UNIVERSIDAD CATÓLICA DE LA SANTÍSIMA CONCEPCIÓN

FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

Caracterización de patrones de conducta diaria de residentes de una casa inteligente

Ignacio Nicolas Cabrera Matus

Informe de Habilitación Profesional para optar al título de:
Ingeniero Civil Eléctrico

Profesor Patrocinante:
Dra. Silvia E. Restrepo M.

Profesores Comisión:
Dr. Roberto E. Aedo G.
Dr. Jaime A. Cariñe. C.

Concepción, Agosto de 2023

UNIVERSIDAD CATÓLICA DE LA SANTISIMA CONCEPCION
Facultad de Ingeniería
Departamento de Ingeniería Eléctrica

Profesor Patrocinante:
Dra. Silvia E. Restrepo M.

Caracterización de patrones de conducta diaria de residentes de una casa inteligente

Ignacio Nicolas Cabrera Matus

Informe de Habilitación Profesional
para optar al Título de

Ingeniero Civil Eléctrico

Agosto 2023

Agradecimientos

Como persona creyente, quiero agradecer primero a Dios por acompañarme siempre, por ayudarme a lograr mis metas y apoyarme en los momentos en que mis esfuerzos no bastaban.

A mi familia, familiares por apoyarme de manera incondicional en todos mis años de vida, por siempre buscar lo mejor para mí, por siempre estar atentos a mi desarrollo universitario y ofrecerme toda la ayuda que estaba a su alcance.

A la profesora Silvia Restrepo Medina, quien me acompaña en mi desarrollo académico desde 2020, cuando me permitió realizar la práctica de laboratorio, que ahora se llama Proyecto de Iniciación en Ingeniería Eléctrica. Quiero agradecer todo su compromiso a lo largo de estos casi 4 años, por su empatía, por su comprensión, por siempre incentivarme a dar lo mejor de mí y por supuesto, quiero agradecer todo el conocimiento entregado y el sobre esfuerzo que realizó para siempre resolver todas mis dudas. Le deseo el mayor de los éxitos en su vida y espero nunca cambie.

Al proyecto ANID Programa FONDECYT 11221231 por el financiar la investigación, apoyo que fue sumamente útil para realizar el proyecto.

Agradezco a todos los profesores de la institución por su constante respeto hacia mí y mis compañeros. Especialmente, quiero destacar la dedicación de los profesores del departamento de Ingeniería Civil Eléctrica, quienes han demostrado un fuerte compromiso en la formación de los estudiantes. También agradezco por el valioso conocimiento impartido en diversas áreas de la carrera, ampliando mi comprensión de la Ingeniería Civil Eléctrica.

A mis compañeros, a aquellos que me acompañaron y aun me acompañan en este proceso que ya finaliza si Dios quiere, quiero agradecerles la paciencia y la disposición por ayudarme a lo largo de todos estos años.

A la Universidad Católica de la Santísima Concepción por entregarme todas las herramientas que me permiten estar en esta instancia y por siempre brindarme el apoyo mediante diferentes cursos para sobrellevar la vida universitaria.

A cualquier lector, te agradezco y deseo que este documento se de gran ayuda para tus futuros proyectos.

RESUMEN

La domótica es una nueva tecnología que ayuda en muchos aspectos a las personas en su día a día, por lo cual ya no es una opción tan lejana querer incursionar en esta nueva tecnología. Por ende, en este proyecto se busca examinar, entender y caracterizar patrones de comportamiento de una persona dentro de un ambiente común, como lo es la casa donde una persona reside y pasa en medida, poco o gran parte de su tiempo, para en un futuro poder optimizar esta tecnología que ya está presente en la actualidad.

Se realizó una investigación utilizando una base de datos sobre actividades diarias junto con 6 modelos de aprendizaje automático. Estos modelos, que incluyen K-vecinos más cercanos (KNN), Árbol de decisión (DT), Bosques aleatorios (RF), Maquina de soporte vectorial (SVM), Perceptrón multicapa (MLP) y Maquina de potenciación de gradiente (GBM), fueron entrenados y evaluados con diferentes conjuntos de datos. Cinco de ellos se construyeron siguiendo pasos de segmentación de datos, ajuste de hiperparámetros y pruebas finales para maximizar su rendimiento. El sexto modelo se generó mediante una herramienta del software usado. Se empleó la matriz de confusión para comparar los modelos y analizar los errores. Cuatro métricas (Accuracy, Precision, Recall y F1-score) se calcularon a partir de esta matriz para cuantificar numéricamente los errores en la clasificación de datos.

La GBM demostró el mejor rendimiento entre los modelos evaluados, logrando resultados sobresalientes en las métricas y aproximándose al resultado ideal. El segundo lugar lo ocupó el modelo KNN, a pesar de algunos errores en su matriz de confusión y un rendimiento inferior al GBM. Sin embargo, se mantuvo consistentemente positivo en todas las pruebas. En tercer puesto se ubicó el modelo MLP, con métricas inferiores pero una notable capacidad de generalización de datos. Por otro lado, el RF presentó el peor rendimiento, mostrando resultados decrecientes a medida que se sometía a distintos conjuntos de datos.

Los resultados muestran que el modelo entregado por el propio software fue capaz de generar mejores criterios de clasificación en función de la base de datos en comparación con los modelos manuales, los cuales solo basaron su criterio en los valores de las características de cada clase

Tabla de Contenidos

Lista de Tablas

Lista de Figuras

Abreviaciones

Mayúsculas

A.E : average energy.
B.P.T.T. : back propagation through time.
C.A.R.T. : classification and regression trees.
C.H.A.I.D. : chi-square automatic interaction detection.
C.N.N. : convolutional neural networks.
D.F. : dominant frequency.
D.T. : decision tree.
D.T.E. : decision tree with entropy.
E.S.N. : echo state network.
G.B.D.T. : gradient boosting decision tree.
K.N.N. : k-nearest neighbor.
L.D.A. : linear discriminant analysis.
L.R. : logistic regression.
L.V.Q. : learning vector quantization.
M.A.D : mean absolute deviation.
M.L.P. : multilayer perceptron.
P.I.R. : pasive infrared.
Q.U.E.S.T : quick, unbiased, efficient statistical tree.
R.B.F. : radial basis function.
R.F. : random forest
R.M.S. : root mean square.
R.N.A. : red neuronal artificial.
R.T.R.L. : real time recurrent learning.
S.G.D. : stochastic gradient descent.
S.M.A. : signal magnitude area.
S.O.F.M. : self-organizing feature map.
S.Q.L : structured query language.
S.V.M. : support vector machine

1. Introducción

La domótica está siendo utilizada para apoyar la calidad de vida de las personas, ya sea para encontrar seguridad dentro del hogar, al identificar comportamientos anómalos o actividades sospechosas. O también para ayudar a la salud y bienestar de los residentes, al proponer una detección temprana de problemas de salud, realizando el seguimiento de enfermedades crónicas y la fomentación de estilos de vida saludables.

La rutina de las personas se vuelve la fuente de información fundamental para llevar a cabo cualquier plan de mejora en cualquiera de las áreas descritas previamente. Por lo cual la detección y sobre todo el reconocimiento de las actividades cotidianas de las personas se convierte en el eje principal para establecer la rutina que envuelve a cada integrante de la casa. La detección de una actividad resulta más directa al aplicar la tecnología de diferentes sensores, pero el reconocimiento especifico de la actividad que se está realizando es el verdadero desafío que se debe afrontar. Para llevar a cabo el reconocimiento de actividades es muy práctico realizar una caracterización de patrones de conducta de las personas, por medio de la ciencia de datos y el machine learning, los cuales mediante diferentes algoritmos y procesamientos de datos permiten clasificar (reconocer) las diferentes actividades realizadas. Inclusive algunos algoritmos podrían ser utilizados para predecir las actividades rutinarias.

En el capítulo 2 se establece el marco teórico de la investigación, donde se incluye información sobre diversos modelos de clasificación, sensores y la relación entre la domótica y las actividades de la vida diaria enfocada en el reconocimiento de las diferentes actividades. También se define como se construirá y evaluaran los modelos.

Luego en el capítulo 3 se presentará y establecerá la base de datos con la cual se trabajará durante todo el proyecto, luego se definen los diferentes modelos de aprendizaje supervisado. También se presentarán los métodos para mejorar los modelos en busca de mejores resultados, con el fin de comparar su desempeño y encontrar el mejor método de aprendizaje supervisado que hará el reconocimiento de patrones de conducta diaria.

Finalmente, en el capítulo 4 se realiza el sumario del proyecto y las conclusiones correspondientes en función de los resultados obtenidos en el capítulo 3. Se finaliza el trabajo proponiendo trabajo futuro para mejorar los resultados obtenidos en la investigación.

1.1. Trabajos Previos

1.1.1 Wearable System for Daily Activity Recognition Using Inertial and Pressure Sensors of a Smart Band and Smart Shoes

- Truong, Phuc & You, Sujeong & Sang Hoon, Ji & Jeong, Gu-Min. (2019). Wearable System for Daily Activity Recognition Using Inertial and Pressure Sensors of a Smart Band and Smart Shoes. International Journal of Computers Communications & Control. 14. 726-742. [2].

Un grupo de investigadores se propuso desarrollar un sistema menos invasivo para estudiar la rutina de las personas, aprovechando avances tecnológicos. Diseñaron un sistema de detección de actividades compuesto por una pulsera y 2 zapatos inteligentes, conectados a un celular vía bluetooth. La pulsera midió velocidad y aceleración triaxial de la muñeca, mientras que los zapatos registraron aceleración triaxial y puntos de presión. Voluntarios realizaron nueve actividades que se dividieron en tres grupos temáticos. El grupo 1 estuvo compuesto por actividades relacionadas a los pies, como estar de pie, caminar y correr, el grupo 2 estuvo enfocado en actividades relacionadas a las manos como lo es, cepillarse los dientes, limpieza de ventanas y uso de la computadora, finalmente el grupo 3 se constituyó de un equilibrio de actividades entre movimientos con las manos y pies, en este caso abrir puertas, fregar el piso y llevar libros de un punto a otro.

Los datos recopilados se analizaron y visualizaron, considerando precisión temporal. Se extrajeron características estadísticas, de correlación, frecuencia y magnitud, y se usó el método Relief-F para seleccionar las más importantes.

Se construyeron clasificadores KNN y CART para reconocer las actividades. KNN demostró mayor precisión que CART. El modelo KNN presentó una precisión de 89.1% usando solo datos de la pulsera, 77% con un zapato, y 80% con ambos. La máxima precisión alcanzada fue 90.7%, reconociendo actividades con solo 18 características de las 39 características reconocidas por el modelo, lo que agiliza el proceso.

1.1.2 Human Daily Activity Recognition Performed Using Wearable Inertial Sensors Combined With Deep Learning Algorithms

- C. -T. Yen, J. -X. Liao and Y. -K. Huang, "Human Daily Activity Recognition Performed Using Wearable Inertial Sensors Combined With Deep Learning Algorithms," in IEEE Access, vol. 8, pp. 174105-174114, 2020 [3].

Al igual que en el estudio previo [1], el avance tecnológico motivó a un grupo de investigadores a explorar el comportamiento humano a través de un sistema propio de recolección, procesamiento y análisis de datos. Para llevar a cabo el estudio, se recolectaron datos de 30 voluntarios usando un celular con acelerómetro y giroscopio, mientras realizaban 6 actividades como caminar, subir y bajar escaleras, sentarse, ponerse de pie y acostarse. Los datos se dividieron aleatoriamente en dos grupos: el 70% para entrenar un modelo y el 30% para probar su fiabilidad. Se empleó una red neuronal convolucional (CNN) para extraer automáticamente características y clasificar las actividades.

Después de validar el modelo, se realizó un estudio similar al anterior, pero con 21 participantes de alrededor de 22 años, todos hombres. Utilizaron un hardware diseñado por los investigadores que incluía un sensor inercial de acelerómetro y giroscopio, Raspberry Pi 3 y una fuente de alimentación. Los datos de estas 6 actividades fueron procesados mediante una CNN para reconocimiento de movimientos.

La estructura de la red neuronal incluyó capas convolucionantes, capas de descarte y capas conectadas, finalizando con una capa softmax para clasificación. La precisión del modelo en la fase de prueba fue del 93.77%, demostrando su efectividad en el reconocimiento de actividades humanas en base a datos obtenidos de los participantes.

1.1.3 Smart homes for the elderly dementia sufferers: identification and prediction of abnormal behaviour

- Lotfi, A., Langensiepen, C., Mahmoud, S.M. *et al.* Smart homes for the elderly dementia sufferers: identification and prediction of abnormal behaviour. *J Ambient Intell Human Comput* **3,** 205–218 (2012). [3].

El estudio se propuso investigar la relación entre el comportamiento en un hogar inteligente y posibles enfermedades degenerativas, como la demencia. Para lograr esto, se implementaron sensores

de bajo nivel para monitorear las rutinas de adultos mayores y detectar anomalías que podrían estar relacionadas con alguna enfermedad. Se utilizaron cuatro tipos de sensores: un sensor infrarrojo pasivo (PIR), un sensor de entrada para puertas y ventanas, un sensor de presión para muebles y un sensor de desbordamiento.

Dos de los sensores, el PIR y el de entrada, se colocaron estratégicamente en la casa y se enviaron sus datos a una estación base y luego a una base de datos central. Para interpretar los datos, se emplearon técnicas como árboles de rutinas, n-gramas y rutas de actividad. Los datos binarios se ordenaron en función de la hora de activación y la duración en minutos. Una representación gráfica en dos dimensiones mostró las horas de activación y la duración.

Tres criterios se usaron para analizar visualmente los datos:

1) datos normales en clústeres

2) datos cercanos al centroide del clúster como normales

3) grupos más grandes como normales.

Se exploraron métodos de predicción, como echo state network (ESN), back propagation through time (BPTT) y real time recurrent learning (RTRL), siendo método ESN el preferido.

Se realizó una prueba de tres días para detectar anomalías en la rutina de una persona mayor. Los sensores en el salón y el baño revelaron que deambulaba en horas nocturnas y pasaba más tiempo en el baño, posiblemente debido a un medicamento nuevo que estaba consumiendo. Una vez se procedió a cambiar el medicamento, las anomalías disminuyeron, indicando una mejora en su bienestar.

1.1.4 Combining wearable physiological and inertial sensors with indoor user localization network to enhance activity recognition

- Fiorini, L., Bonaccorsi, M., Betti, S., Esposito, D., & Cavallo, F. R. (2018). Combining wearable physiological and inertial sensors with indoor user localization network to enhance activity recognition. Journal of Ambient Intelligence and Smart Environments, 10(4), 345-357. [4].

El objetivo de la investigación fue construir un sistema de reconocimiento de 8 actividades diarias, sistema que se constituye de 3 capas, el hardware, la comunicación de hardware con el sistema de almacenamiento de los datos, y por último el procesamiento de los datos obtenidos.

El hardware estaba compuesto por sensores portátiles, uno situado en el pecho conformado por un sensor de electrocardiograma, un sensor inercial y un nodo móvil para sensar la ubicación

respecto al entorno y un segundo sensor se ubicó en la parte baja de la espalda, donde solo se consideró un sensor inercial.

La comunicación entre sensores y el sistema de almacenamiento se hizo mediante bluetooth.

El reconocimiento de las actividades se produce mediante diferentes modelos de clasificación; un árbol de decisión, máquina de vectores de soporte y redes neuronales artificiales. Previamente se hizo un preprocesamiento de los datos obtenidos para eliminar el ruido producto de las diferentes mediciones y también para etiquetar los datos.

Como resultado se obtuvo que la precisión de los modelos se veía mermada cuando no se consideraban los datos de ubicación de los participantes (10 en total), esto se veía reflejado en la precisión de cada modelo, siendo la red neuronal el modelo optimo con la mejor precisión, sin dejar de lado los buenos resultados de los otros modelos.

1.1.5 Human Activity Recognition: A Comparison of Machine Learning Approaches

- Sbai, Samir & Kraisakdawat, Anan. (2019). Human Activity Recognition, comparison of machine learning approaches. 10.13140/RG.2.2.26639.94881 [5].

La investigación está enfocada en comparar el rendimiento de diferentes técnicas de machine learning, dentro de las cuales se encuentran Naive Bayes, Máquina de Vectores de Soporte (SVM), K- Vecinos más Cercanos (KNN), Regresión Logística (LR), Descenso de Gradiente Estocástico (SGD), Árbol de Decisión (DT), Árbol de Decisión con Entropía (DTE), Bosques Aleatorios (RF), Árboles de Decisión Potenciados por Gradiente (GBDT) y el algoritmo XGBoost.

Estas técnicas se evaluaron con 3 bases de datos obtenidas de diferentes repositorios, llamadas Pampa2, Mhealth y Swell, las cuales están construida con información extraída de sensores inerciales que se usaron para registrar diversas actividades físicas.

Para evaluar el rendimiento de cada uno de los modelos se usaron diferentes métricas en conjunto con sus matrices de confusión correspondientes, las métricas seleccionadas fueron el accuracy, la precision, el recall y el F1-score, todas estas métricas acompañadas por el tiempo que se demora cada modelo en realizar la predicción.

Al juzgar los modelos según los resultados de sus métricas, se obtuvo que el árbol de decisión con entropía fue el modelo que mejores métricas obtuvo al evaluarlo con las 3 bases de datos por separado. Luego se encontraban como mejores modelos los bosques aleatorios y la regresión logística, pero solo al evaluarlos con los datos de Swell y Mhealth. Por último, se obtiene un buen resultado con

el modelo XGBoost, pero solo con los datos de Swell. Otros de los modelos que mostró buen rendimiento fue el de k-vecinos más cercanos, pero se tardaba demasiado en arrojar los resultados, al igual que el modelo SVM.

1.2. Discusión

La revisión realizada da a conocer las diversas formas de afrontar el desafío de reconocer las actividades realizadas por las personas dentro de su hogar, pero también muestra que, para llevar a cabo una buena investigación, se debe seguir una metodología muy detallada.

La obtención de los datos puede ser desarrollada con una cantidad muy reducida de sensores, ya que con solo un sensor inercial se pueden conseguir los datos esenciales para la investigación [1][2][4][5], o también con la combinación de sensores de presión [3] y apertura de puertas [3]. Al momento de generar nuevos datos, parece una mejor opción limitar la cantidad de actividades en una lista muy acotada y definida [1][2], inclusive agruparlas para buscar una mayor correlación entre ellas (actividades) [1]. Dentro de las opciones para clasificar los datos surge como alternativas más amigables, el algoritmo KNN [1][5] y el método de clasificación árbol de decisión usando entropía [5], los bosques aleatorios [5], dado que el método de CNN [2], está ligado a las redes neuronales (CNN), lo que se presenta como una desventaja en el caso de la próxima investigación debido a desconocimiento específico sobre cómo utilizar el método de redes neuronales convolucionales. También cabe destacar que todos los modelos de aprendizaje con los que se pretende trabajar como el modelo KNN y el árbol de decisión, requiere formar una base de datos adecuada que especifique las variables atributo y variables objetivos, y que también cuenten con un formato normalizado que mantenga los datos entre rangos similares para evitar confusiones al momento de refinar el modelo mediante segmentación de los datos entre entrenamiento y prueba [2].

Por último, se establece como una buena forma de visualizar el rendimiento de los modelos mediante la matriz de confusión y las métricas accuracy, precisión, recall y F1-core.[5].

1.3. Hipótesis de trabajo

Las técnicas de aprendizaje automático supervisado son una buena base para caracterizar los patrones de comportamiento humano en búsqueda de buscar mejorar la domótica y calidad de vida.

1.4. Objetivos

1.4.1 Objetivo General

Examinar y caracterizar patrones de comportamiento de los residentes de una casa inteligente, basado en información recolectada por sensores, para comprender las bases de la domótica y sus beneficios.

1.4.2 Objetivos Específicos

- Obtener información preliminar sobre patrones y algoritmos relacionados con la conducta diaria del ser humano.
- Buscar y seleccionar una base de datos que incluya información recolectada mediante sensores, en consecuencia de no poder trabajar con datos experimentales.
- Implementar algoritmos que permitan caracterizar patrones de conducta humana dentro de la casa inteligente.
- Evaluar los resultados mediante métricas para obtener la eficiencia de los algoritmos.

1.5. Alcances y Limitaciones

En el proyecto se espera tomar una serie de datos ya sea de una simulación, elaboración propia o de algún estudio externo a pequeña escala, relacionar técnicas de reconocimiento de patrones, que permitan descubrir si los datos indican algún patrón especifico que se pueda asociar a una respuesta equivalente a un sistema que se ajuste para facilitar la rutina de la persona dentro de la casa.

Este proyecto está acotado al estudio de técnicas y análisis de datos, no se busca generar un hardware que cumpla con los objetivos mencionados anteriormente.

La opción principal es trabajar con una base de datos externa, como por ejemplo la subsidiaria KAGGLE DATASETS de GOOGLE LLC, la cual es una plataforma que posee una gran cantidad de conjuntos de datos consecuentes de propios desafíos propuestos por la plataforma, donde proponen una serie de problemas con el fin de que diferentes participantes logren solucionarlos usando la ciencia de datos, el análisis predictivo y el machine learning. El análisis de las técnicas será mediante una máquina virtual para facilitar el trabajo desde computadores con menor capacidad de procesamiento de datos, de ser necesario se hará uso de un computador proporcionado por el Laboratorio de Redes y Sistemas Inteligentes de la UCSC.

1.6. Metodología

Se revisan documentos previos con el propósito de establecer un marco teórico sobre enfoques destinados a detectar pautas de comportamiento humano a través de algoritmos.

Para desarrollar los algoritmos, se emplea la plataforma Google Colaboratory, aprovechando su facilidad para la programación en Python y la obtención de diversas representaciones visuales de los resultados generados por el software.

Se procede a crear varios modelos (algoritmos), incluyendo KNN, Árbol de decisión, Bosques aleatorios, SVM y MLP, como parte de la metodología de investigación.

Una base de datos relacionada con las rutinas y comportamientos diarios de individuos se adquiere de KAGGLE. Esta base de datos se emplea para entrenar, evaluar y validar los diversos modelos previamente desarrollados.

Adicionalmente, se incorpora otra herramienta al software de programación con el propósito de generar automáticamente un conjunto de modelos. El objetivo es identificar el modelo que mejor se adapte a la base de datos en uso.

La comparación de resultados se realiza a través de métricas y matrices, con el fin de determinar qué modelos de clasificación resultan más eficaces en el reconocimiento de patrones de comportamiento humano.

2. Marco teórico

2.1. introducción

En el actual capitulo se establecerá la teoría sobre los elementos más relevante y necesaria para llevar a cabo la investigación, por lo cual se definirán algunos modelos de clasificación y sensores. También se analiza brevemente la relación entre lo anterior mencionado y las actividades de la vida cotidiana.

2.2. Clasificadores

La base de las investigaciones sobre la conducta humana corresponde a la obtención de datos que puedan ayudar a encontrar los patrones que caracterizan a las personas tanto individual como colectivamente, y si bien, entre más datos se puedan obtener, más cerca se está de entender al ser humano. De nada sirve una gran base de datos si no se puede ordenar esa información, por lo cual también surge como gran pilar para la investigación el poder darle forma a la información mediante algoritmos clasificadores.

A continuación, se mencionan algunos de los clasificadores que han ayudado a llevar a cabo distintas investigaciones. El resto de los clasificadores se pueden encontrar junto a los siguientes en la referencia [6].

2.2.1 Máquina de soporte vectorial

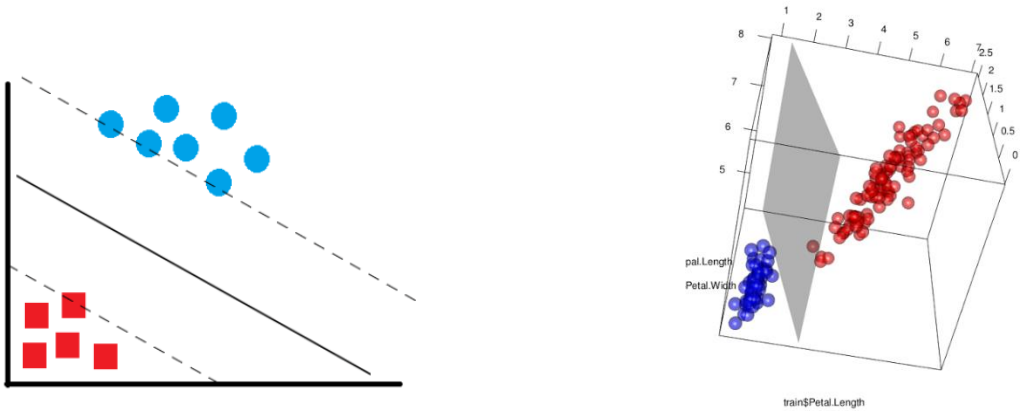
Support Vector Machine (SVM), es un algoritmo de aprendizaje supervisado relacionado con problemas de clasificación y regresión.

Para este modelo los datos son vistos como un vector p-dimensional con p números. Dado un nuevo conjunto de datos, en que cada dato pueda pertenecer a una de dos posibles categorías, un algoritmo basado en SVM, genera un modelo capaz de discernir si un nuevo elemento pertenece a una categoría u otra.

El SVM es un modelo que, partiendo de un conjunto de datos de entrenamiento, busca otorgar una clasificación correcta a los datos, separando lo más posible ambas categorías entre ellas, para que un nuevo dato pueda ser bien referido a la categoría que le corresponde.

La característica principal del SVM es generar una recta o hiperplano que separe ambas categorías, de manera que estas categorías de igual manera queden a una misma distancia hacia la recta, que se espera que sea una distancia considerable y no tan cercana a los datos (Fig.2.1.a).

Existen ocasiones en que el conjunto de datos no puede ser separado por una recta o hiperplano debido a la dispersión de los datos o la existencia de más de 2 categorías, por lo cual existen métodos para suplir la falencia del modelo, y es a través de funciones núcleo o Kernel, que proyecta la información a un espacio de características de mayor dimensión (Fig.2.1. b) el cual aumenta la capacidad computacional de la máquina de aprendizaje lineal.



(a) **(b)**

Fig. 2.1 Representación gráfica SVM. Fuente:[6]

(a) representación 2D ;**(b)** representación 3D.

2.2.2 Algoritmo k-vecinos más cercanos (KNN)

Es un algoritmo de aprendizaje supervisado que a partir de un conjunto de datos bases, los cuales están agrupados en clases, busca clasificar nuevos datos de entrada en una de las clases ya definidas.

El algoritmo clasifica los datos nuevos dependiendo de los k vecinos que tenga más cerca y su distancia con ellos. Para medir la distancia entre el dato nuevo y sus vecinos ya existentes generalmente se usa la distancia euclidiana.

Para crear el modelo de aprendizaje, se particiona el set de datos en un 70-30, 70% de los datos se utilizarán para entrenar el modelo, y el 30% para validar el modelo.

Este método es propenso a caer en el efecto de sobre entrenamiento (overfitting), lo que implica que el algoritmo solo modele los datos de entrenamiento y que no se pueda extrapolar a otro conjunto de datos nuevos.

El modelo es sensible al valor k ya que, si se considera un número muy pequeño, puede clasificar el nuevo dato en un grupo erróneo, y si el valor de k es muy grande, el dato nuevo puede ser apto para clasificarse en cualquiera de las clases definidas.

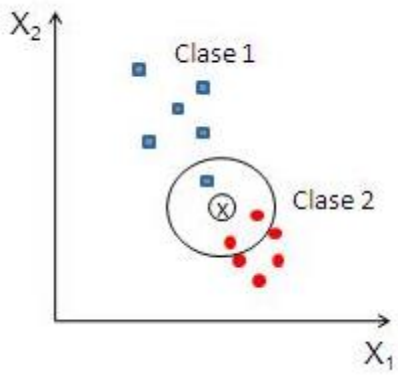


Fig. 2.2 Clasificación KNN. Fuente:[6]

La Fig. 2.2 muestra un ejemplo del algoritmo KNN, donde se presenta una muestra de 12 datos pertenecientes a 2 clases distintas, 6 datos son azules, cuadrados y corresponden a la clase 1, los otros 6 datos son rojos, redondos y corresponden a la clase 2. Para clasificar el nuevo dato "x", el modelo KNN lo clasifica como clase 2 usando los tres vecinos más cercanos (k =3) o clase 1 si k =1

2.2.3 Árbol de decisión o clasificación

En principio se puede describir un árbol de decisión o clasificación como "un algoritmo para clasificar, usando particiones sucesivas [6]''. Este modelo es apropiado cuando existe un conjunto de datos de gran tamaño. Al ser un modelo descriptivo es fácil entender e interpretar las decisiones del modelo, mejorando el análisis de datos que no se pueden detectar con métodos de regresión convencional.

Este modelo está basado en el aprendizaje automático (machine learning), donde desde una base de datos genera diagramas lógicos con objetivo de clasificar los datos. Este modelo también lleva el nombre de segmentación jerárquica, cuya técnica utiliza división secuencial, iterativa y de forma descendente. A partir de una variable dependiente, forma grupos homogéneos con las variables independientes y sus posibles combinaciones.

Se debe construir un criterio de clasificación con la información del grupo al que pertenece cada caso, a partir de una muestra de entrenamiento. El modelo se inicia con un nodo principal que tendrá 2 ramas, en cada rama se considerara una característica y se tomará un conjunto de datos lo más homogéneo posible y se observará si el conjunto tiene una minoría o mayoría de datos que

cumplen con la característica deseada, si el conjunto presenta impurezas, es decir elementos que no pertenezcan a la característica de clasificación correspondiente, se vuelve a generar dos ramas con los datos nuevamente segmentados para analizar si se alcanzó la totalidad de datos bien clasificados o siguen existiendo impurezas, con lo cual se debe seguir iterando y generando ramas, hasta conseguir una clasificación lo más precisa posible.

Los árboles de decisión están compuestos por nodos, ramas y hojas. El nodo son las variables de entrada, las ramas son posibles valores para las variables de entrada y las hojas son posibles valores para las variables de salida. El nodo principal representa la variable de mayor relevancia para el proceso de clasificación.

Los algoritmos de los árboles de decisión presentan modelos complejos que se basan en la evidencia, si los datos presentan alguna incoherencia o ruido, el modelo se ajustará a tales incoherencias perjudicando la precisión en el comportamiento predictivo, a este fenómeno se le conoce como sobre ajuste, para evitar este fenómeno o disminuirlo de alguna manera, es necesario limitar el número de iteraciones y considerar un modelo más general y no tan preciso [6].

Los modelos de árboles de decisión o clasificación más comunes son:

-CHAID (Chi-square Automatic Interaction Detection)
-CART (Classification and Regression Trees)
-QUEST (Quick, Unbiased, Efficient Statistical Tree)

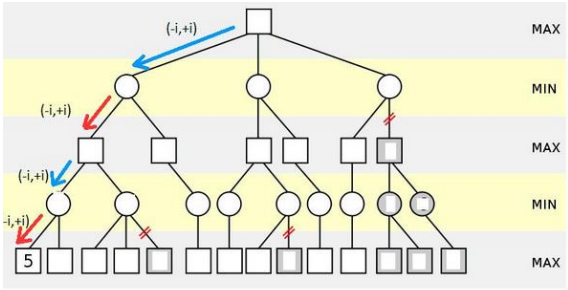


Fig. 2.3 Árbol de Decisión. Fuente:[6]

La Fig.2.3 muestra una de las estructuras más básicas pertenecientes a un árbol de decisión, presentando el orden que se le puede dar a medida que el modelo crece o según la cantidad de características.

2.2.4 Redes neuronales

"*Una Red Neuronal Artificial (RNA) es un modelo matemático inspirado en el comportamiento biológico de las neuronas y en cómo se organizan formando la estructura del cerebro* [6]", la cual se puede observar en la Fig.2.4. Las redes neuronales mediante ensayos repetitivos buscan mejorar la estructura del propio modelo con el fin de lograr una mejor eficiencia predictiva que se espera para los datos con los que se esté trabajando.

El modelo de este algoritmo se compone de nodos que actúan como entradas, salidas o procesadores intermedios, y cada nodo se conecta mediante enlaces con información de estimaciones predictivas. En base a un modelo de aprendizaje, este algoritmo toma una estimación inicial, evalúa el error de predicción y ajusta el modelo para mejorar la dicha predicción, luego se evalúa el nuevo modelo con las nuevas estimaciones predictivas y se vuelven a hacer ajustes en el modelo, este proceso se repite hasta alcanzar un modelo lo más ajustado a las necesidades de quien lo requiera, por lo cual la red puede ser más compleja o simple, dependiendo de cuál sea el caso. Cuando el modelo este ajustado a las necesidades, se procede a evaluarlo con un set de datos de prueba para corroborar su fiabilidad [6].

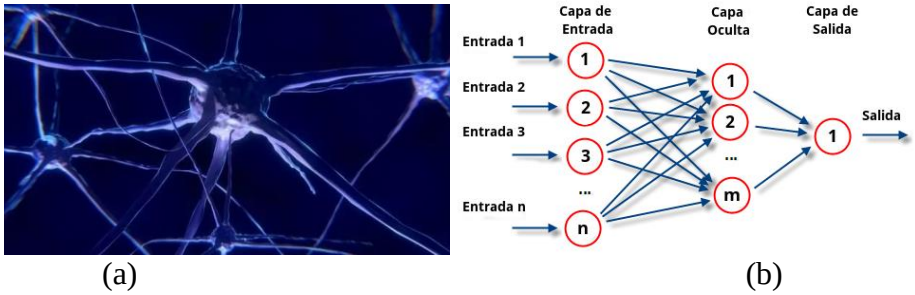


Fig. 2.4 Redes Neuronales

(a) red neuronal humana. Fuente: Fotolia, **(b)** red neuronal artificial. Fuente:[7]

La Fig. 2.4. compara las redes neuronales (a) y (b), donde se puede ver que presentan similitudes en su estructura como lo son los nodos y las ramas, pero con la diferencia de que la primera presenta una infinidad de conexiones más de las que se pueden alcanzar con opción artificial

Se estima que cuatro tipos de arquitecturas componen el 90% de los modelos de redes neuronales, esas arquitecturas son:

-El modelo perceptrón multicapa (MLP)

-Los mapas autoorganizados de kohonen (SOFM)

-El vector de cuantificación (LVQ)

-Redes de base radial (RBF)

Una ventaja es el aprendizaje adaptativo, ya que es un modelo que es capaz de corregirse a sí mismo basándose en sus propios errores, aumentando o disminuyendo la cantidad de enlaces entre cada uno de los nodos, al mismo tiempo que va modificando la información que se trasmite por los enlaces. Otra ventaja es su tolerancia a fallos, dado que es capaz de retener información aun cuando existan daños en la red neuronal, además de considerarse una estructura robusta ante información redundante e imprecisa. La última ventaja está referida a su capacidad de tratar los datos en tiempo real, aumentando la eficiencia en el procesado de datos de manera constante a medida que corrige sus propios errores [6].

2.3. Bosques aleatorios

El bosque aleatorio es un clasificador conformado por un conjunto de clasificadores del tipo árbol de decisión, cada uno de los árboles es entrenado con una muestra finita y equitativa de los datos. A medida que se genera cada árbol, se va disminuyendo el error del bosque aleatorio, ya que el modelo considera el error del árbol predecesor para generar cada nuevo árbol. Los resultados entregados por el bosque aleatorio es el promedio de las respuestas entregadas por cada árbol que conforma este modelo [19].

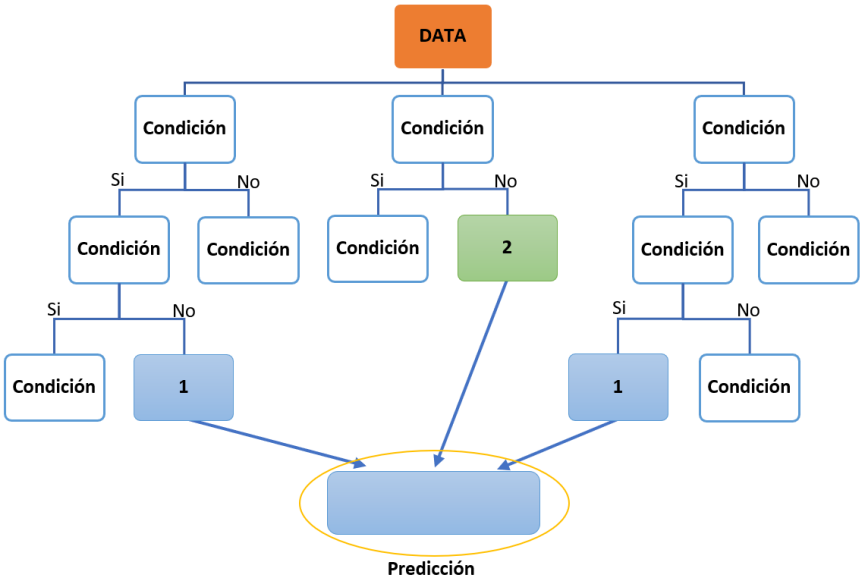


Fig. 2.5 Bosque Aleatorio. Fuente: [autoría propia]

2.4. Actividades de la vida diaria

Si bien la caracterización de patrones de conducta diaria mediante algoritmos es lo que motiva esta investigación, es necesario definir las actividades diarias de las personas que se pretenden

caracterizar, ya que son la fuente de la cual se obtendrán los datos que se utilizan para llevar a cabo la construcción de los diferentes modelos de predicción.

La comprensión de la rutina diaria de las personas supone un gran beneficio al momento de caracterizar actividades, en primer lugar, permite ajustar los modelos según la predominancia de algunas actividades por parte de los residentes de la casa. La información obtenida mediante sensores puede ser acompañada por una variable tiempo que establezca entre que horarios se realiza cada actividad, saber esta información permite que los modelos puedan formar criterios para poder descartar posibles respuestas erróneas y aumentar la probabilidad de que el modelo detecte que actividad se está realizando. Todo lo comentado anteriormente ayuda directamente a mejorar la precisión del modelo, precisión que permitiría tomar mejores decisiones al momento de entregar alguna respuesta en función de la actividad que se detecta, ya sea en el área de la telemedicina o al momento de domotizar una casa.

Hacer un reconocimiento previo y detallado de las actividades se realizan dentro de la casa es importante para poder detectar la existencia de similitudes entre actividades, saber esta información permite adelantarse a un posible problema en el criterio de los modelos al momento de reconocer entre una actividad u otra. Así mismo se puede planear mejores métodos para obtener la información y caracterizar cada actividad de mejor manera, ya sea agregando sensores o mejorando la ubicación de ellos.

En la Fig. 2.6 se puede visualizar actividades cotidianas y las similitudes que pueden existir, por ejemplo, el caso de una persona que se recuesta en un sofá o en su cama, saber a cuál de las 2 situaciones corresponde puede ser complejo si no se tiene suficiente información, otra situación similar que se puede dar es saber si la persona está frente a la computadora o frente a la televisión.

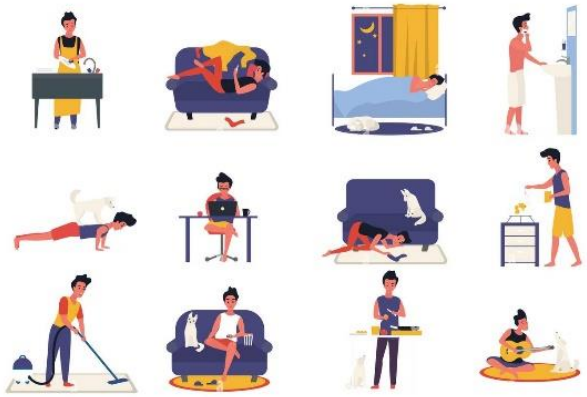


Fig. 2.6 Rutina Diaria. Fuente: [Alamy]

2.5. Sensores

Es una buena opción revisar algunos de los dispositivos sensoriales con los cuales se obtienen los datos en trabajos previos en el reconocimiento de actividades diarias (subir escaleras, sentarse, acostarse o ir desde una habitación a otra), con el fin de tener una mejor idea de cómo tratar la información obtenida sabiendo que no todos los sensores entregan la información de la misma manera.

Por lo cual a continuación se presentan algunos de los sensores que se usaron en trabajos previos.

2.5.1 Sensor inercial

Dispositivo compuesto por acelerómetros, giróscopos y magnetómetros que mide aceleración y velocidad angular, se utiliza en aplicaciones de captura y análisis de movimiento [8].

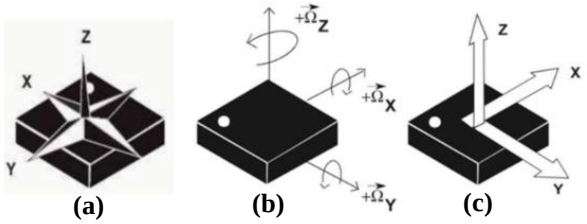


Fig. 2.7 Ejes de Trabajo del Sensor Inercial. Fuente:[9]

(a) magnetómetro, **(b)** giroscopio, **(c)** acelerómetro

2.5.2 Sensor de pulso cardiaco

Es un sensor de contacto fotoeléctrico el cual se puede posicionar en el dedo índice o en el lóbulo de del oído, siendo así capaz de medir las pulsaciones de una persona. El dispositivo emite una luz la cual se refleja en la sangre y retorna al sensor, la variación del pulso y el flujo de sangre permite que la luz reflejada varíe, así mismo varia la señal que el sensor detecta, la cual posteriormente se amplifica y se entrega como una señal analógica que debe ser procesada para hacer una lectura de la información entregada por el sensor [10].

Fig. 2.8 sensor de pulso cardiaco Fuente: [10]

2.6. Google Colaboratory-Python

Colaboratory, o “Colab” para abreviar, es un producto de Google Research [12]. Permite a cualquier persona escribir y ejecutar códigos arbitrarios de Python desde el navegador. Es especialmente útil para tareas de aprendizaje automático, análisis de datos y educación. Colab es un servicio de cuaderno alojado de Jupyter que no requiere instalación y que ofrece acceso sin coste adicional a recursos informáticos incluidas las librerías más usadas, el servicio ofrece, la funcionalidad de una máquina física, con 12 GB de memoria ram y una capacidad de almacenamiento de 107 GB.

Se decidió trabajar con este servicio porque muchos de los algoritmos de clasificación se trabajan con Python, por lo cual se hace mucho más fácil trabajar desde la fuente de información hasta obtener los resultados correspondientes considerando los recursos mencionados en el párrafo anterior.

2.7. Python v.3.11.

Python es un software y lenguaje de programación utilizado para construir softwares, sitios webs o para realizar análisis de datos.

Debido a que se necesita utilizar un software externo al cual Google colab no tiene acceso, se optó por instalar el software Python en un computador, permitiendo la conexión con el software externo. El computador utilizado poseía las siguientes características para poder desarrollar la tarea de crear un nuevo modelo de aprendizaje automático.

- Memoria Ram: 16 gb
- Tarjeta de Video: NVIDIA RTX 3060
- Procesador: i7 11va generación

2.7.1 H2O

H2O es un software basado en Java para el modelado de datos, Python posee un módulo capaz de generar una conexión con el software para acceder a los algoritmos de aprendizaje automático que este puede proporcionar y así generar mediante Python diferentes modelos de aprendizaje automático. H2O posee algoritmos avanzados como boosting and bagging ensembles, pero también posee algoritmos comunes como regresión lineal, regresión logística y random forest [13][14].

2.8. Gridsearch

Es un proceso que busca minuciosamente los mejores hiperparametros de un algoritmo dentro de un conjunto de posibles valores asignados manualmente, los posibles valores se evalúan en el

modelo al mismo tiempo que se realiza una validación cruzada para obtener la mejor puntuación de prueba [15].

2.9. Validación cruzada

Es un método y proceso estadístico para evaluar algoritmos de aprendizaje mediante la división de los datos en 2 conjuntos, un conjunto de entrenamiento y un segundo conjunto de prueba o validación, la segmentación de los datos se repite k veces, buscando siempre variar los datos correspondientes a cada segmento de datos [16].

2.10. Métricas

Las métricas son sistemas de medición que sirven para cuantificar y evaluar el rendimiento de un modelo de aprendizaje automático en tareas de clasificación [17].

2.10.1 Matriz de confusión

La matriz de confusión, la cual se puede observar en la Fig. 2.9., es una herramienta que permite visualizar el desempeño de un algoritmo de aprendizaje supervisado. Cada columna de la matriz representa el número de predicciones de cada clase, mientras que cada fila representa a las instancias en la clase real, la cual permite ver qué tipos de aciertos y errores está teniendo el modelo a la hora de pasar por el proceso de aprendizaje con los datos.

		predicción	
		0	**1**
realidad	**0**	TN	FP
	1	FN	TP

Fig. 2.9 Matriz de confusión. Fuente:[17]

La Fig. 2.9 representa una matriz binaria, es decir una matriz que considera 2 clases, pero que puede ser de mayores dimensiones dependiendo de la cantidad de clases que presente el modelo de clasificación. Las siglas que componen la matriz son TN, FP, FN y TP.

True Negative (TN) o Verdaderos Negativos que son las instancias predichas como clase 0 y que en realidad pertenecen a esa clase.

False Positive (FP) o Falsos Positivos son las instancias predichas como clase 1 pero que en realidad pertenecen a la clase 0.

False Negative (FN) o Falsos Negativos son las instancias predichas como clase 0 pero que en

realidad pertenecen a la clase 1.

True Positive (TP) o Verdaderos Positivos son las instancias predichas como clase 1 y que en realidad pertenecen a dicha clase.

No importa que una clase se designe como positiva o negativa, ya que esto solo las diferencia y no representa precisamente que una clase sea mejor que la otra, solo se debe considerar que sean verdaderas, ya que representa que son predicciones correctas.

2.10.2 Precision

La métrica de precision es utilizada para saber qué porcentaje de valores que se han clasificado como positivos son realmente positivos.

$$Precision = \frac{verdaderos\ positivos}{verdaderos\ positivos + falso\ positivos} \tag{2.1}$$

2.10.3 Recall

La métrica de recall, también conocida como la proporción de verdaderos positivos, es utilizada para saber cuántos valores positivos son correctamente clasificados.

$$Recall = \frac{verdaderos\ positivos}{verdaderos\ positivos + falsos\ negativos} \tag{2.2}$$

2.10.4 F1

Esta es una métrica muy utilizada en problemas en los que el conjunto de datos a analizar está desbalanceado. Esta métrica combina la métrica de precision y recall, para obtener un valor mucho más objetivo.

$$F1\ score = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} \tag{2.3}$$

2.10.5 Accuracy

La métrica accuracy representa el porcentaje total de valores correctamente clasificados, tanto positivos como negativos.

$$Accuracy = \frac{verdaderos\ positivos + verdaderos\ negativos}{verdaderos\ positivos + falsos\ positivos + verdaderos\ negativos + falsos\ negativos} \tag{2.4}$$

3. Caracterización de patrones de comportamiento humano

3.1. Introducción

Los modelos de aprendizaje supervisado son el primer acercamiento para poder reconocer los patrones de conducta diaria, por lo cual a continuación se establecerá el proceso y las herramientas necesarias para poder llevar a cabo la investigación, tanto para poder construir los modelos de aprendizaje supervisado, como para poder evaluar los resultados futuros. Se analizarán los resultados de los diferentes modelos seleccionados en función de la base de datos con la cual se trabajará en la investigación.

3.2. Base de datos.

Como se mencionó en el apartado 1.5, se optó por trabajar con las técnicas de reconocimiento de patrones usando una base de datos externa de KAGGLE DATASETS llamada MHEALTH Dataset [11].

La base de datos usada consiste en un estudio sobre 10 sujetos, a los cuales se les pidió realizar 12 actividades diferentes durante un tiempo determinado, dotados de un conjunto de sensores ubicados en el pecho, las muñecas y tobillos. La base de datos consiste en 24 columnas de información (características), 23 de las columnas tienen los valores en el eje x, y, z de cada sensor inercial además del valor de cada sensor de electrocardiograma y la columna número 24 corresponde a la clase que se le asigna a cada conjunto de datos representada por cada fila (ver Tabla 3.1).

Para el primer acercamiento a los modelos de aprendizaje supervisado, se considerarán los datos de solo un sujeto. El número de filas que componen la base de datos de este sujeto es de 161.280, donde el 78% de las filas tiene como salida la actividad número 0, también llamada clase número 0, el resto de las filas se divide equitativamente entre las otras actividades, teniendo cada una aproximadamente un 2% de la totalidad de los datos, exceptuando la actividad número 12 que posee el 1% de la totalidad de los datos como se puede observar en la Tabla 3.2.

Tabla 3.1 Variables de la Base de Datos

Número de columna	Sensor	Descripción
1	acx	Aceleración del sensor de pecho (eje X)
2	acy	Aceleración del sensor de pecho (eje Y)
3	acz	Aceleración del sensor de pecho (eje Z)
4	Ec1	Señal de electrocardiograma (derivación 1)
5	Ec2	Señal de electrocardiograma (derivación 2)
6	alx	Aceleración del sensor del tobillo izquierdo (eje X)
7	aly	Aceleración del sensor del tobillo izquierdo (eje Y)
8	alz	Aceleración del sensor del tobillo izquierdo (eje Z)
9	glx	Giroscopio del sensor del tobillo izquierdo (eje X)
10	gly	Giroscopio del sensor del tobillo izquierdo (eje Y)
11	glz	Giroscopio del sensor del tobillo izquierdo (eje Z)
12	mlx	Magnetómetro del sensor del tobillo izquierdo (eje X)
13	mly	Magnetómetro del sensor del tobillo izquierdo (eje Y)
14	mlz	Magnetómetro del sensor del tobillo izquierdo (eje Z)
15	arx	Aceleración del sensor del antebrazo derecho (eje X)
16	ary	Aceleración del sensor del antebrazo derecho (eje Y)
17	arz	Aceleración del sensor del antebrazo derecho (eje Z)
18	grx	Giroscopio del sensor del antebrazo derecho (eje X)
19	gry	Giroscopio del sensor del antebrazo derecho (eje Y)
20	grz	Giroscopio del sensor del antebrazo derecho (eje Z)
21	mrx	Magnetómetro del sensor del antebrazo derecho (eje X)
22	mry	Magnetómetro del sensor del antebrazo derecho (eje Y)
23	mrz	Magnetómetro del sensor del antebrazo derecho (eje Z)

Tabla 3.2 Base de Datos

Actividades/Clases	Numero asignado	Numero de filas de cada clase	Porcentaje real normalizado	Porcentaje aproximado
Hacer nada	0	126106	0.781	78%
De pie	1	3072	0.019	2%
Sentarse y relajarse	2	3072	0.019	2%
Acostado	3	3072	0.019	2%
Caminar	4	3072	0.019	2%
Subir escaleras	5	3072	0.019	2%
Inclinarse desde la cintura	6	3072	0.019	2%
Elevar los brazos de manera frontal	7	3072	0.019	2%
Doblar las rodillas	8	3379	0.021	2%
Ciclismo	9	3072	0.019	2%
Trotar	10	3072	0.019	2%
Correr	11	3072	0.019	2%
Saltar adelante y atrás	12	1075	0.007	1%
TOTAL	----------	161280	1	100%

Se seleccionaron 12 columnas de datos, las cuales se muestran en la Tabla 3.3 en una forma más reducida, con el fin de reducir el tiempo de entrenamiento y complejidad del modelo.

Tabla 3.3 Variables Seleccionadas

Número de columna	abreviación de sensor	descripción
6	alx	Aceleración del sensor del tobillo izquierdo (eje X)
7	aly	Aceleración del sensor del tobillo izquierdo (eje Y)
8	alz	Aceleración del sensor del tobillo izquierdo (eje Z)
9	glx	Giroscopio del sensor del tobillo izquierdo (eje X)
10	gly	Giroscopio del sensor del tobillo izquierdo (eje Y)
11	glz	Giroscopio del sensor del tobillo izquierdo (eje Z)
15	arx	Aceleración del sensor del antebrazo derecho (eje X)
16	ary	Aceleración del sensor del antebrazo derecho (eje Y)
17	arz	Aceleración del sensor del antebrazo derecho (eje Z)
18	grx	Giroscopio del sensor del antebrazo derecho (eje X)
19	gry	Giroscopio del sensor del antebrazo derecho (eje Y)
20	grz	Giroscopio del sensor del antebrazo derecho (eje Z)

El motivo de seleccionar solo 12 columnas es para disminuir la carga de datos en el software con el cual se hará el análisis de datos y patrones de conducta humana, por lo cual se consideró suficiente los valores correspondientes al acelerómetro y giroscopio de los sensores ubicados en el tobillo izquierdo y el antebrazo derecho.

3.3. Preprocesamiento

Debido al desbalance significativo entre la clase “Hacer nada” y el resto de las clases, se busca abordar este problema aplicando diversas estrategias en los modelos de clasificación. Inicialmente, se decide realizar una reasignación de la base de datos, definiendo solo 2 clases. La primera clase, denominada clase 0 (Hacer nada), se mantiene sin aplicar alguna modificación. Luego, se crea una nueva clase, la clase 1, que se formará al combinar los datos de las demás actividades presentes en la base de datos. Esta nueva clase representa una actividad en general, en contraste con la clase 0 que denota la inactividad, es decir, el acto de no realizar ningún movimiento. Por lo cual se ha decidido hacer un análisis de uno vs todos (clase 0 vs nueva clase 1), para observar la respuesta de los modelos ante una base de datos más balanceada, buscando evitar que se produzca un sobre entrenamiento en función de la clase mayoritaria.

Tabla 3.4 Distribución de clases.

Clase	N° de Datos	Proporción	N° de datos de entrenamiento	N° de Datos de Prueba
0	707.122	72%		
1	275151	28%	687.591	294.682
Total	982.273	100%		

Como se puede observar en la Tabla 3.4., la clase 1 solo corresponde a un 28% de la totalidad de los datos y si comparamos ambas clases, la clase 1 es aproximadamente un poco menos de una tercera parte de la totalidad de los datos correspondientes a la clase 0.

La distribución de los datos dentro del conjunto de entrenamiento es de 494.558 datos correspondientes a la clase 0 y 193.033 datos a la clase 1.

Después de evaluar la nueva base de datos en los modelos de aprendizaje y debido a que aún es notable el desbalance entre ambas clases, se aplicarán 2 técnicas para balancear la nueva base de datos, con el fin de realizar nuevas pruebas y comparar los resultados obtenidos. Las técnicas que se aplicarán son el submuestreo y el sobremuestreo.

- **Submuestreo o Undersamplig.**

Este método consiste en eliminar de manera aleatoria datos de la clase mayoritaria hasta alcanzar un número de muestras igual o cercana a la o las clases minoritarias [18]. En este caso la clase mayoritaria disminuirá de 494.558 a 193.033 datos, esperando conseguir una menor cantidad de falsos negativos y positivos al mismo tiempo que aumentan los verdaderos negativos y positivos, con el fin de conseguir el menor error posible al momento de clasificar las instancias.

- **Sobremuestreo u Oversampling.**

El sobre muestreo es una técnica que permite duplicar el valor de las muestras minoritarias con el fin de alcanzar el número de datos de la clase mayoritaria y crear una base de datos más balanceada [18]. En el caso de estas pruebas las clases minoritarias pasaran de tener 193.033 a 494.558 datos. Al igual que en el caso del submuestreo, se espera conseguir una baja en los falsos negativos y positivos, al mismo tiempo que aumentes los verdaderos negativos y positivos.

A continuación, en la Fig. 3.1 se presenta el diagrama correspondiente a la estructura que poseen los modelos para llevar a cabo el proceso de clasificación.

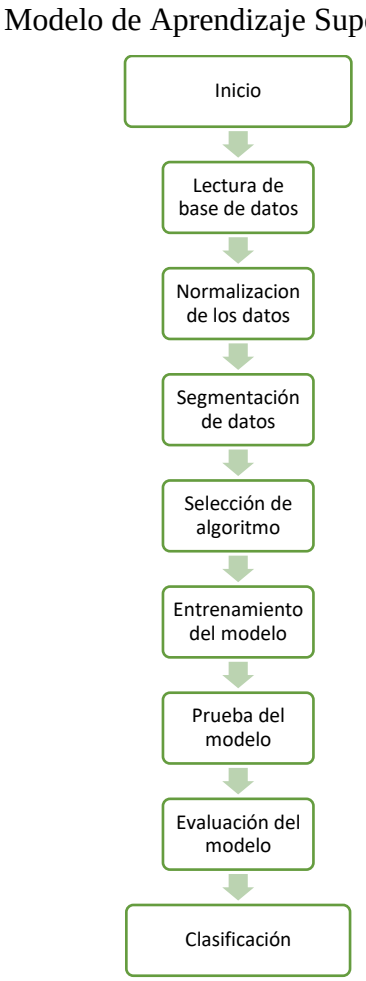


Fig. 3.1 Diagrama de bloques de clasificadores. Fuente: [autoría propia]

Tanto el algoritmo de árbol de decisión como el algoritmo KNN comparten una serie de pasos en su proceso de análisis de patrones. Estos pasos incluyen la normalización de los datos para evitar problemas con la clasificación, la carga de la base de datos para entrenamiento, la verificación de la correcta lectura de los datos y la segmentación de la base de datos en un 70% para entrenamiento y un 30% para pruebas.

Una vez realizada la segmentación, se eligen los algoritmos a trabajar, en este caso, el algoritmo de árbol de decisión y el KNN. Junto con esta elección, se definen las constantes necesarias para cada algoritmo, como el número de vecinos para KNN y la profundidad máxima de los nodos para el árbol de decisión respectivamente. Para el árbol de decisión, se selecciona una profundidad de 20 nodos para evitar un excesivo tiempo de procesamiento debido a la cantidad de datos, mientras que

para KNN se opta por 30 vecinos cercanos, basándose en la cantidad de datos procesados.

Es importante destacar que la segmentación de datos se realiza de manera aleatoria para construir los conjuntos de entrenamiento y prueba.

Como se mencionó con anterioridad, el siguiente paso es entrenar los respectivos algoritmos con el set de datos de entrenamiento para luego evaluarlos con el set de prueba respectivo a cada algoritmo de manera conjunta con una posterior validación cruzada.

Para evaluar los algoritmos, se construye la matriz de confusión y a partir de ella se evalúan las métricas mencionadas anteriormente, considerando estas métricas se pueden obtener los primeros indicios de cual algoritmo es mejor para ciertas condiciones.

Finalmente, tras evaluar los algoritmos, se puede dar por construidos los modelos, por lo cual se pueden ingresar nuevos datos para que sean clasificados, lo cual se pretende con esta investigación.

3.4. Resultados de uno vs todos.

3.4.1 Clasificador Árbol de decisión.

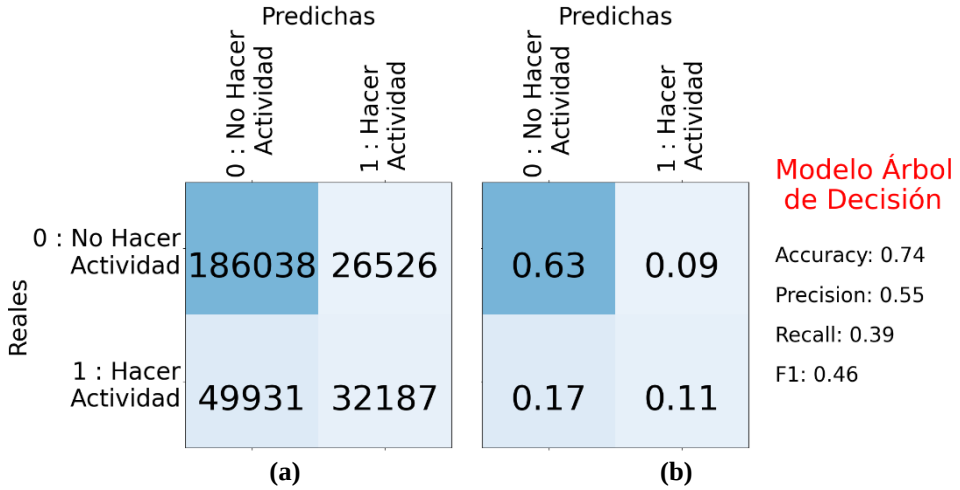


Fig. 3.2 Matriz de confusión binaria, árbol de decisión sin modificación. Fuente: [autoría propia]

(a) Matriz sin normalización **(b)** Matriz normalizada

En la Fig. 3.2, se ve la clasificación binaria de la nueva base de datos, con 76.457 datos predichos erróneamente (26% del total). La mayoría de las clasificaciones correctas corresponden a la clase 0 debido a su predominancia en los datos. Los errores están sesgados hacia falsos negativos (23,405 más que falsos positivos). Esta distribución desbalanceada afecta alguna de las métricas, el "recall" es el más afectado debido a la gran cantidad de datos que son falsos negativos, datos que incluso superan en un 55% a la cantidad de verdaderos positivos.

3.4.2 Clasificador K Nearest Neighbors (KNN).

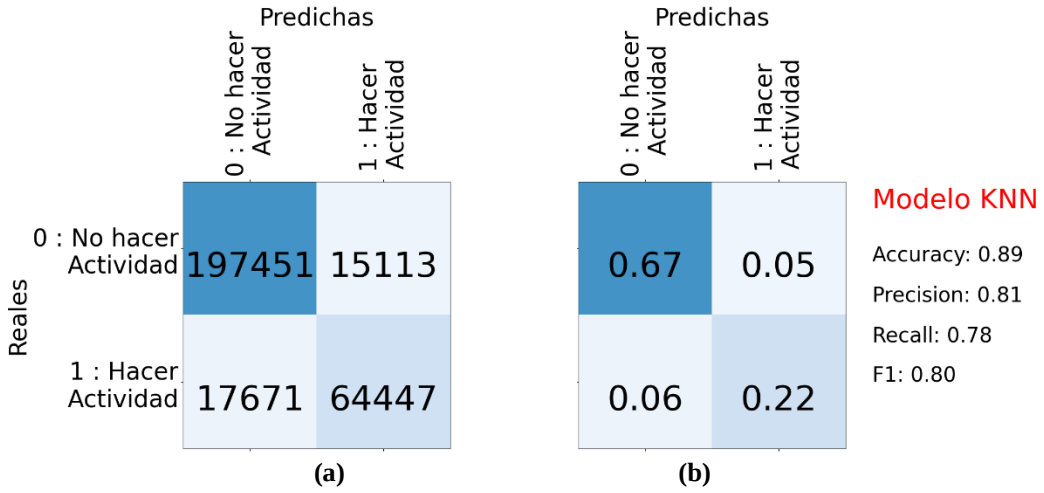


Fig. 3.3 Matriz de confusión, algoritmo KNN sin modificación. Fuente: [autoría propia]

(a) Matriz sin normalización **(b)** Matriz normalizada

En la **Figura 3.3** se puede observar que tanto los verdaderos positivos como los verdaderos negativos son mayores que en la **Figura 3.2** y que los datos clasificados erróneamente son menores en comparación con la misma figura, el error en este caso es de aproximadamente 11%, lo cual demuestra que el modelo es menos propenso a errar cuando la base de datos es desbalanceada, inclusive en este resultado se observa que los errores están más balanceados, escapando de los comportamientos que tenían los anteriores modelos respecto a una mayor cantidad de falsos negativos en comparación con los falsos positivos.

3.4.3 Clasificador KNN con Submuestreo.

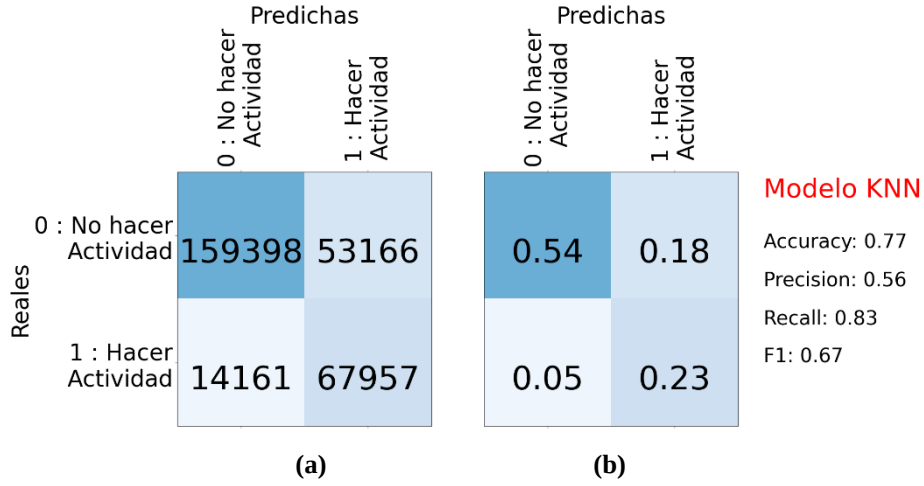


Fig. 3.4 Matriz de confusión binaria Normalizada, KNN con submuestreo. Fuente: [autoría propia]

(a) Matriz sin normalización **(b)** Matriz normalizada

En la Figura 3.4, reducir los datos de la clase 0 resultó en un aumento de errores en las predicciones de la clase 1 en comparación con la Figura 3.3. Esto generó más falsos positivos, con una proporción desfavorable en comparación con la reducción de falsos negativos. Los falsos negativos se redujeron en un 19.9%, mientras que los falsos positivos aumentaron en un 251.8%. En cuanto a los verdaderos negativos y positivos, los verdaderos positivos aumentaron un 5.4%, pero los negativos disminuyeron en un 19.27% en relación con la Figura 3.3. Estos cambios resultaron en una disminución general de las métricas, indicando un empeoramiento en la capacidad del modelo para clasificar correctamente.

3.4.4 Clasificador Árbol de Decisión con Submuestreo.

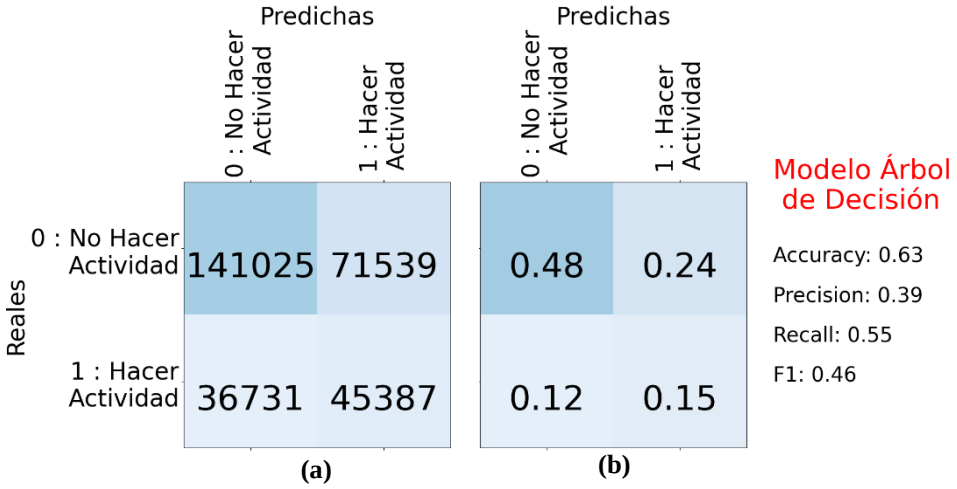


Fig. 3.5 Matriz de confusión binaria, Árbol de decisión con submuestreo. Fuente: [autoría propia]

(a) Matriz sin normalización **(b)** Matriz normalizada

Los resultados de la Figura 3.5 muestran un cambio significativo respecto a la Figura 3.2, el submuestreo redujo los falsos negativos en alrededor del 26%, una mejora respecto a la Figura 3.4, pero afectó a los verdaderos negativos, disminuyendo un 24.2% en comparación con la Figura 3.2. Reducir los errores en la clase 0 resultó en un aumento notable en la cantidad de datos presentes en la columna de la clase 1 de las matrices, con un incremento del 169.7% de los falsos positivos. Solo una métrica se benefició con el submuestreo, el recall aumentó de 0.39 a 0.55 en comparación con la Figura 3.2.

3.4.5 Clasificador KNN con Sobremuestreo.

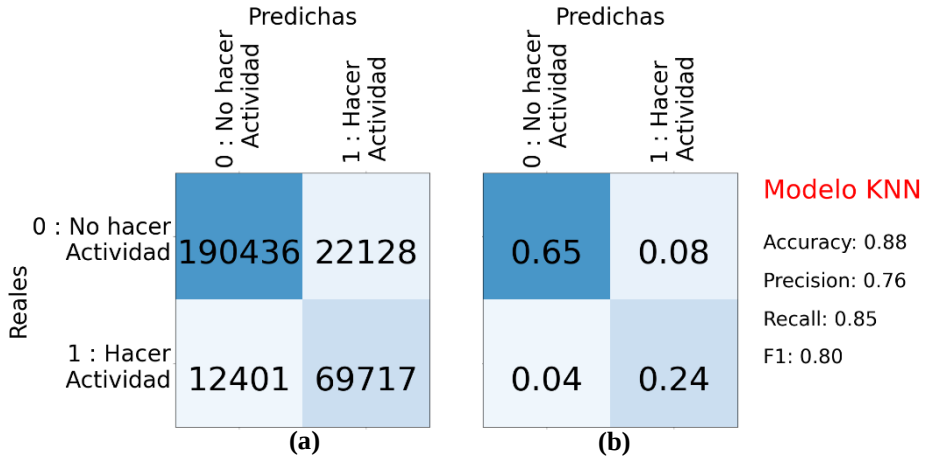


Fig. 3.6 Matriz de confusión binaria Normal, KNN con sobremuestreo. Fuente: [autoría propia]

(b) Matriz sin normalización **(b)** Matriz normalizada

Se observa en Fig. 3.6 que los valores asociados a las predicciones de la clase 0 disminuyeron y las predicciones asociadas a la clase 1 aumentaron respecto a la Fig. 3.3. En la columna 0, los verdaderos negativos disminuyeron un 3.5% y los falsos negativos un 28.2%. En el caso de la columna 1, los falsos positivos aumentaron un 46 % y los verdaderos positivos un 7.7%. El número de datos clasificados correctamente sigue favoreciendo a la clase 0, pero con teniendo una menor cantidad de datos erróneos asociados a la misma clase, la cual es menor a la cantidad de datos erróneos en las predicciones asociadas a la clase 1. Por ultimo las métricas se ven afectadas, siendo la precisión la única que disminuyó su valor de 0.81 a 0.76 respecto a la Fig. 3.3.

3.4.6 Clasificador Árbol de Decisión con Sobremuestreo.

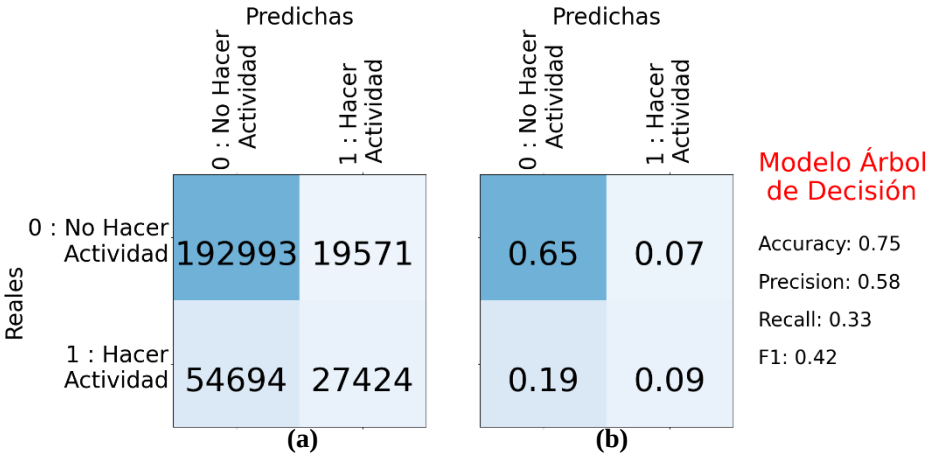


Fig. 3.7 Matriz de confusión binaria, Árbol de decisión con sobremuestreo. Fuente: [autoría propia]

(a) Matriz sin normalización **(b)** Matriz normalizada

La Fig. 3.7. refleja peores resultados en comparación con lo obtenido en la Fig. 3.2, pero mejores respecto a la Fig. 3.5, ya que se obtuvieron menos errores asociados a la clase 1 clase aunque se obtuvo mayor cantidad de falsos negativos, existiendo un 58.3% más de errores asociados a la clase 0. Los verdaderos positivos también disminuyeron respecto a la Fig. 3.2, de 0.11% a 0.09% de la totalidad de los datos con los que se evaluó el modelo, mientras que los verdaderos negativos solo aumentaron de 0.63% a 0.65%. El desempeño regular del modelo se ve reflejado en las métricas, las cuales mejoraron respecto a los otros modelos (Accuracy, Precision) y empeoraron respecto a los mismos (Recall, F1).

Tabla 3.5 Resumen de resultados.

Modelo	Accuracy	Precision	Recall	F1
Árbol de Decisión	0.74	0.55	0.39	0.46
KNN	0.89	0.81	0.78	0.80
KNN con Submuestreo	0.77	0.56	0.83	0.67
Árbol de Decisión con Submuestreo	0.63	0.39	0.55	0.46
KNN con Sobremuestreo	0.88	0.76	0.85	0.80
Árbol de Decisión con Sobremuestreo	0.75	0.58	0.33	0.42

Al examinar las métricas en la Tabla 3.5, se destaca que el KNN con sobremuestreo es el modelo de mejor rendimiento, superando a los otros, especialmente a los modelos de árbol de decisión que son los menos efectivos, siendo el árbol de decisión con submuestreo el menos favorable. Entre los modelos KNN, el de sobremuestreo es el más eficiente como se mencionó anteriormente, mientras que el de submuestreo es el peor. El submuestreo no beneficia a los modelos (KNN y Árbol de Decisión), pero el sobremuestreo sí. Es necesario recordar que estos resultados son válidos para una base de datos binaria con ajuste en cantidades de datos en clases mayoritarias y minoritarias, pero no reflejan la complejidad que surgiría al considerar más clases con menos datos representativos, dificultando la generalización precisa de los modelos para cada clase.

3.5. Base de datos balanceada.

Luego de realizar las pruebas en los modelos, se determinó que los datos referentes a la clase 0 no eran adecuados para entrenar y evaluar los modelos, debido a que pueden provocar que el modelo no considere de manera adecuada a las clases minoritarias, por lo cual se procedió a tomar la base de datos original y generar una nueva en la cual solo se considerará 8 actividades y se elimina por completo la clase 0, dejando así la base de datos prácticamente balanceada. A partir de la información de los 10 sujetos que contiene la base de datos, se consideraron los primeros 9 sujetos para

entrenamiento y prueba (Ver Tabla 3.6) y el décimo sujeto se consideró para realizar la validación del modelo, el cual puede tener un rango de datos diferente y evidenciar la generalización de los modelos. También se optó por considerar todas las características disponibles (23 columnas de características).

Una vez generada la nueva base de datos, se procede a evaluar en 5 modelos de clasificación, sumándole a los modelos KNN y Árbol de decisión otros 3 modelos, los cuales corresponden a los modelos Random Forest, Support Vector Machine y Multilayer Perceptron.

Tabla 3.6 Tabla de clases de nueva base de datos.

N° Asignado	Actividades/Clases	N° de Datos	N° de Datos de Entrenamiento	N° de Datos de Prueba	N° de Datos Validación
1	De Pie	27.648			
2	Sentado	27.648			
3	Acostado	27.648			
4	Caminar	27.648			
5	Subir escaleras	27.648	152.070	65.173	23.450
6	Inclinarse hacia adelante	25.857			
7	Elevar los brazos	26.676			
8	Doblar rodillas	26.470			
Total		217.243			

3.5.1 Resultados de entrenamiento de base de datos balanceada.

Los modelos fueron evaluados considerando sus valores predeterminados y usando la técnica de gridsearch para obtenerlos mejores hiperparametros para cada modelo.

Los resultados obtenidos para cada modelo se muestran a continuación, donde las matrices de confusión están organizadas de tal manera que la primera columna corresponde a las matrices de confusión luego de evaluar los modelos con los conjuntos de entrenamiento, prueba y validación, y la segunda columna corresponde a sus versiones normalizadas.

- **Modelo KNN sin gridsearch**

  El modelo tiene como hiperparámetros de interés el número de k-vecinos, cuyo valor por defecto es de 5 vecinos.

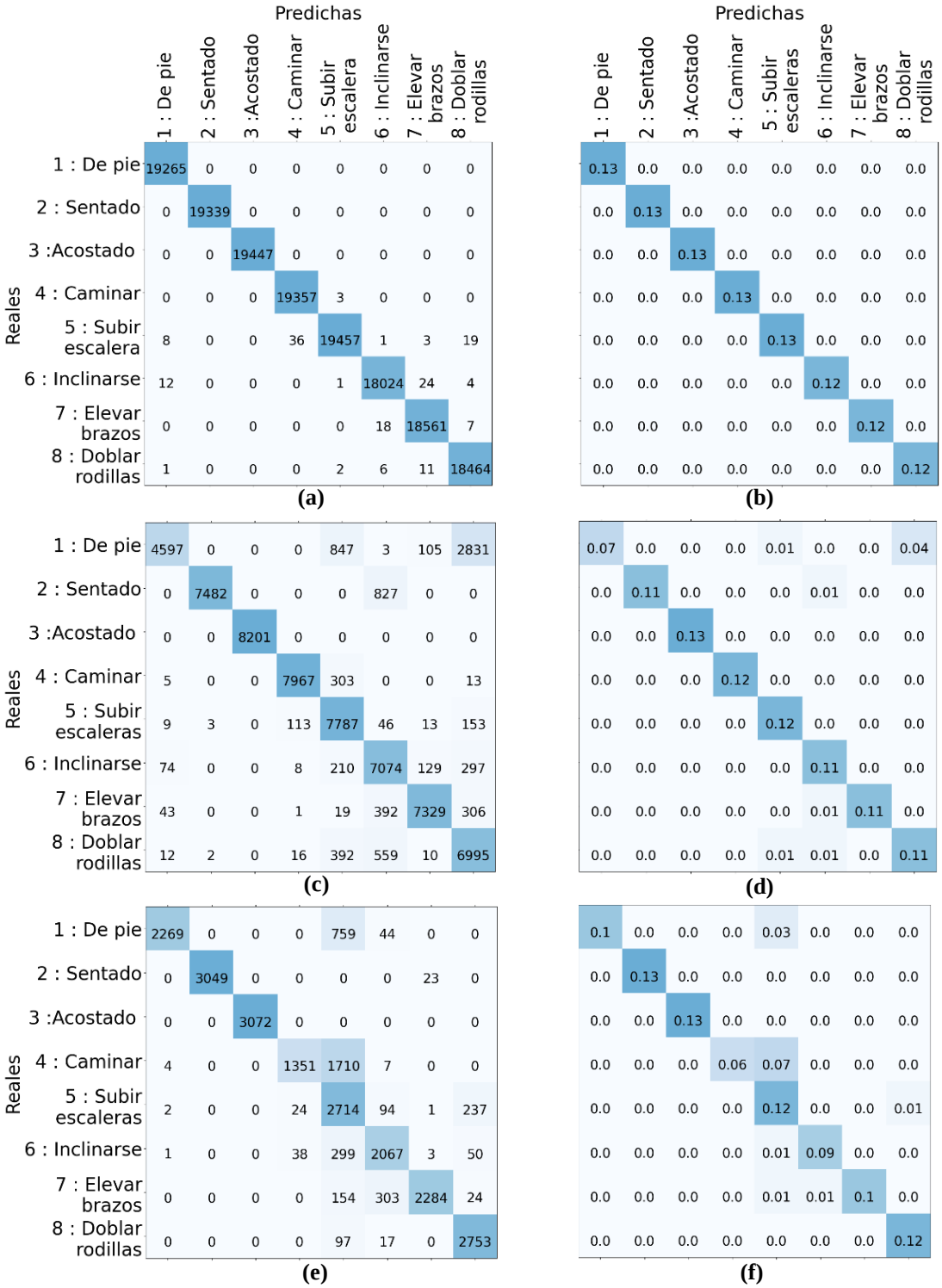


Fig. 3.8 Matriz de confusión modelo KNN sin gridsearch, Fuente: [autoría propia].

(a), (b) Entrenamiento (c),(d) Prueba (e),(f) Validación.

Tabla 3.7 Métricas de Algoritmo KNN sin Gridsearch.

KNN	Entrena	Prueba	Validación
Accuracy	1	0.88	0.84
Precision	1	0.90	0.89
Recall	1	0.88	0.84
F1-score	1	0.88	0.85

En las Fig. 3.8a y Fig. 3.8b se pueden observar los resultados de evaluar el modelo usando los mismos datos con los cuales fue entrenado el modelo, obteniendo así una primera visualización del error presente en el modelo. En este caso el error es muy bajo que solo es perceptible en la matriz de confusión no normalizada y que representa un número de datos erróneos que no son significativos para disminuir la capacidad de clasificaciones correctas. Al contar con un valor tan bajo de errores como se visualiza en las métricas de entrenamiento de la Tabla 3.7, puede significar que el modelo está sobre ajustado a los datos con los que fue entrenado y eso puede provocar que el modelo se vea afectado negativamente al momento de clasificar instancias que son desconocidas para el modelo al no estar en el proceso de entrenamiento del clasificador.

Para corroborar lo explicado anteriormente se presentan las matrices de la Fig. 3.8c y Fig. 3.8d en conjunto con las métricas de prueba visibles en la Tabla 3.7. Se puede observar que las métricas disminuyeron levemente, siendo menos afectada la precisión, la cual disminuyó solo un 10% de su valor mientras que el resto de las métricas tuvieron una disminución de 12% en su valor respecto a las métricas de entrenamiento, mostrando un resultado más acorde a la eficiencia real del modelo.

En la Tabla 3.7 se observa que el valor de las métricas al evaluarlo con los datos de validación, no disminuyeron más de 0.04 su valor, lo que quiere decir que el modelo fue capaz de identificar la mayoría de los nuevos datos, aun cuando estos podían tener un rango más amplio de valores, los cuales pueden confundir al modelo con otras clases.

- **Modelo KNN con gridsearch**

El proceso de gridsearch arrojó que el número recomendado de vecinos para el modelo es de solamente 1 vecino. Esto se debe a la cantidad de los datos de entrenamiento y su distribución.

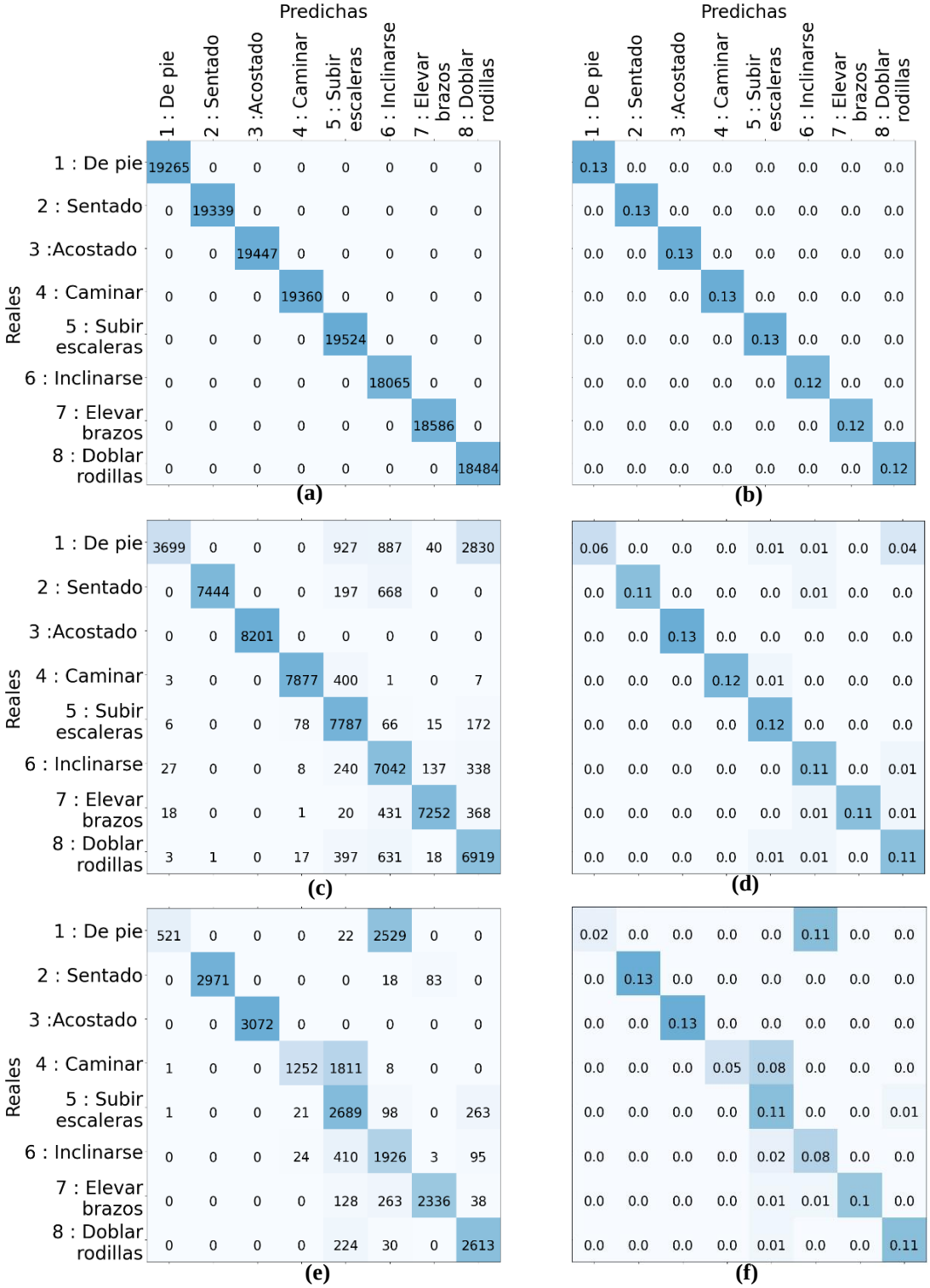


Fig. 3.9 Matriz de confusión modelo KNN con gridsearch. Fuente: [autoría propia].

(a), (b) Entrenamiento (c),(d) Prueba (e),(f) Validación.

Tabla 3.8 Métricas de Algoritmo KNN con Gridsearch

	Entrenamiento	Prueba	Validación
Accuracy	1	0.86	0.74
Precision	1	0.89	0.84
Recall	1	0.86	0.74
F1	1	0.86	0.73

En la Tabla 3.8 se puede observar que las métricas de entrenamiento poseen el mismo valor que las obtenidas en la Tabla 3.7 del modelo sin gridsearch, pero con la diferencia de que en las matrices de entrenamiento de la Fig. 3.9 no se presentan falsos negativos ni falsos positivos. Al igual que el caso anterior, no es recomendable formar una idea en base a los resultados observados, siendo así necesario evaluar nuevamente el modelo con el conjunto de prueba, cuyos resultados se observan en la Tabla 3.8. Aca se puede visualizar que los valores de las métricas son iguales a los arrojados por el modelo KNN sin gridsearch, pero con una mayor cantidad de instancias mal clasificadas(ver Fig. 3.9c),teniendo el modelo con gridsearch 8.014 datos mal clasificados y el modelo sin gridsearch un poco menos con 7.802 datos mal clasificados. Finalmente los resultados de validación reflejan que el modelo se generalizó bastante bien con los datos de entrenamiento, por lo cual no tuvo una caída significativa en el valor de sus metricas como se puede observar en la Tabla 3.8, donde la precision fue la que se vio menos afectada, disminuyendo aproximadamente un 2% en comparacion con el valor de las metricas de prueba, mientras que las otras métricas solo disminuyeron alrededor de un 6% en el caso de F1 y un 7% en el caso del accuracy y recall, comparados tambien con el conjunto de prueba.

- **Modelo Árbol de Decisión sin gridsearch**

El modelo tiene como hiperparámetros de interés la profundidad del árbol, cuyo valor por defecto no está determinado y el árbol se expande hasta que las hojas poseen un valor bajo de datos para dividir.

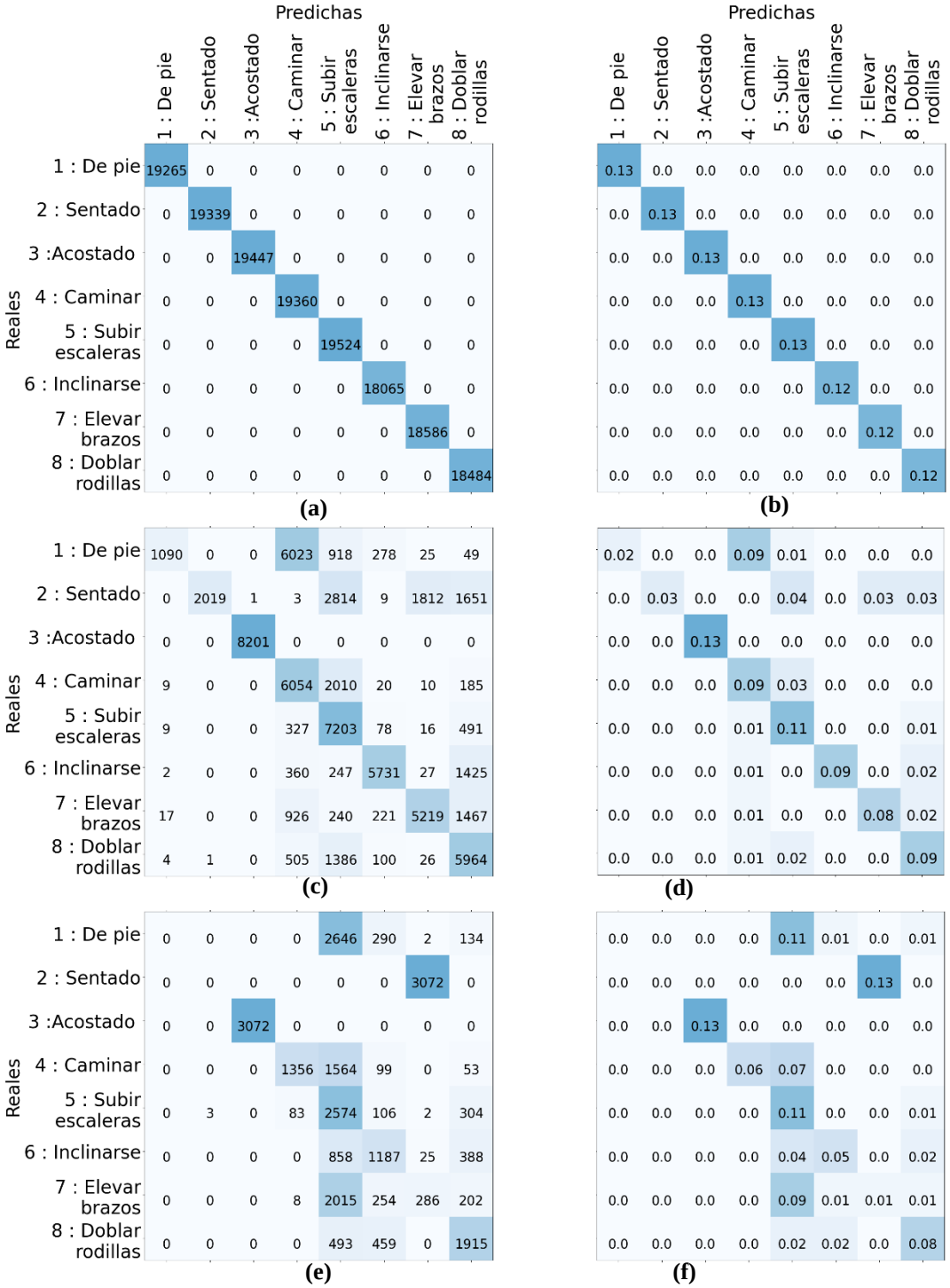


Fig. 3.10 Matriz de confusión modelo árbol de decisión sin gridsearch, Fuente: [autoría propia].

(a), (b) Entrenamiento (c), (d) Prueba (e),(f) Validación.

Tabla 3.9 Métricas de Algoritmo Árbol de decisión sin Gridsearch

	Entrenamiento	Prueba	Validación
Accuracy	1	0.64	0.44
Precision	1	0.75	0.43
Recall	1	0.64	0.44
F1	1	0.61	0.40

Las matrices de entrenamiento de la Fig. 3.10 presentan resultados ideales en un modelo de clasificación con valores en sus métricas igual a 1 (ver Tabla 3.9), pero que como ya se comentó anteriormente, no son resultados fiables ya que se asume que el modelo se ajustó muy bien a los datos con los que fue entrenado, algo que se considera solo como un primer acercamiento a el modelo de clasificación óptimo para los datos con los que se trabaja. Posteriormente, se tiene en las matrices de confusión del conjunto de prueba (ver Fig. 3.10) del modelo, resultados cuyos valores representan de mejor manera la capacidad real de clasificación del modelo, cuyos valores de las métricas disminuyeron de forma considerable, siendo menos afectada la precisión la cual disminuyó un 25% respecto al entrenamiento, mientras que el resto de las métricas disminuyeron un 36% en el caso del accuracy junto con el recall y un 39% en el caso de la métrica F1. Esta disminución en los valores indica que el modelo no está generalizado para rangos de valores distintos a los datos de entrenamiento o para la similitud de valores de los datos de prueba de las diferentes clases. Lo anterior se puede corroborar al observar los valores de las métricas de validación reflejadas en la Tabla 3.9, donde los valores tuvieron una disminución de más de la mitad del valor ideal, alcanzando valores entre 0.40-0.44, donde cabe destacar que la precisión fue la métrica que más disminuyó su valor respecto a los resultados de prueba del mismo modelo. En la Fig. 3.10e y Fig. 3.10f se puede observar que el modelo tuvo una mayor cantidad de errores asociado a la clase 5, donde se obtuvieron más falsos positivos en comparación con el resto de las clases.

- **Modelo Árbol de Decisión con gridsearch**

El proceso de gridsearch, entrega que la máxima profundidad del árbol debe ser de 15 niveles bajo el nodo principal.

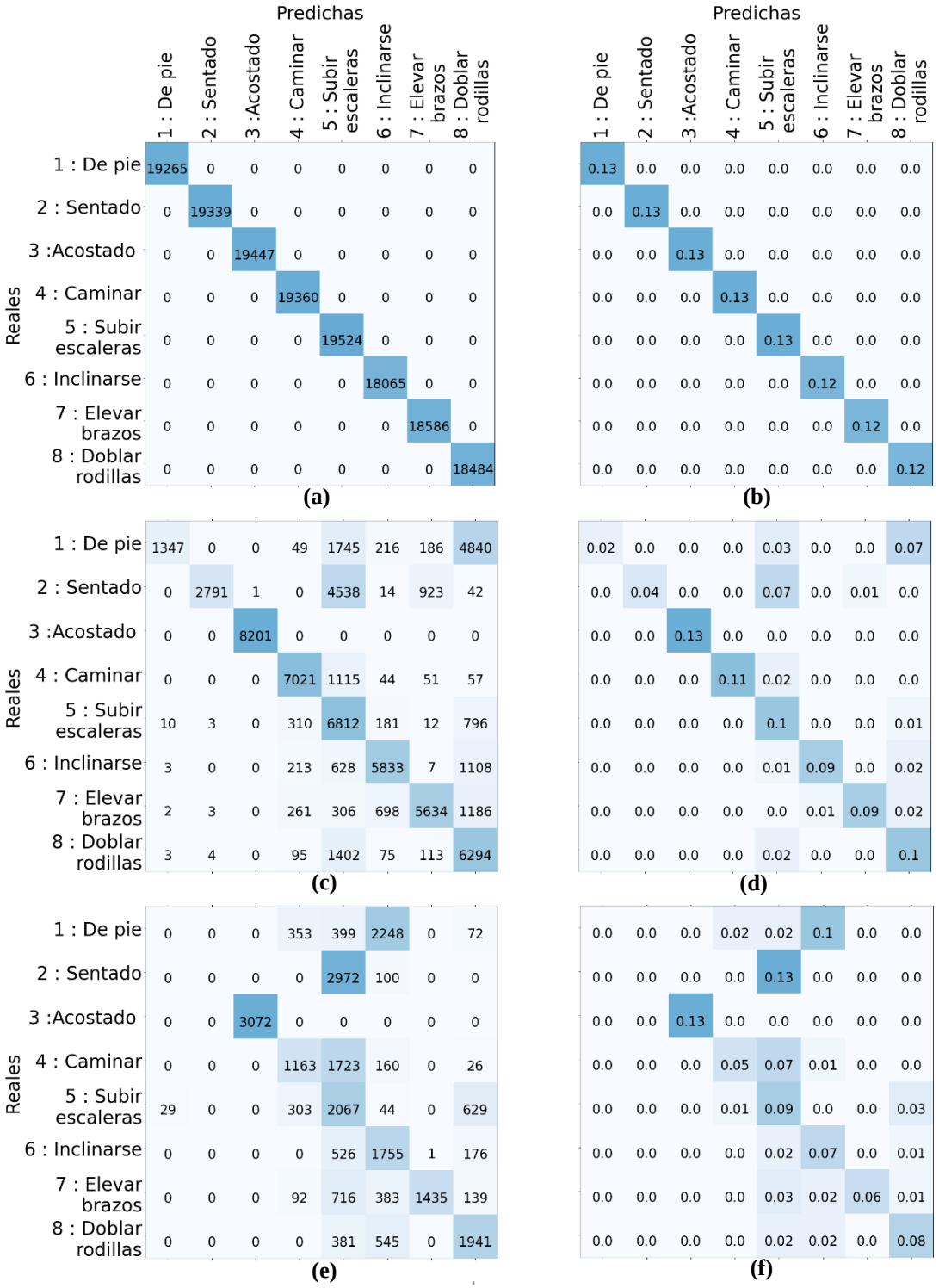


Fig. 3.11 Matriz de confusión modelo árbol de decisión con gridsearch, Fuente: [autoría propia].

(a), (b) Entrenamiento (c), (d) Prueba (e),(f) Validación.

Tabla 3.10 Métricas de Algoritmo Árbol de decisión con Gridsearch

	Entrenamiento	Prueba	Validación
Accuracy	1	0.67	0.49
Precision	1	0.79	0.48
Recall	1	0.68	0.50
F1	1	0.66	0.45

Se puede observar que evaluar el nuevo modelo mejorado con los datos con los que fue entrenado, entrega resultados esperados y ya vistos en los modelos anteriores, por lo cual se mantiene el mismo criterio de realizar más pruebas con los otros 2 conjuntos (Prueba y Validación), donde se puede observar en la Tabla 3.10, que las métricas mejoraron su valor, pero no de manera significativa.

También se puede observar en la Fig. 3.11c y Fig. 3.11d que los errores se distribuyeron en otras clases en comparación con el modelo sin gridsearch evaluado con el conjunto de prueba, ya que falsos negativos de la clase 2 que antes estaban asociados a las clases 7 y 8, ahora están asociados a la clase 5. En la evaluación del modelo con el conjunto de validación, se obtuvieron mejores métricas que sin considerar gridsearch, pero al igual que en el caso del conjunto de prueba, estos valores solo mejoraron de manera mínima, siendo los valores de la validación un 11.6% más alto que los obtenidos con el mismo conjunto en el modelo anterior. En el caso de los errores, se puede observar en la Fig. 3.11e y Fig. 3.11f que hay clases que poseen mayor cantidad de falsos positivos, donde la clase 5 tiene la mayor cantidad de falsos positivos los cuales representan casi un 39% de los errores de clasificación del modelo, luego la clase 6 representa un 14% de errores. También se puede observar que el modelo tiene dificultades en la validación para detectar y clasificar correctamente los datos correspondientes a la clase 1 y 2.

- **Modelo Random Forest sin gridsearch**

  El modelo tiene como hiperparámetros de interés el número de árboles que se deben generar y la profundidad de estos, los valores por defecto son 100 árboles y una profundidad no definida, expandiéndose las hojas de los árboles hasta alcanzar un valor mínimo de datos.

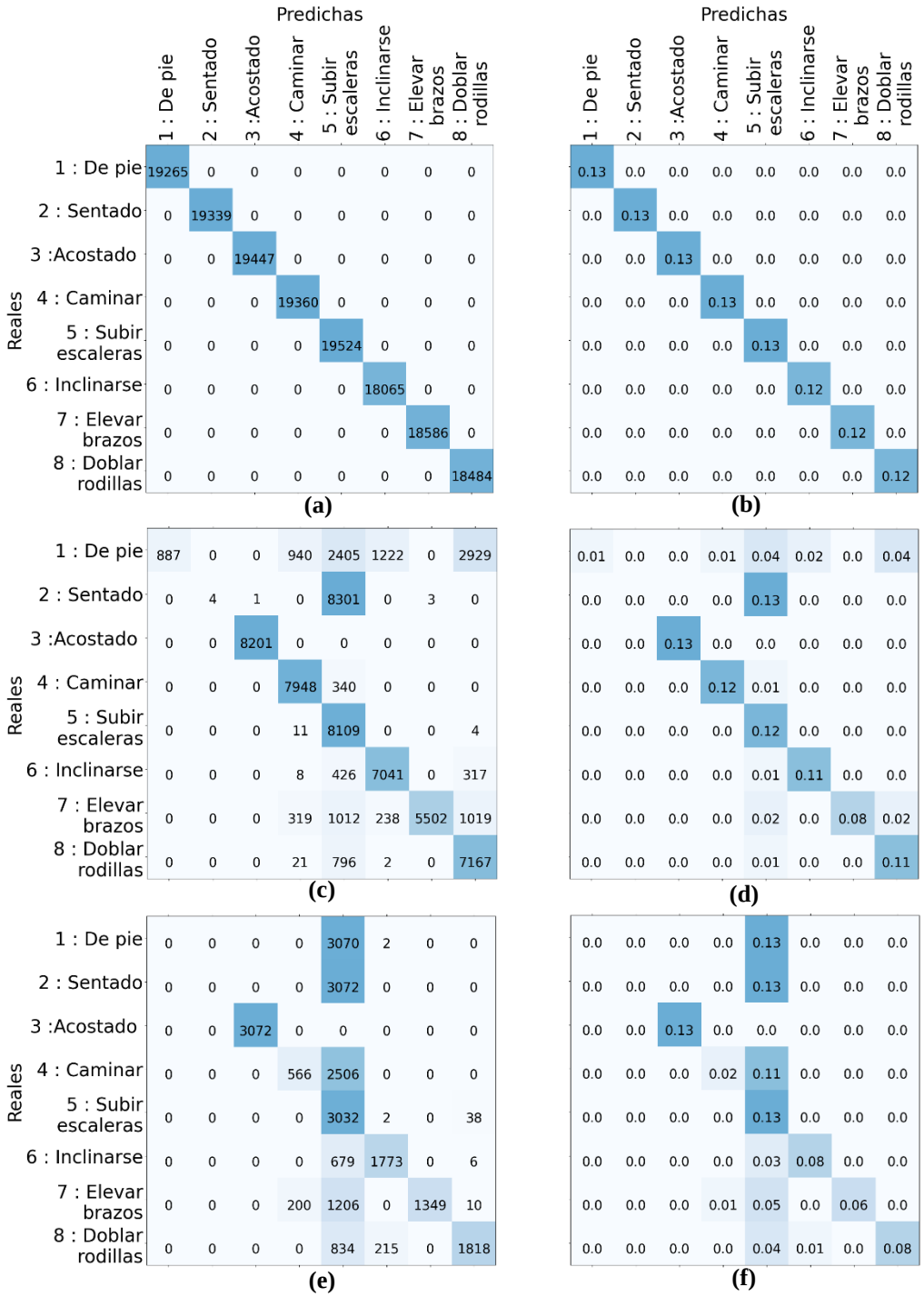


Fig. 3.12 Matriz de confusión modelo random forest sin gridsearch, Fuente: [autoría propia].

(a), (b) Entrenamiento (c), (d) Prueba (e),(f) Validación.

Tabla 3.11 Métricas de Algoritmo Random Forest sin Gridsearch

	Entrenamiento	Prueba	Validación
Accuracy	1	0.69	0.50
Precision	1	0.84	0.60
Recall	1	0.69	0.50
F1	1	0.63	0.48

El modelo random forest y modelos previos exhiben resultados similares en la evaluación del conjunto de entrenamiento, con métricas óptimas (Tabla 3.11) y matrices de clasificación sin errores (Fig. 3.12a). Sin embargo, esta evaluación inicial no refleja su verdadera efectividad, sino solo su adaptación a los datos de entrenamiento. El modelo carece de generalización para datos no vistos y necesita mejoras, dado que las métricas de la prueba (Tabla 3.11) son regulares, aunque la precisión destaque superando el valor de 0.80. En el conjunto de validación, las métricas muestran disminución en la efectividad y menor capacidad para datos diversos.

- **Modelo Random Forest con gridsearch**

  El proceso de gridsearch propone 314 árboles con una profundidad máxima de 19 niveles bajo el nodo principal.

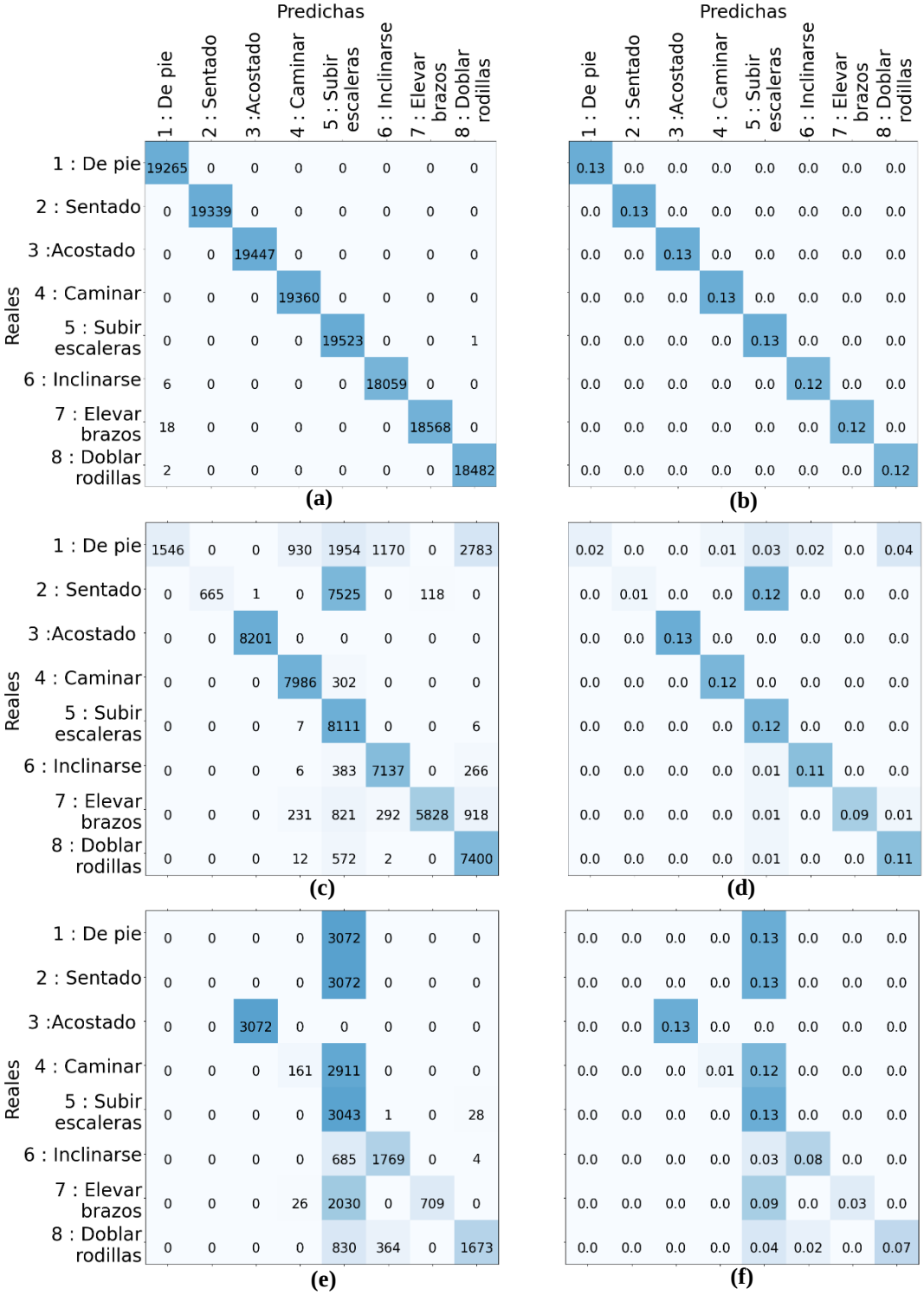


Fig. 3.13 Matriz de confusión modelo random forest con gridsearch, Fuente: [autoría propia].

(a), (b) Entrenamiento (c), (d) Prueba (e),(f) Validación.

Tabla 3.12 Métricas de Algoritmo Random Forest con Gridsearch

	Entrenamiento	Prueba	Validación
Accuracy	1	0.72	0.44
Precision	1	0.84	0.61
Recall	1	0.72	0.45
F1	1	0.68	0.42

Una vez ajustado el modelo, se observa en la Tabla 3.12 que con los datos de entrenamiento vuelve a presentar valores ideales en las métricas, las cuales se deben siempre cuestionar, ya que es difícil llegar a esos valores en la realidad y prueba de eso es que la matriz de la Fig. 3.13a muestra que existen errores en el modelo, aun cuando los datos sean los mismos con los que fue entrenado, por lo cual es necesario hacer las otras evaluaciones correspondientes. La prueba arrojó valores similares a los observados en la Tabla 3.11, aumentando solo entre 1.6%-4.8%, por lo cual el ajuste no mejoró lo suficiente el modelo, en las matrices de la Fig. 3.13c y Fig. 3.13d se observa una mejoría en los verdaderos positivos asociados a la clase 1, pero una redistribución en los falsos negativos asociados a la misma clase. Por último, se puede observar que el ajuste del modelo provocó una disminución en la capacidad de clasificar los datos del conjunto de validación, provocando así una reducción en los valores de las métricas, valores que se visualizan en la Tabla 3.11 y que representan una disminución en los verdaderos positivos de la clase 7 y un aumento de los falsos positivos de la clase 5, modificaciones que se pueden corroborar al observar la Fig. 3.13e y Fig. 3.13f.

- **Modelo SVM sin gridsearch**

El modelo tiene como hiperparámetros de interés el valor de la constante C cuyo valor por defecto es 1 y el kernel del modelo, que por defecto es una función de base radial.

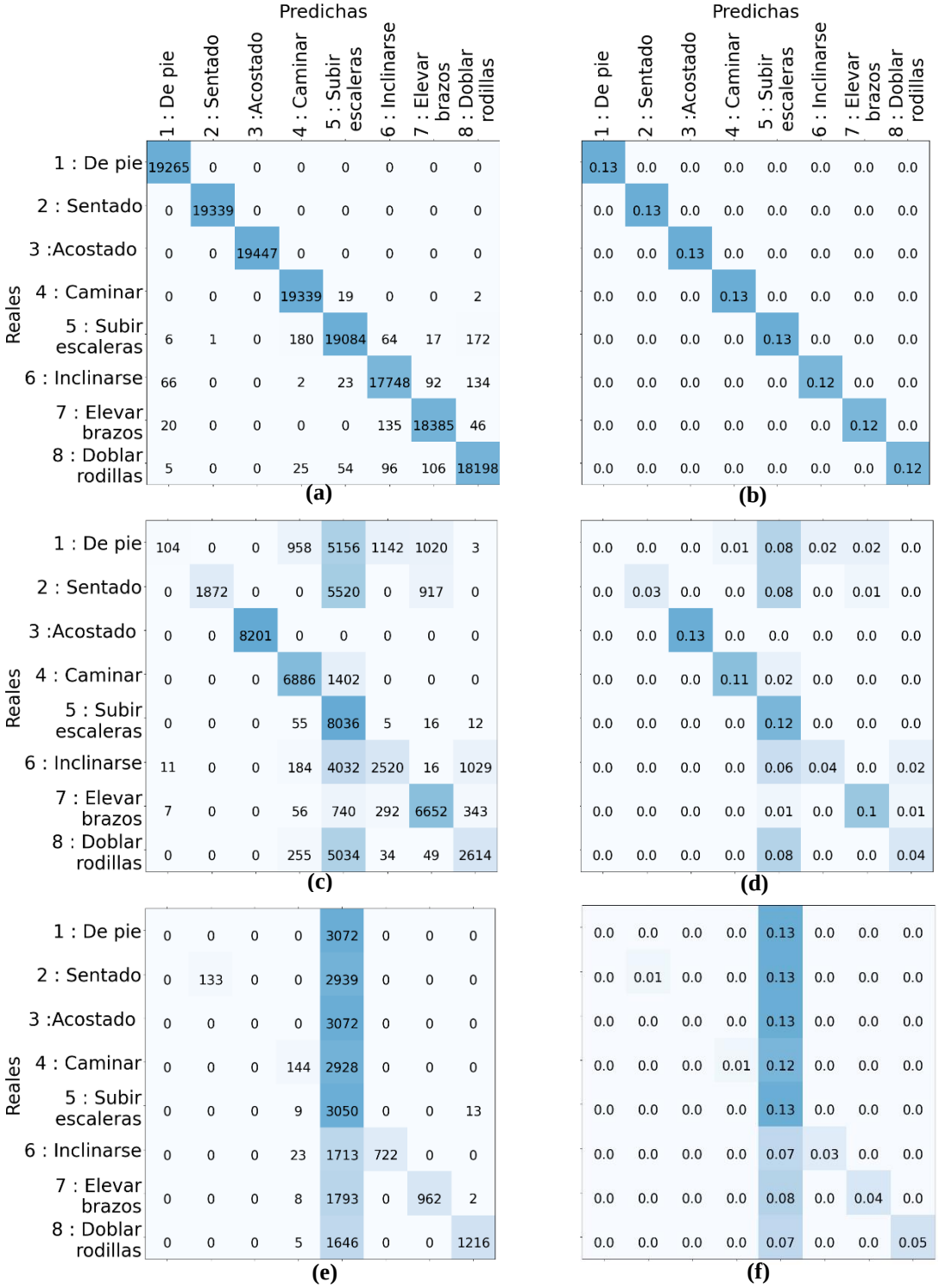


Fig. 3.14 Matriz de confusión modelo SVM sin gridsearch, Fuente: [autoría propia].

(a), (b) Entrenamiento (c), (d) Prueba (e),(f) Validación.

Tabla 3.13 Métricas de Algoritmo SVM sin Gridsearch

	Entrenamiento	Prueba	Validación
Accuracy	0.99	0.57	0.27
Precision	0.99	0.75	0.61
Recall	0.99	0.57	0.27
F1	0.99	0.54	0.25

Al visualizar la Tabla 3.13, se puede apreciar que este nuevo modelo sale inmediatamente de la tendencia de los modelos anteriores, al poseer valores en sus metricas igual a 0.99 al momento de ser evaluado con los datos de entrenamiento, esto mismo se refuerza al observar la Fig. 3.14a, donde se muestra que hay falsos positivos y negativos asociados a las clases 5, 6, 7 y 8, los cuales exponen el error asociado al modelo de manera más evidente.

El conjunto de prueba evidencia la menor capacidad que tiene el modelo SVM para clasificar los datos con los que se está trabajando, prueba de ello son los valores de las métricas las cuales son menores a lo obtenido en los modelos anteriormente vistos, también se puede observar en la Fig. 3 .14c que los falsos negativos y positivos están sesgados a las clases enumeradas en el párrafo anterior.

Por último, la validación expuso lo ineficaz que es el modelo frente a datos totalmente externos, aún cuando se normalizaron los datos de prueba, el modelo no fue capaz de distinguir las diferentes clases y se sesgó a clasificar 86% de los datos como clase 5 (ver Fig. 3.14e), cuando el porcentaje máximo e ideal de los verdaderos positivos es de un 13% de la totalidad de los datos de prueba.

- **Modelo SVM con gridsearch**

  El proceso gridsearch establece que el mejor valor para C es de 0.1 y el mejor kernel según la distribución de datos, es una función polinomial.

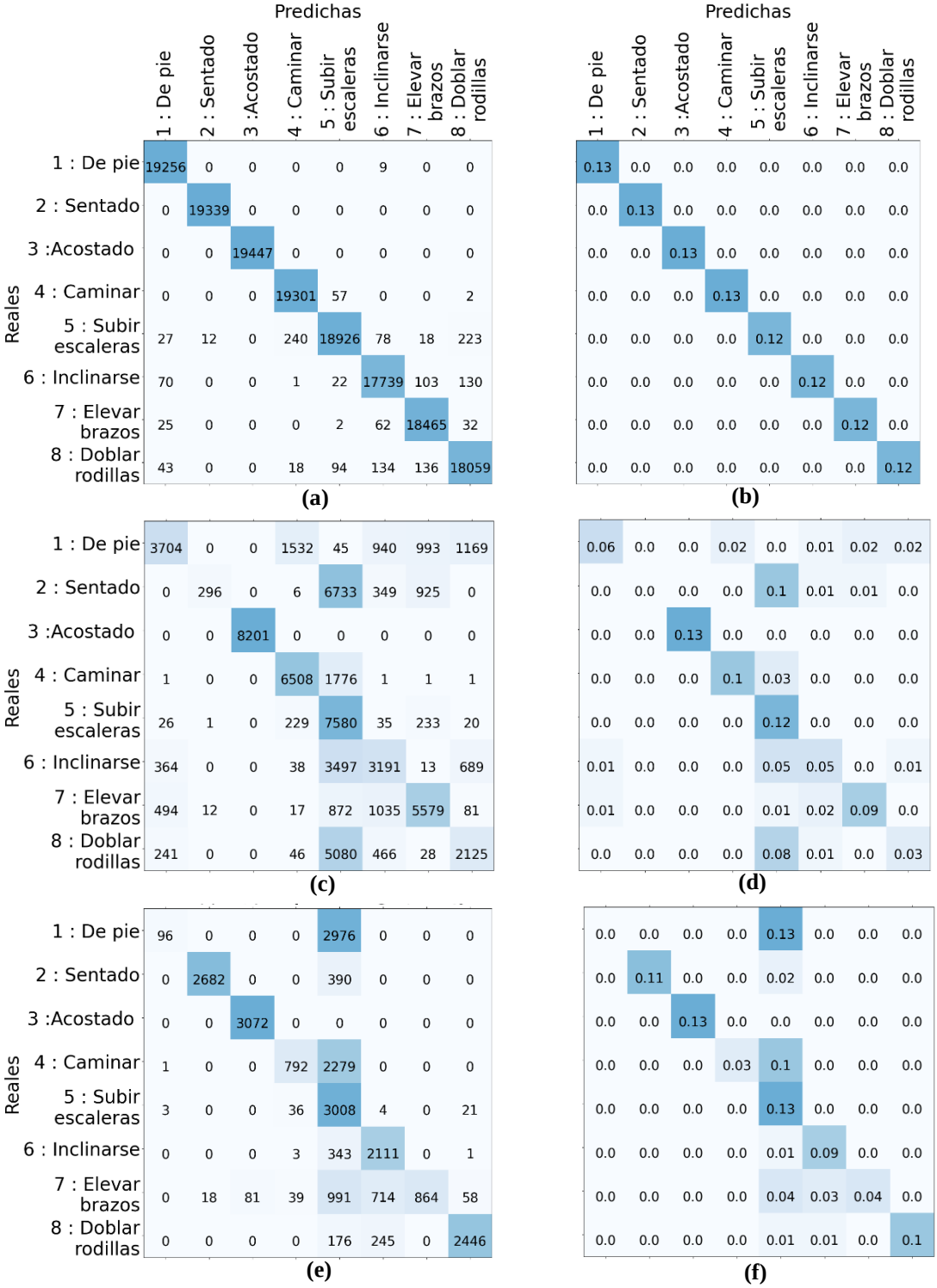


Fig. 3.15 Matriz de confusión modelo SVM con gridsearch, Fuente: [autoría propia].

(a), (b) entrenamiento (c), (d) Prueba (e),(f) Validación.

Tabla 3.14 Métricas de Algoritmo SV1M con Gridsearch

	Entrena	Prueba	Validación
Accuracy	0.99	0.57	0.64
Precision	0.99	0.70	0.85
Recall	0.99	0.57	0.65
F1	0.99	0.55	0.62

El modelo ajustado presenta un mejor resultado comparado con su versión previa, al observa la Tabla 3.14 se muestra un resultado similar a la Tabla 3.13 al momento de evaluar el modelo con el conjunto de entrenamiento, manteniendo ese error evidenciado en el modelo sin ajuste, inclusive al observar la Fig. 3.15a se muestran más errores de clasificación, teniendo 1.538 datos mal clasificados en comparación con la Fig. 3.14a, la cual posee un poco menos, con 1.265 datos mal clasificados. En el caso del conjunto de prueba, se obtuvieron métricas con valores semejantes al modelo SVM sin gridsearch, por lo cual el ajuste para obtener mejores resultados en el modelo no es notable hasta que se evalúa el conjunto de validación (Ver Tabla 3.14), donde se puede observar una mejor generalización del modelo el cual puede clasificar de mejor manera los nuevos datos.

- **Modelo MLP sin gridsearch**

El modelo tiene como hiperparámetros de interés el número de capas ocultas y el número de neuronas para cada capa oculta, el valor por defecto de capas ocultas es de 1 con 100 neuronas dentro de ella.

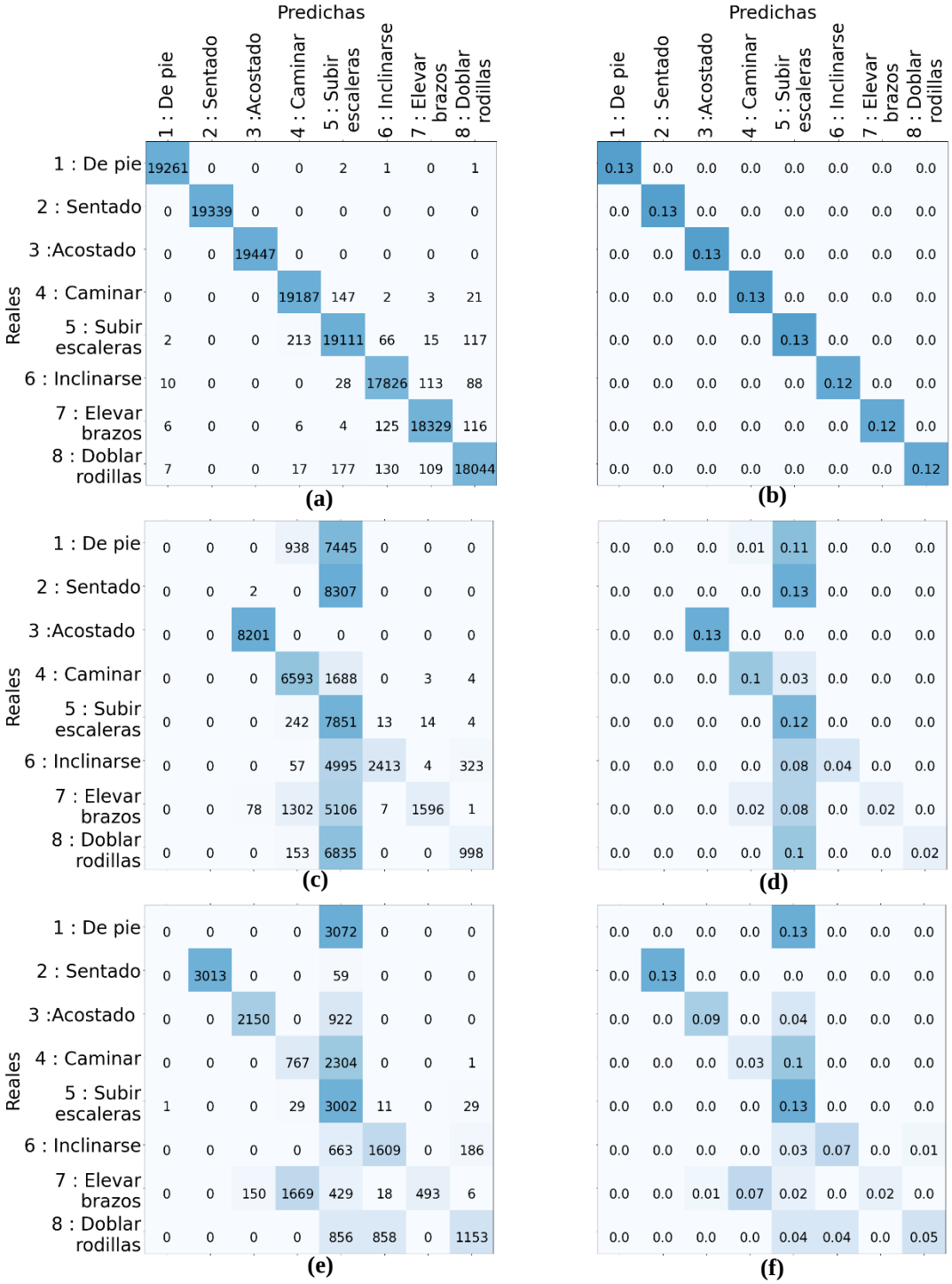


Fig. 3.16 Matriz de confusión modelo MLP sin gridsearch, Fuente: [autoría propia].

(a), (b) Entrenamiento (c), (d) Prueba (e),(f) Validación.

Tabla 3.15 Métricas de Algoritmo MLP sin Gridsearch

	Entrenamiento	Prueba	Validación
Accuracy	0.99	0.42	0.52
Precision	0.99	0.58	0.62
Recall	0.99	0.42	0.52
F1	0.99	0.38	0.50

Los resultados obtenidos al evaluar el modelo con el conjunto con el que fue entrenado es similar a lo obtenido en el modelo SVM, valores en las métricas que no alcanzan el valor ideal como se puede observar en la Tabla 3.15, demostrando una mayor dificultad al generalizar el modelo mediante sus diferentes criterios en función del rango de datos de cada clase, algo que se comprueba al momento de hacer la evaluación con el primer conjunto de prueba, el cual entrega los valores de métricas más bajos obtenidos con este conjunto de datos en todos los modelos evaluados anteriormente. Por lo anterior se hace evidente que el modelo necesita un ajuste que le permita mejorar sus criterios de clasificacion, considerando que el conjunto de prueba está construido bajo los mismo rangos de datos que tiene el conjunto de entrenamiento. Finalmente al evaluar con el último conjunto de prueba se obtienen mejores resultados, siendo el primer modelo sin ajustar cuyos resultados de validación superan a los de prueba, manteniendo la tendencia de que la precision alcanza el mejor valor entre las métricas.

- **Modelo MLP con gridsearch**

  El proceso de gridsearch entregó que la mejor combinación de neuronas y capas ocultas era de 2 capas ocultas con 33 neuronas en cada una de ellas.

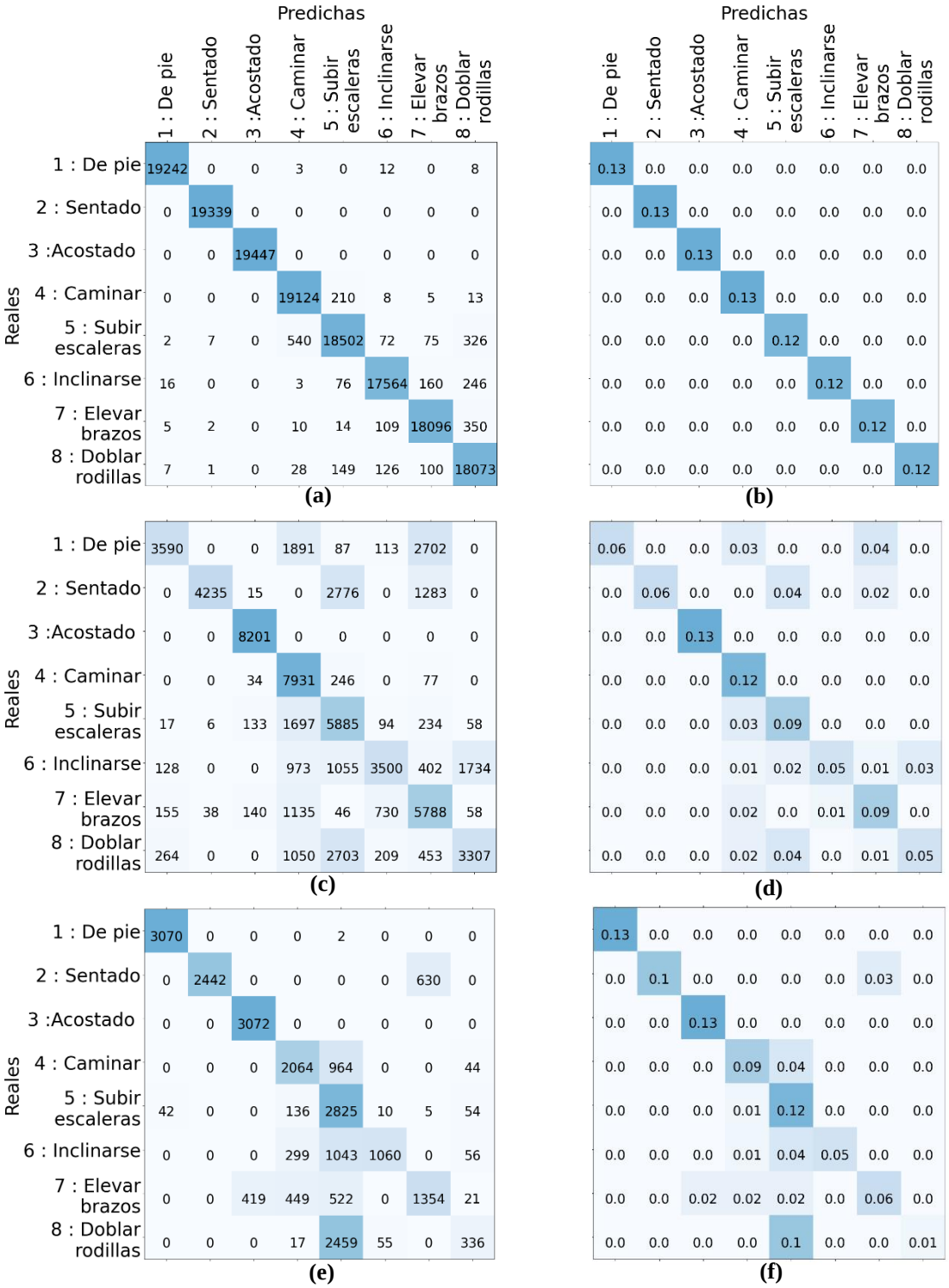


Fig. 3.17 Matriz de confusión modelo MLP con gridsearch, Fuente: [autoría propia].

(a), (b) Entrenamiento (c), (d) Prueba (e),(f) Validación.

Tabla 3.16 Métricas de Algoritmo MLP con Gridsearch

	Entrenamiento	Prueba	Validación
Accuracy	0.98	0.65	0.69
Precision	0.98	0.72	0.78
Recall	0.98	0.65	0.68
F1	0.98	0.64	0.67

El modelo al ser ajustado presenta un menor valor en las métricas resultantes de todos los modelos al momento de evaluarlo con los datos de entrenamiento, como se puede apreciar en la Tabla 3.16, en la cual se registra mayores errores asociados, desde la clase 4 hasta la 8. Al evaluar con el segundo conjunto se muestra una mejoría en comparación con el modelo MLP anterior, consiguiendo así mejores resultados que demuestran que el modelo pudo entrenarse de mejor manera generalizando mejor los rangos de datos a los que enfrenta. La tercera evaluación presenta mejores resultados que la segunda, pero superándola por un valor no mayor a un 8.3% en todas las métricas. Aunque no se puede considerar una mejoría considerable, cabe destacar que no existe mucha diferencia entre los resultados de prueba y validación, demostrando que con el ajuste el modelo se generalizó mejor y es capaz de hacer frente a nuevos datos, y con buenos valores en sus métricas.

Tabla 3.17 Tiempo de entrenamiento de los modelos

Modelo	Tiempo 1 (s)	Tiempo 2 (s)
KNN	0.034	0.11
Árbol de Decisión	7.29	5.54
Random Forest	92.97	244.9
SVM	184.66	205.8
MLP	516	354.7

La Tabla 3.17 presenta tiempos de entrenamiento para varios modelos, con hiperparámetros por defecto y optimizados por gridsearch. KNN destaca por su breve tiempo y buen rendimiento, dependiendo de su enfoque e hiperparámetros.

El árbol de decisión tiene el segundo menor tiempo, con dos tiempos significativamente bajos. Su rendimiento lo ubica en cuarto lugar, superando solo al modelo random forest en el conjunto de validación. Sin embargo, su desempeño sugiere dificultades en generalizar criterios de clasificación.

El SVM, con mejores hiperparámetros, ocupa el tercer menor tiempo, más de 3 minutos, y tiene el tercer mejor rendimiento, con buena generalización en criterios de clasificación y métricas adecuadas, destacando en las métricas de validación.

El random forest tarda aproximadamente 4 minutos, medio minuto más que el SVM, y tiene

un rendimiento inferior, ubicándose en último lugar en términos de desempeño, mostrando disminución en las métricas en validación.

El modelo multilayer perceptrón tiene el mayor tiempo, alrededor de 6 minutos, pero con mejores hiperparámetros, su rendimiento es equilibrado, obteniendo buenas métricas en prueba y validación. Su rendimiento lo coloca como el segundo mejor modelo en el estudio.

3.6. Depuración de base de datos

Con el fin de mejorar el rendimiento en la clasificación de actividades cotidianas cuyos resultados fueron mostrados en la sección anterior, se realizó una limpieza de los datos. Para esto se siguió una serie de pasos para reducir la cantidad de columnas y filas, que supusieran una baja representatividad para distinguir las diferentes clases.

Primero se tomó la base de datos en busca de valores únicos en cada una de las columnas, es decir se buscó ver la variabilidad en el valor de cada uno de los datos y al mismo tiempo establecer cuantas veces se repetían los diferentes valores. Si la variabilidad era baja y el número de repeticiones alta entonces se consideraba que la columna no era representativa para distinguir las diferentes clases por lo cual se podía eliminar. Ninguna columna presentó una baja representatividad por lo cual no se eliminó ninguna.

Luego se consideró eliminar las columnas que tuvieran una baja varianza, para eso se estableció un umbral de varianza igual a 0.5, y toda columna que tuviera una varianza menor a ese umbral fue eliminada, en este caso 8 columnas fueron eliminadas dejando solo 15 características para entrenar los modelos.

Posteriormente, una vez eliminadas las columnas en el proceso anterior, se procedió a identificar filas que tuvieran datos idénticos, para buscar eliminar las filas duplicadas, este proceso no fue productivo pues no se detectaron filas idénticas.

Por último, sé utilizo la detección de valores atípicos dentro de la base de datos mediante la desviación de densidad de los datos respecto a sus vecinos, donde los datos con menor densidad se consideraban valores atípicos, al aplicar este método, se redujo el número de filas en un 3.3%.

	acx	acy	acz	alx	aly	alz	mlx	mly	mlz	arx	ary	arz	mrx	mry	mrz	Actividad
0	-8.7952	0.36553	-0.90394	-0.409370	-6.0615	-5.1613	-18.0950	7.8874	7.49800	-2.0496	-7.8307	-1.06860	-2.9309	-5.83710	12.274	5
1	-8.4579	0.17011	-1.22680	-0.030183	-4.8701	-4.8271	-9.1922	10.3670	5.56360	-2.0719	-8.4213	-1.12940	-2.7625	-6.94560	10.484	5
2	-8.8564	0.39709	-1.09060	1.045500	-4.8157	-4.3449	-1.3652	10.1050	3.38950	-2.1899	-8.3788	-0.87727	-2.5611	-4.76270	11.188	5
3	-9.3906	0.33803	-0.76958	1.327300	-4.0696	-3.9923	7.3845	17.7240	0.54087	-2.5724	-8.3779	-0.79390	-2.6985	-0.55144	14.753	5
4	-8.8890	0.51293	-1.59590	0.567660	-4.1039	-3.4599	18.2390	18.1660	-0.25784	-3.4526	-8.4178	-0.77505	-3.8102	-3.99930	13.682	5

Fig. 3.18 Representación de base de datos nueva. Fuente: [autoría propia, software Python].

En la Fig. 3.18 se puede observar las columnas que serán utilizadas para hacer las nuevas pruebas a los modelos con una base de datos más limpia, donde el número de instancias corresponde a un total de 147.052 filas para entrenar el modelo. En el caso del conjunto de validación se debió ajustar la cantidad de columnas y las características resultantes de la limpieza de la base de datos, dejando solo las columnas que se pueden visualizar en la Fig. 3.18.

3.6.1 Resultados de evaluación de base de datos depurada.

Se presentarán los resultados de los modelos evaluados con la nueva base de datos en tablas, enfocándose en métricas y excluyendo las matrices de confusión, las cuales están disponibles en el Anexo A. Los valores por defecto de los modelos no se mencionarán aquí, dado que se discutieron en la sección 3.10.1, pero sí se proporcionarán los resultados del proceso de gridsearch.

Con fines prácticos, la comparación de los modelos se llevó a cabo exclusivamente mediante el análisis de las métricas previamente establecidas.

- **Modelo KNN sin gridsearch**

Tabla 3.18 Métricas de Nuevo Algoritmo KNN sin Gridsearch

	Entrenamiento	Prueba	Validación
Accuracy	1	0.62	0.54
Precision	1	0.79	0.66
Recall	1	0.62	0.55
F1	1	0.61	0.52

Los resultados obtenidos al aplicar el modelo KNN muestran que los datos de entrenamiento fueron clasificados de manera correcta obteniendo un valor ideal en las métricas, algo que ya se había visto en los anteriores resultados, y que de la misma forma se debe cuestionar, por lo cual disminuyen los valores de las métricas resultantes, las cuales contrastan con los obtenidos en ambos modelos KNN evaluados sin la limpieza de datos, donde los resultados de validación están por debajo de los resultados de prueba del actual modelo y el modelo asociado a la Tabla 3.7.

- **Modelo KNN con gridsearch**

  El proceso de gridsearch determinó que el número de vecino necesario para hacer las predicciones es de solo 1.

Tabla 3.19 Métricas de Nuevo Algoritmo KNN con Gridsearch

	Entrenamiento	Prueba	Validación
Accuracy	1	0.61	0.51
Precision	1	0.68	0.54
Recall	1	0.61	0.52
F1	1	0.60	0.48

Este modelo presenta una menor efectividad al momento de clasificar correctamente cada instancia, algo que no se ve reflejado en la columna de entrenamiento de la Tabla 3.19, pero se hace evidente en el resto de las columnas, donde las pruebas arrojan valores de sus métricas levemente menores y que siguen el mismo comportamiento analizado en los resultados de la tabla anterior.

- **Modelo Árbol de Decisión sin gridsearch**

Tabla 3.20 Métricas de Nuevo Algoritmo Árbol de Decisión sin Gridsearch

	Entrenamiento	Prueba	Validación
Accuracy	1	0.54	0.42
Precision	1	0.70	0.35
Recall	1	0.54	0.43
F1	1	0.52	0.38

Se puede apreciar en la Tabla 3.20 que el modelo se ajusta muy bien a los datos de entrenamiento y que no hubo una buena generalización en el modelo, lo que lo hizo propenso a cometer errores ya que no es capaz de distinguir de buena manera las distintas clases. También es necesario mencionar que las pruebas arrojaron peores resultados en comparación con los anteriores modelos de árbol de decisión, siendo más bajo el resultado de validación.

- **Modelo Árbol de Decisión con gridsearch**

  El proceso de gridsearch determinó que la profundidad máxima del árbol debe ser de 30 niveles bajo el nodo principal.

Tabla 3.21 Métricas de Nuevo Algoritmo Árbol de Decisión con Gridsearch

	Entrenamiento	Prueba	Validación
Accuracy	1	0.54	0.47
Precision	1	0.5	0.44
Recall	1	0.54	0.48
F1	1	0.48	0.43

La Tabla 3.21 nos muestra que el ajuste del modelo anterior propone nuevamente un sobreajuste a los datos de entrenamiento, y se obtuvo una reducción en el valor de algunas métricas en el conjunto de prueba.

En el caso de la validación se obtuvo una mejora de todas sus métricas en contraste con el modelo sin ajuste, donde la precisión es la métrica más beneficiada, subiendo su valor en un 9%. Aun cuando las métricas superan a las métricas del anterior modelo, no lo hace significativamente, lo que hace que el modelo siga siendo ineficaz y sea menos preferible para la actividad de clasificación.

- **Modelo Random Forest sin gridsearch**

Tabla 3.22 Métricas de Nuevo Algoritmo Random Forest sin Gridsearch

	Entrenamiento	Prueba	Validación
Accuracy	1	0.59	0.5
Precision	1	0.52	0.52
Recall	1	0.6	0.51
F1	1	0.53	0.47

Observando la Tabla 3.22, se evidencia que el modelo sufre del sobre ajuste de los datos de entrenamiento. Los resultados de evaluar el modelo con el conjunto de prueba evidencian una menor eficacia al momento de clasificar los datos de la nueva base de datos producto de la limpieza realizada, esta misma respuesta se ve reflejada en los resultados de la validación, resultados que de la misma forma se vieron afectados por la menor cantidad de características a considerar, teniendo así un valor en sus métricas menor a los obtenidos con el conjunto de prueba.

- **Modelo Random Forest con gridsearch**

El proceso de gridsearch determinó que la cantidad de árboles debe ser de 279, con una profundidad máxima de 17 niveles bajo su nodo principal.

Tabla 3.23 Métricas de Nuevo Algoritmo Random Forest con Gridsearch

	Entrenamiento	Prueba	Validación
Accuracy	1	0.59	0.51
Precision	1	0.64	0.57
Recall	1	0.59	0.52
F1	1	0.53	0.5

En la Tabla 3.23, el ajuste del modelo provoca sobreajuste en el entrenamiento, pero mejora resultados en prueba y validación. En la prueba, la precisión aumenta de 0.52 a 0.64, destacando sobre la leve disminución del recall de 0.60 a 0.59. En la validación, todas las métricas mejoran,

especialmente la precisión de 0.52 a 0.57. Aunque hay mejora, no es sobresaliente, ya que el resultado de la prueba aún es inferior al del random forest ajustado con todas las características. El modelo random forest ajustado muestra mejora en la validación respecto a los resultados de la Tabla 3.12, excepto en precisión, que disminuye de 0.60 a 0.57.

- **Modelo SVM sin gridsearch**

Tabla 3.24 Métricas de Nuevo Algoritmo SVM sin Gridsearch

	Entrenamiento	Prueba	Validación
Accuracy	0.95	0.41	0.23
Precision	0.95	0.57	0.36
Recall	0.95	0.4	0.23
F1	0.95	0.36	0.17

Al observar la Tabla 3.24, se hace evidente que el modelo SVM se ve afectado en mayor medida respecto a los modelos analizados anteriormente, pues solo basta con observar la primera columna para reconocer que el error del modelo es más evidente, pues sus métricas ya presentan un valor no ideal que está un 5% más bajo de lo observado en los otros modelos.

Sin considera la precisión en la prueba, se puede evidenciar que el valor de las métricas decreció hasta valores menores a la mitad de lo obtenido en la primera evaluación de este modelo, solo la precisión está sobre el 50% del valor ideal de las métricas, siendo 1 el mayor valor obtenible y el ideal.

Finalmente, en la validación se alcanza el valor más bajo hasta el momento en todas las métricas, mostrando su baja capacidad de clasificar datos correctamente, resultando un modelo deficiente para trabajar con la base de datos actual.

- **Modelo SVM con gridsearch**

  El proceso de gridsearch determinó que los mejores hiperparámetros son un kernel polinomial y un valor de C igual a 0.1.

Tabla 3.25 Métricas de Nuevo Algoritmo SVM con Gridsearch

	Entrenamiento	Prueba	Validación
Accuracy	0.95	0.41	0.45
Precision	0.95	0.51	0.49
Recall	0.95	0.41	0.45
F1	0.95	0.37	0.41

El ajuste del modelo no provocó una diferencia apreciable en la primera evaluación ya que se puede observar en la Tabla 3.25, que los valores de sus métricas de entrenamiento son iguales a los obtenidos en la Tabla 3.24 manteniendo el mismo porcentaje de error asociado, error que aumenta

considerablemente al evaluar con el conjunto de prueba, obteniendo un mejor valor en el F1, pero un peor valor en la precisión en comparación con el modelo sin ajustar. La validación evidencia una mejora en todas las métricas, pero que sigue sin ser suficiente para considerar realmente el modelo SVM como uno de los mejores, al contrario, sigue siendo uno de los modelos con menor rendimiento.

- **Modelo MLP sin gridsearch**

Tabla 3.26 Métricas de Nuevo Algoritmo MLP sin Gridsearch

	Entrenamiento	Prueba	Validación
Accuracy	0.96	0.39	0.4
Precision	0.96	0.43	0.62
Recall	0.96	0.39	0.4
F1	0.96	0.31	0.35

Los valores proporcionados por la Tabla 3.26 dan cuenta de una disminución en el rendimiento del modelo en comparación con la Tabla 3.15 ya que todas las evaluaciones hechas al actual modelo han arrojado menores valores, exceptuando la precisión en la validación que toma el mismo valor.

Los resultados de las diferentes pruebas son deficientes y muy similares a los obtenidos en los modelos SVM analizados previamente, por lo cual es necesario realizar un ajuste en los parámetros para buscar obtener mejor modelo que pueda resolver la tarea de clasificar correctamente una mayor cantidad de instancias.

- **Modelo MLP con gridsearch**

  El proceso gridsearch determinó que la mejor combinación de capas ocultas y neuronas está dada por 2 capas ocultas y 20 neuronas en cada una de ellas.

Tabla 3.27 Métricas de Nuevo Algoritmo MLP con Gridsearch

	Entrenamiento	Prueba	Validación
Accuracy	0.94	0.55	0.52
Precision	0.94	0.47	0.54
Recall	0.94	0.56	0.52
F1	0.94	0.49	0.48

Una vez ajustado el modelo, se puede observar en la Tabla 3.27, que el error aumentó, debido a que las métricas resultantes de evaluar el modelo con el conjunto de entrenamiento disminuyeron su valor de 0.96 a 0.94. Al realizar la prueba se registra una mejoría en el valor de sus métricas, siendo la precisión la menos favorecida, aumentando solo un 9.3% su valor, mientras que el resto de las métricas aumentaron entre 41%-58% respecto a los resultados del modelo sin ajuste de la Tabla 3.26. Por último, la validación muestra un aumento en el valor de 3 de sus métricas a costa de la disminución

de la cuarta, siendo la precisión la métrica que decreció su valor en un 12.9 % de 0.62 a 0.54, el accuracy y recall aumentaron un 30% de 0.40 a 0.52 y el F1 aumentó un 37% de 0.35 a 0.48.

Se esperaba tener un resultado mejor al eliminar características, pero los resultados evidencian que al usar la nueva base de datos es más difícil construir un modelo generalizado capaz de reconocer la cantidad de datos correspondiente a cada clase, considerando siempre la similitud que pueda existir entre clases, la cual se ve reflejada en las métricas debido al aumento de falsos negativos y positivos en consecuencia de la disminución de verdaderos positivos, disminución que afecta a todas las métricas.

En última instancia, al analizar los resultados métricos al reducir el número de características, se identifica al KNN como el modelo más destacado, seguido por el Random Forest que obtiene el segundo mejor rendimiento. El tercer puesto lo ocupa el modelo de árboles de decisión, aunque su rendimiento disminuyó en la validación. El modelo MLP se sitúa en cuarto lugar debido a métricas globalmente inferiores. Finalmente, el SVM se posiciona como el modelo menos efectivo.

3.7. Implementación de H20

Debido a los resultados deficientes en el apartado anterior, se hará uso del módulo H20 en Python para que genere de manera automática distintos modelos de clasificación, en busca de un modelo que tenga la capacidad de reconocer las actividades con mayor exactitud.

Una vez generados todos los modelos posibles, el mismo software entrega una lista con los mejores modelos obtenidos y algunas métricas asociadas. También entrega resultados más detallado del mejor modelo, para poder realizar un mejor análisis de este.

Al ejecutar el módulo propio de H2O, se entregan 9 modelos que se pueden observar a continuación en la Tabla 3.28 con sus respectivos errores asociados resultantes de evaluar cada modelo con los mismos datos de entrenamiento.

Tabla 3.28 Modelos de Clasificación

Modelo	Error asociado	logloss	rmse	mse
GBM_1	0.0070067	0.0221489	0.0765919	0.00586632
StackedEnsemble_BestOfFamily_1	0.00713805	0.0240082	0.0797117	0.00635395
StackedEnsemble_BestOfFamily_2	0.00727712	0.0321823	0.0886947	0.00786674
StackedEnsemble_AllModels_1	0.00732599	0.0321956	0.088733	0.00787355
DRF_1	0.0243249	0.174944	0.170273	0.0289929
GBM_4	0.0331363	0.489548	0.388572	0.150988
GBM_3	0.0395594	0.403122	0.342184	0.11709
GBM_2	0.0473222	0.399145	0.342332	0.117191
GLM_1	0.284415	0.92993	0.552927	0.305728

Se puede observar en la Tabla 3.28 que el modelo con el menor error asociado es un GBM el cual es un conjunto de modelos de regresión o clasificación que está basado en arboles de decisión, de este conjunto de modelos se obtienen resultados predictivos mediante estimaciones mejoradas gradualmente, es decir que el resultado de un árbol permite ajustar los siguientes árboles que se construyan, donde el modelo GBM seleccionado posee los siguientes parámetros representados en la Tabla 3.29.

Tabla 3.29 Modelo Seleccionado

Número de árboles	Número de árboles internos	Profundidad máxima	Profundidad media	Máximo de hojas	Media de hojas
41.0	328.0	15.0	14.957317	729.0	506.60367

En la Tabla 3.29 se puede observar que el número de árboles necesarios para ajustar el modelo no es muy alto, pero si es alto el número máximo de hojas necesarias en el nivel más bajo del árbol para cumplir de manera óptima la tarea de clasificar los datos.

3.7.1 Matrices de confusión

A continuación, se presenta la matriz de confusión resultante de evaluar el modelo con los datos de entrenamiento.

Tabla 3.30 Matriz de Confusión de Entrenamiento

Clases	1	2	3	4	5	6	7	8	Error
1	19259	0	0	0	0	3	1	2	0.0003114
2	0	19338	0	0	0	0	1	0	0.0000517
3	0	0	19447	0	0	0	0	0	0
4	0	0	0	19312	37	4	0	7	0.0024793
5	0	0	0	63	19402	14	0	45	0.0062487
6	7	0	0	0	6	17997	7	48	0.0037642
7	10	2	0	0	0	2	18555	17	0.0016679
8	8	0	0	7	81	48	23	18317	0.0090348
Total	19284	19340	19447	19382	19526	18068	18587	18436	0.0029131

En este caso, la matriz de confusión se representa como se observa en la Tabla 3.30, donde las columnas representan las predicciones del modelo y las filas representan la clase verdadera de cada dato. Se puede observar que al igual que los anteriores modelos, el resultado de hacer una evaluación al modelo con el mismo conjunto de entrenamiento arroja un resultado con muy bajo error, el cual se puede corroborar en la columna de error y última fila, mostrando el buen criterio que se construyó para clasificar este conjunto de datos.

Tabla 3.31 Matriz de Confusión de Prueba

Clases	0	1	2	3	4	5	6	7	Error
1	8367	0	0	0	0	12	2	2	0.0019086
2	0	8309	0	0	0	0	0	0	0
3	0	0	8201	0	0	0	0	0	0
4	0	0	0	8229	53	1	0	5	0.0071187
5	0	0	0	99	7925	25	0	75	0.0244953
6	12	0	0	0	18	7679	9	74	0.0145021
7	10	0	0	0	0	7	8047	26	0.0053152
8	14	0	0	16	125	78	29	7724	0.0328074
Total	8403	8309	8201	8344	8121	7802	8087	7906	0.0106179

En la Tabla 3.30 se puede observar que, al evaluar con el conjunto de prueba, la proporción de datos erróneos respecto a la totalidad de datos es similar a la obtenida con el conjunto de entrenamiento, aunque en este caso es levemente superior el error asociado a las predicciones. Aun así, se evidencia el buen rendimiento que posee el modelo y su buena generalización de los datos.

3.7.2 Curva de aprendizaje del modelo GBM

Se puede visualizar en la Fig. 3.19 como al aumentar el número de árboles va disminuyendo el valor de la métrica de logloss al entrenar el modelo, incluso se puede observar que la curva de entrenamiento sin validación cruzada tiene el mismo comportamiento que la curva con validación cruzada, teniendo un comportamiento diferente al llegar a los 41 árboles, donde la curva de entrenamiento sin validación cruzada se detiene para evitar un sobre ajuste, confirmando de manera gráfica el número de árboles necesarios para obtener un buen modelo de clasificación, el cual se entrega en la Tabla 3.29.

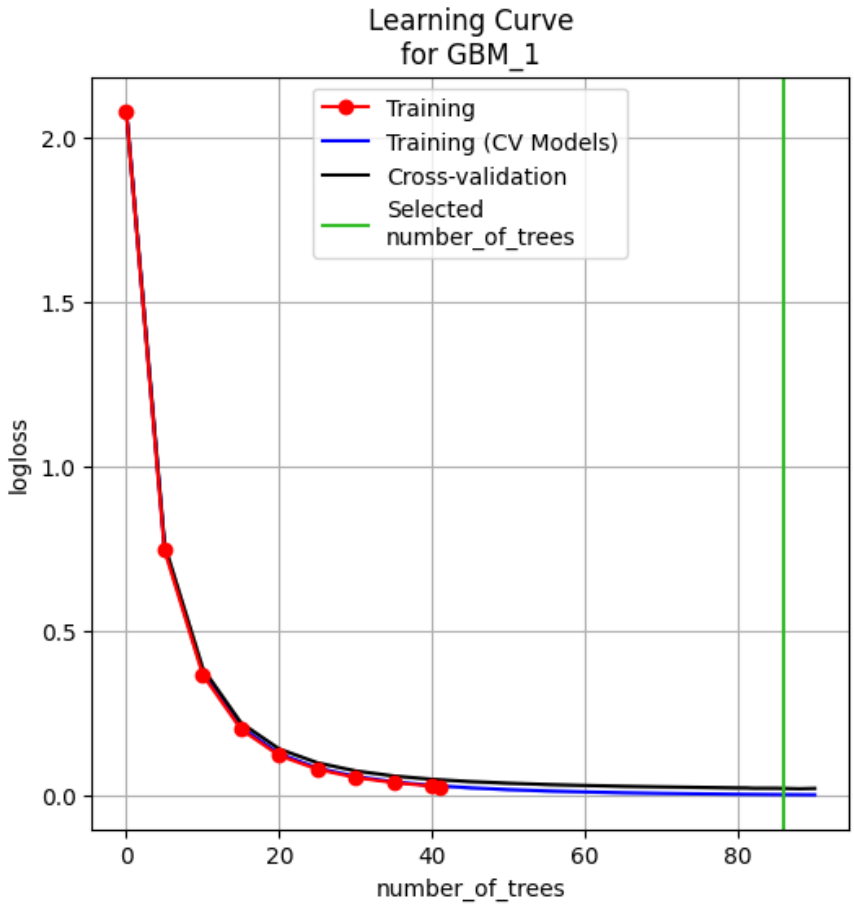


Fig. 3.19 Curva de aprendizaje, modelo GBM. Fuente: [autoría propia].

3.7.3 Variables más importantes

Por otro lado, la herramienta H2O entrega la importancia de cada variable, es decir, aquellas que más influyen en el modelo y que permiten generalizar de mejor manera el modelo para obtener mejores criterios de clasificación.

En la Fig. 3.20 se muestra el orden de importancia de las variables y cuáles son las que afectan en menor y mayor medida a las predicciones del modelo, es decir, muestra que variables son las más determinantes al momento de elegir una clase u otra. Se puede observar que la variable **acz** es la más determinante y con bastante diferencia en la evaluación general del modelo, la cual corresponde al valor en el eje Z del acelerómetro de un sensor inercial ubicado en el pecho de los sujetos. Como segunda variable más importante se tiene la **alz**, que es el valor en el eje Z de un acelerómetro ubicado en el tobillo izquierdo de los sujetos, la cual supera por una leve diferencia a la siguiente variable más determinante que corresponde a la aceleración en el eje Y del sensor inercial ubicado en el antebrazo derecho.

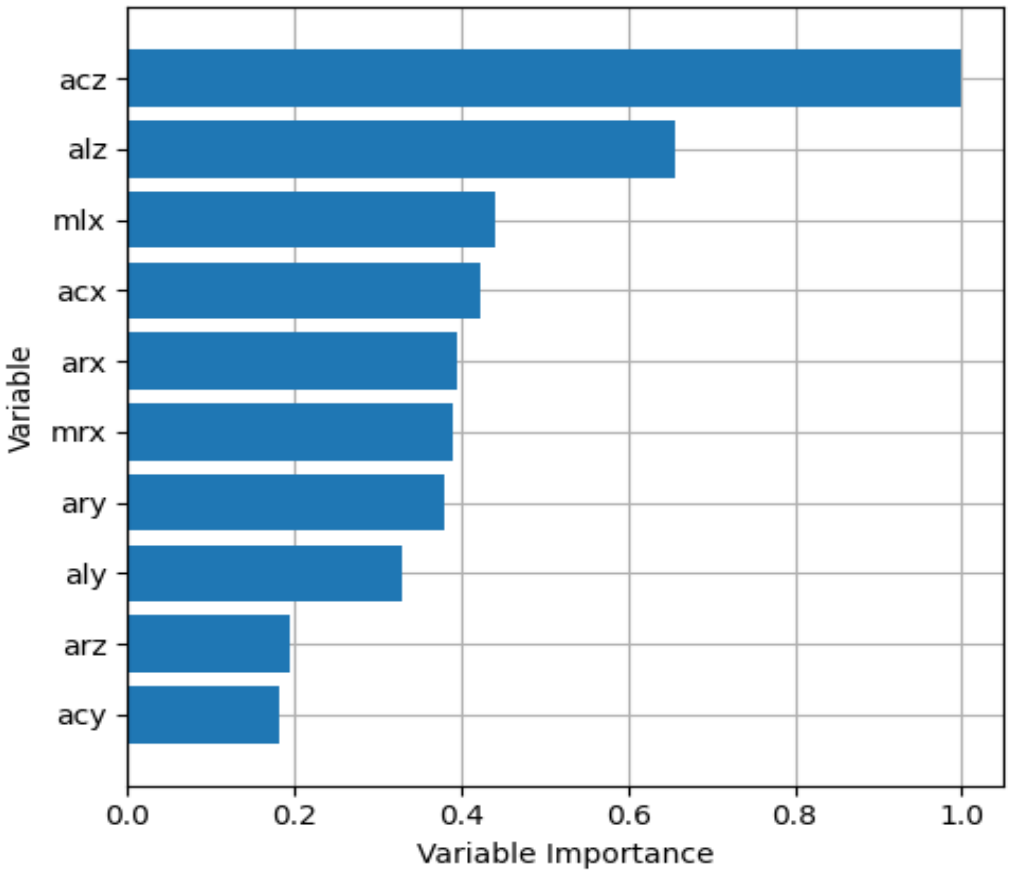


Fig. 3.20 Clasificación de la importancia de las variables. Fuente: [autoría propia].

3.7.4 Mapa de calor

También se obtuvo un mapa de calor que muestra la relación entre las características y algunos de los modelos observados en la Tabla 3.28.

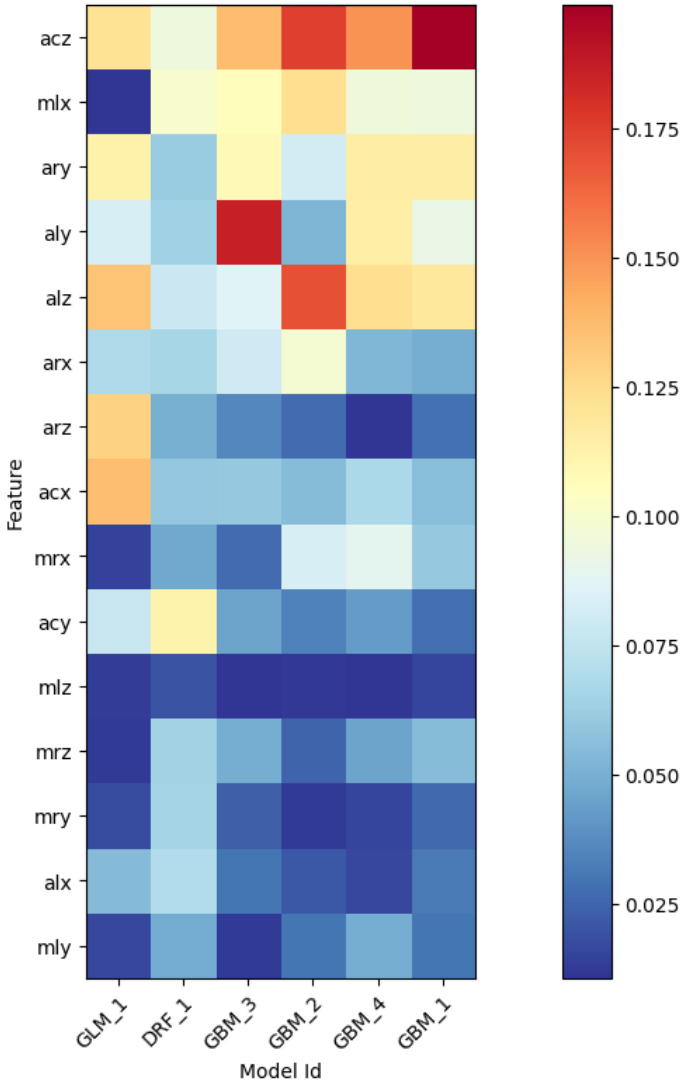


Fig. 3.21 Mapa de calor, importancia de variable vs modelos. Fuente: [autoría propia].

Al visualizar la Fig. 3.21 se puede apreciar que las variables más influyentes para los diferentes modelos se ubican entre la variable acz y la variable alz, aunque hay excepciones como en el caso del modelo DRF, cuyo variable más influyente es acy, y la variable acx para el modelo GLM. El resto de los modelos tienen sus variables más influyentes entre la variable arx y acz, siendo aly la variable más importante para el modelo GBM 3 y acz la más importantes para los modelos GBM 1, GBM 2 y GBM 4. Finalmente se puede notar que la variable acz afecta en mayor medida a los modelos GBM, con la excepción de GBM 3.

3.7.5 Correlación entre modelos

La Fig. 3.22 presenta la matriz de correlación entre todos los modelos generados por el módulo H20, la correlación está basada en los resultados predictivos que se obtuvieron de cada modelo, es decir que su correlación está estructurada según la similitud de sus resultados predictivos, por lo cual al observa la matriz, se observa que los resultados obtenidos por el modelo GLM difieren en gran medida a los obtenidos por el resto de modelos, siendo los modelos StackedEnsemble_BestOfFamily_2 y StackedEnsemble_AllModels_1 los que menos difieren, por el contrario se observa que el modelo StackedEnsemble_BestOfFamily_1 presente en la misma tabla tiene resultados más acorde a lo obtenido por todos los modelos GBM y el modelo DRF, sin llegar a ser resultados idénticos.

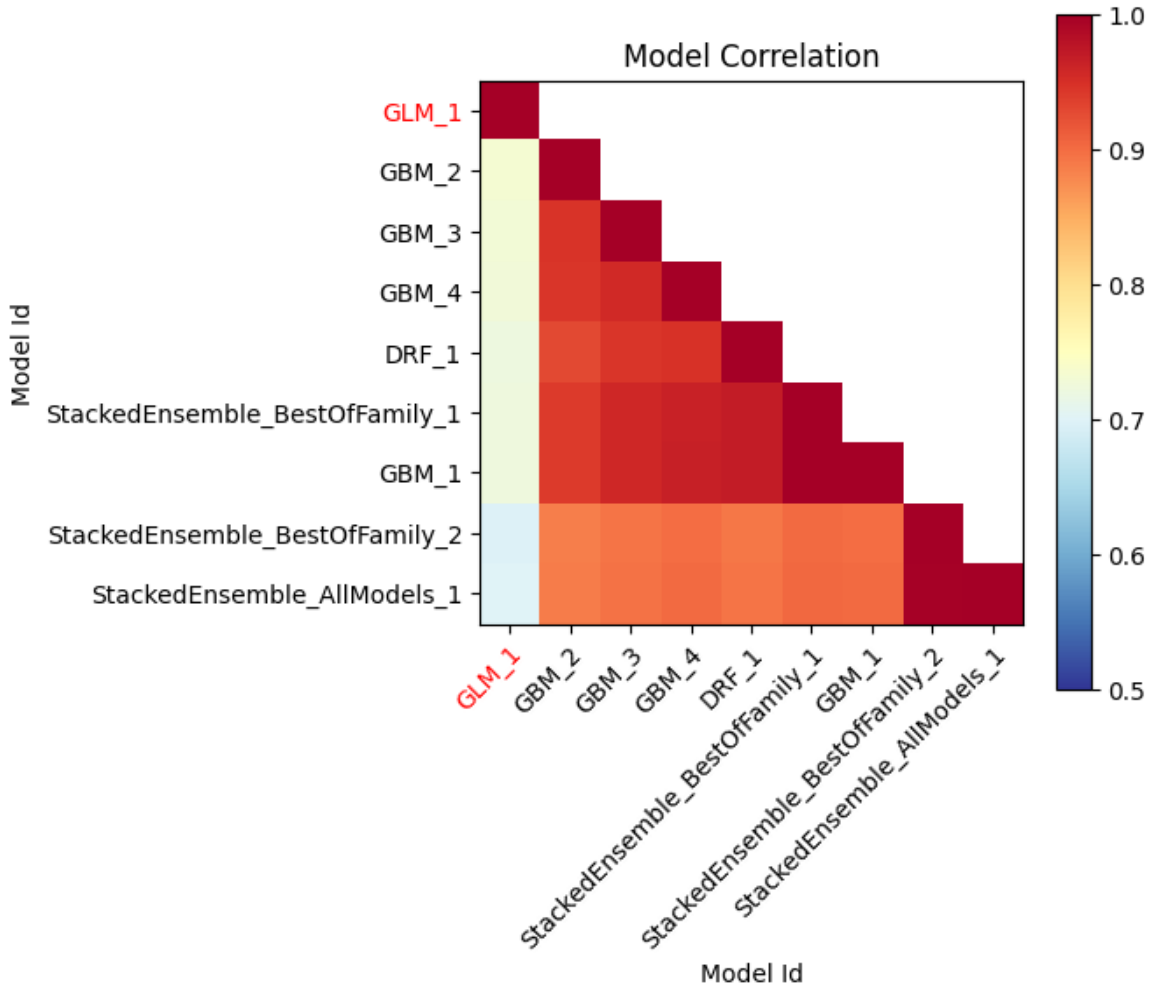


Fig. 3.22 Matriz de correlación. Fuente: [autoría propia].

3.7.6 Análisis de dependencia parcial

A continuación, se presentan los gráficos de dependencia parcial, cuyo objetivo es representar la interacción que sucede entre una variable y la clase predicha cuando la variable varía su valor mientras el resto de las variables mantienen un valor fijo. En el eje horizontal se puede observar el rango de valores que posee la variable y en el eje vertical representa la respuesta media por parte de la variable, es decir representa de manera numérica la influencia sobre la clase en cuestión. Las curvas serán ilustradas para las cuatro categorías de modelos: GBM, StackedEnsemble, DRF y GLM. Estos gráficos reflejarán la relación entre estos modelos y la variable "acz", que, como se analizó previamente, es la más influyente en los resultados.

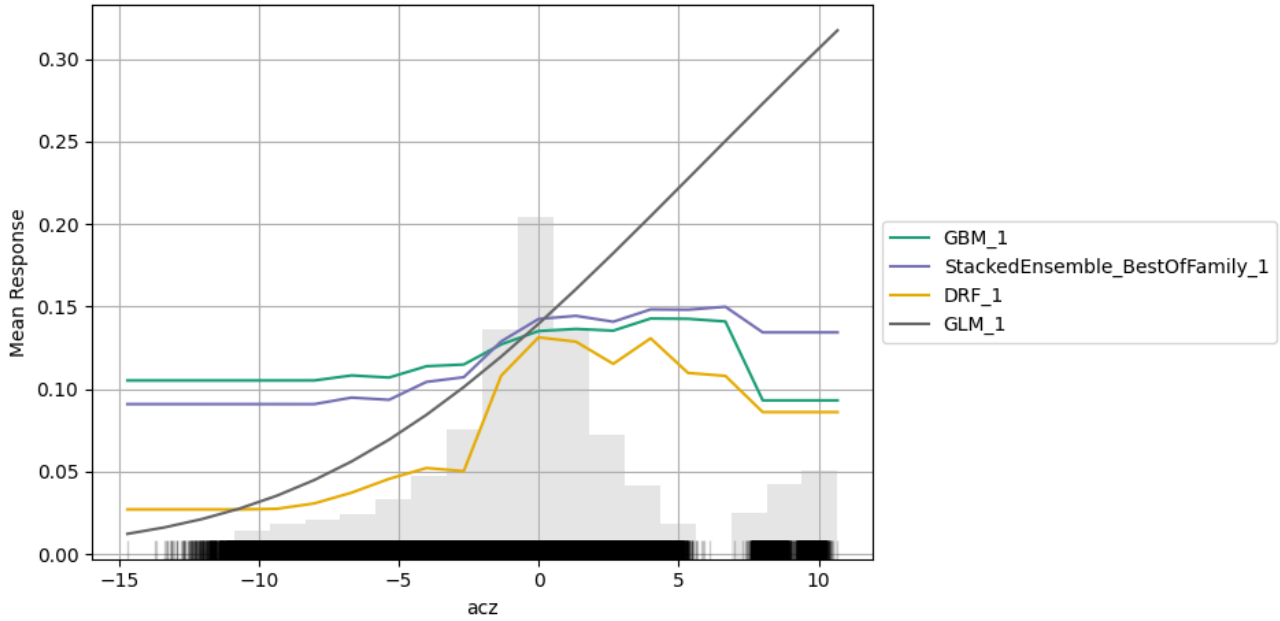


Fig. 3.23 Influencia de acz sobre la clase “De pie”. Fuente: [autoría propia].

En la Fig. 3.23, se observa la respuesta de la clase 1 de cada modelo a medida que varía la variable "acz". Los modelos GBM y StackedEnsemble tienen comportamientos similares en el rango de -15 a -2, con el GBM ligeramente más alto. Entre -2 y 11, el StackedEnsemble es más influenciado por "acz". El modelo DRF tiene un patrón similar, pero con variaciones ascendentes y descendentes que no superan a los anteriores. El modelo GLM tiene comportamientos exponenciales y lineales entre -15 a 0 y 0 a 11, respectivamente, superando a los otros modelos. El GBM alcanza una respuesta de alrededor de 0.145 sobre la clase “De pie”.

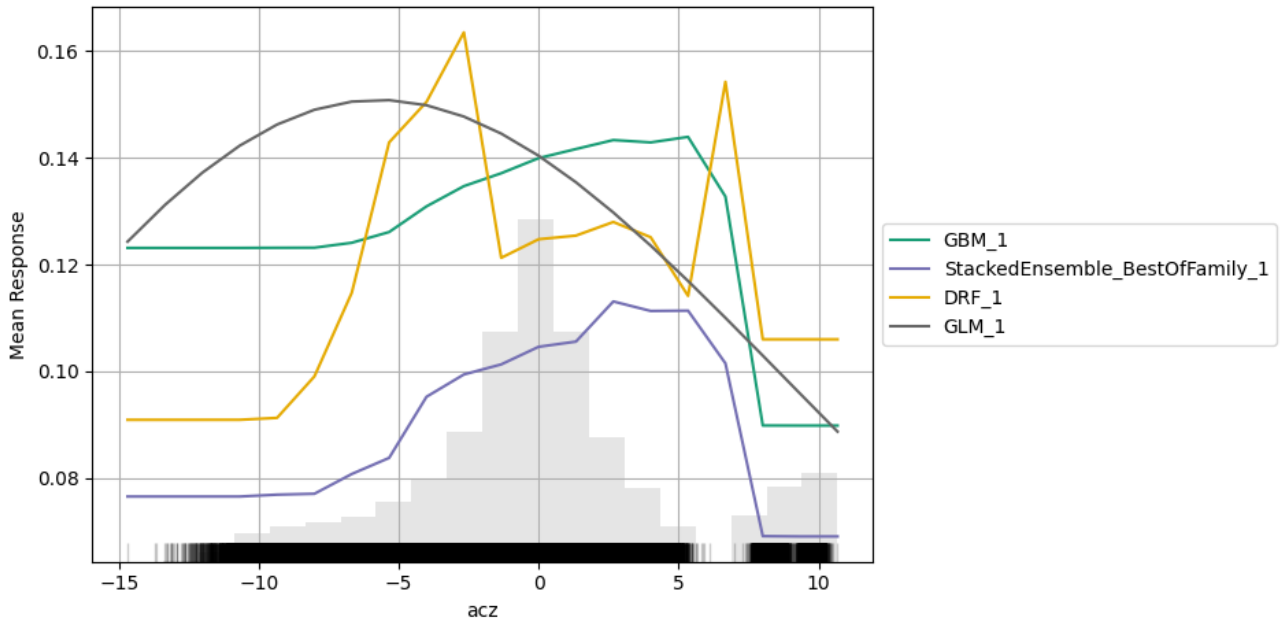


Fig. 3.24 Influencia de acz sobre la clase “Sentado”. Fuente: [autoría propia].

Se puede observar en la Fig. 3.24 que la dependencia de la clase "sentado" muestra una baja en los rangos de -15 a -8 para DRF y StackedEnsemble, con StackedEnsemble presentando la respuesta más baja. Los modelos restantes tienen comportamientos distintos entre sí y frente a los dos mencionados. El modelo GBM muestra una respuesta constante cerca de 0.126 a medida que aumenta acz. Aproximadamente en -5, GBM y StackedEnsemble aumentan su dependencia en la variable acz. Por su parte DRF tiene el mayor crecimiento, pero en un rango limitado, mientras que StackedEnsemble también aumenta, pero en menor medida. Cerca del valor 7.5 de acz, la influencia de la variable disminuye en todos los modelos. Se destaca la alta y constante dependencia del modelo GBM en un amplio rango de valores de acz. Por otra parte, el modelo GLM tiene un comportamiento parabólico similar a la Fig. 3.23, creciendo y luego decreciendo con acz.

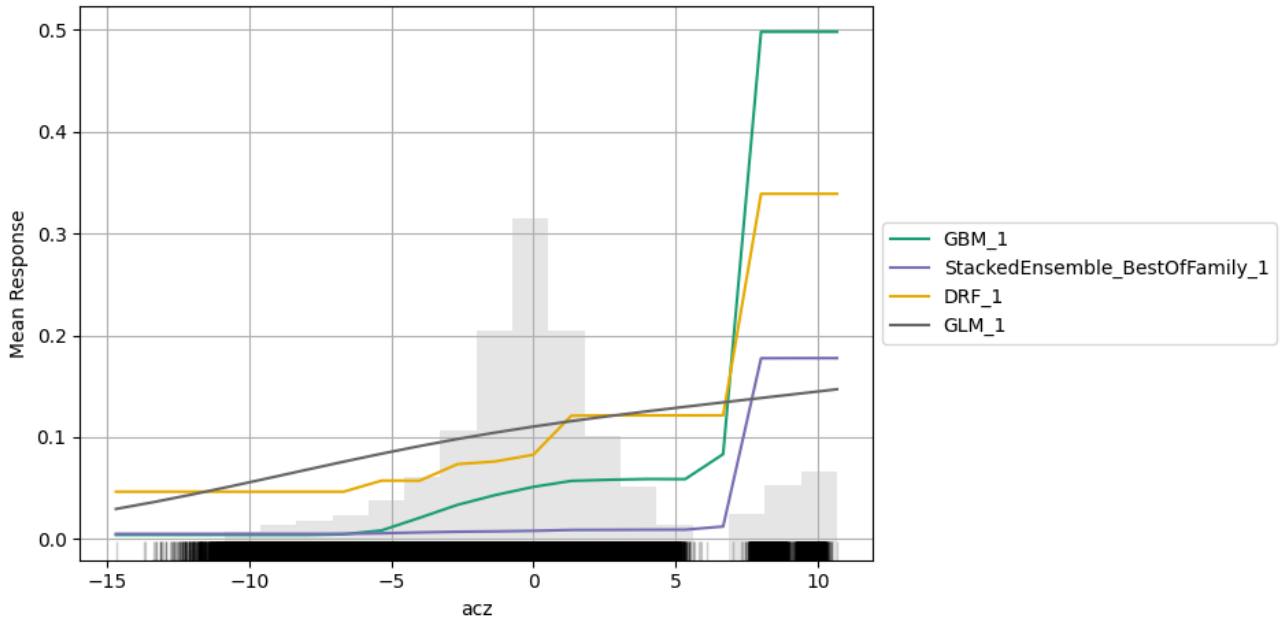


Fig. 3.25 Influencia de acz sobre la clase "Acostado". Fuente: [autoría propia].

Se observa que los cuatro modelos tienen una baja dependencia inicial en la clase "acostado", incluso menor que en la Fig. 3.24. Los modelos GBM y StackedEnsemble muestran una dependencia constante cercana a 0 hasta que acz llega a -5. El modelo DRF también tiene una dependencia constante en ese rango, pero ligeramente mayor. A medida que acz aumenta, la dependencia de la clase en los diferentes modelos también incrementa. La respuesta de GBM crece de manera logarítmica, diferenciándose del StackedEnsemble, cuya dependencia crece suavemente en una línea con pendiente baja. Alrededor de acz igual a 7, la respuesta de los tres modelos aumenta significativamente, especialmente DRF y GBM, superando el valor de 0.3 de respuesta. El modelo GLM muestra un comportamiento lineal con aumento gradual.

Hasta el momento, la clase "acostado" es la que ha sido más influenciada por la variable acz, para el modelo previamente seleccionado, GBM.

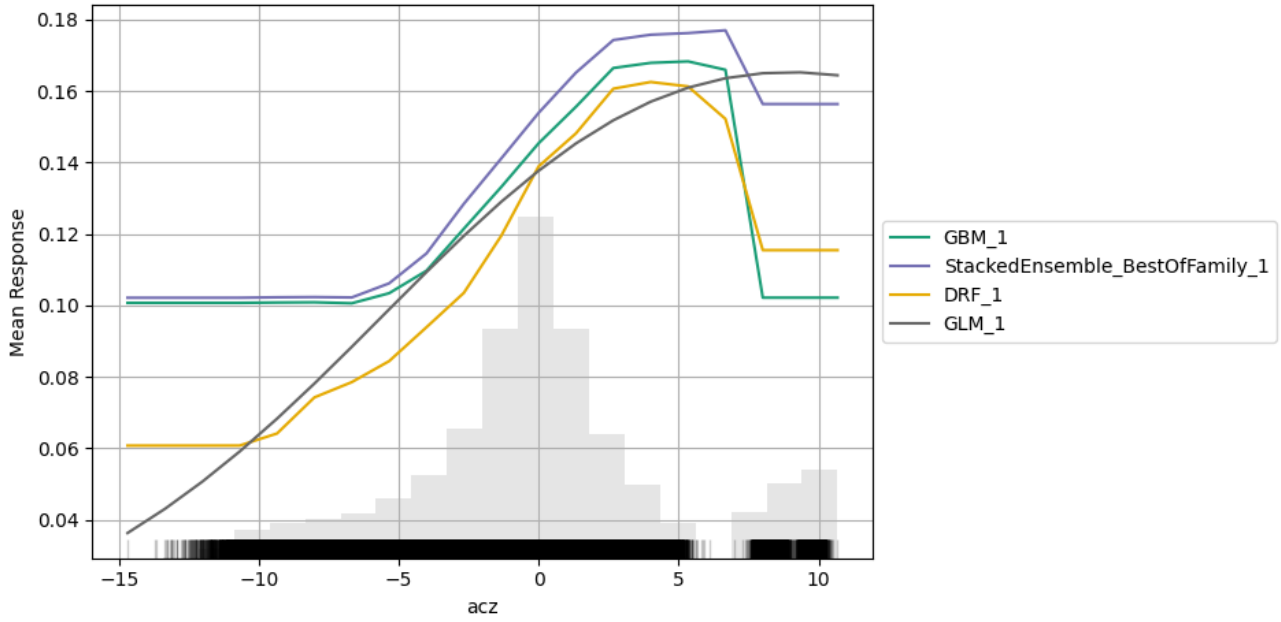


Fig. 3.26 Influencia de acz sobre la clase “Caminar”. Fuente: [autoría propia].

En la Fig. 3.26, el modelo GBM y StackedEnsemble muestran un comportamiento similar en la respuesta media a lo largo de todo el rango de valores de acz. Inician con un valor constante entre 0.10 y 0.12 en el eje vertical, siendo StackedEnsemble el que tiene mayor dependencia de la clase sobre la variable. Esta diferencia persiste mientras acz crece, con cambios más pronunciados en DRF reflejados en rectas de distintas pendientes. En valores altos de acz, DRF tiene la menor dependencia visible en la figura.

El modelo GLM sigue un patrón similar a la Fig. 3.25, creciendo su dependencia a medida que crece acz, siendo este modelo el de menor dependencia en valores bajos de acz y mayor dependencia en valores altos de acz. El valor máximo de dependencia alcanzado por el modelo principal GBM en la actual figura, es menor a la mitad del valor alcanzado en la Fig. 3.25.

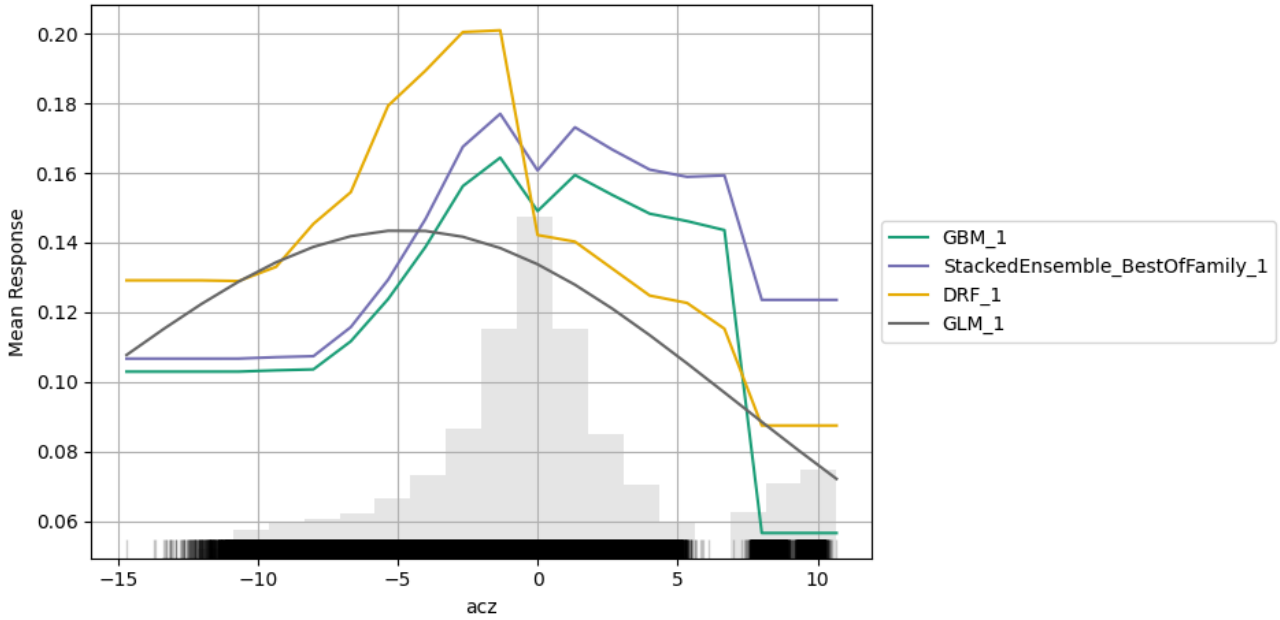


Fig. 3.27 Influencia de acz sobre la clase “Subir escaleras”. Fuente: [autoría propia].

De la Fig. 3.27 se observa que el comportamiento de los modelos GBM y StackedEnsemble en los valores más bajos de acz, es el mismo que se observó en la figura anterior, alcanzando valores casi constantes en el eje vertical, siendo ahora levemente superior la dependencia por parte del modelo DRF, luego de superar el valor aproximado de -8 en el eje horizontal, los 3 modelos comienzan a elevar su dependencia en gran medida en un rango de valores acotados de la variable acz, luego de alcanzar el valor máximo de dependencia, los 3 modelos comienzan a comportarse de manera similar, disminuyendo algunos su valor en el eje vertical en menor o mayor medida, hasta alcanzar la menor dependencia asociada a la clase 5, alcanzando el valor más bajo el modelo GBM.

El modelo GLM presenta el mismo comportamiento observado en la Fig. 3.24, una parábola que no llega a superar la dependencia máxima alcanzada por los otros modelos, excepto entre los rangos -15 a -5 donde logra superar al modelo GBM y StackedEnsemble.

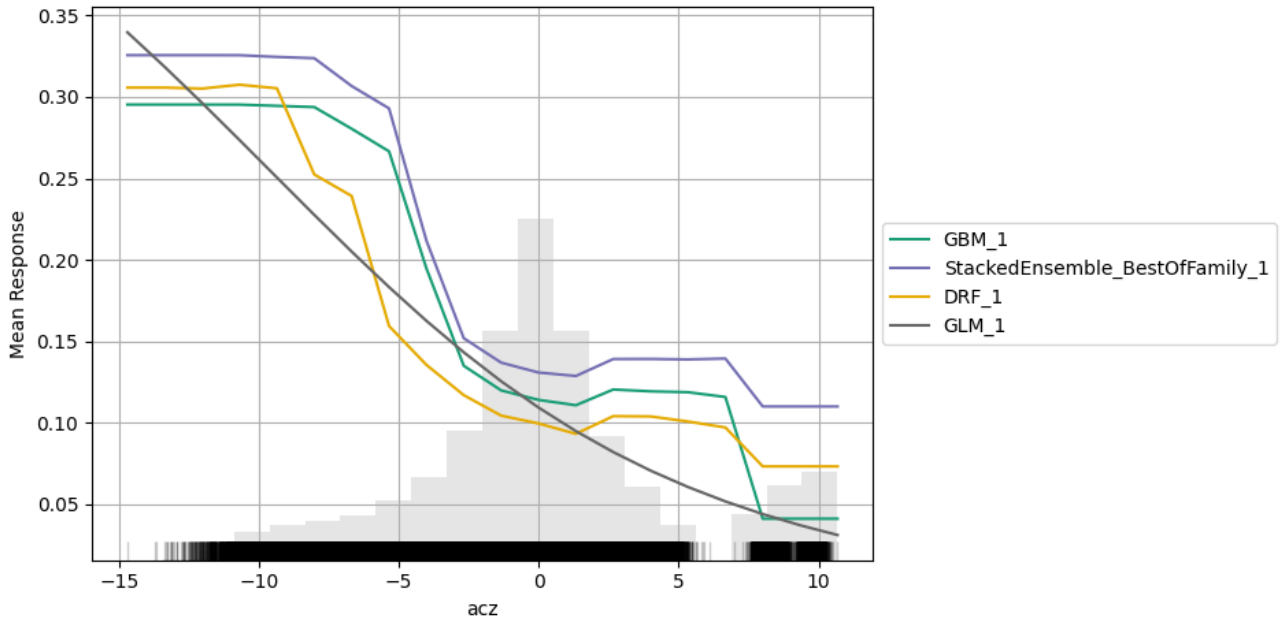


Fig. 3.28 Influencia de acz sobre la clase "Inclinarse". Fuente: [autoría propia].

La Fig. 3.28 muestra curvas altamente similares entre los modelos, incluyendo GLM. Todos los modelos tienen máxima dependencia en el valor más bajo de acz, seguido por un descenso en el valor de respuesta media. GLM inicia su descenso inmediatamente cuando acz en aumenta, con una pendiente negativa constante hasta el valor 0 de acz, donde la pendiente disminuye antes de llegar al mayor valor de acz.

Para GBM y StackedEnsemble, la dependencia es constante al aumentar acz, pero cambia al acercarse al valor -5 de acz, luego descienden con pendiente negativa considerable, llegando a menos de la mitad de la respuesta media inicial. Hay un pequeño aumento antes de descender a un valor de respuesta media constante en valores altos de acz.

DRF tiene un comportamiento similar, con ligera diferencia en el valor de acz donde disminuye la dependencia de clase actual, comenzando a disminuir cerca del valor -11 de acz.

GBM es el segundo modelo más influido por acz al predecir la clase "Inclinarse".

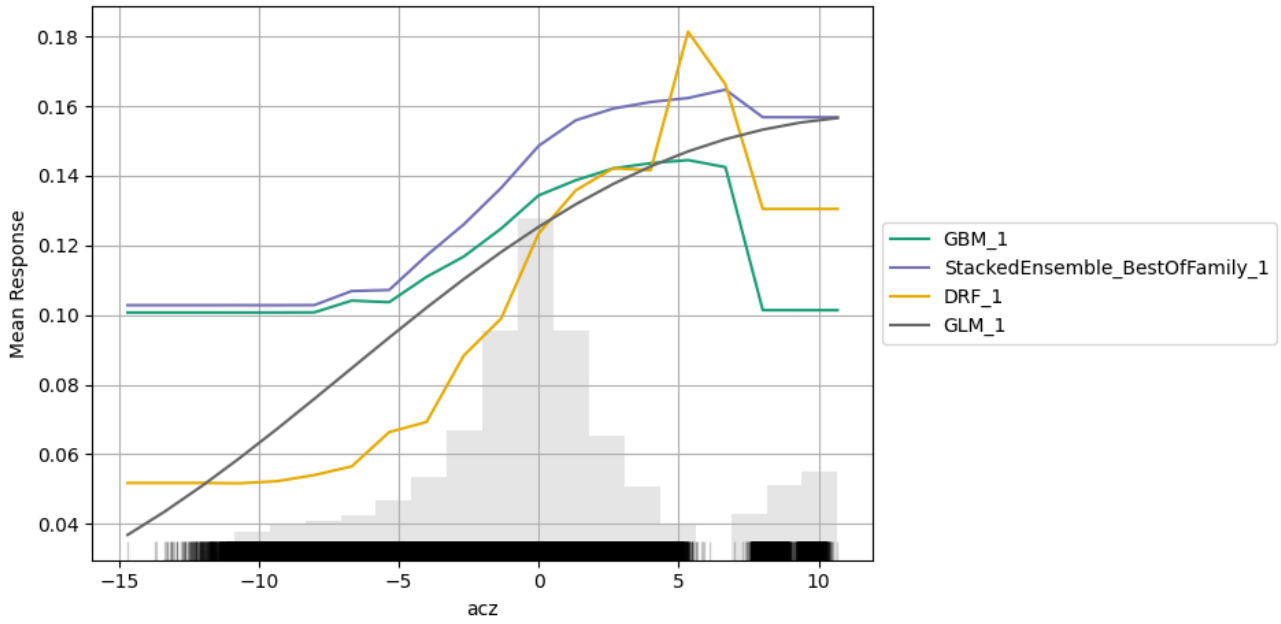


Fig. 3.29 Influencia de acz sobre la clase "Elevar brazos". Fuente: [autoría propia].

La mayor influencia de la variable acz sobre la séptima clase se ve en el modelo StackedEnsemble, el cual presenta un mayor valor en el eje vertical durante gran parte del recorrido de los valores de acz, valor de respuesta que es superado por el modelo DRF en un rango acotado de valores de acz, posteriormente este último modelo vuelve a tener un valor de respuesta menor, ubicándose entre el valor de respuesta del modelo StackedEnsemble y GBM.

La respuesta de la clase 7 del modelo GBM se ubica como la segunda más alta durante el rango desde -15 hasta 2 de la variable acz, estando por debajo del modelo destacado en el párrafo anterior, luego el modelo tiene la peor respuesta entre los valores más altos de acz.

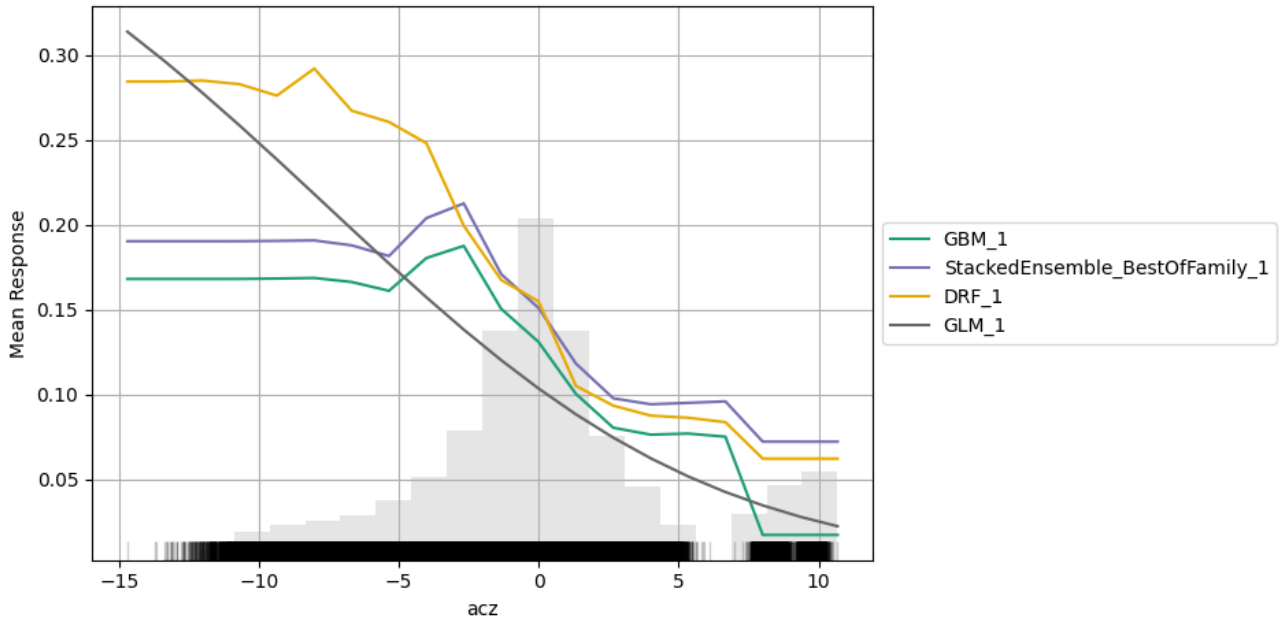


Fig. 3.30 Influencia de acz sobre la clase *Doblar rodillas*. Fuente: [autoría propia].

La dependencia de la clase “doblar rodillas” en todos los modelos tiene un comportamiento similar al de la Fig. 3.28. En el modelo GLM, la dependencia es mayor cuando acz es menor a -13, pero otros modelos tienen mayor respuesta cuando acz supera ese valor, relegando al modelo GLM a la última posición en dependencia.

En el modelo DRF, la influencia de acz en la clase es mayor en valores negativos de la variable, disminuyendo cuando acz se acerca a su valor más alto. El modelo StackedEnsemble es el modelo más influido al predecir esta última clase, en el intervalo de valores más altos de acz.

El modelo GBM muestra la segunda menor influencia en la clase por la variable acz, superando solo al modelo GLM.

Finalmente se puede observar en las figuras previamente analizadas, que, si bien el modelo GBM se presentó como el mejor modelo, este no presento una superioridad evidente en la dependencia de las clases asociadas a la variable acz, la cual también se presentó como la variable más determinante al momento de hacer las predicciones. Por lo cual el variar solo el valor de acz no es suficiente para que el modelo pueda discriminar correctamente cada clase.

4. Conclusiones

4.1. Sumario

Se obtuvo una base de datos desde una página externa, mediante un software de programación se construyeron 5 modelos de aprendizaje automático previamente seleccionados. Luego, mediante un segundo software enlazado al primero, se generó una serie de nuevos modelos ajustados a la base de datos, de los cuales se seleccionó el mejor, para que se convirtiera en el sexto modelo en ser considerado dentro de la investigación. Los 6 modelos se pusieron a prueba y se ajustaron respectivamente con el objetivo de obtener los mejores resultados posibles.

Los resultados fueron comparados mediante ciertas métricas, para establecer que modelos presentaron los mejores resultados de cara al objetivo de caracterizar los patrones de conducta de las personas, a través de la predicción o clasificación de las actividades realizadas por las personas en su día a día.

4.2. Conclusiones

Las técnicas de aprendizaje automático supervisado son una buena base para comenzar a comprender sobre los patrones presentes en la rutina diaria de una persona. Si bien existen muchas técnicas para comprender o reconocer ciertos patrones, los modelos GBM, KNN y MLP demostraron ser buenas técnicas para incursionar en el reconocimiento de actividades asociadas al ser humano. Los 3 modelos presentaron fallas al momento de clasificar un conjunto de datos, pero se debe tener en cuenta que la similitud entre los datos de cada clase pudo generar un criterio más deficiente de clasificación en los modelos. También se debe considerar que la dimensionalidad de los datos afecta a algunos modelos como el KNN, aunque no se vio demostrado en los resultados debido al criterio que usa este modelo. Si bien se buscó reducir el tamaño de la base de datos, esto terminó por perjudicar a los modelos, exceptuando al modelo GMB, que logró generalizarse en función de la base de datos reducida. El resto de los modelos se vieron perjudicados debido a que se eliminaron características por lo cual la ausencia de esas características limito la cantidad de criterios que se podían considerar para reconocer cada clase.

Respecto a la eficacia de los clasificadores, el modelo GBM se considera el más apto para aplicar en la tarea de reconocimiento de actividades dentro de una casa inteligente por su bajo error asociado, luego también es válido considerar los modelos KNN y MLP, que presentaron un buen desempeño aunque con más complicaciones de lo esperado, para que estos últimos 2 modelos puedan

ser aplicados, se debería hacer al menor 3 consideraciones, la primera consideración es poner mayor atención en los sensores necesarios para entrenar los modelos, también se debe considerar las características de las personas que residen en la casa, y por último se debe considerar el tiempo que toma entrenar cada modelo y el tiempo que se demoran en reconocer cada actividad dentro de la casa.

Para los modelos que arrojaron menor rendimiento se debería considerar hacer un mejor procesamiento de los datos antes de ser entrenados, ya que las características ofrecidas por la base de datos no fueron suficientes para alcanzar un rendimiento considerable.

4.3. Trabajo futuro

Como trabajo futuro se espera trabajar en el proceso de obtención de datos de manera local, lo que conlleva diseñar y simular un ambiente dentro de una casa, la cual tendrá una serie de sensores con los cuales obtener los datos. Se debe considerar en el proceso de diseño, la disposición de los sensores, ya que no deben ser invasivos pues alteraran las conductas diarias de las personas. También se espera poder realizar un mejor procesamiento de los datos, ya que se pueden encontrar características implícitas dentro de las mismas características obtenidas por los sensores. La selección de las mejores características es un proceso que igual se debe mejorar, ya que de esto depende que un modelo sea entrenado de manera eficiente o deficiente, considerando también el tiempo que le lleva al modelo procesar los datos.

Se buscará el adaptar los sistemas de reconocimientos a las nuevas tecnologías relacionadas a la domótica y la obtención de información sobre las personas en su vida diaria.

Por último, sería un gran avance poder incorporar técnicas de aprendizaje profundo, debido a que se desenvuelven mejor cuando las bases de datos son muy grandes.

Bibliografía

[1] Truong, Phuc & You, Sujeong & Sang Hoon, Ji & Jeong, Gu-Min. (2019). Wearable System for Daily Activity Recognition Using Inertial and Pressure Sensors of a Smart Band and Smart Shoes. International Journal of Computers Communications & Control. 14. 726-742.

[2] C. -T. Yen, J. -X. Liao and Y. -K. Huang, "Human Daily Activity Recognition Performed Using Wearable Inertial Sensors Combined With Deep Learning Algorithms," in IEEE Access, vol. 8, pp. 174105-174114, 2020.

[3] Lotfi, A., Langensiepen, C., Mahmoud, S.M. *et al.* Smart homes for the elderly dementia sufferers: identification and prediction of abnormal behaviour. *J Ambient Intell Human Comput* **3,** 205–218 (2012).

[4] Fiorini, L., Bonaccorsi, M., Betti, S., Esposito, D., & Cavallo, F. R. (2018). Combining wearable physiological and inertial sensors with indoor user localization network to enhance activity recognition. Journal of Ambient Intelligence and Smart Environments, 10(4), 345-357.

[5] Sbai, Samir & Kraisakdawat, Anan. (2019). Human Activity Recognition, comparison of machine learning approaches.. 10.13140/RG.2.2.26639.94881. [5].

[6] *"Estadística y Machine Learning con R"*, Francisco Parra, libro electrónico, 2019.

[7] Redes neuronales. (22 oct 2019). ATRIA INNOVATION.

[8] Sensor inercial. (2019, 13 de octubre). *Wikipedia, La enciclopedia libre*.

Fecha de consulta: 00:50, mayo 24, 2022 desde https://es.wikipedia.org

[9] D. Segovia. *Adquisición de datos desde un sensor inercial*. Universidad Carlos III de Madrid, 2018.

[10] Sensor de pulso. (s.f**.).** World Famous Electronics llc. Disponible: https://pulsesensor.com/

Disponible: https://www.atriainnovation.com/que-son-las-redes-neuronales-y-sus-funciones/

[11] *Banos,Oresti, Garcia,Rafael, and Saez,Alejandro. (2014). MHEALTH Dataset. UCI Machine Learning Repository.* Disponible:https://doi.org/10.24432/C5TW22.

[12] Google Colaboraty. (s.f.). IArtificial.net.

Disponible: https://colab.research.google.com/?hl=es

[13] *The H2O Python Module — H2O documentation*. (s. f.-c).

Disponible: https://docs.h2o.ai/h2o/latest-stable/h2o-py/docs/intro.html

[14] *Gradient Boosting Machine with H2O | H2O.ai. (s. f.).*
Disponible:https://h2o.ai/resources/booklet/gradient-boosting-machine-with-h2o/

[15] Govindaraju, Venu & Raghavan, Vijay & Rao, C.R.. (2015). Big Data Analytics: Handbook of statistics.

[16] Refaeilzadeh, P., Tang, L., & Liu, H. (2009). Cross-Validation. En Springer eBooks (pp. 532-538). Disponible: https://doi.org/10.1007/978-0-387-39940-9_565

[17] Métricas. (s.f.). IArtificial.net.

Disponible: https://www.iartificial.net/precision-recall-f1-accuracy-en-clasificacion/

[18] Ellis, C. (2023, 21 enero). *Oversampling vs undersampling for machine learning. Crunching the Data.* Disponible: https://crunchingthedata.com/oversampling-vs-undersampling/

[19] Breiman, L. Random Forests. *Machine Learning* **45**, 5–32 (2001).

Anexo A. Matrices de Confusión de Base de Datos Limpia

A.1. Matriz de confusión KNN sin gridsearch.

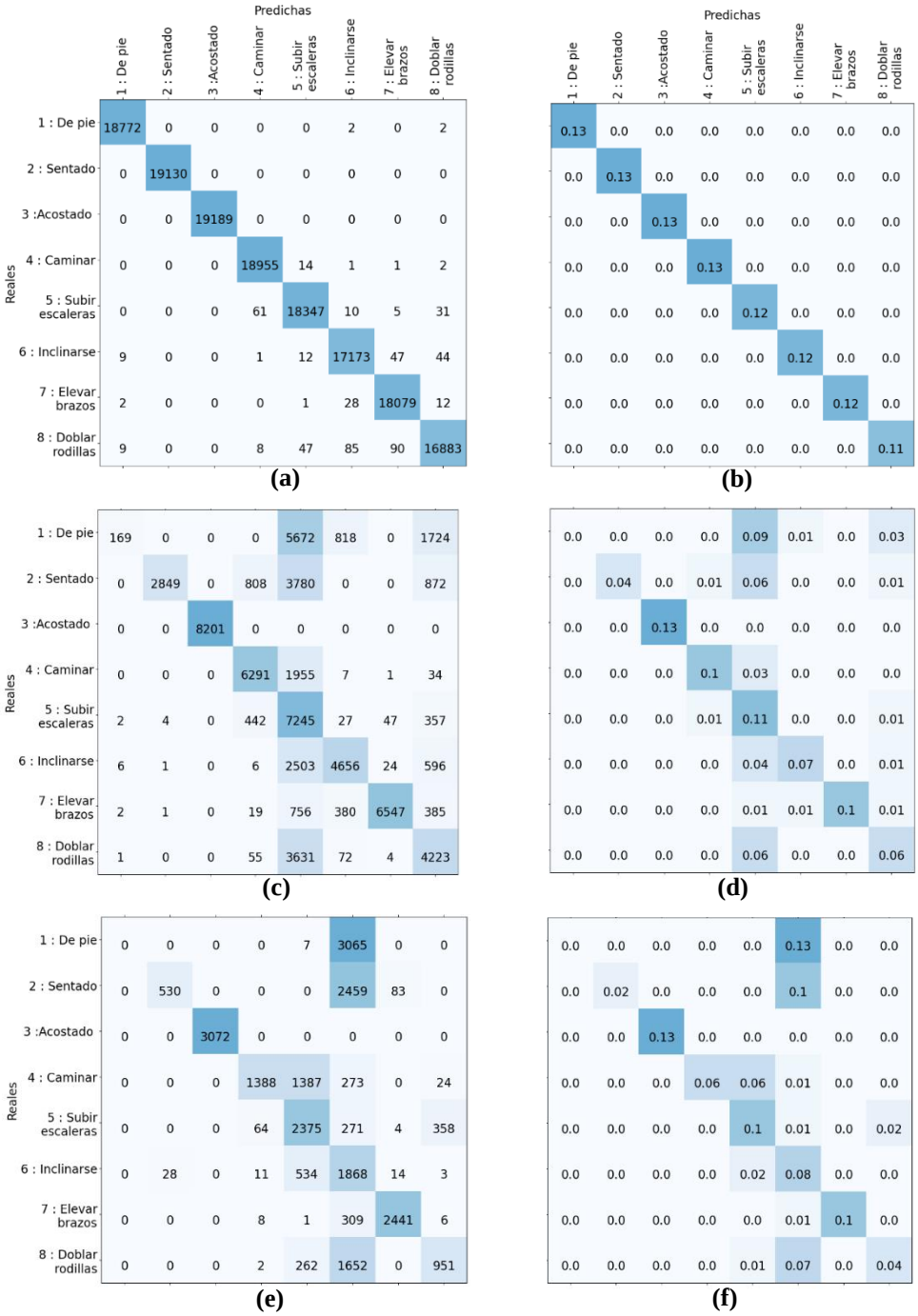


Fig. A.1 Modelo KNN 1 con Base de Datos Limpia, Fuente: [autoría propia].

(a), (b) Entrenamiento (c), (d) Prueba (e),(f) Validación.

A.2. Matriz de confusión KNN con gridsearch.

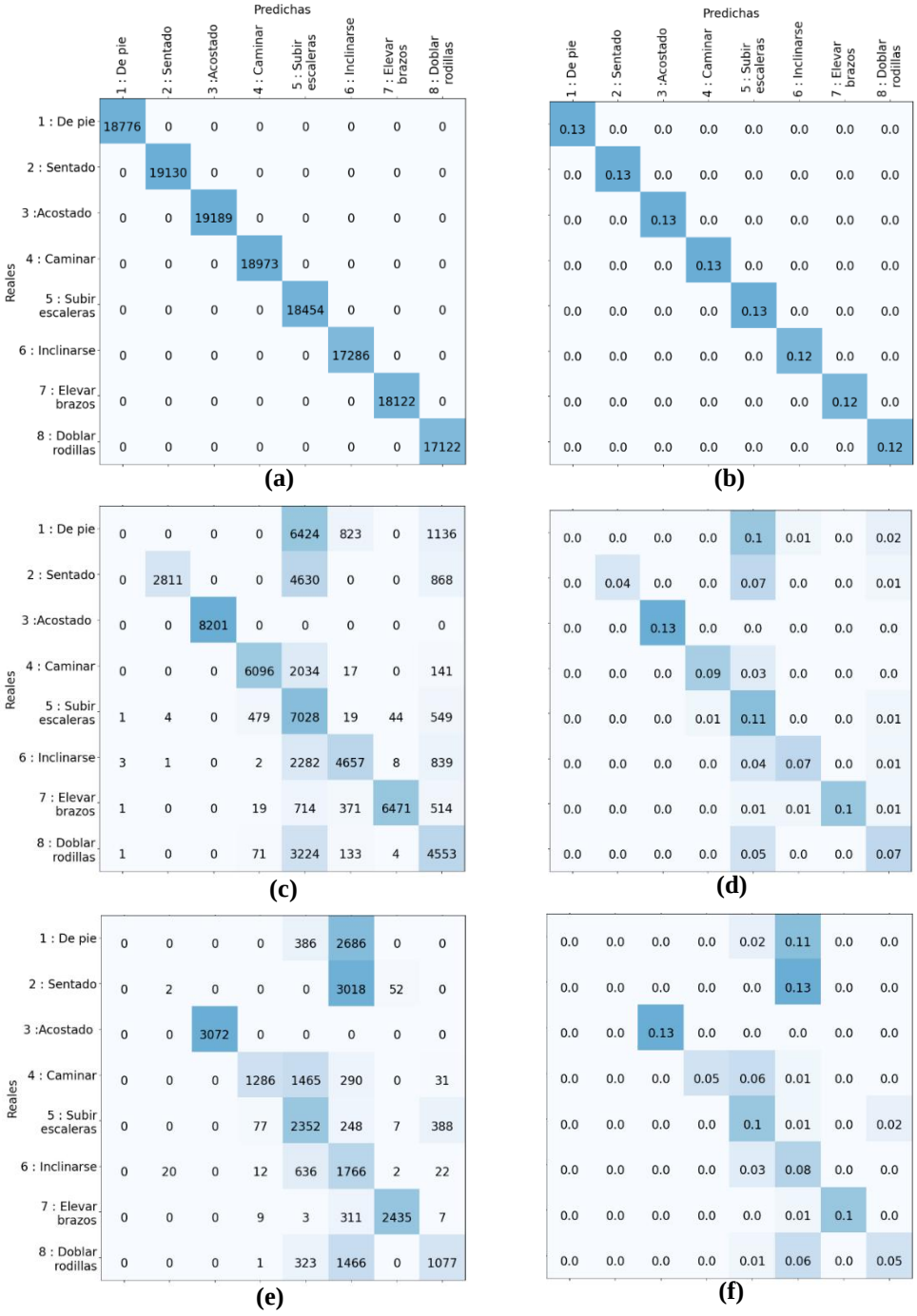


Fig. A.2 Modelo KNN 2 con Base de Datos Limpia, Fuente: [autoría propia].

(a), (b) Entrenamiento (c), (d) Prueba (e),(f) Validación.

A.3. Matriz de confusion Árbol de Decisión sin gridsearch.

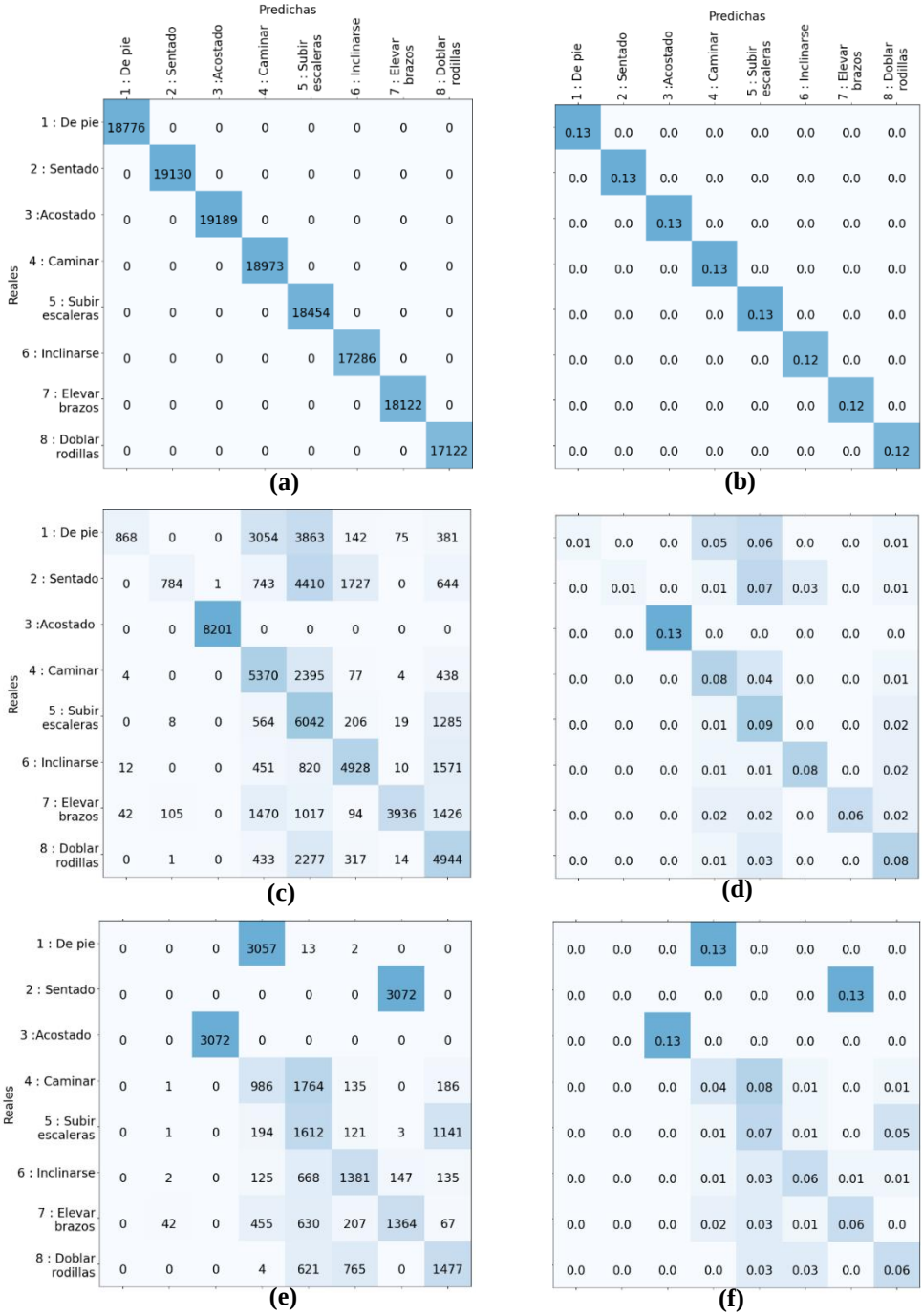


Fig. A.3 Modelo DT 1 con Base de Datos Limpia, Fuente: [autoría propia].

(a), (b) Entrenamiento (c), (d) Prueba (e),(f) Validación.

A.4. Matriz de Confusión Árbol de Decisión con Gridsearch.

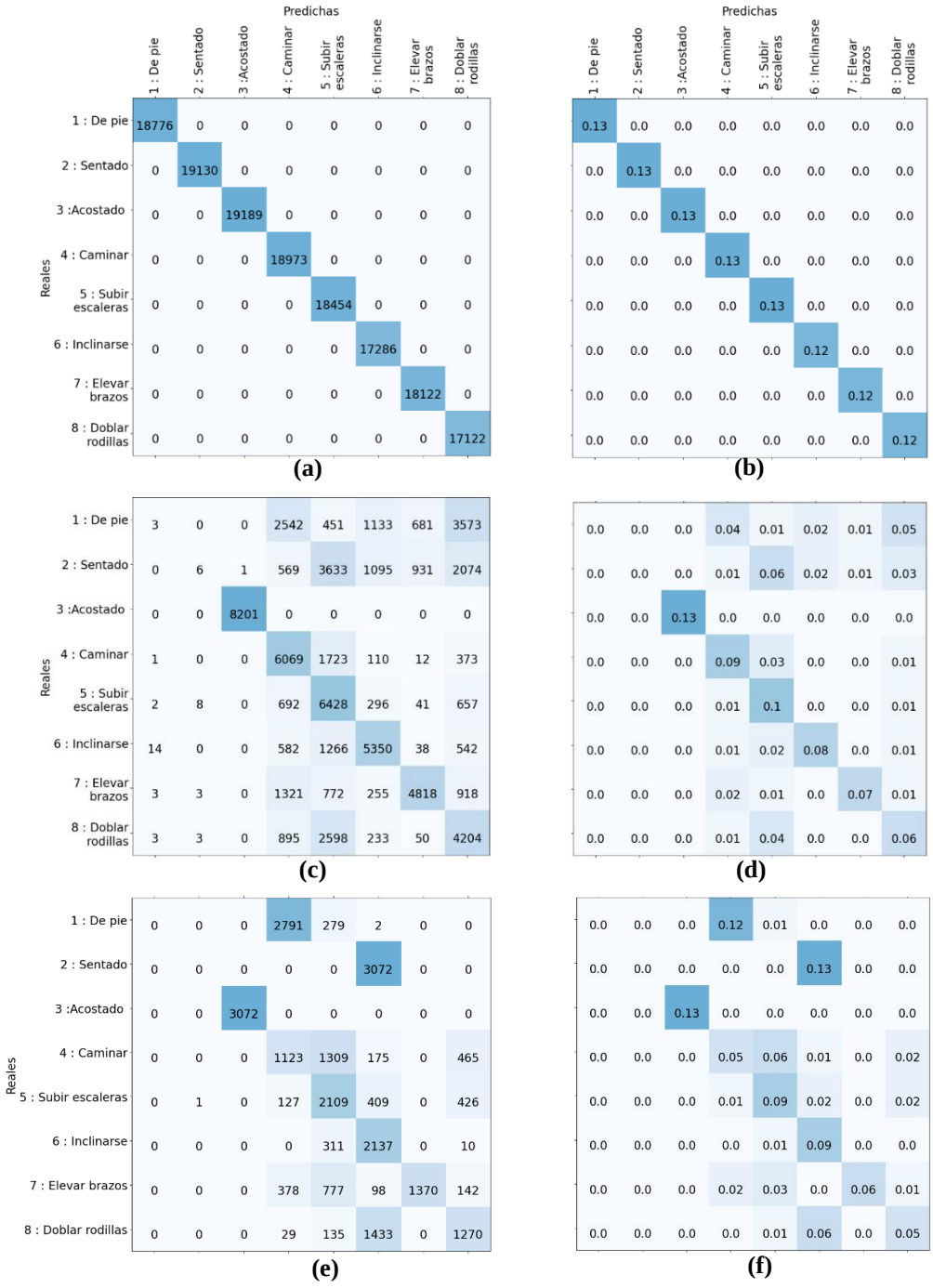


Fig. A.4 Modelo DT 2 con Base de Datos Limpia, Fuente: [autoría propia].

(a), (b) Entrenamiento (c), (d) Prueba (e),(f) Validación.

A.5. Matriz de Confusión Random Forest sin Gridsearch.

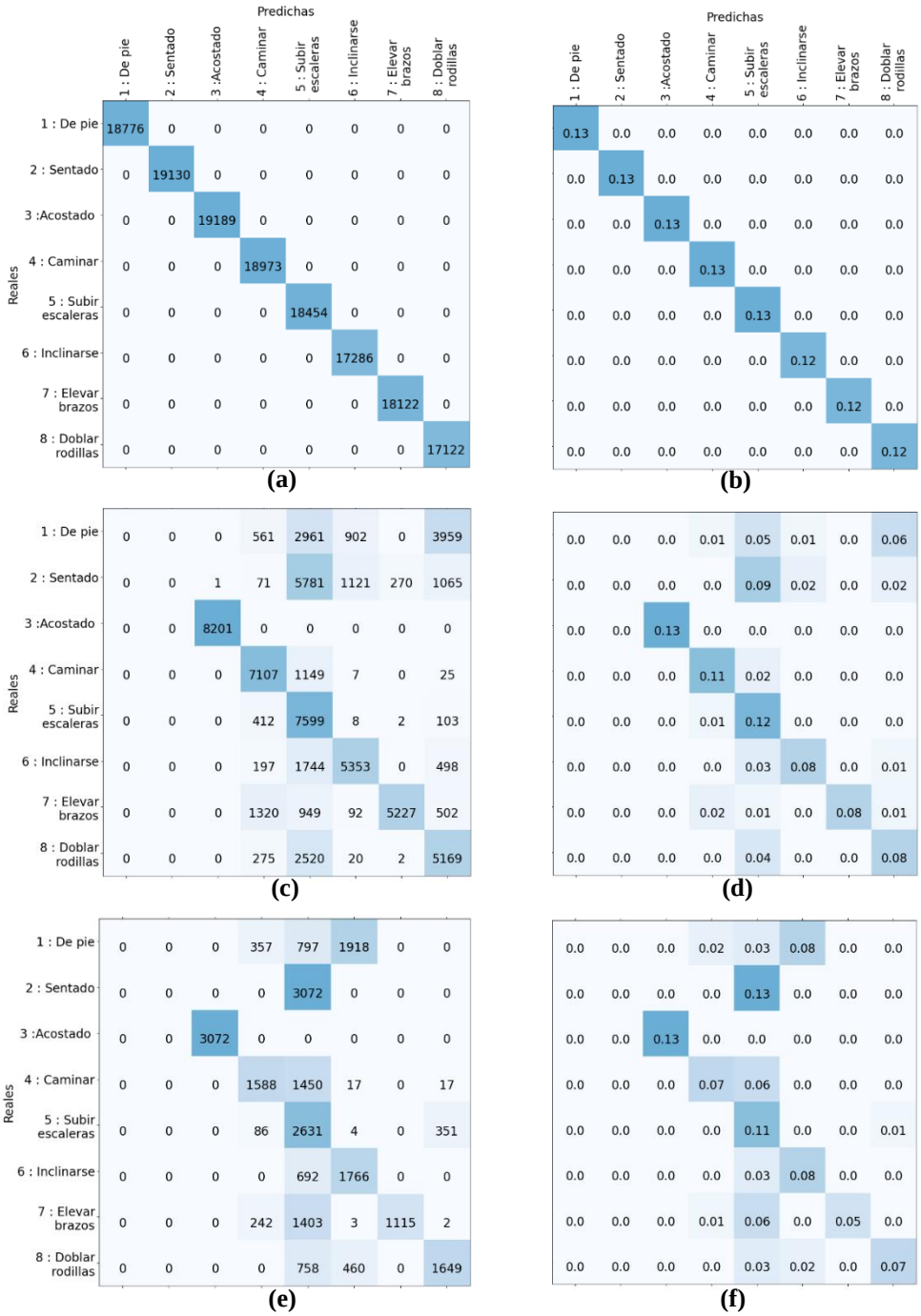


Fig. A.5 Modelo RF 1 con Base de Datos Limpia, Fuente: [autoría propia].

(a), (b) Entrenamiento (c), (d) Prueba (e),(f) Validación.

A.6. Matriz de Confusión Random Forest con Gridsearch.

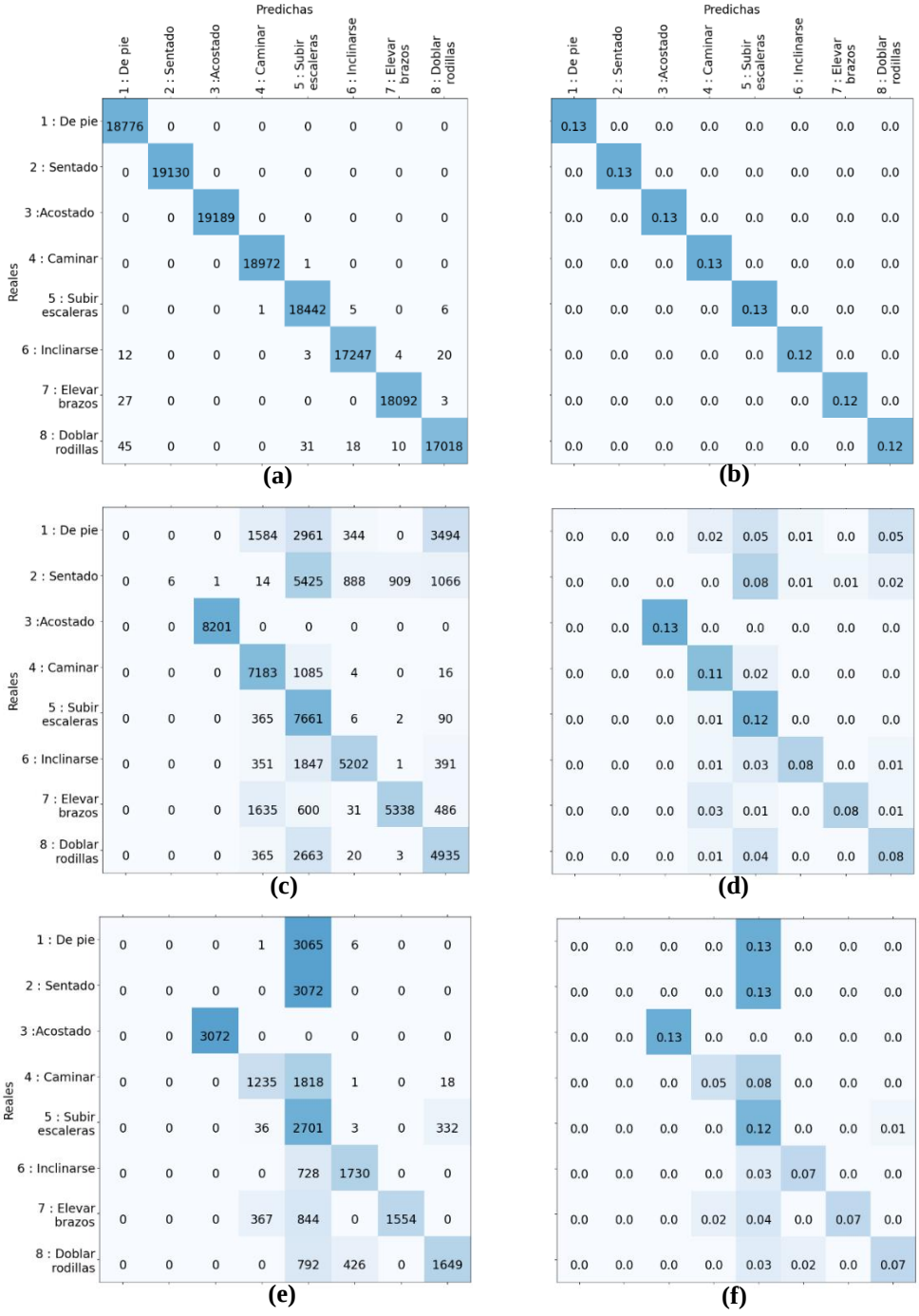


Fig. A.6 Modelo RF 2 con Base de Datos Limpia, Fuente: [autoría propia].

(a), (b) Entrenamiento (c), (d) Prueba (e),(f) Validación.

A.7. Matriz de Confusión SVM sin Gridsearch.

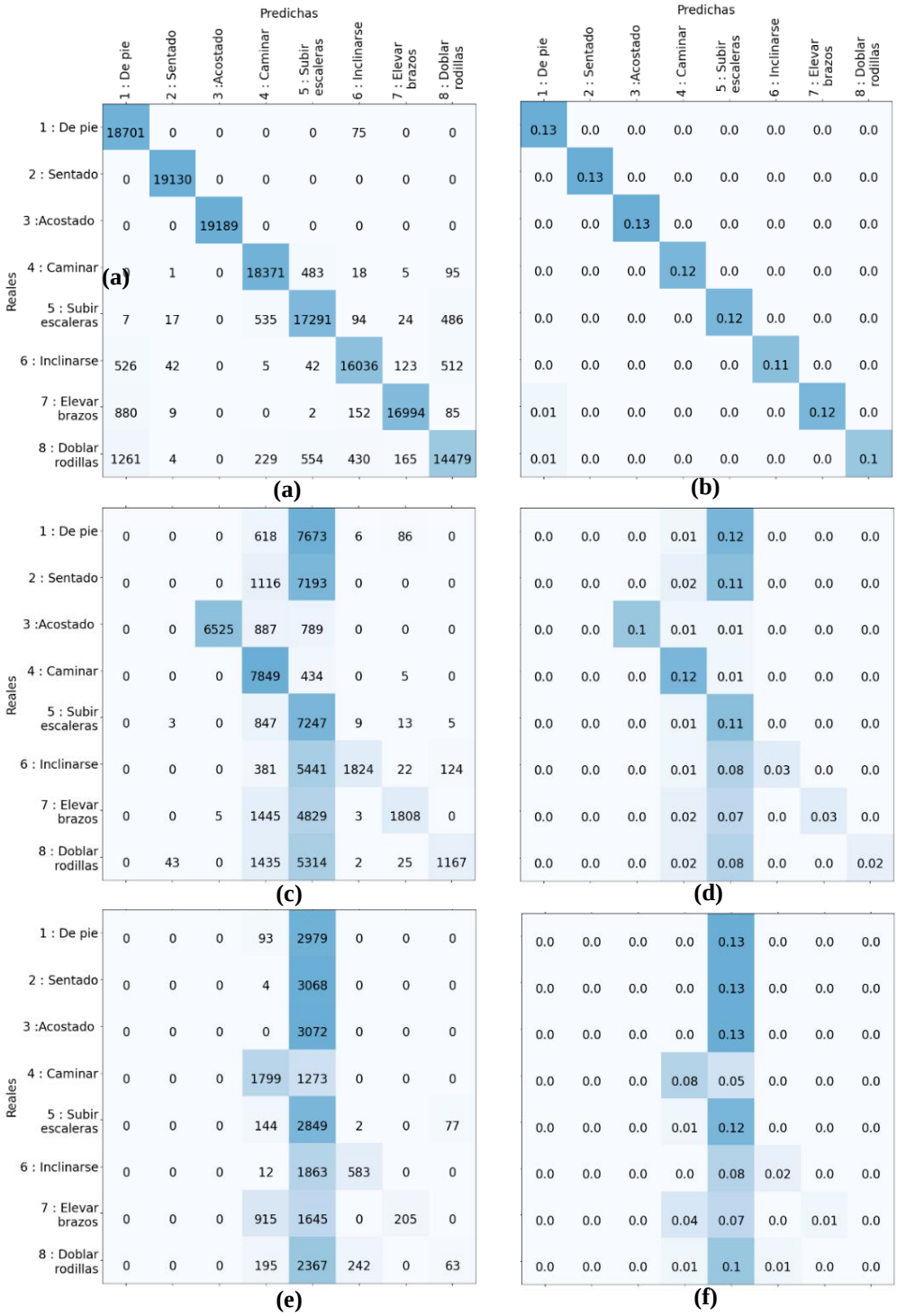


Fig. A.7 Modelo SVM 1 con Base de Datos Limpia, Fuente: [autoría propia].

(a), (b) Entrenamiento (c), (d) Prueba (e),(f) Validación.

A.8. Matriz de Confusión SVM con Gridsearch.

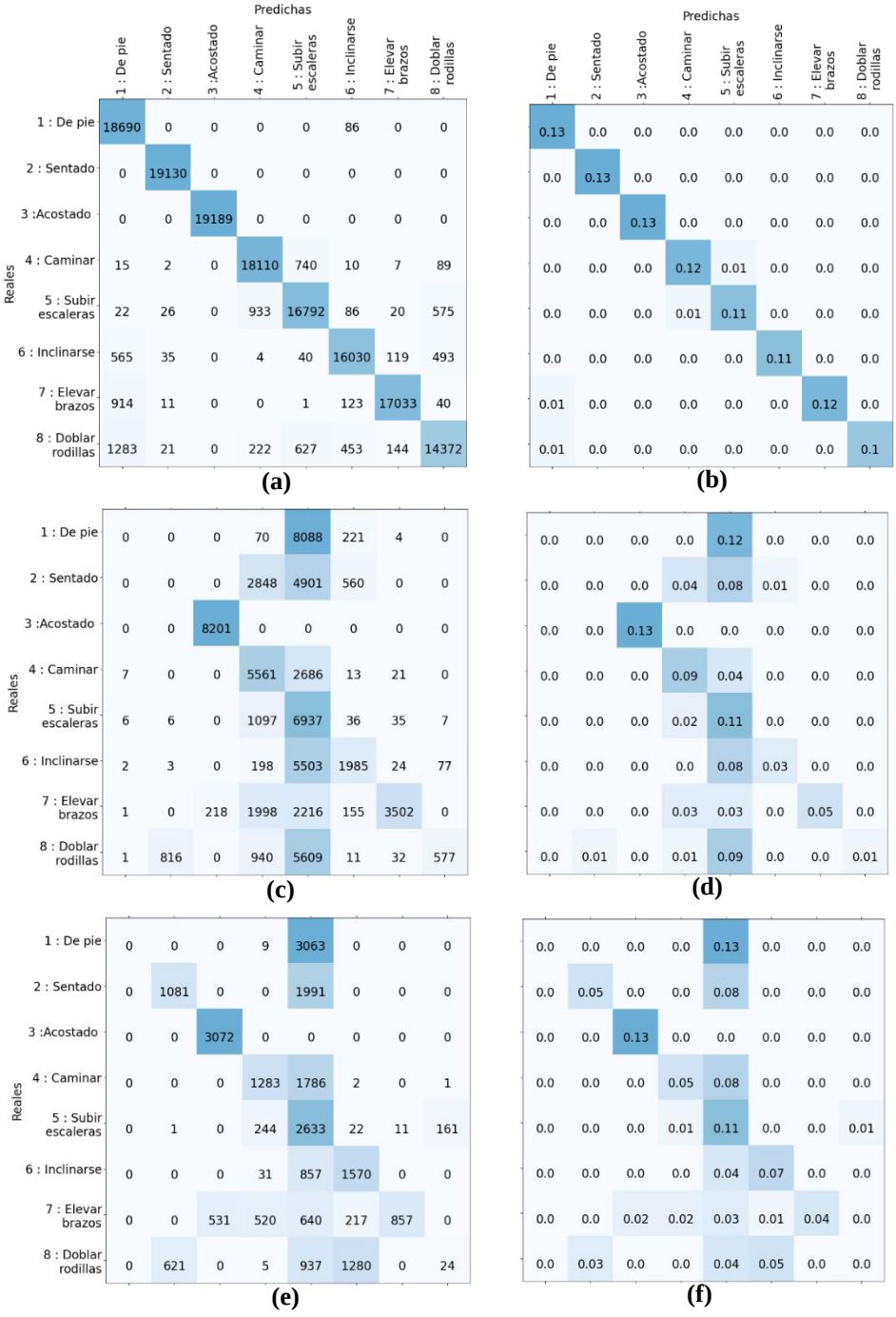


Fig. A.8 Modelo SVM 2 con Base de Datos Limpia, Fuente: [autoría propia].

(a), (b) Entrenamiento (c), (d) Prueba (e),(f) Validación.

A.9. Matriz de Confusión MLP sin Gridsearch.

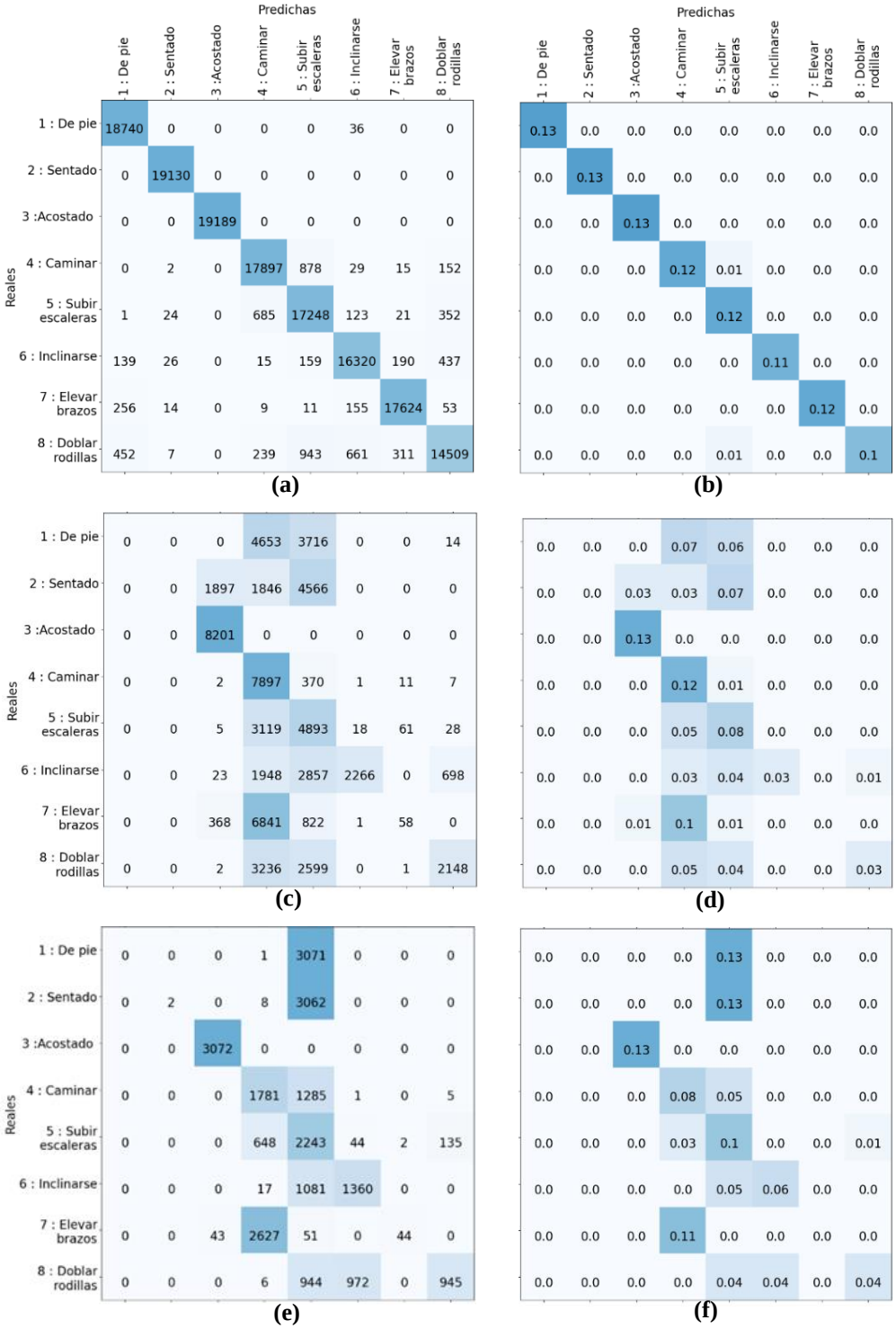


Fig. A.9 Modelo MLP 1 con Base de Datos Limpia, Fuente: [autoría propia].

(a), (b) Entrenamiento (c), (d) Prueba (e),(f) Validación.

A.10. Matriz de Confusión MLP con Gridsearch.

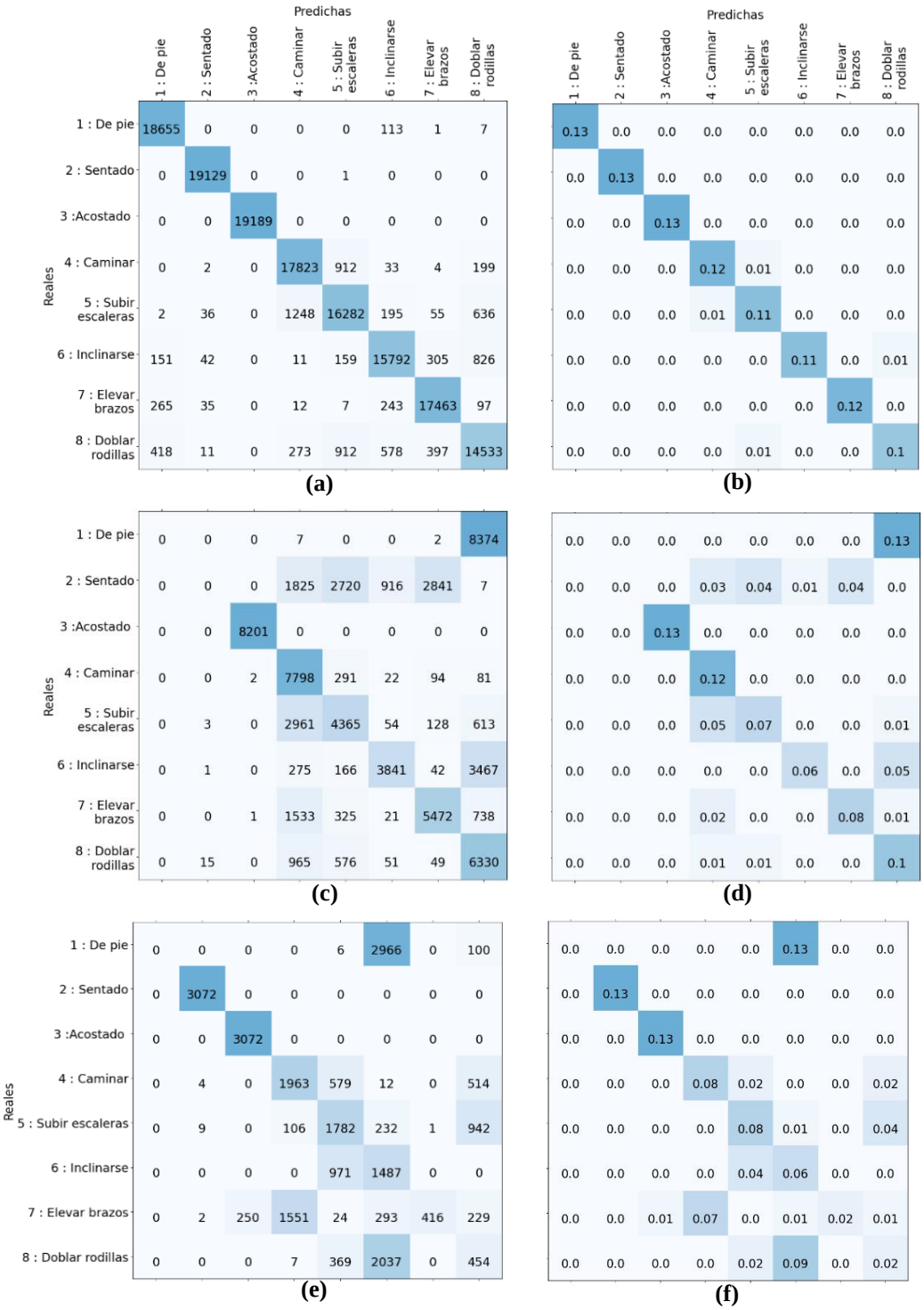


Fig. A.10 Modelo MLP 2 con Base de Datos Limpia, Fuente: [autoría propia].

(a), (b) Entrenamiento (c), (d) Prueba (e),(f)Validación.