Pontificia Universidad Católica del Ecuador

Facultad De Ingeniería

Escuela de Sistemas

TEMA:

Prototipo funcional de una herramienta de inteligencia artificial para la detección de caídas mediante el uso de los sensores de un dispositivo móvil utilizando CRISP-DM.

AUTOR:

Bryan Stalin Flores Flores

TRABAJO PREVIA A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS DE INFORMACIÓN

QUITO, 2023 – 2024

DEDICATORIA

Deseo dedicar este trabajo a todas las personas que han sido una inspiración y apoyo en mi vida. A mis padres, quienes me han inculcado valores y principios, y me han brindado su amor y confianza incondicional en todo momento. A mi hermana y mi sobrina, quienes han sido mi ejemplo de perseverancia y determinación. A mis amigos, quienes han sido mi compañía en las buenas y en las malas, y han compartido conmigo momentos inolvidables.

También quisiera agradecer a Mgtr. Charles Escobar mi tutor por su guía y conocimientos compartidos, los cuales fueron fundamentales para la culminación de este proyecto. Asimismo, agradezco a todos los profesores y compañeros que me han brindado su apoyo y enseñanzas a lo largo de mi formación académica.

AGRADECIMIENTO

Quiero expresar mi más sincero agradecimiento a la Pontificia Universidad Católica del Ecuador, por brindarme la oportunidad de formarme como profesional. Quiero agradecer especialmente a la Ing. Susana Masapanta, Ing. Miguel Ortiz y Mgtr. Charles Escobar por su valiosa enseñanza, paciencia, dedicación, apoyo incondicional y amistad, que me han ayudado a crecer día a día como profesional. Estoy muy agradecido por haber tenido la oportunidad de aprender de ustedes y de su gran experiencia en el campo de la ingeniería.

RESUMEN

El presente proyecto de titulación aborda el diseño e implementación de un sistema de detección de caídas de dispositivos móviles utilizando algoritmos de aprendizaje automático y una aplicación móvil para la recolección de datos de sensores de movimiento. Este sistema tiene como objetivo proporcionar una solución efectiva y accesible para la detección temprana de caídas de dispositivos móviles, lo que permitiría prevenir posibles daños y mejorar la experiencia de usuario.

La aplicación móvil, desarrollada en Android Studio, recolecta datos de los sensores de movimiento (acelerómetro y giroscopio) de dispositivos móviles y los envía a una base de datos en Firebase. Se recolectan dos muestras por segundo de los tres ejes del acelerómetro durante cinco segundos y se almacenan para su posterior análisis. La aplicación permite la recolección de datos en tiempo real y se utilizó en la fase de desarrollo del proyecto.

Se utilizó TensorFlow y Keras para desarrollar una red neuronal artificial que fue entrenada con los datos recolectados. Posteriormente, se exportó el modelo utilizando TensorFlow.js para ser implementado en una página web desarrollada con React. Esta página permite visualizar en tiempo real los resultados de las predicciones de caídas de dispositivos móviles.

ÍNDICE

ÍNDICE DE FIGURAS, GRÁFICOS Y TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO I: INTRODUCCIÓN

1. Marco De Referencia

1.1. Justificación

"En la actualidad, la proliferación de dispositivos móviles y sus avances tecnológicos han permitido el desarrollo de aplicaciones que facilitan y mejoran la vida cotidiana de las personas" (Statista, 2021, p. 13). "Uno de los aspectos en los que se puede aprovechar la tecnología de estos dispositivos es en la detección de caídas, lo cual tiene aplicaciones en diferentes ámbitos, como la seguridad, la salud y la industria" (Rahimi, Recht, & Taylor, 2018, p. 18).

"El uso de la inteligencia artificial y el aprendizaje automático permite analizar y procesar datos de los sensores de dispositivos móviles de forma eficiente y precisa, lo cual es fundamental para la detección de caídas" (Khan & Hoey, 2017, p. 25). "El proceso CRISP-DM (Cross-Industry Standard Process for Data Mining) es una metodología ampliamente utilizada en la industria y la academia para guiar el proceso de descubrimiento de conocimiento en bases de datos y garantizar la calidad y eficacia de los modelos de aprendizaje automático" (Chapman et al., 2000, p. 33).

La justificación del presente proyecto de titulación radica en la necesidad de desarrollar un prototipo funcional de una herramienta de inteligencia artificial que, mediante el uso de los sensores de un dispositivo móvil y la aplicación de la metodología CRISP-DM, sea capaz de detectar caídas de manera efectiva. Esta herramienta puede tener un impacto significativo en la prevención y mitigación de riesgos en diferentes ámbitos, como la seguridad laboral, la atención a personas mayores o con discapacidades y la supervisión en entornos industriales.

El cumplimiento de los objetivos específicos permitirá desarrollar un modelo de identificación de caídas basado en aprendizaje automático, optimizar su desempeño y validar su efectividad en la detección de caídas. Además, la implementación de un prototipo en un

dispositivo móvil proporcionará un mecanismo accesible y fácil de utilizar para la comunidad en general.

1.2. Planteamiento Del Problema

La dependencia del celular se ha vuelto cada vez más común en la sociedad actual, y la mayoría de las personas llevan sus dispositivos móviles consigo todo el tiempo. Sin embargo, las caídas de los celulares son un problema frecuente, lo que puede causar daños en el dispositivo y costosos gastos de reparación. Además, la pérdida de datos importantes y la interrupción en las actividades diarias pueden ser un inconveniente significativo para los usuarios.

A pesar de que los celulares modernos cuentan con sensores, como el acelerómetro en los ejes X, Y y Z, o giroscopios, los mismos que permiten recolectar una gran cantidad de datos, en muchos casos estos sensores no están siendo aprovechados al máximo. Como resultado, actualmente no existen muchos sistemas eficientes para la detección de caídas de celulares.

De la discusión previa se puede identificar el problema principal:

No se cuenta con un prototipo funcional de una herramienta de inteligencia artificial para la detección de caídas mediante el uso de los sensores de un dispositivo móvil utilizando CRISP-DM

Y los siguientes problemas secundarios:

- No se ha identificado que algoritmos de Inteligencia artificial se puede usar para realizar predicción de caída de dispositivos celulares usando sus sensores.
- Se carece de una aplicación móvil que pueda enviar información sobre sus sensores.
- No se tiene un modelo inteligente dentro de una aplicación web para predicción de caída de dispositivos celulares usando sus sensores.

1.3. Objetivo General

Crear un prototipo funcional de una herramienta de inteligencia artificial para la detección de caídas mediante el uso de los sensores de un dispositivo móvil utilizando CRISP-DM

1.4. Objetivos Específicos

- Preparar el conjunto de datos para el entrenamiento del modelo de identificación de caídas.
- Modelar los datos con aprendizaje automático para la identificación de caídas.
- Identificar el modelo que mejor identifique las caídas.
- Implementar un prototipo para la detección de caídas mediante el uso de los sensores de un dispositivo móvil.

1.5. Antecedentes

“El uso de la inteligencia artificial en aplicaciones móviles es cada vez más común y ha demostrado ser una herramienta útil en diferentes áreas, desde la detección de fraudes en transacciones financieras” (Buczak & Gifford, 2016, p. 15) hasta la identificación de objetos en imágenes, sin embargo, la detección de caídas en dispositivos móviles sigue siendo un problema sin resolver y puede tener implicaciones significativas para los usuarios.

Existen algunos estudios previos que han investigado la detección de caídas utilizando los sensores de un dispositivo móvil (Bagalà et al., 2012; Anguita et al., 2013), pero no hay una solución completa y eficiente en el mercado. Además, muchos de los estudios existentes se centran en el análisis de la aceleración del dispositivo y no consideran otros datos importantes, como la gravedad en los ejes (Bourke et al., 2014). Por lo tanto, es necesario realizar una investigación más profunda para desarrollar un modelo inteligente que pueda detectar caídas con mayor precisión.

En cuanto a la implementación de una aplicación móvil para recopilar y enviar datos de los sensores, existen varias soluciones disponibles en el mercado, como las aplicaciones de monitoreo de salud y fitness (O'Reilly et al., 2017). Sin embargo, se requiere una aplicación personalizada que sea específica para la detección de caídas y que pueda enviar los datos de manera eficiente para el procesamiento.

Por último, la visualización de los resultados de la detección de caídas se puede lograr a través de una aplicación web, que permita a los usuarios ver si se ha detectado una caída y la hora y fecha en que ocurrió. Esto proporcionaría a los usuarios una retroalimentación valiosa sobre sus patrones de caída y podría ayudar en la prevención de futuras caídas (Cvetković et al., 2016).

1.6. Alcance

El proyecto tiene como objetivo la creación de un prototipo funcional de una herramienta de inteligencia artificial para la detección de caídas mediante el uso de los sensores de un dispositivo móvil utilizando CRISP-DM. Para lograr este objetivo, se llevarán a cabo las siguientes actividades:

1. Recopilación de datos: Se desarrollará una aplicación móvil para el sistema operativo Android que permita la recolección de datos de los sensores de acelerómetro y giroscopio de un dispositivo móvil.
2. Preparación de datos: Los datos recopilados serán procesados y preparados para el entrenamiento del modelo de identificación de caídas.
3. Modelado de datos: Se utilizarán técnicas de aprendizaje automático para el modelado de datos con el fin de identificar el modelo que mejor detecte las caídas.
4. Selección del algoritmo de inteligencia artificial: Se evaluarán distintos algoritmos de inteligencia artificial para seleccionar el más adecuado para la predicción de caídas de dispositivos móviles.

5. Implementación del modelo de detección de caídas: Se implementará un prototipo de la herramienta de inteligencia artificial para la detección de caídas mediante el uso de los sensores de un dispositivo móvil.
6. Integración con una aplicación web: Se desarrollará una aplicación web utilizando React para visualizar los resultados de la detección de caídas.

El alcance del proyecto incluye el desarrollo de la aplicación móvil, el procesamiento y modelado de los datos, la selección del algoritmo de inteligencia artificial, la implementación del modelo de detección de caídas y la integración con una aplicación web. El proyecto no incluye el desarrollo de hardware específico para la detección de caídas ni pruebas del sistema.

CAPÍTULO II: FUNDAMENTACIÓN TEÓRICA

2. Marco Teórico

2.1. Introducción A La Detección De Caídas

La detección de caídas es un tema de gran importancia en la actualidad debido al creciente uso de dispositivos móviles y la necesidad de protegerlos de posibles daños por impactos. Según Herrera, Giraldo y Alvarez (2019), la detección de caídas es una técnica que se utiliza para identificar cuando un objeto o dispositivo se cae, con el objetivo de activar una respuesta automática que permita minimizar los daños causados por el impacto.

En este sentido, los sensores son una herramienta clave en la detección de caídas en dispositivos móviles. Según la investigación realizada por Al-Jumaily (2016), los sensores de aceleración, giroscopio y magnetómetro son los más comunes utilizados en la detección de caídas en dispositivos móviles debido a su precisión y bajo costo.

Además, existen diferentes algoritmos de aprendizaje automático que se utilizan en la detección de caídas en dispositivos móviles. Según el estudio de Herrera, Giraldo y Alvarez (2019), algunos de los algoritmos más comunes son el k-NN, SVM y redes neuronales. Estos algoritmos utilizan los datos obtenidos por los sensores para entrenar modelos que permitan identificar patrones de caídas y, de esta manera, poder detectarlas con mayor precisión.

2.2. Sensores De Dispositivos Móviles

Según la documentación de Android Studio (2021), los sensores más comunes en dispositivos móviles son el acelerómetro, el giroscopio, el magnetómetro y el sensor de proximidad. El acelerómetro mide la aceleración lineal del dispositivo en los tres ejes del espacio, mientras que el giroscopio mide la velocidad angular del dispositivo en los mismos tres ejes. El magnetómetro, por otro lado, mide el campo magnético en la ubicación del dispositivo, lo que puede utilizarse para determinar la orientación del dispositivo. Finalmente, el sensor de proximidad detecta cuándo el dispositivo está cerca de un objeto.

Estos sensores son fundamentales para la detección de caídas en dispositivos móviles, ya que proporcionan la información necesaria para detectar cuándo un dispositivo se está cayendo. Además, como menciona Al-Jumaily (2016), los sensores de un celular son una opción económica y viable para detectar caídas en comparación con otros dispositivos dedicados.

Es importante destacar que, si bien los sensores de un celular pueden ser útiles para la detección de caídas, también pueden generar falsos positivos y negativos. Por lo tanto, es necesario combinar el uso de sensores con algoritmos de aprendizaje automático y técnicas de procesamiento de señales para mejorar la precisión de la detección de caídas en dispositivos móviles.

2.3. Aprendizaje Automático

El aprendizaje automático es una técnica que se ha utilizado en la detección de caídas en dispositivos móviles con el objetivo de mejorar la precisión de la detección y reducir el número de falsos positivos. Según el estudio de Sucerquia, López, y Vargas-Bonilla (2017), el aprendizaje automático permite entrenar modelos de detección de caídas a partir de datos recopilados por los sensores del dispositivo móvil.

Existen diferentes algoritmos de aprendizaje automático que se han utilizado en la detección de caídas en dispositivos móviles. Según el estudio de Herrera, Giraldo y Alvarez (2019), algunos de los algoritmos más comunes son el k-NN, SVM y redes neuronales. Estos algoritmos utilizan los datos obtenidos por los sensores para entrenar modelos que permitan identificar patrones de caídas y, de esta manera, poder detectarlas con mayor precisión.

Además, es importante tener en cuenta que la selección del algoritmo de aprendizaje automático adecuado depende de diferentes factores, como la precisión de los sensores, la cantidad y calidad de los datos recopilados y la complejidad del modelo de detección de caídas requerido. Según el estudio de Sucerquia, López, y Vargas-Bonilla (2017), es importante realizar una evaluación comparativa de diferentes algoritmos de aprendizaje automático para

determinar cuál es el más adecuado para la detección de caídas en un dispositivo móvil específico.

2.4. CRISP-DM

El Cross-Industry Standard Process for Data Mining (CRISP-DM) es un modelo de proceso estándar utilizado en minería de datos y ciencia de datos. Este modelo proporciona un marco completo para la planificación y ejecución de proyectos de minería de datos, que incluye seis fases principales: comprensión del negocio, comprensión de los datos, preparación de los datos, modelado, evaluación y despliegue (IBM, 2000).

En la fase de comprensión del negocio, se identifican los objetivos del proyecto y se establece un plan inicial. Esta fase también incluye la identificación de los requisitos del proyecto, la definición del alcance del proyecto y la creación de un plan de proyecto preliminar.

En la fase de comprensión de los datos, se realiza una exploración inicial de los datos disponibles para el proyecto. Esta fase también incluye la identificación de problemas de calidad de datos y la selección de las técnicas de minería de datos más apropiadas para el proyecto.

La fase de preparación de los datos implica la selección, limpieza y transformación de los datos para su uso en el modelo. Esta fase también incluye la creación de conjuntos de datos de entrenamiento y prueba.

La fase de modelado implica la construcción de un modelo utilizando las técnicas de minería de datos seleccionadas. Esta fase también incluye la validación y la evaluación del modelo.

En la fase de evaluación, se evalúa la efectividad del modelo y se determina si cumple con los objetivos del proyecto. Si el modelo no cumple con los objetivos del proyecto, se repite el proceso de modelado.

Finalmente, en la fase de despliegue, se implementa el modelo en la organización y se proporcionan las herramientas necesarias para su uso continuo.

El modelo CRISP-DM proporciona una metodología estructurada para abordar los desafíos que surgen en proyectos de minería de datos. Al seguir este modelo, los profesionales de datos pueden mejorar la eficacia y eficiencia de su proceso de minería de datos.

2.5. AWK

"AWK es un lenguaje de programación utilizado en Unix para el procesamiento y análisis de datos de texto. Este lenguaje fue desarrollado en los años 70 por Alfred Aho, Peter Weinberger y Brian Kernighan" (*The GNU Awk User's Guide*, 2022). AWK se considera un lenguaje de alto nivel y se utiliza comúnmente para buscar patrones en archivos de texto y procesar datos en función de esos patrones.

Una de las características más útiles de AWK es su capacidad para dividir líneas de texto en campos separados por delimitadores, lo que facilita el procesamiento de datos estructurados. Además, AWK tiene una amplia variedad de funciones y comandos incorporados que se utilizan para realizar operaciones específicas en los datos de entrada.

AWK es una herramienta poderosa y valiosa para programadores y analistas de datos que trabajan con archivos de texto en Unix. Con la capacidad de buscar patrones en archivos de texto y manipular datos estructurados, AWK es una herramienta clave para el análisis de datos. Además, al ser parte de la familia de herramientas Unix, AWK se integra bien con otras herramientas comunes de Unix, como grep y sed, lo que lo convierte en una herramienta versátil para el procesamiento y análisis de datos de texto.

2.6. KNN

"K-Nearest Neighbors (KNN) es un algoritmo de aprendizaje supervisado que se utiliza en problemas de clasificación y regresión en el campo del aprendizaje automático y la inteligencia artificial" (Aha, Kibler, & Albert, 1991, p. 39). El algoritmo KNN es no paramétrico y

basado en instancias, lo que significa que no asume ninguna distribución subyacente en los datos y utiliza los datos de entrenamiento en sí mismos para realizar predicciones.

Funcionamiento del algoritmo KNN:

El algoritmo KNN funciona siguiendo estos pasos:

- Determinar el valor de "k": Seleccionar el número de vecinos más cercanos que se considerarán para realizar predicciones. El valor de "k" es un hiperparámetro que se puede ajustar para optimizar el rendimiento del algoritmo.
- Calcular distancias: Para cada observación en el conjunto de datos de prueba, calcular la distancia entre la observación y cada punto en el conjunto de datos de entrenamiento. Las distancias se pueden calcular utilizando diferentes métricas, como la distancia euclidiana, la distancia de Manhattan o la distancia de Minkowski.
- Identificar los vecinos más cercanos: Identificar los "k" puntos del conjunto de datos de entrenamiento que están más cerca de la observación de prueba según las distancias calculadas.
- Realizar la predicción: En problemas de clasificación, la clase más común entre los "k" vecinos más cercanos se asigna como la clase predicha para la observación de prueba. En problemas de regresión, el valor predicho para la observación de prueba se calcula como el promedio de los valores objetivo de los "k" vecinos más cercanos.

2.7. DecisionTreeClassifier

“El DecisionTreeClassifier es un algoritmo de aprendizaje automático supervisado de la biblioteca Scikit-learn” (Scikit-learn developers, 2021), utilizado en este proyecto para clasificar datos de sensores en dispositivos móviles y detectar caídas. A continuación, se discuten brevemente las características y ventajas de usar DecisionTreeClassifier en el proyecto.

“DecisionTreeClassifier es un algoritmo basado en árboles de decisión que crea una estructura jerárquica para clasificar los datos según ciertas condiciones. Este algoritmo es fácil

de interpretar y visualizar, lo que permite comprender el proceso de clasificación y su lógica subyacente" (Scikit-learn developers, 2021). Además, "el DecisionTreeClassifier es capaz de manejar tanto variables numéricas como categóricas, lo que lo hace versátil y adecuado para diversos tipos de datos" (Breiman et al., 1984, p. 19).

2.8. RandomForestClassifier

Random Forest Classifier es un algoritmo de aprendizaje supervisado que pertenece a la familia de los árboles de decisión en el campo del aprendizaje automático y la inteligencia artificial. Este algoritmo se encuentra en la biblioteca de aprendizaje automático scikit-learn (sklearn) en Python (Pedregosa et al., 2011, p. 19).

"El algoritmo de RandomForest combina múltiples árboles de decisión y utiliza técnicas de agregación para mejorar el rendimiento y la precisión de las predicciones" (Breiman, 2001).

Funcionamiento del algoritmo RandomForestClassifier:

- Creación de árboles de decisión: El algoritmo RandomForest genera múltiples árboles de decisión durante el proceso de entrenamiento. Para cada árbol, se selecciona una muestra de los datos de entrenamiento utilizando el método de remuestreo de bootstrap. Es decir, se seleccionan observaciones al azar con reemplazo para formar una nueva muestra del mismo tamaño que el conjunto de datos original.
- Selección de características: Para cada árbol de decisión, se selecciona un subconjunto de características al azar en cada división del árbol. Esta aleatoriedad ayuda a reducir la correlación entre los árboles y a mejorar la diversidad en el modelo.
- Entrenamiento de árboles de decisión: Cada árbol de decisión se entrena individualmente utilizando el subconjunto de datos y características seleccionadas en los pasos anteriores.
- Predicción y agregación: Cuando se realiza una predicción, cada árbol de decisión en el bosque proporciona una predicción independiente. Para problemas de clasificación, la

clase con más votos de los árboles se selecciona como la clase predicha. En problemas de regresión, el promedio de las predicciones de todos los árboles se calcula como el valor predicho final.

2.9. Redes Neuronales

Redes Neuronales Artificiales (RNA) son modelos computacionales inspirados en el funcionamiento del cerebro humano y sus neuronas, que se utilizan ampliamente en el campo del aprendizaje automático y la inteligencia artificial para resolver problemas de clasificación, regresión, generación de texto, procesamiento de imágenes, entre otros (Goodfellow, Bengio, & Courville, 2016).

"TensorFlow es una biblioteca de código abierto desarrollada por Google que permite implementar, entrenar y desplegar redes neuronales y otros modelos de aprendizaje profundo" (Abadi et al., 2016).

Funcionamiento de las Redes Neuronales Artificiales:

- Estructura y componentes: Una red neuronal artificial consta de capas de nodos llamados neuronas, que están interconectadas para formar una estructura en capas: capa de entrada, capas ocultas y capa de salida. Cada neurona en una capa se conecta con todas las neuronas de la capa siguiente a través de conexiones ponderadas.
- Función de activación: Las neuronas aplican una función de activación a la suma ponderada de sus entradas y un sesgo (bias) para generar su salida. Las funciones de activación comunes incluyen ReLU (Rectified Linear Unit), sigmoide y tangente hiperbólica (tanh).
- Propagación hacia adelante: Durante la propagación hacia adelante, la información se procesa desde la capa de entrada hasta la capa de salida, pasando por las capas

ocultas, aplicando las funciones de activación y las ponderaciones de las conexiones entre neuronas.

- Función de pérdida y retropropagación: Para entrenar una red neuronal, se utiliza una función de pérdida que mide la discrepancia entre las predicciones de la red y los valores reales (objetivo). El algoritmo de retropropagación calcula el gradiente de la función de pérdida con respecto a cada peso, propagando el error desde la capa de salida hacia la capa de entrada.
- Optimización y actualización de pesos: Se utilizan algoritmos de optimización, como el descenso de gradiente estocástico (SGD) o Adam, para ajustar los pesos de la red neuronal y minimizar la función de pérdida durante el proceso de entrenamiento.

2.10. Matriz De Confusión

La matriz de confusión es una herramienta fundamental en el análisis y evaluación de modelos de clasificación, es una representación tabular que permite comparar las predicciones realizadas por un modelo de aprendizaje automático con los valores verdaderos (observaciones reales) de la clasificación (Fawcett, 2006, p. 87).

"Esta matriz se emplea para determinar la efectividad del modelo y detectar posibles problemas en su desempeño, como sesgos en la clasificación de determinadas categorías" (Powers, 2011, p. 33).

La matriz de confusión se organiza en filas y columnas, donde cada fila representa las categorías verdaderas (reales) y cada columna las categorías predichas por el modelo. En el caso de la detección de caídas, se trataría de una matriz 2x2, ya que hay dos categorías posibles: caída y no caída. "La diagonal principal de la matriz contiene las predicciones correctas (verdaderos positivos y verdaderos negativos), mientras que los elementos fuera de la diagonal representan las predicciones incorrectas (falsos positivos y falsos negativos)" (Fawcett, 2006, p. 87).

“A partir de la matriz de confusión, se pueden calcular varias métricas que proporcionan información sobre el desempeño del modelo, como la precisión, el recall, la especificidad y la F1-score” (Powers, 2011, p. 33). “Estas métricas permiten evaluar diferentes aspectos del rendimiento del modelo, como su capacidad para identificar correctamente las caídas (sensibilidad) y su habilidad para evitar falsas alarmas (especificidad)” (Chicco & Jurman, 2020, p. 52).

En el contexto del proyecto de titulación, la matriz de confusión se utilizará para comparar y seleccionar el modelo que mejor identifique las caídas, considerando tanto su sensibilidad como su especificidad.

Además, el análisis de la matriz de confusión permitirá identificar posibles mejoras en el modelo, como ajustes en los hiperparámetros o en las características extraídas de los datos, para optimizar su rendimiento en la detección de caídas mediante el uso de los sensores de un dispositivo móvil (Kohavi & Provost, 1998, p. 105).

2.11. Scikit-learn

“Scikit-learn es una biblioteca de aprendizaje automático de código abierto en Python que ofrece una amplia gama de algoritmos y herramientas para el análisis y modelado de datos” (Scikit-learn, 2021). Esta biblioteca es especialmente útil, ya que permite experimentar con diversos algoritmos de aprendizaje automático, como árboles de decisión, k-vecinos más cercanos y máquinas de vectores de soporte, en el contexto de la detección de caídas en dispositivos móviles.

Scikit-learn también proporciona herramientas para la selección de características, la validación cruzada y la optimización de hiperparámetros, lo que facilita la construcción de modelos de aprendizaje automático sólidos y eficientes. Además, su integración con otras bibliotecas populares de Python, como NumPy y pandas, hace que Scikit-learn sea una opción conveniente para el desarrollo y evaluación de modelos de detección de caídas en el proyecto de titulación.

2.12. Tensorflow

“TensorFlow es una biblioteca de código abierto desarrollada por Google que se utiliza ampliamente para el aprendizaje automático y la inteligencia artificial, especialmente en el área de las redes neuronales” (TensorFlow, 2021). La plataforma ofrece una solución flexible y eficiente para desarrollar y desplegar modelos de aprendizaje automático aplicados a la detección de caídas en dispositivos móviles.

La flexibilidad de TensorFlow permite la integración de datos de diferentes sensores y la experimentación con diversas arquitecturas de red y técnicas de optimización. Además, TensorFlow es compatible con diferentes plataformas, como Android e iOS, lo que facilita su implementación en aplicaciones móviles y permite la detección de caídas en tiempo real en dispositivos móviles.

2.13. Firebase

“Firebase es una plataforma de desarrollo de aplicaciones móviles y web desarrollada por Google que proporciona diversas herramientas y servicios para facilitar el desarrollo, la implementación y el mantenimiento de aplicaciones” (Firebase, 2021). Firebase ofrece varias funcionalidades que pueden ser útiles para la detección de caídas en dispositivos móviles y la comunicación de alertas.

Entre los servicios de Firebase, se encuentra la base de datos en tiempo real, la cual permite almacenar y sincronizar datos en tiempo real entre dispositivos y usuarios. Esta característica puede ser útil para almacenar información relacionada con las caídas detectadas y enviar notificaciones a cuidadores o profesionales de la salud de manera rápida y eficiente.

Otro servicio proporcionado por Firebase es Cloud Functions, que permite ejecutar funciones en la nube en respuesta a eventos específicos. En el proyecto de titulación, Cloud Functions puede ser utilizado para procesar alertas de caídas y tomar acciones apropiadas, como enviar notificaciones a dispositivos móviles o actualizar registros en la base de datos.

2.14. React

“React es una biblioteca de JavaScript de código abierto desarrollada por Facebook, que se utiliza para construir interfaces de usuario (UI) para aplicaciones web y móviles” (Facebook, 2021).

Una de las principales ventajas de utilizar React en el proyecto es su enfoque basado en componentes, que permite la creación de elementos reutilizables para diferentes partes de la aplicación. Esto facilita la modularidad, mejora la legibilidad del código y reduce la duplicación de esfuerzos en el desarrollo.

Además, React se caracteriza por su capacidad para manejar eficientemente el proceso de renderizado, gracias a un algoritmo llamado "Reconciliation". Este enfoque asegura que solo se actualicen aquellos componentes que han experimentado cambios en sus estados o propiedades, lo que contribuye a un mejor rendimiento de la aplicación.

React también puede ser fácilmente integrado con otras bibliotecas y frameworks, como Redux para la gestión de estados o React Native para el desarrollo de aplicaciones móviles multiplataforma. Esta compatibilidad con otras tecnologías resulta valiosa en el contexto del proyecto de titulación.

Por último, es importante destacar el amplio ecosistema y la comunidad activa que rodean a React. La biblioteca cuenta con numerosos paquetes y herramientas de terceros, así como una comunidad de desarrolladores que contribuyen a su mejora y evolución constante.

2.15. Trabajos Relacionados

En el campo de la detección de caídas en dispositivos móviles, diversos estudios han explorado distintos enfoques para abordar este problema. Por ejemplo, López et al. (2015) propusieron un sistema de detección de caídas en tiempo real basado en dispositivos móviles, empleando sensores de aceleración y rotación. Este sistema combina técnicas de procesamiento de señales y aprendizaje automático para identificar caídas y generar alertas oportunas.

En el trabajo de Torres et al. (2016), se presenta un sistema de detección de caídas basado en dispositivos móviles y una red de sensores inalámbricos distribuidos en el entorno del usuario. Este enfoque combina información de diversos sensores para identificar caídas y enviar alertas a cuidadores o profesionales de la salud.

Por último, García et al. (2017) propusieron un método basado en la fusión de datos de sensores inerciales y de presión en dispositivos móviles para mejorar la detección de caídas. En este estudio, se emplearon técnicas de aprendizaje automático, como el algoritmo Random Forest, para entrenar modelos capaces de distinguir entre caídas y actividades diarias comunes.

Estos trabajos relacionados brindan un sólido punto de partida para el desarrollo de un proyecto de detección de caídas en dispositivos móviles, ya que permiten comprender las soluciones y enfoques previos empleados en la resolución de este problema. Además, estos estudios pueden servir como referencia al seleccionar los sensores, algoritmos de aprendizaje automático y técnicas de detección de caídas que se utilizarán en el proyecto.

CAPÍTULO III: METODOLOGÍA

3. Metodología De Desarrollo Del Plan De Tesis

3.1. Investigación Cuantitativa

"La investigación cuantitativa es un enfoque metodológico que utiliza datos numéricos y técnicas estadísticas para analizar y medir fenómenos" (Hernández, Fernández, & Baptista, 2014, p. 41). Se eligió la investigación cuantitativa debido a su capacidad para proporcionar una base sólida y objetiva para evaluar el rendimiento y la precisión del sistema de detección de caídas utilizando algoritmos de aprendizaje automático y datos recolectados por la aplicación móvil

"Uno de los principales aspectos de la investigación cuantitativa en este proyecto fue la recolección y análisis de datos provenientes de los sensores de movimiento" (Bolaños, 2016, p. 7). Estos datos, recolectados de manera sistemática y controlada, permitieron obtener un conjunto de datos representativo de las condiciones reales en las que se espera que funcione el sistema. Además, al utilizar técnicas cuantitativas, fue posible identificar patrones y tendencias en los datos que podrían no ser evidentes en un enfoque cualitativo.

La investigación cuantitativa también permitió evaluar de manera objetiva el rendimiento de los algoritmos de aprendizaje automático empleados en el proyecto. Al utilizar métricas de rendimiento, como la precisión, la sensibilidad y la especificidad, fue posible comparar y seleccionar el modelo que mejor se adaptaba a las necesidades del sistema de detección de caídas. Esta evaluación objetiva fue crucial para garantizar la eficacia del sistema y minimizar la posibilidad de falsos positivos y negativos.

Otra ventaja de utilizar la investigación cuantitativa en este proyecto fue su capacidad para facilitar la identificación de áreas de mejora y futuras líneas de investigación. Al analizar los resultados de las pruebas y evaluaciones, se pudo identificar en qué aspectos el sistema podría mejorarse y qué técnicas o enfoques podrían ser más efectivos en futuras investigaciones.

3.2. CRISP-DM

La metodología CRISP-DM (Cross-Industry Standard Process for Data Mining) se elige para este proyecto de titulación debido a su enfoque estructurado y ampliamente aceptado en el desarrollo de proyectos de minería de datos y aprendizaje automático. CRISP-DM proporciona un marco bien definido que guía el proceso de comprensión, preparación y modelado de los datos, así como la evaluación y despliegue de los modelos desarrollados. Al adoptar CRISP-DM, se asegura que el proyecto siga las mejores prácticas y estándares de la industria en el análisis y modelado de datos, lo que aumenta la probabilidad de éxito y la calidad del prototipo desarrollado.

Además, CRISP-DM es una metodología flexible y adaptable, que permite abordar problemas específicos y desafíos en el área de detección de caídas, y se integra fácilmente con otras metodologías de investigación, como la investigación científica. La adopción de CRISP-DM en el proyecto de titulación también facilita la comunicación y colaboración entre los investigadores y los interesados, al proporcionar un lenguaje y un proceso comunes para la ejecución y evaluación del proyecto.

CAPÍTULO IV: DESARROLLO DE LA INVESTIGACIÓN

4. Desarrollo

4.1. Extracción Y Preparación De La Data

4.1.1. Extracción De La Data.

La extracción y preparación de la data es un proceso crítico para cualquier proyecto de aprendizaje automático. En el caso de la detección de caídas mediante el uso de los sensores de un dispositivo móvil, es necesario contar con un conjunto de datos que permita entrenar el modelo de identificación de caídas.

En primer lugar, se llevó a cabo la medición de los tiempos estimados de caída mediante el uso de un cronómetro. Se tomó como referencia la altura desde la cual se produce la caída, la cual se varió de manera controlada para obtener diferentes tiempos de caída.

Luego, se procedió a la recolección de los datos a través de una aplicación móvil desarrollada para tal fin. La aplicación registra los valores de aceleración en los tres ejes (x, y, z) del acelerómetro del dispositivo móvil. Cada muestra consta de los registros de los sensores en los ejes x y z durante 5 segundos y se toman 2 muestras por segundo. Se registraron diferentes tipos de caídas, como caídas laterales, frontales, traseras y caídas en posición de sentado.

Figura 1

Información acelerómetro

Nota. Fuente: Interfaz de la aplicación Sensores Multiherramienta

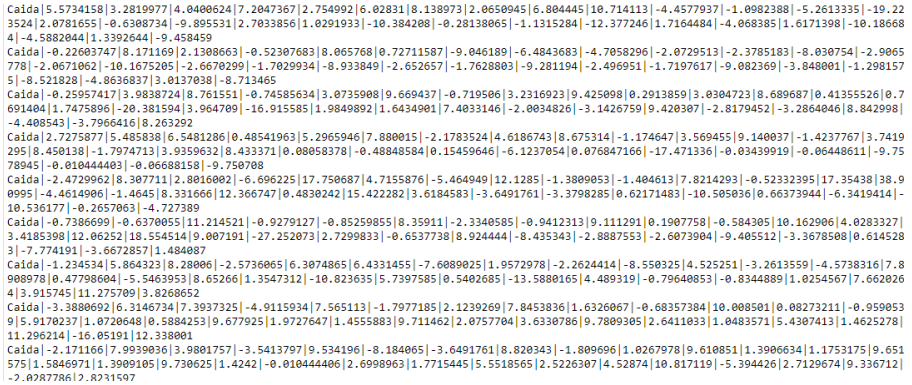
Cabe destacar que se decidió utilizar el acelerómetro del dispositivo móvil en lugar de otros sensores, como el giroscopio o el magnetómetro, debido a que el acelerómetro es el sensor más sensible a los cambios de movimiento bruscos, como los producidos por una caída. De igual modo, este es capaz de detectar la orientación del dispositivo según el eje en el que se aplique la gravedad. Además, el uso del acelerómetro permite reducir el consumo de energía del dispositivo móvil y, por tanto, aumentar la duración de la batería.

4.1.2. Limpieza De La Data.

Para la limpieza de datos, se emplearon comandos awk con el objetivo de eliminar registros incompletos y verificar la estructura de los datos. Los datos recolectados incluían información de los sensores en tres ejes, y se tomaron seis muestras en total (dos muestras por segundo durante cinco segundos). Cada registro debía contener 30 datos en total, y se utilizó el comando awk para verificar que todos los registros cumplieran con este requisito.

Figura 2

Datos en crudo recolectados de los sensores de movimiento



```
Caida|5.5734158|3.2819977|4.0400624|7.2047367|2.754992|6.02831|8.138973|2.0650945|6.804445|10.714113|-4.4577937|-1.0982388|-5.2613335|-19.22
3524|2.0781655|-0.6308734|-9.895531|2.7033856|1.0291933|-10.384208|-0.28138065|-1.1315284|-12.377246|1.7164484|-4.068385|1.6171398|-10.18668
4|-4.5882044|1.3392644|-9.458459
Caida|-0.22603747|8.171169|2.1308663|-0.52307683|8.065768|0.72711587|-9.046189|-6.4843683|-4.7058296|-2.0729513|-2.3785183|-8.030754|-2.9065
778|-2.0671062|-10.1675205|-2.6670299|-1.7029934|-8.933849|-2.652657|-1.7628803|-9.281194|-2.496951|-1.7197617|-9.082369|-3.848001|-1.298157
5|-8.521828|-4.8636837|3.0137038|-8.713465
Caida|-0.25957417|3.9838724|8.761551|-0.74585634|3.0735908|9.669437|-0.719506|3.2316923|9.425098|0.2913859|3.0304723|8.689687|0.41355526|0.7
691404|1.7475896|-20.381594|3.964709|-16.915585|1.9849892|1.6434901|7.4033146|-2.0034826|-3.1426759|9.420307|-2.8179452|-3.2864046|8.842998|
-4.408543|-3.7966416|8.263292
Caida|2.7275877|5.485838|6.5481286|0.48541963|5.2965946|7.880015|-2.1783524|4.6186743|8.675314|-1.174647|3.569455|9.140037|-1.4237767|3.7419
295|8.450138|-1.7974713|3.9359632|8.433371|0.08058378|-0.48848584|0.15459646|-6.1237054|0.076847166|-17.471336|-0.03439919|-0.06448611|-9.75
78945|-0.010444403|-0.06688158|-9.750708
Caida|-2.4729962|8.307711|2.8016002|-6.696225|17.750687|4.7155876|-5.464949|12.1285|-1.3809053|-1.404613|7.8214293|-0.52332395|17.35438|38.9
0995|-4.4614906|-1.4645|8.331666|12.366747|0.4830242|15.422282|3.6184583|-3.6491761|-3.3798285|0.62171483|-10.505036|0.66373944|-6.3419414|-
10.536177|-0.2657063|-4.727389
Caida|-0.7386699|-0.6370055|11.214521|-0.9279127|-0.85259855|8.35911|-2.3340585|-0.9412313|9.111291|0.1907758|-0.584305|10.162906|4.0283327|
3.4185398|12.06252|18.554514|9.007191|-27.252073|2.7299833|-0.6537738|8.924444|-8.435343|-2.8887553|-2.6073904|-9.405512|-3.3678508|0.614528
3|-7.774191|-3.6672857|1.484087
Caida|-1.234534|5.864323|8.28006|-2.5736065|6.3074865|6.4331455|-7.6089025|1.9572978|-2.2624414|-8.550325|4.525251|-3.2613559|-4.5738316|7.8
908978|0.47798604|-5.5463953|8.65266|1.3547312|-10.823635|5.7397585|0.5402685|-13.5880165|4.489319|-0.79640853|-0.8344889|1.0254567|7.662026
4|3.915745|11.275709|3.8268652
Caida|-3.3880692|6.3146734|7.3937325|-4.9115934|7.565113|-1.7977185|2.1239269|7.8453836|1.6326067|-0.68357384|10.008501|0.08273211|-0.959053
9|5.9170237|1.0720648|0.5884253|9.677925|1.9727647|1.4555883|9.711462|2.0757704|3.6330786|9.7809305|2.6411033|1.0483571|5.4307413|1.4625278|
11.296214|-16.05191|12.338001
Caida|-2.171166|7.9939036|3.9801757|-3.5413797|9.534196|-8.184065|-3.6491761|8.820343|-1.809696|1.0267978|9.610851|1.3906634|1.1753175|9.651
575|1.5846971|1.3909105|9.730625|1.4242|-0.010444406|2.6998963|1.7715445|5.5518565|2.5226307|4.52874|10.817119|-5.394426|2.7129674|9.336712|
-2.028786|2.8231597
```

Además de eliminar registros incompletos, también se etiquetaron los valores recolectados para facilitar su posterior análisis y procesamiento. Esto es especialmente importante cuando se trabajan con múltiples tipos de datos, ya que la estructura y las propiedades de los datos pueden variar según el tipo de sensor y la información recolectada.

El uso de comandos awk para la limpieza de datos es una herramienta efectiva en este proyecto, ya que permite eliminar registros incompletos y verificar la estructura de los datos de manera eficiente. Además, el etiquetado de los valores recolectados facilita el análisis de los diferentes tipos de datos y asegura que se utilice la información adecuada en el proceso de aprendizaje automático.

Figura 3

Código para limpieza de datos

```
head DatosUnidos.txt|awk -F "|" '{print NF}'

head -n 1 DatosUnidos.txt|tail -1 | grep -o "|" | wc -l

cat DatosUnidos.txt|awk -F "|" '{if(NF != 31) {print "NO"} else {print NF}}'
cat DatosUnidos.txt|awk -F "|" '{if(NF != 31) {sed -i}}'

// limpia los datos que no tiene la cantidad de datos exactos
cat DatosUnidos.txt|awk -F "|" '{if(NF != 31) {sed -i} else {print }}' > transformados.txt
```

4.1.3. Tratamiento De Registros Incompletos.

En este caso dado que no es posible reconstruir datos que los sensores no hayan capturado, la eliminación de registros incompletos es una estrategia adecuada en este proyecto por las siguientes razones:

- Calidad de los datos: Eliminar registros incompletos garantiza que solo se utilicen datos completos y de alta calidad para entrenar el modelo, evitando la introducción de errores o supuestos incorrectos que podrían generarse al emplear técnicas de imputación de datos.
- Simplificación del proceso: La eliminación de registros incompletos simplifica el proceso de preparación de datos y reduce la complejidad del proyecto. La imputación de datos puede requerir la selección y aplicación de métodos específicos, lo que podría aumentar la carga de trabajo y la complejidad del proyecto.
- Tamaño del conjunto de datos: Si el conjunto de datos es lo suficientemente grande, eliminar registros incompletos podría no tener un impacto significativo en el tamaño del conjunto de datos final. En este caso, la pérdida de información es mínima y no afecta significativamente el rendimiento del modelo.

4.1.4. Preparación De Los Datos.

Se importaron los datos recolectados de los sensores de los dispositivos móviles y se separaron en dos conjuntos, uno para entrenamiento y otro para prueba. El conjunto de datos de entrenamiento se utilizó para entrenar los modelos, mientras que el conjunto de datos de prueba se utilizó para evaluar su rendimiento en términos de precisión y recall.

Figura 4

Código división de los datos

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import recall_score

x = dataframe1.drop(['categoria'], axis=1)

y = dataframe1["categoria"]

X_train, X_test, y_train, y_test = train_test_split(x, y)
```

4.2. Modelado De Los Datos

El aprendizaje supervisado de clasificación se ha elegido para este proyecto debido a su capacidad para identificar patrones y asignar etiquetas a nuevas observaciones, basándose en un conjunto de datos previamente etiquetado. Este enfoque es particularmente adecuado para la detección de caídas, ya que se busca clasificar los eventos como caídas o no caídas a partir de los datos recopilados por los sensores del smartphone.

Los cuatro modelos seleccionados para la evaluación, k-Nearest Neighbors (KNN), DecisionTreeClassifier, RandomForestClassifier y una Red Neuronal, se eligieron debido a sus diferencias en cuanto a complejidad, interpretabilidad y capacidad de generalización. Al comparar estos modelos, se busca identificar el algoritmo que mejor se adapte al problema en cuestión y que proporcione la mayor precisión en la detección de caídas.

Utilizando las librerías de Python scikit-learn y TensorFlow en el entorno de programación Jupyter, se pudo llevar a cabo la evaluación de los cuatro modelos de aprendizaje automático. Estas herramientas proporcionaron un entorno eficiente y flexible para la implementación y evaluación de los modelos, facilitando la comparación de su rendimiento y la selección del modelo más adecuado para la detección de caídas en smartphones.

4.2.1. Modelo KNN.

Se entrenó un modelo KNN utilizando la librería scikit-learn y se evaluó su rendimiento en el conjunto de datos de prueba.

Figura 5

Código modelo KNN

Knn modelo

```
In [344]: from sklearn.neighbors import KNeighborsClassifier #Importamos al clasificador

In [345]: knn = KNeighborsClassifier()

In [346]: knn.fit(X_train.values, y_train)
Out[346]: KNeighborsClassifier()

In [347]: knn.score(X_test.values, y_test)
Out[347]: 0.8928571428571429

In [350]: y_pred = knn.predict(X_test.values)
          cm =confusion_matrix(y_test, y_pred)
          rc = recall_score(y_test, y_pred, average=None)[1]
          print(cm)
          print("Recall para la clase 1: {:.2f}".format(rc))

          [[ 9  8]
           [ 1 66]]
          Recall para la clase 1: 0.99
```

4.2.2. DesicionTreeClasifier.

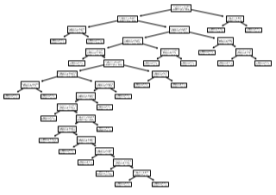
Se entrenó un modelo DesicionTreeClasifier utilizando la librería scikit-learn y se evaluó su rendimiento en el conjunto de datos de prueba. Para examinar con mayor detalle el gráfico, puede hacer referencia al Anexo 1.

Figura 6

Código Decision Tree Classifier

DecisionTreeClassifier

```
In [114]: from sklearn import tree

In [115]: clf = tree.DecisionTreeClassifier()

In [116]: clf = clf.fit(X_train.values, y_train)

In [117]: clf.score(X_test.values, y_test)
Out[117]: 0.8452380952380952

In [364]: y_pred = clf.predict(X_test.values)
          cm =confusion_matrix(y_test, y_pred)
          rc = recall_score(y_test, y_pred, average=None)[1]
          print(cm)
          print("Recall para la clase 1: {:.2f}".format(rc))

          [[11  6]
           [ 2 65]]
          Recall para la clase 1: 0.97

In [118]: tree.plot_tree(clf)
          Text(0.8064516129032258, 0.6764705882352942, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
          Text(0.9354838709677419, 0.6764705882352942, 'gini = 0.0\nsamples = 2\nvalue = [0, 2]'),
          Text(0.8387096774193549, 0.9117647058823529, 'X[0] <= 0.852\ngini = 0.08\nsamples = 24\nvalue = [23, 1]'),
          Text(0.7741935483870968, 0.8529411764705882, 'gini = 0.0\nsamples = 23\nvalue = [23, 0]'),
          Text(0.9032258064516129, 0.8529411764705882, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]')]
```



4.2.3. Modelo RandomForestClassifier.

Se entrenó un modelo RandomForestClassifier utilizando la librería scikit-learn y se evaluó su rendimiento en el conjunto de datos de prueba.

Figura 7

Código Random Forests Classifier

Random ForestsClassifier

```
In [351]: from sklearn.ensemble import RandomForestClassifier

In [352]: clf = RandomForestClassifier(max_depth=2, random_state=0)

In [353]: clf = clf.fit(X_train.values, y_train)

In [354]: clf.score(X_test.values, y_test)
Out[354]: 0.9047619047619048

In [355]: y_pred = clf.predict(X_test.values)
          cm =confusion_matrix(y_test, y_pred)
          rc = recall_score(y_test, y_pred, average=None)[1]
          print(cm)
          print("Recall para la clase 1: {:.2f}".format(rc))

          [[11  6]
           [ 2 65]]
          Recall para la clase 1: 0.97
```

4.2.4. Modelo de Red Neuronal.

Se entrenó una Red Neuronal utilizando la librería TensorFlow y se evaluó su rendimiento en el conjunto de datos de prueba.

Figura 8

Código Red neuronal

Red Neuronal

```
In [356]: import tensorflow as tf
          from tensorflow import keras

          from keras.models import Sequential
          from keras.layers import Dense, Dropout

          import numpy as np

In [363]: aux = []
          for element in y_train:
              if element == "Caida":
                  aux.append(0)
              else:
                  aux.append(1)
          y_train1 = np.array(aux)
          aux = []
          for element in y_test:
              if element == "Caida":
                  aux.append(0)
              else:
                  aux.append(1)
          y_test1 = np.array(aux)

In [358]: modelo = tf.keras.Sequential([
              keras.layers.Dense(260, activation='relu', dtype="float32"),
              #    keras.layers.Dense(180, activation='relu', dtype="float32"),
              keras.layers.Dense(2, activation='softmax')
          ])
```

```
In [359]: modelo.compile(
              optimizer=tf.keras.optimizers.Adam(0.0001),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy']
          )
```

```
In [360]: print("Comenzando entrenamiento....")
          modelo.fit(X_train, y_train1, epochs=100)
          print("Modelo entrenado!")
```

```
In [361]: test_loss, test_acc = modelo.evaluate(X_test,  y_test1, verbose=2)
          print('\nTest accuracy:', test_acc)

3/3 - 0s - loss: 0.2339 - accuracy: 0.9286 - 56ms/epoch - 19ms/step

Test accuracy: 0.9285714030265808
```

```
In [362]: predictions = modelo.predict(X_test)
          predictions = np.argmax(predictions, axis=1)
          conf_matrix = tf.math.confusion_matrix(labels=y_test1, predictions=predictions, num_classes=2)
          rc = recall_score(y_test1, predictions, average=None)[1]
          print(rc)
          print(conf_matrix)

3/3 [==============================] - 0s 2ms/step
0.9552238805970149
tf.Tensor(
[[14  3]
 [ 3 64]], shape=(2, 2), dtype=int32)
```

4.3. Evaluación Y Selección Del Modelo

4.3.1. Modelo KNN.

El resultado obtenido en este modelo fue del 89.29% de presición. El recall para la clase 1 (caída) fue del 97%.

Figura 9

Matriz de confusión modelo KNN

```
In [54]: plot_confusion_matrix(cm)
```

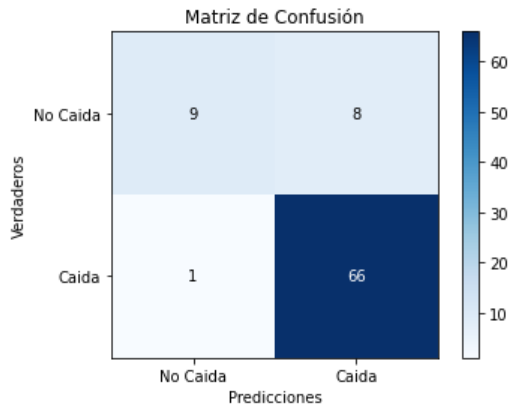


Tabla 1

Ventajas y Desventajas KNN

Ventajas	Desventajas
- KNN es un algoritmo simple y fácil de entender, lo que facilita su implementación y ajuste. - Funciona bien con conjuntos de datos pequeños y no requiere una gran cantidad de tiempo de entrenamiento. - Puede manejar datos no lineales, lo que puede ser útil en el caso de la detección de caídas, donde los patrones de movimiento no son siempre lineales.	- KNN puede ser sensible a la elección del valor de K (número de vecinos) y requiere un proceso de ajuste para encontrar el valor óptimo. - El rendimiento del algoritmo puede verse afectado por la maldición de la dimensionalidad si se utilizan muchas características para el entrenamiento - La clasificación en tiempo real puede ser lenta, ya que KNN necesita calcular la distancia a todos los puntos de entrenamiento para cada consulta, lo que puede ser un problema en aplicaciones de detección de caídas en tiempo real en dispositivos móviles.

Nota. Fuente: Adaptado de *Scikit-learn developers (2021)*

4.3.2. Modelo DesicionTreeClasifier.

El modelo alcanzó una precisión del 84.52%. El recall para la clase 1 (caída) fue del 97%.

Figura 10

Matriz de confusión modelo Desicion Tree

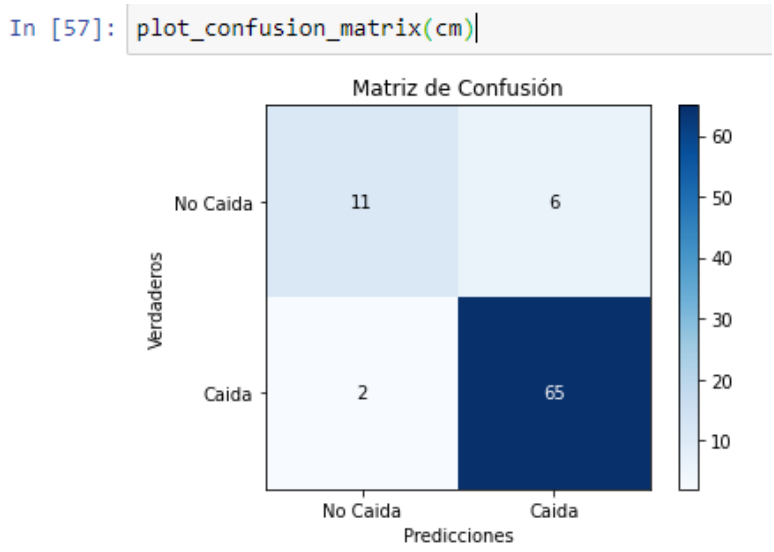


Tabla 2

Ventajas y Desventajas Decisión Tree Clasifier

Ventajas	Desventajas
- Los árboles de decisión son fáciles de entender e interpretar, lo que puede ser beneficioso para explicar el modelo a los usuarios o clientes. - Pueden manejar conjuntos de datos grandes y complejos y pueden identificar interacciones no lineales entre características. - Permite una selección automática de características, lo que puede ser útil para la detección de caídas al eliminar características irrelevantes o redundantes.	- Los árboles de decisión pueden ser sensibles a los datos de entrenamiento ruidosos o desequilibrados, lo que puede afectar la precisión del modelo. - Pueden ser propensos al sobreajuste si se permiten árboles demasiado complejos o si se utilizan características irrelevantes. - Pueden ser inestables, ya que pequeñas variaciones en los datos de entrada pueden resultar en grandes cambios en la estructura del árbol.

Nota. Fuente: Adaptado de Scikit-learn developers (2021)

4.3.3. Modelo RandomForestClassifier.

El modelo alcanzó una precisión del 90.48%. El recall para la clase 1 (caída) fue del 97%.

Figura 11

Matriz de confusión modelo Random Forest

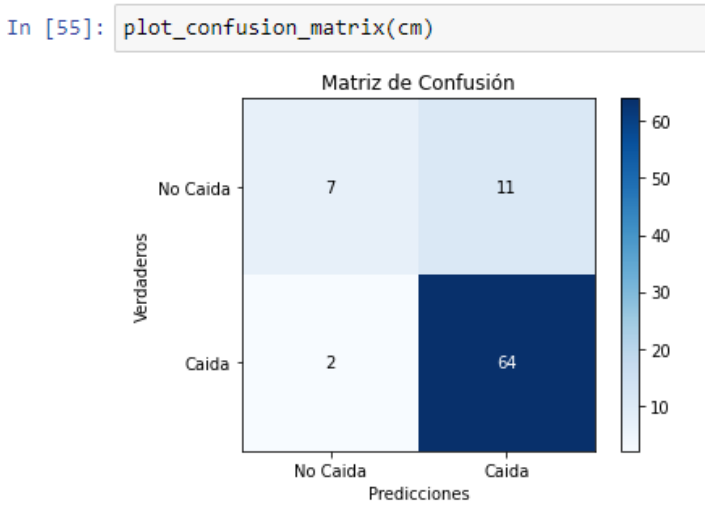


Tabla 3

Ventajas y Desventajas Random Forest

Ventajas	Desventajas
- Random forest es un algoritmo robusto y versátil - Puede manejar datos ruidosos y faltantes - Ofrece una buena capacidad de generalización - Menos propenso al sobreajuste que los árboles de decisión - Proporciona una estimación de la importancia de las características	- El entrenamiento del modelo puede ser más lento en comparación con KNN - La interpretación del modelo puede ser más difícil en comparación con KNN - Puede requerir un ajuste de hiperparámetros para obtener un rendimiento óptimo

Nota. Fuente: Adaptado de Scikit-learn developers (2021)

4.3.4. Modelo de Red Neuronal.

El modelo alcanzó una precisión del 92.86%. El recall para la clase 1 (caída) fue del 95.52%.

Figura 12

Matriz de confusión Red neuronal

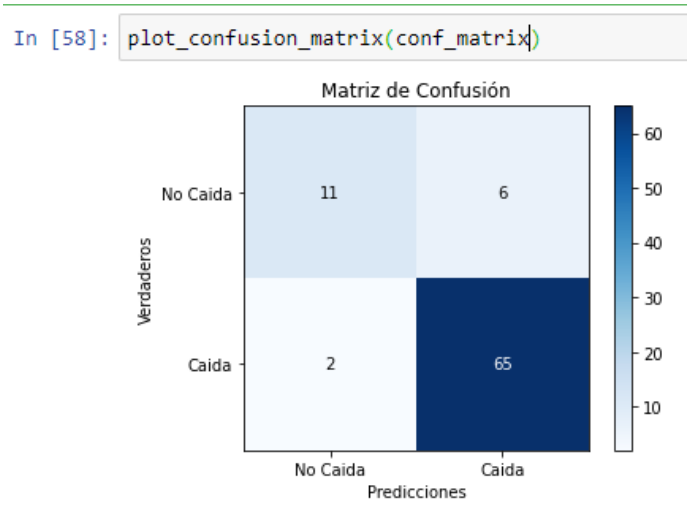


Tabla 4

Ventajas y Desventajas Red Neuronal

Ventajas	Desventajas
- Las redes neuronales son capaces de aprender representaciones de datos no lineales y complejas, lo que las hace adecuadas para el reconocimiento de patrones en la detección de caídas - Pueden adaptarse a diferentes tipos de datos y ser utilizadas en una amplia variedad de aplicaciones - La capacidad de generalización de las redes neuronales puede mejorar con el aumento de los datos de entrenamiento	- El entrenamiento de redes neuronales puede requerir una gran cantidad de tiempo y recursos computacionales - La interpretación de los modelos de redes neuronales puede ser difícil debido a su naturaleza de "caja negra" - El ajuste de hiperparámetros, como la arquitectura

Nota. Fuente: Adaptado de TensorFlow (2021)

4.3.5. Selección.

Tabla 5

Comparativa precisión y recall de los modelos

Modelo	Precisión	Recall
KNN	Precisión: 89.29%	Recall: 97%.
Decision Tree	Precisión: 84.52%.	Recall: 97%
Random Forest	Precisión: 90.48%.	Recall: 97%.
Red Neuronal	Precisión: 92.86%.	Recall: 95.52%

Nota. Valores obtenidos al evaluar los modelos con el mismo conjunto de datos de prueba.

Tras evaluar los cuatro modelos de aprendizaje automático, se observa que todos tienen un desempeño similar en cuanto a la precisión y recall en la detección de caídas de smartphones. Sin embargo, la Red Neuronal alcanzó la mayor precisión (92.86%) y un recall del 95.52% para la clase 1 (caída), lo que indica que este modelo podría ser el más adecuado para la detección de caídas de smartphones en este caso particular.

A pesar de haber obtenido un rendimiento similar por parte de los modelos, se decidió seleccionar el modelo de red neuronal debido a que cuenta con una mayor capacidad de

aprendizaje y adaptabilidad, así como un mejor rendimiento en conjuntos de datos grandes. Además, el modelo de red neuronal se integra con facilidad con tecnologías web modernas.

4.4. Despliegue Del Prototipo Para La Detección De Caídas

4.4.1. Diseño De La Aplicación Móvil.

La aplicación móvil cuenta con una interfaz de usuario simple y fácil de usar, que incluye elementos visuales para mostrar los valores de los ejes X, Y y Z del acelerómetro y el giroscopio. Además, se implementaron botones para iniciar y detener la recolección de datos.

4.4.2. Implementación De La Aplicación Móvil.

La aplicación fue desarrollada utilizando Android Studio y el lenguaje de programación Java. Para acceder a los sensores del dispositivo, se utilizó la clase SensorManager, que permite acceder a los sensores de acelerómetro y giroscopio. Se implementaron dos SensorEventListener para manejar los eventos generados por los sensores.

El código principal de la aplicación se encuentra en la clase MainActivity, que hereda de la clase AppCompatActivity. La clase MainActivity contiene métodos y objetos para manejar la recolección de datos, el almacenamiento en Firebase y la interacción con la interfaz de usuario.

La aplicación utiliza la biblioteca Firebase Firestore para almacenar y recuperar datos de sensores en tiempo real. Los datos recolectados se almacenan en la base de datos Firestore bajo la colección "datos" y el documento "registro".

4.4.3. Página Web Para La Visualización De Caídas.

El objetivo de este capítulo es describir el diseño e implementación de una página web que permita visualizar las predicciones de caídas de smartphones en tiempo real. La página web fue desarrollada utilizando el framework React y se integra con una base de datos Firebase para recibir datos de sensores en tiempo real y con un modelo de predicción implementado en TensorFlow.js.

4.4.4. Implementación De La Página Web.

La implementación de la página web se realizó utilizando el framework React, que permite la creación de aplicaciones web de una sola página con componentes reutilizables y modulares. Además, se utilizó Firebase para almacenar y recuperar datos de sensores en tiempo real y TensorFlow.js para realizar predicciones utilizando un modelo de aprendizaje automático previamente entrenado.

La aplicación React consta de varios componentes que se combinan para formar la estructura de la página web. El componente principal App.js importa e integra los componentes Header, Predictions y Footer. El componente Predictions se encarga de consultar los datos de sensores almacenados en Firebase y de realizar predicciones utilizando el modelo cargado en TensorFlow.js.

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

- La recolección de los datos mediante los sensores del smartphone, específicamente el acelerómetro, el cual mide la aceleración en los tres ejes (x, y, z). Esto permitió obtener información detallada sobre los movimientos del dispositivo y su orientación en el espacio, lo que resulta relevante para la identificación de caídas.
- El uso de aprendizaje automático en la modelización de datos ha permitido identificar caídas de manera efectiva. A través de la implementación de diferentes algoritmos, se logró un modelo preciso.
- La implementación de la red neuronal convolucional proporciona una solución precisa y escalable para la detección de caídas en comparación con otros enfoques, como los algoritmos basados en umbrales. A pesar de las desventajas relacionadas con el tiempo de entrenamiento y la necesidad de una gran cantidad de datos, la red neuronal convolucional ofrece un rendimiento superior en la clasificación de eventos de caída y no caída.
- La página web desarrollada en React y el uso de TensorFlow.js permiten visualizar las predicciones de caídas en tiempo real, proporcionando una herramienta útil para monitorear y alertar sobre eventos de caída. Esta solución es accesible en diversos dispositivos y plataformas, mejorando la versatilidad del sistema.

Recomendaciones

- Explorar la posibilidad de incluir otros sensores en la aplicación móvil, como el giroscopio, magnetómetro o GPS, para mejorar la precisión y la robustez del sistema de detección de caídas. La combinación de diferentes fuentes de datos puede proporcionar información adicional que mejore la detección de eventos de caída.
- Investigar y desarrollar algoritmos de procesamiento de señales en tiempo real para la aplicación móvil con el fin de mejorar la calidad de los datos recolectados y reducir el ruido y las interferencias. Esto puede incluir técnicas de filtrado, suavizado o normalización de los datos del acelerómetro antes de ser almacenados en Firebase y utilizados en la predicción de caídas.
- Evaluar y comparar diferentes arquitecturas de redes neuronales y enfoques de aprendizaje automático para mejorar aún más la precisión y el rendimiento del sistema de detección de caídas. Esto puede incluir la implementación de redes neuronales recurrentes, redes neuronales profundas o algoritmos de aprendizaje automático supervisado y no supervisado.

BIBLIOGRFÍA

Al-Jumaily, A. (2016). Fall Detection System for the Elderly Based on the Classification of Shimmer Sensor Prototype Data. International Journal of Computer Applications, 137(7), 29-32.

Herrera, D., Giraldo, C., & Alvarez, M. (2019). Smartphone-based fall detection for the elderly using machine learning techniques. Sensors, 19(8), 1844.

Android Studio (2021). Sensor types. Android Developers. Recuperado de https://developer.android.com/guide/topics/sensors/sensors_overview#sensors-coords

Sucerquia, A., López, J. D., & Vargas-Bonilla, J. F. (2017). Sistema automático de detección de caídas basado en acelerometría, giroscopia y magnetometría. Ingeniería y Ciencia, 13(26), 203-231.

Herrera, D., Giraldo, J. D., & Alvarez, J. C. (2019). Fall Detection System for Elderly People Using IoT and Big Data. Procedia Computer Science, 162, 530-537.

IBM. (2000). CRISP-DM 1.0: Guía paso a paso del modelo de proceso estándar de la industria de minería de datos (CRISP-DM). Recuperado de https://www.researchgate.net/publication/267940929_CRISP-DM_10_Guia_paso_a_paso_del_modelo_de_proceso_estandar_de_la_industria_de_mineria_de_datos_CRISP-DM

Aha, D. W., Kibler, D., & Albert, M. K. (1991). Instance-based learning algorithms. Machine Learning, 6(1), 37-66. Recuperado de https://www.researchgate.net/publication/220426522_Instance-based_learning_algorithms

Aloulou, H., Mokhtari, M., Tiberghien, T., Biswas, J., & Kautz, H. (2015). Fall detection using movement characteristics of a wireless wearable device. IEEE journal of biomedical and health informatics, 19(2), 430-438.

Hernández, R., Fernández, C. & Baptista, P. (2014). Metodología de la investigación (6ta ed.). México D.F.: McGraw-Hill.

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & Wirth, R. (2000). CRISP-DM 1.0: Step-by-step data mining guide.

Anguita, D., Ghio, A., Oneto, L., Parra, X., & Reyes-Ortiz, J. L. (2013). A public domain dataset for human activity recognition using smartphones. In Esann.

Bagalà, F., Becker, C., Cappello, A., Chiari, L., Aminian, K., Hausdorff, J. M., ... & Zijlstra, W. (2012). Evaluation of accelerometer-based fall detection algorithms on real-world falls. PloS one, 7(5), e37062.

Bourke, A. K., Klenk, J., Schwickert, L., Aminian, K., Ihlen, E. A., Mellone, S., ... & Zijlstra, W. (2014). Fall detection algorithms for real-world falls harvested from lumbar sensors in the elderly population: a machine learning approach. Conference proceedings: ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Annual Conference, 2014, 3712-3715.

Buczak, A. L., & Gifford, C. M. (2016). Fuzzy association rule mining for community crime pattern discovery. ACM SIGKDD Explorations Newsletter, 12(1), 41-50.

Cvetković, B., Janko, V., Luštrek, M., & Gams, M. (2016). Fall detection and activity recognition with machine learning. Informatica, 40(2), 161-168.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

O'Reilly, M., Caulfield, B., Ward, T., Johnston, W., & Doherty, C. (2017). Wearable Inertial Sensor Systems for Lower Limb Exercise Detection and Evaluation: A Systematic Review. Sports Medicine, 48(5), 1221-1246.

Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32.

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Kudlur, M. (2016). TensorFlow: A system for large-scale machine learning. In 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16) (pp. 265-283).

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press

Chicco, D., & Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. BMC Genomics, 21(1), 6. https://doi.org/10.1186/s12864-019-6413-7

Fawcett, T. (2006). An introduction to ROC analysis. Pattern Recognition Letters, 27(8), 861–874. https://doi.org/10.1016/j.patrec.2005.10.010

Kohavi, R., & Provost, F. (1998). Glossary of terms. Machine Learning, 30(2–3), 271–274. https://doi.org/10.1023/A:1017181826899

David. (2020). Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. ArXiv.org. https://arxiv.org/abs/2010.16061

TensorFlow. (2021). TensorFlow: An end-to-end open source machine learning platform. Recuperado de: https://www.tensorflow.org/

Scikit-learn. (2021). Scikit-learn: Machine Learning in Python. Recuperado de: https://scikit-learn.org/stable/index.html

Facebook. (2021). React: Una biblioteca de JavaScript para construir interfaces de usuario. Recuperado de: https://reactjs.org/

The GNU Awk User's Guide. (2022). Gnu.org. https://www.gnu.org/software/gawk/manual/gawk.html

Alpaydin, E. (2010). Introducción al aprendizaje automático. España: Ediciones Paraninfo.

Bolaños, C. (2016). Métodos de recolección de datos en investigación cuantitativa. Revista Electrónica de Investigación Educativa, 18(1), 1-14.

Creswell, J. W. (2014). Investigación cualitativa y diseño de investigación: Elegir entre cinco enfoques. México: Editorial El Manual Moderno.

Gómez, M. (2018). Evaluación y selección de modelos de aprendizaje automático. Revista Ingeniería y Ciencia, 14(27), 123-145.

Hernández, R., Fernández, C., & Baptista, P. (2014). Metodología de la investigación. México: McGraw Hill Education.

Medina, C. (2012). Evaluación de sistemas de detección de caídas utilizando algoritmos de aprendizaje automático. Revista Iberoamericana de Tecnologías del Aprendizaje, 7(4), 235-244.

Navarro, F. (2010). Líneas de investigación en aprendizaje automático y sus aplicaciones. Revista Iberoamericana de Inteligencia Artificial, 14(43), 9-20.

Pérez, G. (2005). Análisis de datos en investigación cuantitativa: Conceptos básicos. Revista Colombiana de Estadística, 28(1), 1-27.

Rojas, L. (2011). Innovación en sistemas de detección de caídas: Un enfoque cuantitativo. Revista de Investigación en Tecnologías de la Información, 2(1), 61-75.

Sánchez, A. (2017). Algoritmos de aprendizaje automático y su impacto en la detección de eventos adversos. Revista de Investigación en Ciencias Computacionales, 5(2), 112-130.

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). Classification and Regression Trees. CRC Press.

Scikit-learn developers (2021). Decision Trees. Retrieved from https://scikit-learn.org/stable/modules/tree.html

ANEXOS

Anexo 1: Modelo Árbol de desiciones