Reporte Final: Reconocimiento de Actividad en Personas Mayores

Diego Sú Gómez, Estefanía Pérez Yeo, Vanessa Méndez Palacios & Francisco Javier Sánchez Panduro

Instituto Tecnológico y de Estudios Superiores de Monterrey, Campus Guadalajara

Resumen Se cuenta con un conjunto de datos que se compone de registros de 18 adultos mayores, con edades comprendidas entre los 70 y 95 años, quienes llevaron acelerómetros tri-axiales durante 40 minutos en un protocolo de vida diaria semiestructurado. El objetivo principal de esta investigación es entrenar modelos de Machine Learning capaces de predecir y clasificar diferentes actividades en función de las mediciones de los sensores. Las etiquetas representan las actividades reales registradas en cada observación, lo que constituye la base para la generación de varios modelos de Machine Learning que se implementarán posteriormente. Para determinar el modelo óptimo, se llevaron a cabo una serie de procedimientos para garantizar la confiabilidad de los resultados. Una vez obtenidas las puntuaciones de los modelos, se procedió a su clasificación, explicando los criterios utilizados para esta clasificación. El modelo considerado más adecuado para el conjunto de datos fue el de Random Forest. Este informe ofrece una visión general de la metodología empleada y los resultados obtenidos en la búsqueda del modelo de clasificación más apropiado para el conjunto de datos de reconocimiento de actividades en personas mayores. Las conclusiones e interpretaciones se presentan en la sección final del informe.

Keywords: Reconocimiento de Actividad Humana · Personas Mayores · Acelerómetros Triaxiales · Machine Learning · Modelos de Clasificación · Modelos de Machine Learning · Random Forest · K Nearest Neighbor · Gradient Boosting · Regresión Logística · Perceptrón Multicapa · Stochastic Gradient Descent · AdaBoost · Evaluación de Modelos · Conjunto de Datos · Procesamiento de Datos · Predicción de Actividades · Sensores · Protocolo de Vida Libre · Resultados de la Clasificación · Conclusiones de la Investigación

1. Introducción

El conjunto de datos de Reconocimiento de Actividad Humana en Personas Mayores contiene información de 18 adultos mayores desde saludables hasta frágiles (de 70 a 95 años de edad) que llevan puestos dos acelerómetros triaxiales durante aproximadamente 40 minutos en un protocolo de vida libre semiestructurado. Los acelerómetros se encuentran localizados en la parte baja de la espalda y en la parte baja de la cadera derecha. Este conjunto de datos está separado por archivos .csv de cada uno de los adultos participantes en el estudio, y la estructura de cada archivo .csv es la siguiente:

- $Timestamp$: Marca de tiempo del registro realizado
- $Back_x$: Aceleración en el eje x del sensor de la espalda.
- $Back_y$: Aceleración en el eje y del sensor de la espalda.
- $Back_z$: Aceleración en el eje z del sensor de la espalda.
- $Thigh_x$: Aceleración en el eje x del sensor de la cadera.
- $Thigh_y$: Aceleración en el eje y del sensor de la cadera.
- $Thigh_z$: Aceleración en el eje z del sensor de la cadera.
- $Label$: Etiqueta correspondiente a un tipo de actividad, las cuales se clasifican de esta manera:
  - 1: Caminar
  - 3: Arrastrar los pies
  - 4: Subir escaleras
  - 5: Bajar escaleras
  - 6: Estar sentado
  - 7: Estar parado
  - 8: Estar acostado

El objetivo principal de este estudio fue poder entrenar distintos modelos de Machine Learning que pudieran predecir y clasificar cada actividad dependiendo de las medidas de los sensores. Las etiquetas para cada observación dentro de los archivos .csv es el resultado real de la actividad que se estaba llevando a cabo en la marca de tiempo de la respectiva observación, y estos datos fueron la fuente de la generación de varios modelos de Machine Learning que fueron implementados más adelante en el estudio.

No obstante, para este reto el equipo únicamente utilizó los conjuntos de datos del estudio para llevar a cabo un proceso de análisis de datos completo, el cual incluyó la observación y preprocesamiento de los datos, aunado a la generación, implementación, validación, prueba y optimización de varios modelos de clasificación para poder llegar a encontrar el modelo que mejor se ajustara a los datos que fueron provistos.

A lo largo de este reporte se detallarán todos los pasos que fueron seguidos para poder encontrar un modelo adecuado para la clasificación de los datos con los que se trabajaron, junto con explicaciones a profundidad, gráficos y otros recursos visuales que permitirán seguir el camino recorrido durante el desarrollo de este proyecto, además de la respectiva documentación de los resultados de la solución del reto y una sección final con conclusiones e interpretaciones personales de cada miembro del equipo.

2. Metodología

Para poder encontrar el modelo óptimo para el conjunto de datos que se utilizó, se tuvieron que realizar previamente varios procedimientos y métodos para asegurar que los resultados fueran confiables. A lo largo de esta sección se encuentran varias subsecciones las cuales contienen cada uno de estos procesos junto con una explicación a fondo de lo que se realizó, además de una justificación del por qué se realizó. Además de eso, también se incluyen los distintos modelos que fueron seleccionados para ser implementados, junto con una breve explicación de su funcionamiento y algunos de los parámetros más relevantes de cada modelo.

2.1. Preprocesamiento de los datos

El primer paso que se llevó a cabo para encontrar la solución al problema fue el preprocesamiento de los distintos conjuntos de datos con los que se trabajaron. Se sabía que el estudio fue realizado con 18 pacientes distintos; sin embargo, al momento de descargar los datos del lugar donde se subió el reporte del estudio, únicamente se encontraron 15 archivos distintos. Antes de comenzar con el la limpieza, manipulación y exploración de los datos con los que se iban a trabajar, el equipo tomó la decisión de concatenar todos los archivos de los pacientes en uno solo. Esto fue decidido así debido a que el proyecto se llevó a cabo en un entorno de ejecución virtual (Google Colaboratory), por lo que cada vez que este entorno se desconectara, se tenían que volver a cargar los archivos. Tomando en cuenta que eran 15 archivos distintos, se decidió que era más eficiente y rápido formar un solo archivo con todos los registros juntos.

Para poder realizar esto, se utilizaron varias librerías de Python que permitieron generar un archivo conjunto. La primera de estas librerías fue Glob, la cual permite leer varios archivos a la vez desde una ubicación en específico y después iterar sobre ellos. Al tener todos los archivos .csv en un arreglo, lo que se hizo fue seleccionar cada archivo, convertirlo en un DataFrame con la ayuda de Pandas, y concatenarlo en un DataFrame global, añadiendo una columna nueva llamada $patient_{id}$, la cual contenía el número de paciente al que pertenece cada observación. Tras haber iterado sobre todos los archivos, se tuvo un DataFrame con todos los registros juntos, el cual se convirtió a un archivo .csv y se descargó.Este proceso se puede observar en la imagen siguiente:

Tras haber obtenido el archivo .csv con todos los registros concatenados, lo siguiente que se realizó fue la limpieza y manipulación de este archivo, para preparar los datos para el análisis exploratorio y posteriormente la implementación de los modelos de clasificación.

2.2. Limpieza y manipulación de datos

Tras haber obtenido un archivo con todos los registros juntos, lo siguiente que se tuvo que hacer fue limpiar los datos y revisar que estuvieran listos para la implementación de los modelos. Para poder llevar a cabo este procedimiento, se

```
# Código de un solo uso - Leer los archivos .csv de cada paciente y concatenarlos en un DataFrame que será descargado

#Leer los dataframes y añadir una columna con el id del paciente para poder combinarlos
archivos_csv = glob.glob('/content/*.csv')

df_global = pd.DataFrame()
num_paciente = 1

#Leer cada uno de los .csv, añadir la columna patient_id y señalar los registros con el número de paciente correspondiente
for archivo in archivos_csv:
    df_paciente = pd.read_csv(archivo)
    df_paciente.insert(0,"patient_id",num_paciente)
    df_global = pd.concat([df_global, df_paciente])
    num_paciente +=1
```

Figura 1. Código de concatenación de los archivos .csv de cada paciente.

tuvieron que buscar valores duplicados, nulos o registros incompletos. Este paso es de suma importancia, debido a que si no se limpian correctamente los datos, los resultados se verán comprometidos y no se podría encontrar un modelo lo suficientemente confiable como para considerar completado el reto.

Teniendo esto en cuenta, lo que se hizo para limpiar adecuadamente los datos fueron varios procedimientos en el entorno de ejecución para poder garantizar que los datos estén completamente preparados para ser usados en los modelos de clasificación. Lo primero que se hizo fue la búsqueda y detección de valores nulos o registros faltantes. Encontrar este tipo de registros es de suma importancia, debido a que la mayoría de clasificadores no trabajan con valores nulos, además de que la visualización de los datos no sería confiable o incluso, se podrían tomar decisiones en base a estos datos y que fueran erróneas. Para encontrar los valores nulos en el dataset se utilizó Pandas, la librería de Python que incluye varios métodos para poder encontrar la totalidad de registros nulos o incompletos y eliminarlos del DataFrame. Sin embargo, afortunadamente no hubo ni registros con valores nulos ni incompletos, lo que ayudó de gran manera a evitar una carga de trabajo adicional para llegar a la solución del reto.

Finalmente, para concluir con la limpieza y preparación de los datos, lo último que se hizo fue evaluar el dataset para encontrar observaciones duplicadas. El tener registros duplicados en el dataset también puede alterar la clasificación y los resultados del problema, por lo que era de suma importancia llegar a detectar a tiempo estos registros y eliminarlos. Al igual que con los valores nulos, Pandas ofrece varias alternativas y métodos que son útiles para detectar los duplicados y eliminarlos. No obstante, el conjunto de datos tampoco contenía ningún valor duplicado, lo que contribuyó de gran manera a hacer más eficiente el desarrollo de la solución de este problema.

Tras haber asegurado que el conjunto de datos no contaba con valores repetidos ni tampoco tenía registros vacíos o incompletos, se puede decir que el preprocesamiento y limpieza de la información están listos y se puede proceder con el análisis exploratorio de datos y la selección y definición de los modelos a usar y sus criterios de selección y evaluación.

2.3. Análisis exploratorio

Después de haber procesado, limpiado y revisado los datos a detalle, lo último que se debe de hacer antes de comenzar con la selección e implementación de modelos es estudiar, explorar y analizar cuidadosamente los datos con los que se van a trabajar, para así poder tener una idea de qué modelos podrían ser útiles, además de idear un plan para evaluar dichos modelos. El punto más importante y el objetivo principal de un análisis exploratorio es estudiar los datos, entender con qué se está trabajando, cómo se comportan los valores que comprenden este dataset y evaluar distintos aspectos estadísticos de los valores, como podrían ser si los datos están balanceados o no, cómo están distribuidos, entre otros parámetros importantes que se explicarán más a fondo. Además de eso, también es de suma relevancia y se recomienda generar gráficos visuales que ayuden a entender los datos con los que se está trabajando.

Lo primero que se realizó fue, mediante el método de Pandas *shape*, encontrar el tamaño del DataFrame con el que se estaba trabajando, para poder comprender a fondo cuántos registros se tenían, y tener esa cantidad como base para el resto de procesos del desarrollo de la solución. El tamaño del DataFrame después de haber ejecutado este código salió de 1,832,560 observaciones y 9 variables, las cuales eran las 8 variables del dataset original y una novena que se agregó, la cual es una variable categórica que explica la etiqueta de cada registro con el nombre de la actividad en sí.

Además de haber obtenido la estructura del DataFrame, otro de los primeros pasos para este análisis fue utilizar el método *describe* de Pandas para encontrar información acerca de los datos en el DataFrame, como su dispersión, tendencias e información estadística básica de los datos. Este método permite encontrar varios parámetros estadísticos tales como promedio, desviación estándar, cuartiles, valores mínimos y máximos y total de valores.

Los resultados obtenidos al ejecutar este comando fueron los siguientes:

	patient_id	back_x	back_y	back_z	thigh_x	thigh_y	thigh_z	label
count	1.832560e+06	1.832560e+06	1.832560e+06	1.832560e+06	1.832560e+06	1.832560e+06	1.832560e+06	1.832560e+06
mean	8.121649e+00	-8.683814e-01	-3.178814e-02	2.244210e-02	-6.763959e-01	8.185066e-03	-3.858819e-01	3.940692e+00
std	4.196053e+00	2.756643e-01	1.556768e-01	4.279549e-01	5.596829e-01	2.707317e-01	5.087015e-01	2.912512e+00
min	1.000000e+00	-4.333252e+00	-2.031006e+00	-2.204834e+00	-7.942139e+00	-5.142578e+00	-7.593750e+00	1.000000e+00
25%	5.000000e+00	-9.909670e-01	-1.093750e-01	-2.692870e-01	-9.855960e-01	-1.132810e-01	-9.770510e-01	1.000000e+00
50%	8.000000e+00	-9.377440e-01	-1.855500e-02	-9.399400e-02	-9.357910e-01	-1.464800e-02	-1.906740e-01	3.000000e+00
75%	1.200000e+01	-8.344730e-01	5.761700e-02	3.078610e-01	-7.763700e-02	1.230470e-01	-3.174000e-03	7.000000e+00
max	1.500000e+01	3.630370e-01	1.576660e+00	1.179199e+00	3.395264e+00	5.725098e+00	3.953369e+00	8.000000e+00

Figura 2. Estadísticas descriptivas del conjunto de datos.

Tras haber obtenido los valores estadísticos más importantes del conjunto de datos, se pudo observar que las medidas de los acelerómetros suelen estar entre valores de -2 y 2, con algunas excepciones puntuales. Se pudo ver que los valores

más pequeños de la aceleración en los sensores son de -2, -4, -5 y -7, mientras que los valores máximos son de 1, 3 y 5. Otra cosa que se pudo obtener de estos parámetros fue que la mayoría de las mediciones de los acelerómetros son negativas. Esto se obtiene de los cuartiles de los datos, y se observa que la gran mayoría de ellos son negativos, y los que son positivos son con valores cercanos al 0. Con esta información, se podría interpretar que los valores de aceleración son negativos debido a que los sujetos de prueba son adultos mayores, que suelen tener algunas dificultades en términos de movilidad, lo que confirma que los datos del Dataset son congruentes.

Después de haber analizado brevemente las variables numéricas, lo siguiente que se realizó fue generar varios gráficos con las variables numéricas y las variables categóricas del conjunto de datos. El primer gráfico que fue generado fue un diagrama de cajas de cada una de las variables numéricas, para poder tener una percepción de cuántos valores atípicos se encontraban en cada variable y básicamente entender la distribución de los datos dependiendo de la variable que se estaba analizando. La gráfica resultante se puede observar a continuación.

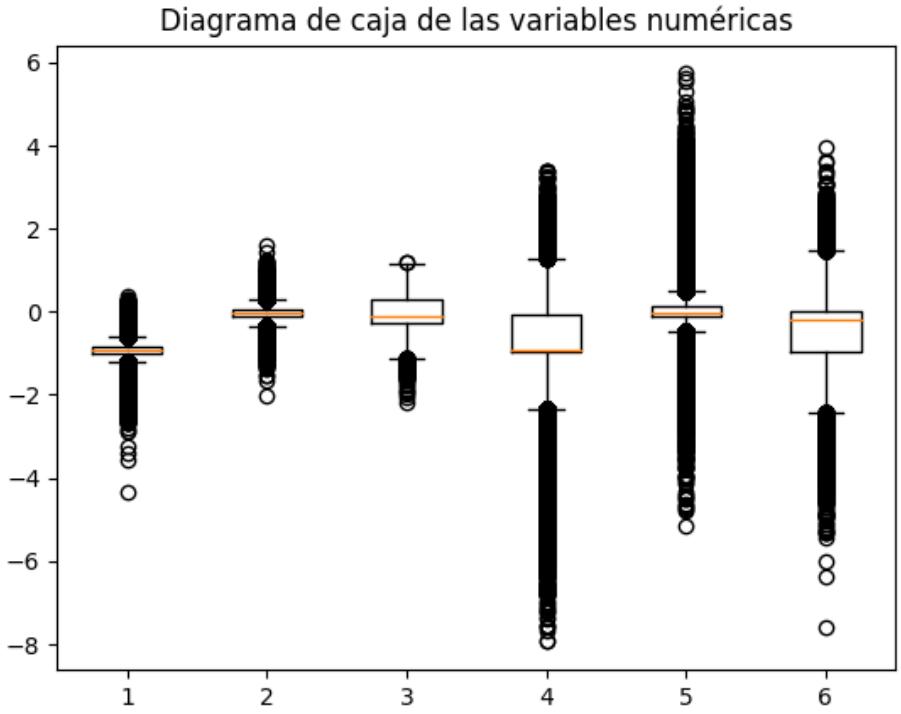

Figura 3. Diagrama de caja de las variables numéricas.

Cómo se puede observar en este diagrama, prácticamente todas las variables numéricas con las que se trabajaron contaban con una gran cantidad de valores atípicos. Esto se debe a que el conjunto de datos utilizado tiene una gran cantidad de registros, y era esperable que los valores estuvieran dispersos de gran forma tanto por la cantidad de observaciones como por la varianza que generan las mediciones de los acelerómetros al ser realizadas cada poco tiempo. Sin embargo, este gráfico también deja entrever que los datos se comportan como una distribución normal, al tener prácticamente en todas las variables numéricas el mismo número de valores atípicos por debajo y por encima de la mayoría donde se condensan los demás datos. Esto no puede ser afirmado aún, y se debe comprobar más adelante, pero es un buen indicio.

Lo siguiente que se realizó fue analizar la distribución de las observaciones entre todos los elementos de la clase que se iba a clasificar. Para esto, se realizó un histograma que muestra la cuenta de observaciones que tiene cada tipo de actividad por realizar. Este histograma es el siguiente:

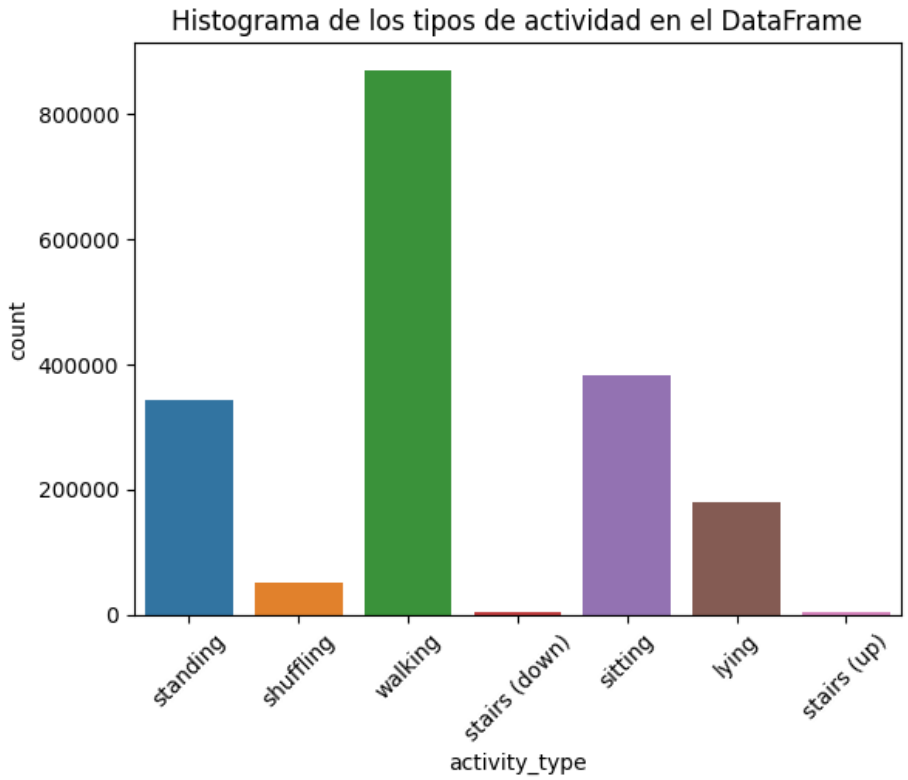

Figura 4. Histograma de cada tipo de actividad.

El histograma muestra que los elementos de la clase predictora se encuentran bastante desbalanceados, donde las observaciones identificadas como caminar tienen la gran mayoría de registros, mientras que actividades como subir o bajar escaleras tienen una cantidad muy pequeña de registros en comparación. Esto es un indicador principal de que probablemente para la implementación de los modelos sea necesario balancear los datos empleando alguna de las técnicas conocidas.

Otra de las gráficas que se realizaron para estudiar los datos con los que se van a trabajar, fue un histograma del número de registros por paciente. Cómo se mencionó previamente, al juntar todos los archivos .csv, se añadió una columna de cada id de paciente, lo que permitió generar este histograma para poder evaluar cuántos registros son correspondientes a cada participante del estudio.

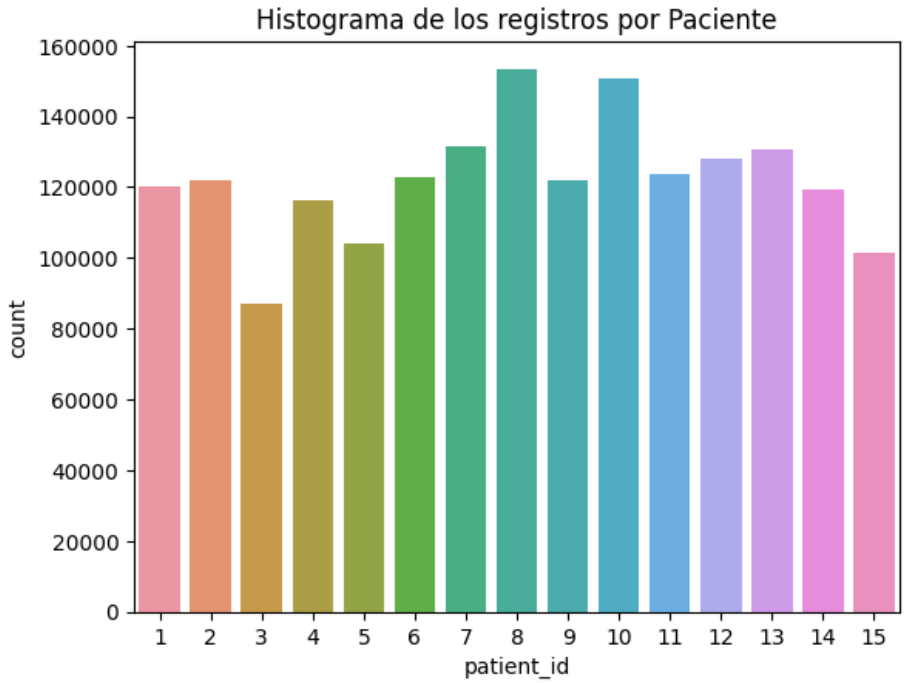

Figura 5. Histograma del número de registros por paciente.

El histograma muestra que el número de registros de cada paciente es un número de registros alto, aproximadamente entre 9000 y 15000 registros, aunque ninguno con la misma cantidad.Sin embargo, todos los pacientes, a excepción del paciente 3 tienen un número de registros relativamente similar, por lo que si se quisiera predecir por paciente, es importante considerar la cantidad de registros. Tomando en cuenta esta representación ahora se comprende qué paciente contri-

buye más al momento de observar los futuros gráficos. Este gráfico también hizo ver que no hay un desbalance significativo entre el número de observaciones de cada paciente, además de que tener esto en cuenta será de utilidad más adelante.

Posterior a haber analizado la distribución de registros tanto por número de paciente como por tipo de actividad, lo siguiente que se realizó en relación a la exploración de los datos fue un análisis bivariante entre las variables predictoras, para buscar encontrar si hay una correlación entre pares de variables. Este análisis genera gráficos de distribución entre todas las combinaciones de dos variables posibles para buscar encontrar posibles correlaciones. Cuando dos variables predictoras están correlacionadas, el gráfico de ambas se asemeja a una gráfica lineal con la ecuación $y = x$. Los gráficos del análisis bivariante se observan en la imagen posterior.

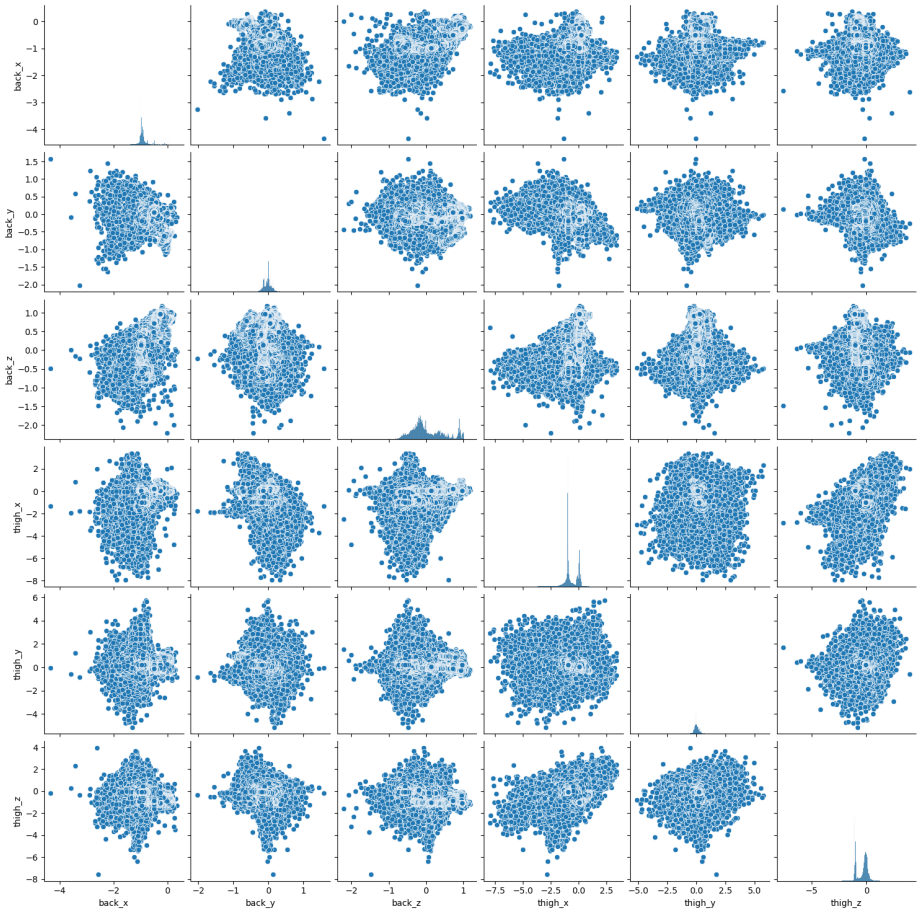

Figura 6. Análisis bivariante de las variables predictoras.

Como se puede observar, el análisis bivariante arrojó resultados que demuestran que ningún par de variables están correlacionados, ya que todas las gráficas de las variables son distintas a una relación lineal, lo que es una indicación relevante que demuestra que las variables predictoras no están correlacionadas. Sin embargo, para asegurar que las variables no están correlacionadas la mejor solución es realizar una matriz de correlación. Otra cosa que arrojó el análisis bivariante fueron los histogramas de cada variable predictora, lo que demuestra cómo están distribuidos los valores de cada uno de los ejes de cada sensor. Estos histogramas se encuentran más a fondo en el código donde se implementó la solución, por si se requiere revisar más a detalle.

A grandes rasgos, el análisis bivariante indicó, además de que no hay variables correlacionadas, que la mayoría de los datos están distribuidos en un rango específico, y se cuentan con muy pocos valores atípicos. Aparte de lo que ya se mencionó, no hay más observaciones relevantes sobre este análisis, por lo que para concluir con esta etapa lo último que se realizó fue la matriz de correlación de las variables para comprobar que todas son significativas para el modelo a generar.

La matriz de correlación es un método de la librería Pandas, la cual permite calcular los coeficientes de correlación entre todas las variables predictoras de un DataFrame. Estos coeficientes de correlación modelan qué tan relacionadas se encuentran dos variables entre sí, siendo lo máximo un valor de 1 o de -1, y lo mínimo un valor de 0. Idealmente, para la implementación de variables en un modelo, los coeficientes de correlación deben estar menores a 0.95 y mayores a -0.95, por lo que cualquier relación con valores que no estén en estos intervalos se debería descartar y no tomarse en cuenta en la implementación de los modelos. La matriz de correlación obtenida fue la siguiente:

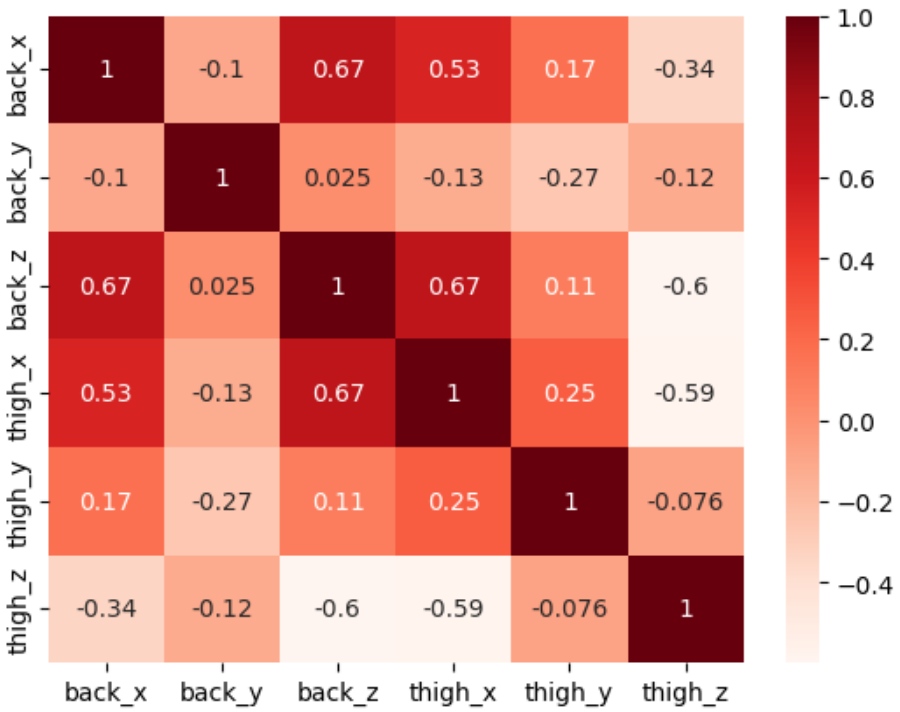

Figura 7. Matriz de correlación de las variables predictoras.

Tras haber obtenido y graficado la matriz de correlación, se puede observar que no hay variables altamente correlacionadas. Se considera que existe una alta correlación cuando el valor se encuentra entre 0.95 y 1 y, por defecto, entre -0.95 y -1. En este caso, en el heatmap graficado no se observan correlaciones en estos rangos, siendo el valor más cercano 0.67 (BackZ y ThighX, BackX y BackZ) Se puede observar que el resto de los coeficientes son más pequeños, siendo algunos muy cercanos a 0 y el resto oscilando entre estos extremos. Esta matriz confirma que no hay variables con alta correlación, por lo que se podrían incluir en el modelo de clasificación que se usará para modelar el tipo de actividad realizada dependiendo de las medidas de aceleración de cada sensor.

Finalmente, después de haber explorado y analizado a detalle los datos, y posteriormente de haberse asegurado de que todas las variables son significativas y no hay correlación, se procedió a seleccionar los diferentes modelos de clasificación que fueron implementados y evaluados para encontrar la solución óptima para este escenario.

2.4. Selección de modelos

Para la selección de los modelos de clasificación que fueron implementados, se buscó seleccionar los que operaran de formas distintas, ya sea mediante Redes Neuronales, Árboles de Decisión, Ensamble, entre otros métodos como optimización o aprendizaje basado en instancias. El objetivo de esto era encontrar una cantidad de modelos que permitiera abarcar la mayor cantidad de modelos de clasificación que operaran de la mayor cantidad de maneras distintas. Posterior a la investigación y estudio de todos los modelos disponibles, se llegó a la decisión de elegir los siguientes 7 modelos.

- Random Forest Classifier
- Stochastic Gradient Descent
- MLP Classifier
- AdaBoost
- Regresión Logística
- K-Nearest-Neighbors
- Gradient Boosting

A continuación se explicará cada uno de los modelos junto con su funcionamiento y algunos de sus parámetros más relevantes.

Random Forest Classifier El RFC, o Random Forest Classifier es un algoritmo de clasificación que funciona mediante la generación de árboles de decisión generados aleatoriamente para poder clasificar los datos en conjuntos. Estos árboles se generan con datos y variables aleatorias, y utilizan las predicciones/resultados de los otros árboles de decisión para generar la predicción final. Cada árbol selecciona una clase y la mayoría de selecciones termina generando los criterios de clasificación. Algunos de sus parámetros más relevantes son los siguientes:

- *nestimators*: Define el número de árboles de decisión que se van a generar.
- *maxdepth*: Define la profundidad de los árboles de decisión, si no se especifica la profundidad será la máxima posible.
- *criterion*: Define el criterio para medir la calidad de las predicciones en un split de datos del árbol respectivo.

Stochastic Gradient Descent El Stochastic Gradient Descent es un algoritmo de clasificación el cual se basa en el descenso de gradiente para encontrar los parámetros del modelo y ajustarlos, al igual que para entrenar los datos Permite procesar únicamente un set de datos aleatorio, generando una función y optimizándola para encontrar la menor pérdida entre los valores actuales y las predicciones. Algunos de los parámetros más importantes del SGD son los siguientes:

- *loss*: Define la función de pérdida que se optimizará, permitiendo elegir entre varias funciones.
- *alpha*: El coeficiente con el que se va a implementar el cambio en el gradiente de la función.
- *maxiter*: El número máximo de iteraciones que utilizará el modelo.

MLP Classifier El clasificador MLP, también conocido como el clasificador del perceptrón multicapa es una red neuronal que permite generar modelos con datos con relaciones complejas. Este algoritmo utiliza distintas capas con cierto número de neuronas para poder encontrar las relaciones entre las variables predictoras y después pasar esa información a la siguiente capa hasta encontrar una función óptima para modelar la pertenencia a cada clase. Algunos de los parámetros más importantes son los siguientes:

- *hiddenlayersizes*: El tamaño de capas y neuronas de la red del clasificador. Define el número de capas y de neuronas por capa.
- *activation*: Función de activación para cada capa de procesamiento, ofreciendo varias funciones dependiendo del conjunto de datos.
- *alpha*: El coeficiente con el que se va a implementar el cambio en el gradiente de la función.

AdaBoost AdaBoostClassifier es un algoritmo de ensamble de máquinas de aprendizaje que se utiliza para mejorar el rendimiento de los modelos de clasificación débiles, los çlasificadores débiles"son modelos de aprendizaje que tienen un rendimiento ligeramente mejor que una suposición aleatoria, como árboles de decisión poco profundos. Parámetros clave de AdaBoostClassifier:

- *baseestimator*: El tipo de clasificador débil que se utilizará en cada iteración. Por defecto, se usa un árbol de decisión con profundidad 1.
- *nestimators*: El número de clasificadores débiles que se entrenarán. Cuantos más estimadores, más preciso puede ser el modelo, pero también puede aumentar el riesgo de sobreajuste.

- *learningrate*: Un parámetro que controla la contribución de cada clasificador débil. Un valor menor reduce la contribución de cada clasificador y puede ayudar a evitar el sobreajuste.
- *randomstate*: Controla la aleatoriedad en la selección de muestras.

Regresión Logística La regresión logística es un algoritmo de clasificación que consiste en el modelado de funciones que determinen la pertenencia a un elemento de una clase en particular. Estas funciones se generan por medio de una función sigmoide, la cual se genera en base a los datos y determina el conjunto de pesos mínimos para cada elemento de cada clase. Los parámetros más importantes de la función son los siguientes:

- *tol*: La tolerancia del criterio de detenimiento.
- *maxiter*: Número máximo de iteraciones a utilizar en la regresión logística.

K-Nearest-Neighbors Es un tipo de aprendizaje basado en instancias o aprendizaje no generalizado. No intenta construir un model interno general, pero simplemente guarda instancias de los datos de entrenamiento. La clasificación es computada de un voto por mayoría simple de los vecinos más cercanos de cada punto. Los hiperparámetros más importantes son los siguientes:

- $Numberofneighbors$: Número de vecinos más cercanos que se consideraran para realizar la clasificación
- $Metric$: Fórmula de medida para calcular la distancia entre vecinos.
- $Weights$: Fórmula o esquema para medir y calcular los pesos entre los vecinos.

Gradient Boosting Gradient Boosting es una técnica de aprendizaje automático para problemas de clasificación y regresión, que construye un modelo predictivo en forma de un ensamblado de modelos débiles, típicamente árboles de decisión. Se trata de un algoritmo que mejora iterativamente la precisión de un modelo al optimizar el error residual de las predicciones previas.

Los parámetros más relevantes de este modelo son los siguientes:

- $Numerodeestimadores$: La cantidad de estimadores o elementos de aprendizaje que serán utilizados para obtener los resultados.
- $LearningRate$: Razón de cambio de cada uno de los estimadores para el ensamble final.
- $Depths$: La profundidad de cada uno de los estimadores en el ensamble.

2.5. Definición de los procesos, actividades y evaluaciones para cada modelo

Finalmente, para concluir con esta etapa, se procedió a definir la forma de implementación de cada uno de los modelos, así como sus métodos y criterios de

evaluación. Para evaluar el desempeño de cada modelo, se realizarán tres pruebas diferentes. Primeramente, se realizará una iteración simple con el conjunto de datos original, después, se aplicará validación cruzada con los originales y, finalmente, se aplicará validación cruzada pero con el conjunto de datos balanceados.

Para determinar qué modelo es mejor, se usará el reporte de clasificación de Sci-Kit Learn, el cual muestra medidas de "Precision", Recallƶ el puntaje F-1, con lo que se podrá diferenciar el modelo más adecuado para los datos trabajados. Este reporte se explicará más a fondo a continuación.

Antes de proceder con la implementación de los modelos y su correspondiente evaluación, se debe generar el conjunto de datos balanceado. Para esto, se debe determinar qué forma de balanceo se utilizará, ya sea oversampling o subsampling. Estas técnicas se utilizan para balancear conjuntos de datos que no están equilibrados, y comparten la misma idea, la cual tiene que ver con regularizar el tamaño de muestras de cada elemento de la clase, ya sea incrementando las muestras para igualar a la clase con más elementos, o reduciendo el tamaño del resto de muestras para igualar a la clase con menor cantidad de elementos.

Para este escenario, al ver que se cuenta con un gran número de registros, se optó por el subsampleo como técnica de balanceo. Para reducir las muestras de cada clase al tamaño de la muestra con menos elementos, se eliminan de forma aleatoria registros del resto de clases hasta que tengan el mismo número de muestras.

Ya con el proceso de balanceo de datos definido, las formas y métodos de implementación, así como los criterios de evaluación de los modelos, se puede proceder con el reporte de la experimentación de cada uno de ellos.

3. Experimentos

Cómo se explicó a fondo en la metodología, para aplicar y evaluar cada uno de los modelos de clasificación se realizarán tres iteraciones diferentes para generar reportes de resultados, aunque únicamente la última será utilizada para la evaluación del rendimiento de cada modelo.

1. Evaluación del modelo con los datos originales
2. Evaluación del modelo con los datos originales utilizando validación cruzada
3. Evaluación del modelo con los datos balanceados utilizando validación cruzada

Estas implementaciones fueron llevadas a cabo en el entorno virtual de Google Colaboratory, el cual cuenta con un motor de Python 3, además de 12.7 GB de RAM y 107.8 GB de Disco Duro. Además, se utilizaron varias librerías adicionales a Python, tales como Pandas, Numpy, Sklearn, Matplotlib, Seaborn y Glob. Todas estas librerías son requisitos necesarios para poder llevar a cabo las implementaciones de los modelos junto con sus respectivas evaluaciones.

Para implementar estas evaluaciones, se va a utilizar un conjunto de datos de entrenamiento, y otro conjunto de datos de prueba. En la primera evaluación

únicamente se va a generar un par de conjuntos, y con esto se va a entrenar el modelo y probarlo para evaluar qué tan bien clasifica los datos desbalanceados.

Para la segunda evaluación, lo que se hará es implementar una validación cruzada de K-pliegues, con 5 pliegues de datos seleccionados al azar, para entrenar el modelo para cada una de las validaciones y encontrar el rendimiento del modelo con los datos desbalanceados de las 5 validaciones juntas.

Finalmente, para la última evaluación, y la más importante, de igual manera se implementará una validación cruzada de K-pliegues, con 5 pliegues con datos al azar. La diferencia es que el conjunto de datos seleccionado es el de los datos balanceados. Con estos datos, se generarán los conjuntos de entrenamiento y prueba para, finalmente encontrar la evaluación del desempeño de cada modelo.

Después de haber realizado las evaluaciones de cada uno de los modelos, se seleccionará el mejor de ellos en base a los resultados de las métricas obtenidas, y tras haber encontrado el modelo que mejor puede predecir los datos balanceados, se optimizará dicho modelo para buscar mejorar los resultados obtenidos previamente.

En cuanto a los métodos que serán implementados para optimizar el modelo seleccionado, se realizarán dos principales para buscar mejorar el rendimiento del modelo más adecuado: Selección de hiperparámetros y selección de características. Estos dos métodos permitirán encontrar las configuraciones óptimas para que el modelo de clasificación pueda estar configurado de manera en que los valores utilizados sean los que arrojen los mejores resultados.

Para encontrar los hiperparámetros óptimos del modelo seleccionado, se generará un diccionario con el nombre de cada uno de los parámetros más importantes del modelo en cuestión como llaves, y un arreglo con las diferentes opciones de valores para cada hiperparámetros. Un ejemplo de lo que se hará es lo siguiente. Se debe suponer que se tiene el siguiente modelo: ModeloA, el cual cuenta con tres hiperparámetros relevantes: A, B y C. El diccionario a utilizar quedaría de la siguiente manera: $parametrosModeloA = "A" : [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], "B" : [True, False], "C" : [1, 2, 3]$

Este diccionario después se usaría en el método de Sci-Kit Learn conocido como GridSearch de Sci-Kit Learn, el cual permite generar combinaciones con diferentes opciones de un hiperparámetro y determinar cuál es la más adecuada. Esto se realiza mediante un proceso de validación cruzada anidada, en el cual primero se evalúa el modelo y se seleccionan ciertos hiperparámetros. Al seleccionarlos, se evalúa el modelo con esos parámetros con otra validación cruzada y así se van almacenando los puntajes del clasificador con los valores de cada hiperparámetro, hasta que se seleccionan los valores con el que el modelo tuvo el mejor desempeño.

El segundo método de optimización del modelo sería la selección de características. Hay tres tipos de métodos de selección de características: Filter, Wrapper y Filter-Wrapper, los cuales permiten encontrar el número óptimo de variables para el modelo de clasificación. En este caso, se implementarán los tres para tratar de encontrar un número óptimo de variables predictoras y determinar si las seis variables utilizadas son realmente significativas. Posterior a haber implemen-

tado los métodos de selección de características, se evaluarán los tres resultados obtenidos y se tomará la decisión de remover o dejar las variables predictoras en base a lo que se haya encontrado.

4. Resultados

Para poder determinar el mejor modelo de clasificación en cuanto a los datos balanceados utilizando validación cruzada, fue necesario mostrar los reportes de clasificación por los 7 modelos determinados en un inicio. A partir de los valores obtenidos, se busca que dentro de las métricas de rendimiento por clase se acerquen a un rendimiento relativamente alto, el cual se mide por su cercanía a 1. Como ya se había mencionado previamente, únicamente se va a tomar en cuenta el puntaje de clasificación del conjunto de datos balanceado con validación cruzada, para evitar sesgar el modelo y tener el mismo número de elementos de cada clase. Los resultados obtenidos de cada modelo de clasificación se muestran en una tabla a continuación.

4.1. Random Forest Classifier

Clase	Precision	Recall	F1-score	Support
1	0.63	0.60	0.62	3726
3	0.56	0.60	0.58	3726
4	0.74	0.76	0.75	3726
5	0.67	0.62	0.64	3726
6	0.79	0.81	0.80	3726
7	1.00	1.00	1.00	3726
8	1.00	1.00	1.00	3726
accuracy			0.77	26082
macro avg	0.77	0.77	0.77	26082
weighted avg	0.77	0.77	0.77	26082

Cuadro 1. Classification Report del Random Forest Classifier

4.2. Stochastic Gradient Descent

Clase	Precision	Recall	F1-score	Support
1	0.21	0.35	0.26	3726
3	0.24	0.17	0.20	3726
4	0.30	0.40	0.34	3726
5	0.34	0.16	0.22	3726
6	0.38	0.28	0.32	3726
7	0.91	0.95	0.93	3726
8	0.96	0.98	0.97	3726
accuracy			0.47	26082
macro avg	0.48	0.47	0.46	26082
weighted avg	0.48	0.47	0.46	26082

Cuadro 2. Classification Report del Stochastic Gradient Descent

4.3. MLP Classifier

Clase	Precision	Recall	F1-score	Support
1	0.61	0.49	0.54	3726
3	0.48	0.51	0.50	3726
4	0.64	0.65	0.65	3726
5	0.55	0.51	0.53	3726
6	0.68	0.81	0.74	3726
7	0.99	0.99	0.99	3726
8	1.00	1.00	1.00	3726
accuracy			0.71	26082
macro avg	0.71	0.71	0.71	26082
weighted avg	0.71	0.71	0.71	26082

Cuadro 3. Classification Report del MLP Classifier

4.4. AdaBoost

Clase	Precision	Recall	F1-score	Support
1	0.41	0.00	0.01	3726
3	0.20	0.59	0.30	3726
4	0.45	0.02	0.05	3726
5	0.20	0.20	0.20	3726
6	0.19	0.19	0.19	3726
7	0.87	0.95	0.91	3726
8	0.96	0.91	0.93	3726
accuracy			0.41	26082
macro avg	0.47	0.41	0.37	26082
weighted avg	0.47	0.41	0.37	26082

Cuadro 4. Classification Report del AdaBoost

4.5. Regresión Logística

Clase	Precision	Recall	F1-score	Support
1	0.56	0.60	0.22	3726
3	0.31	0.20	0.24	3726
4	0.45	0.60	0.52	3726
5	0.42	0.32	0.36	3726
6	0.38	0.56	0.45	3726
7	0.95	0.97	0.96	3726
8	0.97	0.98	0.98	3726
accuracy			0.55	26082
macro avg	0.53	0.55	0.53	26082
weighted avg	0.53	0.55	0.53	26082

Cuadro 5. Classification Report del K Nearest Neighbors

4.6. KNN

Clase	Precision	Recall	F1-score	Support
1	0.56	0.60	0.58	3726
3	0.53	0.50	0.51	3726
4	0.75	0.74	0.74	3726
5	0.68	0.59	0.63	3726
6	0.72	0.81	0.76	3726
7	0.99	1.00	1.00	3726
8	1.00	1.00	1.00	3726
accuracy			0.75	26082
macro avg	0.75	0.75	0.75	26082
weighted avg	0.75	0.75	0.75	26082

Cuadro 6. Classification Report del K Nearest Neighbors

4.7. Gradient boosting

Clase	Precision	Recall	F1-score	Support
1	0.60	0.52	0.56	3726
3	0.49	0.52	0.51	3726
4	0.65	0.66	0.65	3726
5	0.57	0.51	0.54	3726
6	0.69	0.80	0.74	3726
7	0.99	0.99	0.99	3726
8	1.00	1.00	1.00	3726
accuracy			0.72	26082
macro avg	0.71	0.72	0.71	26082
weighted avg	0.71	0.72	0.71	26082

Cuadro 7. Classification Report del Gradient boosting

Después de haber obtenido las tablas con los puntajes de cada uno de los modelos, se decidió ordenarlos del más adecuado al menos; y posterior a su ordenamiento, se explicarán los criterios seguidos para esta clasificación, además de seleccionar el modelo que se consideró más adecuado para el conjunto de datos.

1. Random Forest Classifier
2. K Nearest Neighbor

3. Gradient Boosting
4. MLP Classifier
5. Stochastic Gradient Descent
6. Adaboost

Se decidió ordenar de esta manera los modelos debido al puntaje de accuracy global, además de una evaluación detallada de cada uno de los puntajes de precision y recall. Cómo se pudo ver en las tablas anteriores, el Random Forest Classifier arrojó una exactitud del 0.77, siendo este el valor más alto entre todos los modelos. Sin embargo, también se puede observar que los puntajes de Recall de cada una de las clases no son tan bajos, lo que quiere decir que el modelo puede predecir correctamente la mayoría de las actividades descritas. Si los resultados de este modelo se comparan con el resto, se puede ver que no hay otro modelo que tenga puntajes similares. El más cercano fue el de K-Nearest Neighbors, pero este tiene puntajes de recall más altos y una precisión más baja en la mayoría de clases. En cuanto al resto de los modelos, los valores van decreciendo y no son tan buenos o exactos como estos dos. Es por esto por lo que se decidió como modelo más adecuado el Random Forest Classifier.

Al haber encontrado el modelo que mejor trabaja con los datos balanceados (Random Forest Classifier), lo siguiente que se debe hacer para optimizar su rendimiento es encontrar los hiperparámetros óptimos del mismo. Esto se hace mediante el método GridSearch de Sci-Kit Learn, el cual permite generar combinaciones con diferentes opciones de un hiperparámetro y determinar cuál es la más adecuada. Esto se realiza mediante un proceso de validación cruzada anidada, en el cual primero se evalúa el modelo y se seleccionan ciertos hiperparámetros. Al seleccionarlos, se evalúa el modelo con esos parámetros con otra validación cruzada y así se van almacenando los puntajes del clasificador con los valores de cada hiperparámetro, hasta que se seleccionan los valores con el que el modelo tuvo el mejor desempeño. Los hiperparámetros a seleccionar son los siguientes:

- *nestimators*: Número de árboles de decisión a usar. [100,200,300,400,500]
- *criterion*: Método de criterio de evaluación de cada split: [entropy, gini]
- *maxdepth*: Cantidad de nodos en los que se profundizará cada árbol de decisión. [10,20,30,40,50,60,70,80,90,None]

Tras haber utilizado el GridSearch para encontrar los hiperparámetros óptimos, se obtuvieron los siguientes resultados.

Hiperparámetros Óptimos: 'criterion': 'gini', 'max depth': None, 'n estimators': 300

Clase	Precision	Recall	F1-score	Support
1	0.64	0.60	0.62	3726
3	0.56	0.60	0.58	3726
4	0.74	0.76	0.75	3726
5	0.68	0.62	0.65	3726
6	0.78	0.83	0.80	3726
7	1.00	1.00	1.00	3726
8	1.00	1.00	1.00	3726
accuracy			0.77	26082
macro avg	0.77	0.77	0.77	26082
weighted avg	0.77	0.77	0.77	26082

Cuadro 8. Classification Report de Random Forest Classifier con hiperparámetros

Cómo se puede observar, los resultados del modelo no cambiaron mucho con los hiperparámetros óptimos. Esto se puede deber principalmente a que la cantidad de datos no es lo suficientemente grande como para evaluar correctamente la pertenencia a cada clase, o a que el modelo con los valores por default también es capaz de adaptarlo correctamente, o a que alguna de las variables predictoras no es realmente significativa. Esto se debe comprobar con el siguiente paso de la optimización.

Finalmente, uno de los últimos pasos de la optimización del modelo fue el proceso de selección de características. Se realizaron tres métodos distintos, los cuales fueron Filter, Wrapper y Filter-Wrapper. Estos métodos fueron aplicados con validación cruzada, permitiendo seleccionar de una a 6 variables predictoras y evaluar la exactitud del modelo con cada número de variables y así poder evaluar su desempeño. Para medir los resultados del mismo, se realizaron las siguientes gráficas.

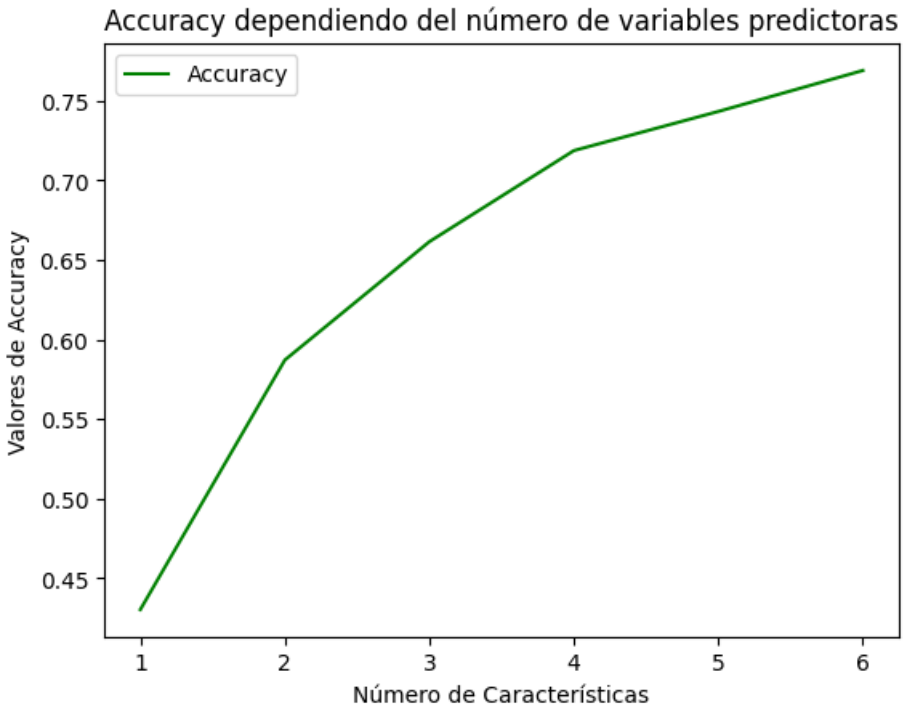

Figura 8. Exactitud dependiendo del número de variables usando Filter.

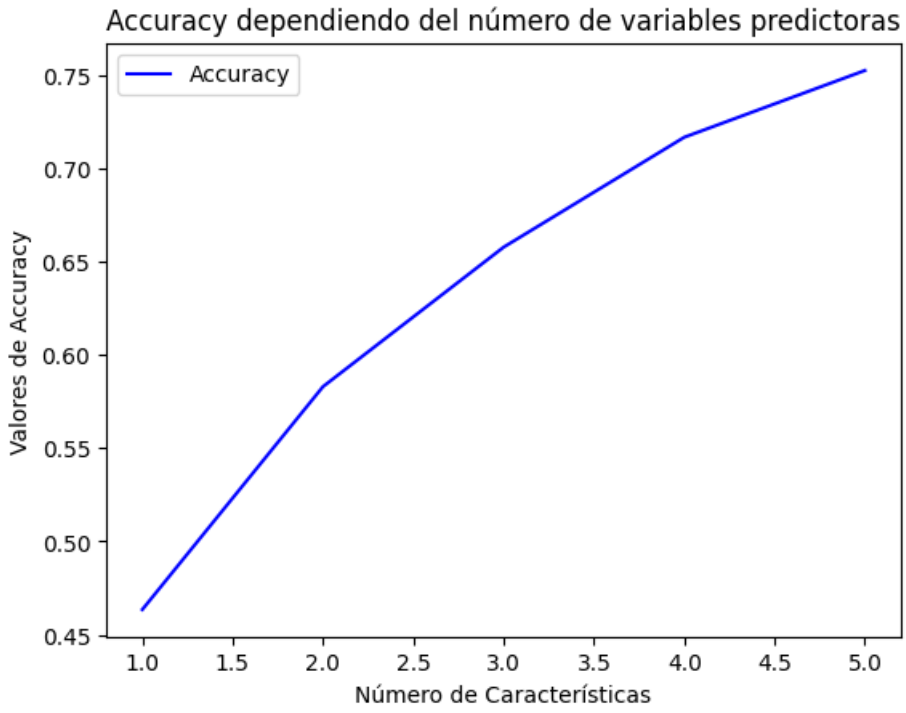

Figura 9. Exactitud dependiendo del número de variables usando Wrapper.

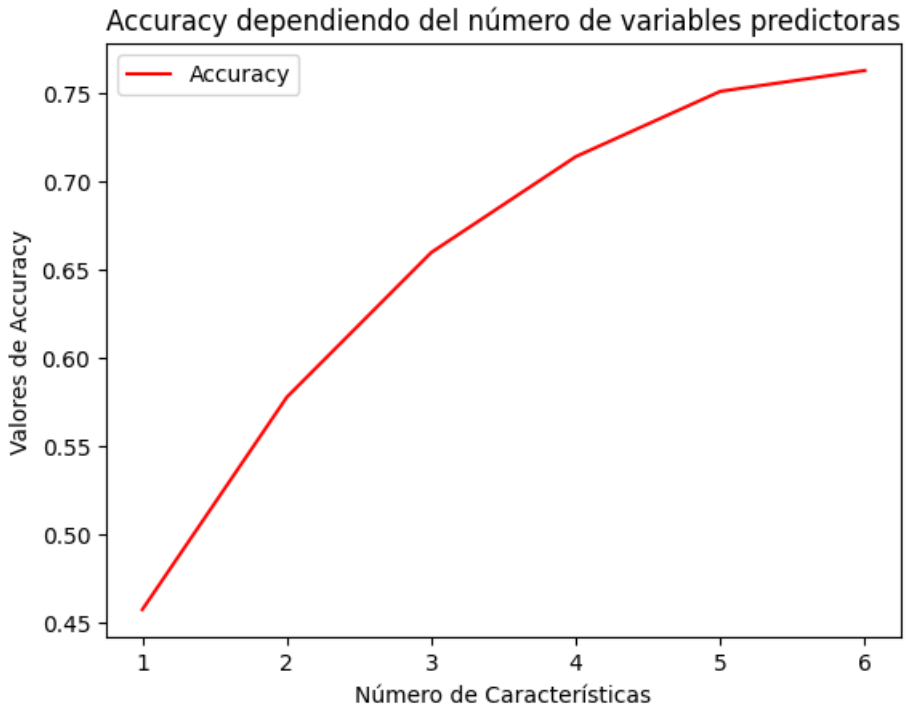

Figura 10. Exactitud dependiendo del número de variables usando Filter-Wrapper.

Cómo se puede observar en las gráficas, las medidas de exactitud del modelo fueron mayores cuando se utilizaron todas las variables. Esto ocurrió con los tres métodos de selección de características, por lo que se puede asumir que el modelo óptimo debe usar todas las variables para obtener los resultados más adecuados al conjunto de datos con el que se trabajó. Finalmente, el último método de optimización que se aplicó fue la regularización de los datos. Para hacer esto, se deben estandarizar los datos, buscando reducir la varianza entre los valores para ver si el modelo se vuelve más confiable. Tras haber hecho esto, se volvió a realizar el entrenamiento y prueba del modelo con validación cruzada para obtener el reporte de clasificación de los resultados, el cual fue el siguiente.

Clase	Precision	Recall	F1-score	Support
1	0.64	0.61	0.62	3726
3	0.56	0.59	0.58	3726
4	0.75	0.76	0.75	3726
5	0.68	0.62	0.65	3726
6	0.79	0.82	0.81	3726
7	1.00	1.00	1.00	3726
8	1.00	1.00	1.00	3726
accuracy			0.77	26082
macro avg	0.77	0.77	0.77	26082
weighted avg	0.77	0.77	0.77	26082

Cuadro 9. Classification Report de Random Forest Classifier con hiperparámetros y datos estandarizados

Se puede observar que el modelo óptimo con los datos estandarizados no mejoró mucho. Si bien es cierto que hay algunas clases con mejor precisión y otras con menos recall, a gran escala el cambio no es tan relevante, por lo que se puede decir que este es el modelo óptimo para modelar los datos. Si se quisiera seguir más a fondo con la optimización, se podría hacer la selección de hiperparámetros con más valores e incluso tomando en cuenta más parámetros, para ver si con eso se podría mejorar aún más el desempeño, o inclusive cambiar la forma de balancear los datos. Sin embargo, con el volumen de información que se está trabajando, esto tomaría mucho espacio y tiempo computacional, lo que haría que fuera muy tardado realizar las implementaciones de los modelos.

5. Conclusiones

5.1. Diego Sú

Después de haber realizado todo el proceso para la resolución del reto, se puede concluir que el modelo óptimo (Random Forest Classifier) puede predecir de forma confiable los resultados, pero no de una manera tan adecuada. Esto se puede deber a que no hay suficientes datos al momento de realizar el balanceo, o a que realmente si es complicado modelar el tipo de actividad que se está realizando dependiendo de las medidas de los sensores en un momento de tiempo específico. Esto mismo puede ser por que un sensor puede tener un rango de valores similares con dos actividades distintas, por ejemplo el estar acostado o parado puede que se tengan los mismos valores en todos los sensores. O el estar levantándose puede que se confunda con subir escaleras. La combinación de ambos puede que haya llevado a que el modelo no fuera tan adecuado.

Otra alternativa que pudo haber sido mejor sería la de incrementar significativamente los parámetros de los modelos, con la desventaja de que esto podría verse limitado debido a las configuraciones del entorno de ejecución, además de

que llevaría una gran cantidad de tiempo ejecutar los modelos con parámetros más amplios. Además de eso, también se podría haber implementado el oversampling en lugar de reducir los registros. Sin embargo, esto también hubiera aumentado el tiempo de ejecución y la complejidad de los algoritmos. Idealmente, si se contara con los recursos computacionales y el tiempo necesario, lo mejor sería hacer oversampling y evaluar los modelos con más pliegues y con parámetros más detallados.

Aún así, se puede concluir que el modelo es bastante confiable y que puede llegar a clasificar correctamente la mayoría de clases. Sin embargo, en lo personal, se puede concluir que hay bastantes formas de buscar un mejor modelo pero estas serían más demandantes, además de que los datos también pueden llegar a confundirse entre cada actividad por lo mismo de que una medida de un sensor puede significar la realización de varias actividades, lo que hace que haya cierto grado de ambigüedad en los registros, lo que influye también en el desempeño de los modelos.

Se puede concluir que fue un proyecto bastante retador pero fructífero, ya que permitió al equipo implementar un modelo de ciclo de vida de los datos, además de también contribuir al pensamiento crítico y al análisis para encontrar la solución más adecuada. También, este proyecto fue bastante útil para aplicar los temas vistos en clase e implementar modelos, técnicas y métodos para mejorar el desempeño de los distintos algoritmos de clasificación, además de también entender cómo aplicar un proceso de ciencia de datos en un escenario real. Fue un reto bastante productivo que dejó mucho aprendizaje.

5.2. Estefanía Pérez

A partir de los 7 modelos utilizados para la clasificación de clases conforme a las clases de actividades que realizan los adultos de tercera edad; el reporte de clasificación proporciona razones clave para concluir que este es el mejor modelo a diferencia del K Nearest Neighbor, Gradient Boosting, MLP Classifier, Stochastic Gradient Descent y Adaboost.

El modelo Random Forest listo para producción ya con los hiperparámetros y features óptimos muestra que la presición y recall demuestran un equilibrio sólido, donde sus instancias se recuperan correctamente generando así buenas predicciones.

La precisión global representado por accuracy marca un 77 por ciento de efectividad general, esto sugiere un modelo incluso después de su regularización no es sesgado por un clase en específico gracias al undersampling aplicado.

También, cabe mencionar, que una de las razones por las cual, usamos la técnica de undersmapling, fue ya que la clase 4 tenía descrito desde un inicio 3,726 registros y a comparación de la clase 1 con 869,690 registros, la diferencia es muy alta; por ende un oversampling podría generar un sesgo al momento de aplicarlo sobre una gran mayoría de las clases, sin incluir el bajo rendimiento que tendría en cuestión de running-time.

5.3. Vanessa Méndez

Después de evaluar los modelos de clasificación que se implementaron, se puede concluir que el Random Forest Classifier se destacó como el más adecuado en términos de precisión y recall en un conjunto de datos balanceado con validación cruzada. Por lo tanto, este modelo es una buena elección para abordar el problema de clasificación. Sin embargo, aún se podría seguir optimizando, considerando un rango más amplio de hiperparámetros o explorando estrategias adicionales de manejo de datos desequilibrados.

Adicionalmente, a pesar de la búsqueda de hiperparámetros óptimos mediante GridSearch, no se observaron mejoras sustanciales en el rendimiento del modelo. Esto podría indicar que el modelo ya tiene configuraciones predeterminadas que se adaptan bien a los datos disponibles. Otra observación relevante, es que los resultados nos sugieren que el uso de todas las variables predictoras es más efectivo en términos de precisión y recall en comparación con la selección de un subconjunto de características. Esto podría deberse a que todas las variables aportan información relevante para el modelo. Por otro lado, la estandarización de datos no condujo a mejoras significativas en el rendimiento del modelo, lo que indica que la variabilidad entre los valores de las variables no era un factor crítico para este conjunto de datos.

5.4. Javier Sánchez

Aunque existen diferentes modelos de clasificación, para este conjunto de datos se puede observar que el clasificador de Random Forest es el más confiable, por lo que si se tienen nuevos datos y se quiere predecir qué tipo de movimiento se está haciendo, este sería el mejor modelo para lograrlo.

Bibliografía

Hastie, T., Tibshirani, R., Friedman, J., & Friedman, J. H. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition. Springer New York.

James, G., Witten, D., Hastie, T., Tibshirani, R., & Taylor, J. (2023). An Introduction to Statistical Learning: With Applications in Python (1st ed.). Springer International Publishing AG.