivas versiones, el cual se encuentra en el disco de instalación de la aplicación adjunto a este documento de tesis.

Para utilizar la biblioteca se debe importar el archivo y llamar al método *generar*, el cual recibe 9 parámetros, que corresponden a vectores de números de punto flotante. Cada vector debe contener 128 elementos, que representan información obtenida a través del acelerómetro y el giroscopio del teléfono móvil:

- *bodyAccX*: Señal del acelerómetro con respecto al cuerpo en el eje X.
- *bodyAccY*: Señal del acelerómetro con respecto al cuerpo en el eje Y.
- *bodyAccZ*: Señal del acelerómetro con respecto al cuerpo en el eje Z.
- *gravityAccX*: Señal del acelerómetro con respecto a la gravedad de la Tierra en el eje X.
- *gravityAccY*: Señal del acelerómetro con respecto a la gravedad de la Tierra en el eje Y.
- *gravityAccZ*: Señal del acelerómetro con respecto a la gravedad de la Tierra en el eje Z.
- *bodyGyroX*: Señal del giroscopio en el eje X.

- *bodyGyroY*: Señal del giroscopio en el eje Y.
- *bodyGyroZ*: Señal del giroscopio en el eje Z.

El método devuelve un vector con 561 características de punto flotante. Un ejemplo de su utilización lo pueden encontrar en el archivo */ejemplos/generador.py* del disco de instalación.

Anexo B

Pseudocódigo de algoritmos utilizados

Parte central del presente proyecto de tesis es la selección del algoritmo de aprendizaje computacional integrado en Apache Spark, que sea adecuado para este problema de clasificación de actividades físicas humanas. En este anexo se muestran precisamente los pseudocódigos de los tres principales algoritmos seleccionados en la metodología presentada en la Sección 1.7, Etapa 2, Fase 1 (Figura 1.2c), mismos que se presentan a continuación.

Pseudocódigo del árbol de potenciación del gradiente

Pseudcódigo 1 (Pseudocódigo del algoritmo de potenciación del gradiente).

```
1  procedure GBT(y, x = {x_1, ..., x_n})
2     F_0(x) = argmin_γ Σ_{i-1}^{N} Ψ(y_i, γ).
3
4     for m = desde 1 hasta M hacer:
5          ỹ_im = −[∂Ψ(y_i, F(x_i)) / ∂F(x_i)]_{F(x)=F_{m-1}(x)}, i = 1, N
6          R_lm 1^L = L - nodo terminal del árbol({ỹ_im, x_i}_1^N)
7          γ_lm = argmin_γ Σ_{x_i ∈ R_lm} Ψ(y_i, F_{m-1}(x_i) + γ)
8          F_m(X) = F_{m-1}(x) + v · γ_lm 1(x ∈ R_lm)
9     end for
10 end procedure
```

Pseudocódigo de los árboles aleatorios

Pseudcódigo 2 (Pseudocódigo del algoritmo de árboles aleatorios).

```
1  Entrada:
2    Un conjunto de entrenamiento D = {(x_1, y_1), ..., (x_n, y_n)}
3    Subconjunto de características de tamaño K
4
5  procedure (D, K)
6    N ← crear un árbol nodo basado en D
7    if todas las instancias están en la misma clase entonces
8      return N
9    end if
10
11   F ← conjunto de características que aún se pueden dividir
12   if F es un conjunto vacío entonces
13     return N
14   end if
15
16   F̃ ← seleccionar K características de F de manera aleatoria
17   N.f ← las características que tengan el mejor punto de
         separación en F̃
18   N.p ← el mejor punto de separación en N.f
19   D_l ← subconjunto de D con los valores de N.f que sean más
         pequeños que los de N.p
20   D_r ← subconjunto de D con los valores de N.f que no sean más
         pequeños que los de N.p
21   N_l ← llamada al proceso con los parámetros (D_l, K)
22   N_r ← llamada al proceso con los parámetros (D_r, K)
23   return N
24 end procedure
```

Pseudocódigo del perceptrón multicapa

Pseudcódigo 3 (Pseudocódigo del algoritmo del perceptrón multicapa).

```
1  procedure MLP(X, T, W)
2    Inicializar W con números pequeños aleatorios
3    Aleatorizar el orden de los ejemplos de entrenamiento en X
4
5    while error no converge hacer
6      for n ← 1, N hacer
7        for k ← 1, K hacer
8          y_k^n ← Σ_{i=1}^{d} W_ki X_i^n + b_k
9          δ_k^n ← y_k^n − t_k^n
10         for i ← 1, d hacer
11           W_ki ← W_ki − η * δ_k^n * X_i^n
12         end for
13         b_k ← b_k − η * δ_k^n
14       end for
15     end for
16   end while
17 end procedure
```

Anexo C

Configuración y manual de usuario de aplicación

De acuerdo con el desarrollo del proyecto del Capítulo 3, el sistema desarrollado para el seguimiento de actividades físicas humanas incluye una aplicación móvil para el teléfono inteligente, un sistema de reconocimiento de patrones de actividades físicas humanas (clasificador) y un sistema de seguimiento de actividades físicas (estos dos últimos instalados en un servidor y en un clúster computacional respectivamente). Para replicar este proyecto, es necesario contar con la configuración y el software adecuado, mismos que son presentados en el presente anexo.

C.1. Configuración

Para la instalación de la aplicación es recomendable utilizar un ambiente virtual, en este caso *virtualenv*. *Virtualenv* es una herramienta usada para crear ambientes aislados de Python con la finalidad de tener distintas configuraciones en cada proyecto. Para esto primero instale PIP (ver Anexo A) en caso de no tenerlo instalado. Ahora, para instalar la herramienta *virutalenv* ejecute el comando de la Figura C.1 en la terminal bash de Ubuntu.

```
1  pip install virtualenv
```

Figura C.1: Comando para la instalación del ambiente virtual (*virtualenv*).

```
1  virtualenv -p venv
2  source venv/bin/activate
3  pip install -r requerimientos/aplicacion.txt
```

Figura C.2: Script para crear un ambiente virtual e instalar las dependencias del proyecto Python.

Antes de ejecutar el proyecto, primero se debe crear el ambiente virtual e instalar las bibliotecas necesarias. Para esto, es necesario cambiarse a la carpeta raíz del proyecto y ejecutar el script de la Figura C.2, con lo cual se tendrá configurado un ambiente virtual completamente nuevo para trabajar e instalar las bibliotecas necesarias. El archivo *requerimientos-aplicacion.txt* se localiza en el disco de instalación de la aplicación, el cual también se encuentra adjunto a este documento de tesis.

Hasta este punto, ya se debería tiene instalado y configurado Apache Kafka, Apache Cassandra y PostgresSQL, tal como se menciona en la Sección 3.2. Además, el usuario necesita contar con una cuenta de correo electrónico que permita conexiones SMTP. Cabe mencionar que actualmente gran parte de los servicios de correo electrónico ya lo proveen de manera gratuita. El siguiente paso es crear un usuario en la base de datos PostgresSQL. También, es necesario crear un *keyspace* en Apache Cassandra y 2 Colas de mensajes en Apache Kafka. La primera cola de mensajes está destinada a almacenar las actividades a reconocer y la segunda a las actividades ya reconocidas.

El siguiente paso es agregarlos a la configuración del sistema. Eso se logra modificando el archivo *config.py*. Un ejemplo de la configuración es el mostrado en la Figura C.3. Asimismo, la configuración del clasificador de actividades se realiza modificando el archivo *application.conf*. Un ejemplo de este archivo de

configuración es el mostrado en la Figura C.4.

Para finalizar la configuración del proyecto, se tienen que crear los modelos de las tablas de datos y migrarlas a los gestores de base de datos. Para esto, ejecute el script de la Figura C.5.

Para iniciar toda la aplicación se deberán ejecutar 3 scripts en distintas instancias de bash. Estos scripts se describen a continuación:

- La aplicación Web. Colocarse en la carpeta raíz de la aplicación e iniciarla ejecutando el script de la Figura C.6.
- Los consimidores de las colas de mensajes. Situarse en el directorio raíz del proyecto e iniciar el consumidor con el script mostrado en la Figura C.7.
- El clasificador de actividades. Una forma de inicarlo es mediante el script de la Figura C.8.

La aplicación Web por defecto funciona en el puerto 5000 y se inicia en modo desarrollador, lo que permite monitorear los mensajes de depuración. Para desplegar el proyecto en un ambiente de producción se recomienda hacer uso de alguna herramienta como Elastic Beanstalk [Amazon, 2010], Mod Wsgi Apache [Ronacher, 2017] o alguna otra herramienta dependiendo del servidor que se vaya a utilizar.

En el caso de la aplicación móvil, se debe configurar los puntos de acceso con los cuales se realizan los intercambios de información entre la aplicación Web y la aplicación móvil. Para realizarlo, primero es necesario situarse en la raíz del proyecto de la aplicación móvil y posteriormente modificar el archivo "app/src/main/res/values/strings.xml". En este archivo, se debe reemplazar la dirección "http://192.168.239.54:5000" con la dirección IP o dominio en el que fue instalado la aplicación Web, los campos a modificar son mostrados en la Figura C.9. Al término de esta modificación se tiene tiene que compilar el proyecto y

```python
# Configuración de proyecto en modo desarrollo
class DevelopmentConfig(Config):
  DEBUG = True
  # Configuración de la base de datos PostgresSQL
  SQLALCHEMY_DATABASE_URI = os.environ.get('DEV_DATABASE_URL') \
    or 'postgresql://usuario:password@127.0.0.1/tesisdb'

  # Configuración de base de datos Cassandra
  CASSANDRA_HOSTS         = ['127.0.0.1']
  CASSANDRA_KEYSPACE      = "dev_tesis"
  CASSANDRA_SETUP_KWARGS  = {'protocol_version': 3}

  # Configuración de las cola de Kafka
  # Cola de mensajes para actividades por clasificar
  KAFKA_TOPIC_CHARACTERISTICS       = 'por_clasificar'
  KAFKA_SERVERS_CHARACTERISTICS     = ['localhost:9092']
  KAFKA_API_VERSION_CHARACTERISTICS = (0,10)
  # Cola de mensajes de actividades ya clasificadas
  KAFKA_TOPIC_CLASSIFIED       = 'ya_clasificados'
  KAFKA_SERVERS_CLASSIFIED     = ['localhost:9092']
  KAFKA_API_VERSION_CLASSIFIED = (0,10)

  # Datos de configuración del pusher. Estos se obtienen
  # al registrar la aplicación en pusher.com
  PUSHER_APP_ID  = '0000'
  PUSHER_KEY     = '0000'
  PUSHER_SECRET  = '0000'
  PUSHER_CLUSTER = '0000'
  PUSHER_SSL     = True

  # Datos de acceso al servidor de correo electrónico
  MAIL_SERVER = 'smtp.googlemail.com'
  MAIL_PORT = 587
  MAIL_USE_TLS = True
  MAIL_USERNAME = 'prueba@gmail.com'
  MAIL_PASSWORD = 'password'

  # Datos de cabecera para los correo electrónicos
  FLASKY_MAIL_SUBJECT_PREFIX = '[Sitema de seguimiento]'
  FLASKY_MAIL_SENDER = 'Sistema seguimiento <contacto@test.com>'
```

Figura C.3: Ejemplo de archivo de configuración del proyecto Python (modo desarrollador).

```
kafka_servidor_a_clasificar = "192.168.1.200:9092"
kafka_a_clasificar = "a_clasificar"
min_max_scaler = "hdfs://192.168.1.200:9000/hopkins/spark/minmax-data-scaler"
rf_model = "hdfs://192.168.1.200:9000/hopkins/spark/model-random-forest"
kafka_servidor_clasificado = "192.168.1.200:9092"
kafka_clasificado = "test002"
backup_checkpoint = "hdfs://192.168.1.200:9000/hopkins/spark/sqs"
```

Figura C.4: Ejemplo de archivo de configuración del clasificador.

```
python manage.py db init
python manage.py db migrate
python manage.py db upgrade
```

Figura C.5: Script para la migración de las bases de datos (PostgresSQL y Cassandra).

```
source venv/bin/activate
python manage.py runserver --host 0.0.0.0
```

Figura C.6: Script para iniciar la aplicación Python.

```
source venv/bin/activate
cd app-seguimiento
python consumer.py
```

Figura C.7: Script para iniciar el consumidor de mensajes de muestras clasificadas.

```
spark-submit \
 --packages org.apache.spark:spark-sql-kafka-0-10_2.11:2.2.0, com.typesafe:config:1.3.3 \
 --files /path/to/application.conf \
 /path/to/liveclassificationusingkafka_2.11-0.1.jar \
 /path/to/application.conf
```

Figura C.8: Script para iniciar el clasificador de actividades.

```
1  <string name="action_index_web">
2      <u>http://192.168.239.54:5000/"</u></string>
3  <string name="API_SIGNUP">
4      http://192.168.239.54:5000/signup</string>
5  <string name="API">
6      http://192.168.239.54:5000/api/v1.0/</string>
7  <string name="API_MAIN">
8      http://192.168.239.54:5000/</string>
```

Figura C.9: Ejemplo de configuración de puntos de acceso en la aplicación móvil.

generar nuevamente el archivo APK. Ésto último puede ser realizado con facilidad con la herramienta de desarrollo Android Studio.

C.2. Manual de usuario

C.2.1. Aplicación Web

Para hacer uso de la aplicación basta con conectarse al servidor que lo aloja, en el caso de usar el modo desarrollador se puede acceder mediante la IP del servidor y apuntando al puerto 5000. Ejemplo, escribir en la barra de navegación de su navegador Web la dirección *http://192.168.239.54:5000*. Al realizar esta acción es redirigido a un página para iniciar sesión (Véase Figura C.10). En el caso de tener una cuenta en el sistema, se puede iniciar sesión con los datos con los que se realizó el registro previo; por el contrario, en la pantalla de inicio de sesión hacer clic en el enlace *"Clic aquí para registrarse"* para ser redirigido al formulario de registro.

Para crear una cuenta en el sistema se tiene que rellenar los datos solicitados en el formulario de registro (Véase Figura C.11) y al terminar dar clic en el botón "Registrarse". Al hacerlo, es redirigido a la página de inicio de sesión. Es importante guardar los datos con los que se realizó el registro, ya que con éstos se accede al sistema desde la plataforma Web y la aplicación móvil. Además, se envía un mensaje de correo electrónico para que el usuario pueda confirmar la

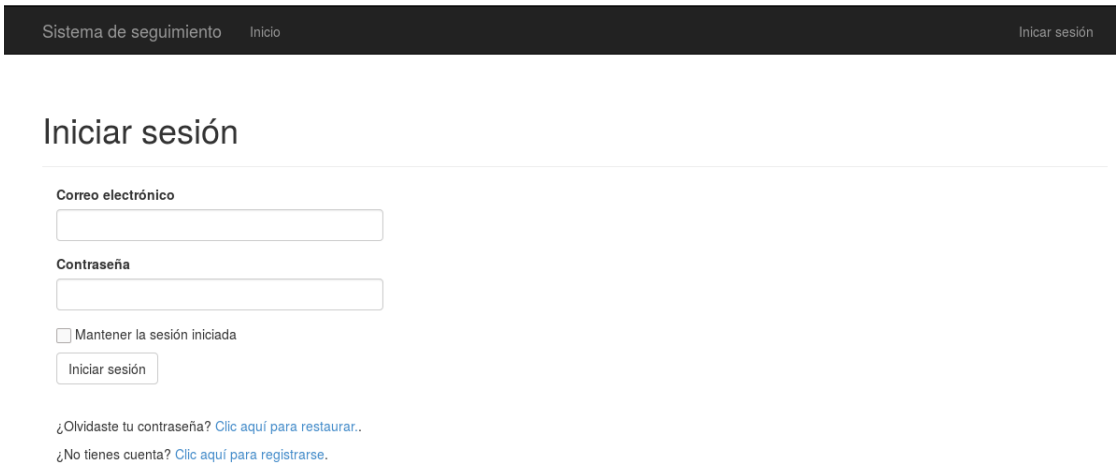


Figura C.10: Pantalla de inicio de sesión en la aplicación Web.

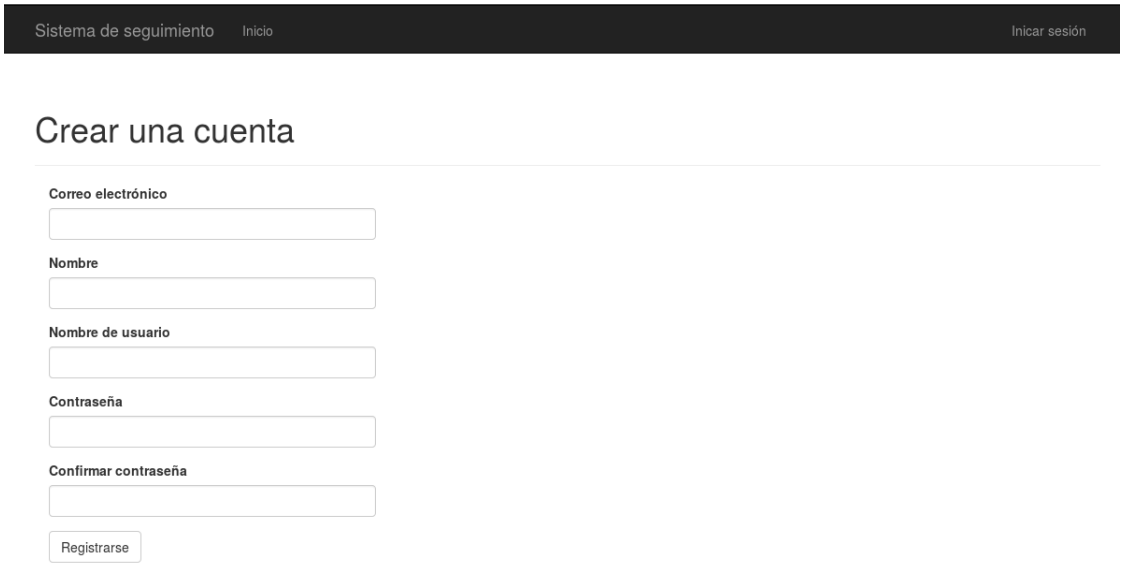


Figura C.11: Pantalla para el registro en la aplicación.

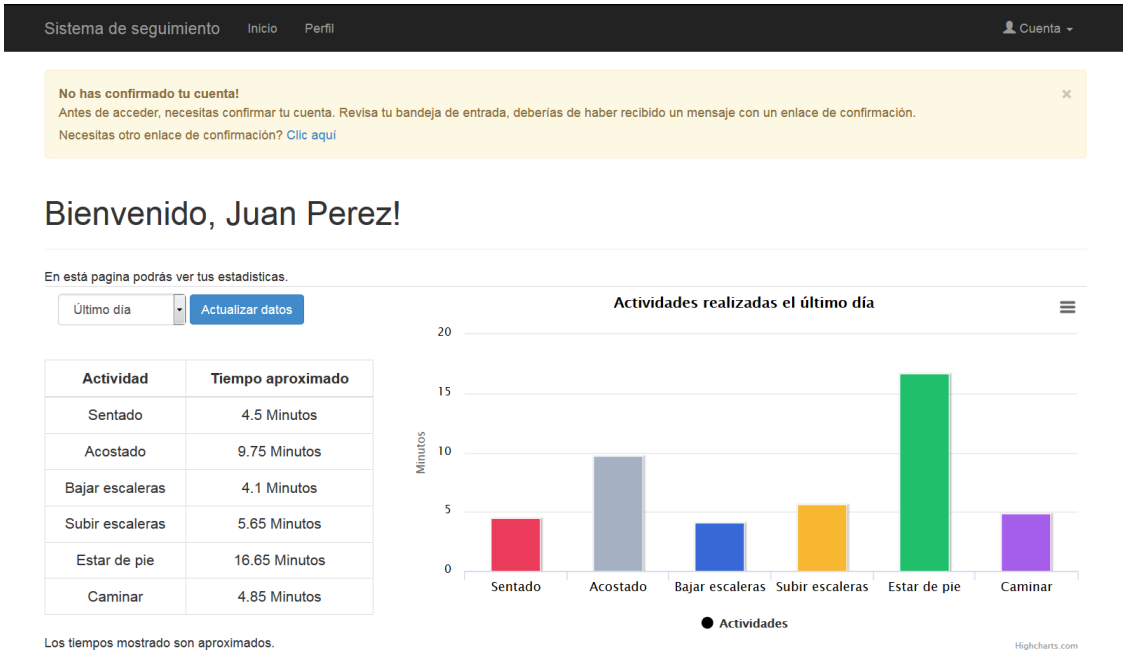


Figura C.12: Pantalla principal de la aplicación Web.

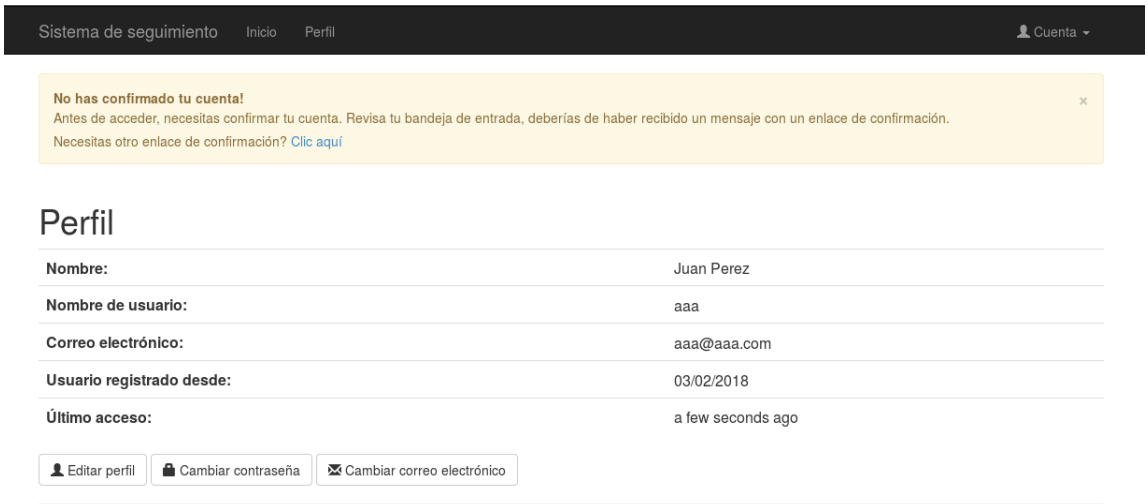


Figura C.13: Pantalla de perfil de usuario.

cuenta creada. Ésto puede solucionar eventualidades como la recuperación de la contraseña, por ejemplo.

Después de iniciar sesión en la aplicación Web, se observa la pantalla principal (Véase Figura C.12), en la cual se muestra el historial de actividad y las notificaciones en caso de tener alguna. Además, desde esta pantalla se puede acceder al perfil de usuario (Véase Figura C.13), en la cual se visualiza los datos con la que se creo la cuenta en el sistema e/o información que puede ser modificada.

C.2.2. Aplicación móvil

La instalación de la aplicación móvil puede hacerse copiando el archivo APK, el cual es generado en la Sección C.1, abriendo el archivo APK en el teléfono y confirmando la instalación. Una vez instalada, ésta es listada en el menú de aplicaciones. En la Figura C.14 se puede ver el icono de la aplicación ya instalada, la cual tiene por nombre *Sistema de seguimiento de actividades*. En la Figura C.15 se muestran las principales pantallas de la aplicación móvil.

Al iniciar la aplicación se observa la pantalla mostrada en la Figura C.15a, en la cual es necesario ingresar sus datos para acceder a la aplicación y empezar a darle seguimiento a sus actividades. En el caso de no tener una cuenta, es posible crear una dando clic sobre la frase *"¿Aún no tienes cuenta? Regístrate"*, con la cual es redirigido a al navegador Web y desde donde es posible gestionarla.

Después de iniciar sesión puede verse una ventana como la mostrada en la Figura C.15b, la cual indica que los datos están siendo validados. Al término de la validación de los datos ingresados, la aplicación es redirigida a la pantalla mostrada en la Figura C.15c, la cual es la pantalla principal e indica que ya se están censando datos. Desde ahí, dando clic en el enlace "*Estadísticas de usuario*", es redirigido a la aplicación Web desde la cual se puede dar un seguimiento a sus actividades mediante las estadísticas mostradas.

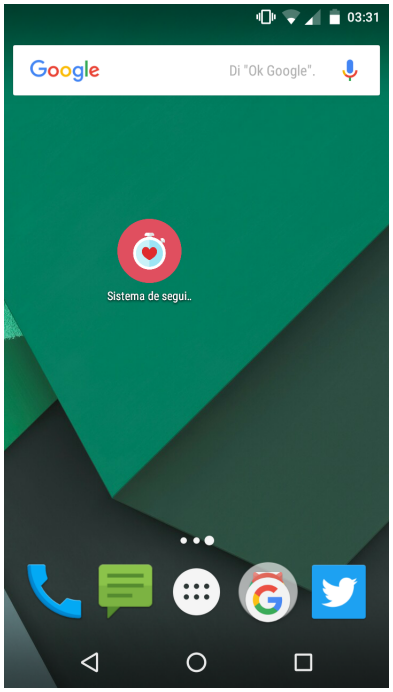
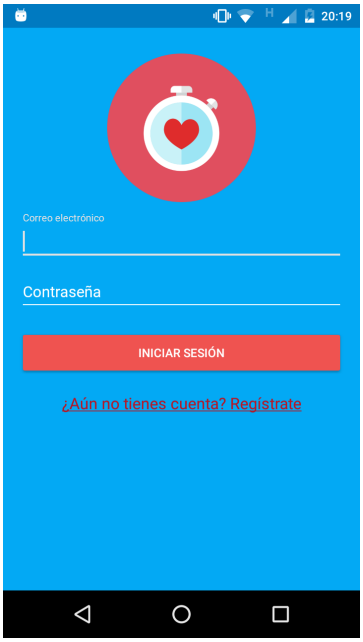
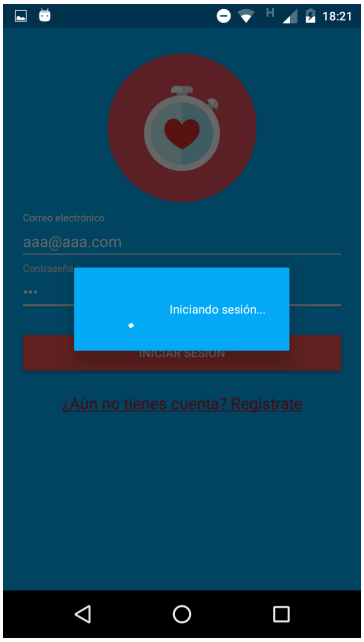


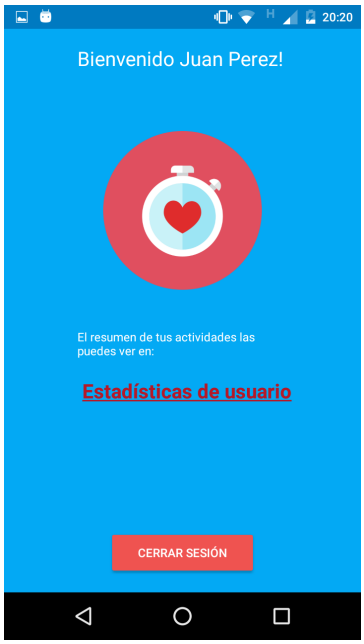
Figura C.14: Captura de pantalla del teléfono móvil con la aplicación instalada.



(a) Pantalla de inicio de sesión de la aplicación móvil.



(b) Pantalla de validación de datos al iniciar sesión de la aplicación móvil.



(c) Pantalla principal de la aplicación móvil.

Figura C.15: Captura de las principales pantallas de la aplicación móvil para la adquisición de datos.