Tesis de Grado

# Modelos de Aprendizaje para la predicción del comportamiento sedentario futuro



# Universidad Nacional del Centro de la Provincia de Buenos Aires

# Facultad de Ciencias Exactas

# Ingeniería de Sistemas

**Alumno:** Santillán Cooper, Martín

**Directores**

Armentano, Marcelo

Tomassel, Antonela

Falta completar

* Trabajos relacionados
* Experimentacion
* Discusion

Voy por la hoja 61

[Introducción](#_bwxenck50ema) 5

[Trabajos relacionados](#_bwxenck50ema) 7

[Monitoreo y modelado del comportamiento humano utilizando smartphones](#_esa6d6panlb7) 7

[Predicción de actividades](#_ukao3nv2lmz0) 10

[Redes Neuronales y time-series](#_p1n00otb4p2g) 10

[Predicción del comportamiento sedentario](#_nkjsb1qugtvx) 11

[Marco Teórico](#_bwxenck50ema) 14

[Machine Learning](#_96a4zfw1rsd0) 16

[Introduccion](#_ziax18pwx0k) 16

[Aprendizaje Supervisado y No Supervisado](#_ziax18pwx0k) 17

[Tipos de tareas:](#_ziax18pwx0k) 17

[Fases de un predictor](#_ziax18pwx0k) 19

[Generalizacion, underfitting y overfitting](#_ziax18pwx0k) 19

[Regularizacion (debajo de funcion de costo/perdida)](#_8mcmbggk6h84) 21

[Penalización de la norma de los parámetros](#_tckp3o1sjf6x) 22

[Modelos lineales y no Lineales](#_ziax18pwx0k) 23

[Clasificación y Regresión (definicion mas precisa)](#_ziax18pwx0k) 23

[Descenso por el gradiente](#_ziax18pwx0k) 23

[Métricas de evaluación](#_fy1hu0h7xblr) 23

[Métricas de regresión](#_2ep9687rdxfr) 24

[Mean Squared Error:https://www.dataquest.io/blog/understanding-regression-error-metrics/](#_teuiygjr0bgd) 24

[Métricas de clasificación](#_5n5dgzp7wkg1) 24

[Accuracy:](#_hffmyfd8n6kz) 25

[Tasa de error:](#_hffmyfd8n6kz) 25

[Precision o Valor de Prediccion Positiva:](#_hffmyfd8n6kz) 25

[Recall o Sensitividad o Tasa de Verdaderos Positivos(TPR):](#_hffmyfd8n6kz) 26

[Rollout o Tasa de Falsos Negativos (FPR):](#_hffmyfd8n6kz) 26

[Especificidad o Tasa de Verdaderos Negativos (TNR):](#_hffmyfd8n6kz) 26

[Rollout o Tasa de Falsos Positivos (FPR):](#_hffmyfd8n6kz) 26

[Metrica F1 o F1 Score:](#_hffmyfd8n6kz) 26

[ROC:](#_hffmyfd8n6kz) 27

[AUC:](#_hffmyfd8n6kz) 27

[Precission - Recall curve:](#_hffmyfd8n6kz) 28

[Reproductibilidad de resultados](#_96a4zfw1rsd0) 28

[Funcion de hipotesis](#_96a4zfw1rsd0) 28

[Funcion de perdida](#_96a4zfw1rsd0) 28

[Función de optimización](#_96a4zfw1rsd0) 28

[Tecnicas de Regularizacion](#_96a4zfw1rsd0) 28

[Deep Learning](#_ziax18pwx0k) 28

[Modelos de DL usados](#_96a4zfw1rsd0) 28

[DNN](#_ziax18pwx0k) 28

[RNN](#_ziax18pwx0k) 29

[CNN](#_ziax18pwx0k) 29

[Combinacion entre ellos](#_ziax18pwx0k) 30

[Hiper-Parametros](#_96a4zfw1rsd0) 30

[Diferencia entre parametros e hiper-parametros](#_ziax18pwx0k) 30

[Numero de capas](#_ziax18pwx0k) 30

[Número de neuronas](#_ziax18pwx0k) 30

[Funcion de acticacion](#_ziax18pwx0k) 30

[Numero de epocas](#_ziax18pwx0k) 30

[Tamaño de bache](#_ziax18pwx0k) 30

[Tuning de hiper-parametros](#_ziax18pwx0k) 30

[Predicción de Comportamiento Sedentario Futuro](#_bwxenck50ema) 32

[Evaluación Experimental](#_bwxenck50ema) 34

[Descripción del Dataset](#_s093l2b6kutm) 34

[Introduccion](#_n87t2tpgyp4o) 34

[Tipos de datos del dataset](#_sz6kovk3e9at) 35

[Datos no disponibles](#_l5x3lmcvuo7) 35

[Datos del acelerómetro](#_uisfuehtonv6) 36

[Datos de sensado](#_mvmik7j0jxx8) 36

[Datasets descartados](#_cc15la8gqxco) 38

[Pre-procesamiento del dataset](#_myvoi0gmjuzq) 38

[Análisis de datos](#_jlx3orj8yxc8) 38

[Tamaño de buckets](#_yvslk3ahs9dp) 38

[Disponibilidad de los datos](#_k7vn35flm781) 40

[Registros desconocidos](#_drbiv2psrh3l) 43

[Depuracion de buckets](#_jlx3orj8yxc8) 46

[Nivel de MET](#_2nqs14x4733i) 47

[Análisis del nivel de MET](#_a7ggrn221cnc) 49

[Inconsistencias encontradas](#_ykt2faadw8ix) 51

[Pipeline del preprocesamiento](#_5akxfwxylnb8) 53

[Seleccion de caracteristicas](#_iu4m6aih2hf4) 53

[Características Discretas](#_sfg3oyy7xfof) 53

[Características obtenidas a partir del GPS [Saeb 2016, cualquiera de los dos]:](#_yjq202r50uhd) 53

[Características basadas en el tiempo:](#_sjg563uez25j) 55

[Características de actividad física:](#_vw0jkvh8j8cr) 56

[Características de audio:](#_fj6szpa9f5r3) 57

[Otras características](#_nccr5fced68a) 57

[Características por intervalo](#_bwxenck50ema) 57

[Variables objetivo](#_y9duidtplvg7) 58

[Tratamiento de variables categóricas](#_dh0p9774ynwl) 59

[Generación de datasets con retrasos](#_d9404x3ykd5) 60

[Separacion entre X e y](#_c0k06mjirh2) 60

[Sepearacion entre entrenamiento y testeo](#_r8nb29o58liv) 60

[Dimensiones de X](#_qriu9j6uzsv4) 61

[Normalización de los datasets generados](#_d8qo7gu7xjoy) 61

[Modelos propuestos](#_pvbz00bagqbj) 61

[Validación de los modelos](#_jwjstti4vmcc) 62

[Tunning de los modelos](#_26g2l0ukjz6i) 64

[Optimizacion de Bayes](#_m8jwp7j0anwz) 64

[Selección de datasets modelo](#_9plflhsz77t8) 65

[Selección de usuarios modelo](#_s66ur8g37jy7) 65

[Selección de hiper parámetros para los modelos](#_9vjcbus25opw) 66

[Bayes Optimization](#_pw0a17sjfzr) 66

[Modelos de regresion](#_vjgh0keks67d) 66

[Validación de los modelos](#_a5gmihgdclb0) 66

[Modelos de regresion](#_8uwnyeyfn26u) 66

[Metricas](#_t628ftwaf045) 66

[Modelos de clasificación](#_mtkyxrbis1jv) 67

[Metricas](#_513j2bgznpyz) 67

[Diseño de los experimentos](#_usyp7uyz8uuf) 67

[Resultados](#_9ycog7juxxqz) 68

[Discusión](#_aoikelspm5xw) 70

[Conclusiones](#_bwxenck50ema) 71

[Bibliografía](#_bwxenck50ema) 72

## Introducción

La definición de comportamiento sedentario ha ido evolucionando a lo largo de los años, al mismo tiempo que lo hizo la forma de medirlo. La Red de investigación del comportamiento sedentario[[1]](#footnote-0) define el comportamiento sedentario como cualquier comportamiento de vigilia caracterizado por un gasto de energía ≤1.5 MET (equivalente metabólico de tareas) mientras se está sentado o reclinado (Sedentary Research Group. 2012). MET mide la intensidad de una actividad en múltiplos del gasto energético en reposo. Ejemplos de actividades sedentarias son mirar televisión (1.0 MET), comer sentado (1.5 MET), jugar videojuegos (1.0 MET) y conducir (1.3 MET).

La investigación realizada en el área demuestra asociaciones sólidas y consistentes entre el tiempo sedentario y la diabetes, las enfermedades cardiovasculares y la mortalidad por todas las causas (Wilmot et al. 2012) (Carter et al. 2017). Sin embargo, las asociaciones informadas fueron en gran medida independientes de la actividad física. Por lo tanto, es importante tener en cuenta que el comportamiento sedentario no representa lo contrario de la actividad física y que es posible que un individuo tenga niveles altos de actividad física moderada a vigorosa (MVPA, por sus siglas en inglés) y comportamiento sedentario. En general, se ha demostrado que el tiempo sedentario está asociado de forma perjudicial con la salud y con marcadores de riesgo metabólico en diversos grupos poblacionales. Además, se ha destacado la importancia de no solo estimular la MVPA, sino también de reducir el tiempo de sedentarismo, ya que la conducta sedentaria es un factor de riesgo para la mortalidad independiente de la MVPA (Koster et al. 2012).

Otra línea de investigación se centró en las asociaciones entre los descansos breves en el tiempo sedentario con los resultados metabólicos (Paing et al. 2018) y con la optimización de las operaciones cognitivas (Paing et al. 2018) (Felez-Nobrega et al. 2018) (Magnon, et al 2018) (Falcket al 2016). Benatti y Ried-Larsen (Benatti y Ried-Larsen 2015) afirmaron que existe evidencia suficiente para demostrar los efectos positivos de romper el tiempo prolongado de estar sentado en los resultados metabólicos.

Los métodos utilizados para medir la conducta sedentaria se pueden clasificar en subjetivos (cuestionarios y diarios de autoevaluación) y objetivos (sensores comunes de dispositivos ubicuos). Los métodos subjetivos están siendo superados por las nuevas tecnologías que pueden proporcionar, para todos los grupos de la población, información de segundo a segundo sobre la postura, el movimiento (o la falta de movimiento) y los patrones dentro y entre días. (Atkin et. Al. 2012).

Si bien los dispositivos móviles pueden considerarse como una de las causas del comportamiento sedentario (He y Agu 2016n1), también ofrecen nuevas oportunidades para prevenirlo. Hoy en día, los dispositivos móviles portátiles, como teléfonos inteligentes, relojes inteligentes y rastreadores de ejercicios están equipados con una amplia variedad de sensores que se pueden usar para la actividad humana y el análisis de comportamiento. El uso de métodos objetivos para evaluar el comportamiento sedentario está creciendo en popularidad a medida que los costos de los dispositivos móviles portátiles disminuyen y son más fáciles de usar. En 2018, el 91% de las personas entre 18 y 49 años en los EE. UU. Posee un teléfono inteligente[[2]](#footnote-1). En este contexto, estos dispositivos pueden verse como una oportunidad para desarrollar métodos objetivos complejos para medir el comportamiento sedentario.

Se han implementado muchas aplicaciones para teléfonos inteligentes con el objetivo de alertar al usuario cuando se reconoce un comportamiento sedentario (He y Agu 2014; Grundgeiger et al. 2017; Fahim et al. 2017). Predecir conductas sedentarias futuras puede ayudar a habilitar intervenciones preventivas, como recordatorios y sugerencias para diferentes actividades basadas en la Teoría de la conducta planificada (TPB) (Ajzen, 1991). TPB postula que es más probable que un sujeto participe en las intervenciones recomendadas para reducir las conductas sedentarias si tales actividades están incluidas en sus planes. (citar a Trapavessi 2015) Siguiendo esta idea, nuestra hipótesis es que si pudiéramos predecir en el momento t que un sujeto será sedentario en el tiempo t + 1, podríamos recomendar actividades para cambiar su rutina sedentaria a largo plazo. Este tipo de intervenciones puede dar lugar a mejores oportunidades para cambiar el comportamiento de los sujetos a resultados más saludables.

[definir MET]Definimos la predicción del comportamiento sedentario futuro (FSBP) como la predicción de si la actividad física de un usuario superará o no, en promedio, 1,5 MET en un futuro próximo. El problema de predecir el comportamiento sedentario futuro se ha abordado previamente analizando solo el tiempo inactivo / estacionario de un sujeto y se ha medido y comparado el rendimiento de varios modelos. Sin embargo, aunque MET es una métrica estándar en el área de la salud para medir la intensidad de una actividad en términos del gasto de energía, el uso de esta métrica para predecir el comportamiento sedentario mediante el uso de dispositivos portátiles y dispositivos móviles permanece inexplorada. En este artículo, presentamos un enfoque novedoso para predecir el comportamiento sedentario futuro de un sujeto en términos de su nivel de MET a partir de diferentes algoritmos de Deep Learning.

En este artículo, se presenta un enfoque novedoso para predecir el comportamiento sedentario futuro de un sujeto en términos de su nivel de MET a partir de diferentes algoritmos de aprendizaje profundo (Deep Learning), basado en la observación de valores obtenidos de múltiples sensores de dispositivos portátiles y móviles.

## Trabajos relacionados

## Monitoreo y modelado del comportamiento humano utilizando smartphones

A partir de que los dispositivos móviles y “wearables”, como smartphones, comenzaron a volverse populares, varios estudios fueron llevados a cabo para encontrar los patrones de uso que permiten relacionar, inferir y predecir diferentes tipos de comportamientos humanos en el contexto de la salud y el bienestar.

Harari et al. [(Harari et al. 2017)](https://paperpile.com/c/FQGDmo/VGeQ), en su investigación, parte del hecho de que los recientes avances en tecnología de sensado móvil ha revolucionado la evaluación del comportamiento al permitir el seguimiento no obstrusivo y continuo del comportamiento a través de dispositivos móviles (como acelerómetros y micrófonos) incluidos en smartphones.

Para examinar la viabilidad de usar métodos móviles de sensado para obtener estilos de vida diarios de comportamiento de estudiantes, los investigadores presentaron un estudio utilizando un dataset generado por una aplicación de sensado en smartphones llamada StudentLife para medir, de forma objetiva, la actividad natural de estudiantes y comportamientos sociales a lo largo de un ciclo académico de 10 semanas de duración. Al hacerlo, abordaron una brecha en la literatura de mHealth existente al proporcionar una descripción de los patrones finos de estabilidad y cambio que caracterizan las conductas relacionadas con la salud de los estudiantes durante un período académico.

Los resultados de este estudio indican que, a medida que el ciclo académico progresa, los estudiantes tendían a participar en más comportamientos sedentarios y relacionarse menos con otros estudiantes, lo que puede estar relacionado a la necesidad de concentrándose en estudiar y preparar exámenes.

En conclusión, Harari et. al. demostraron la viabilidad de utilizar métodos objetivos de sensado basados en smartphones para realizar el seguimientos de comportamientos relacionados a la salud en el contexto de la vida cotidiana de estudiantes. Los investigadores identificaron varias correlaciones significativas entre los datos objetivos de los sensores de los dispositivos móviles y el bienestar mental y el rendimiento académico.

Gong et al. [(Gong et al. 2019)](https://paperpile.com/c/FQGDmo/TxFr), en una colaboración entre ingenieros y psicólogos, utilizaron los datos del acelerómetro y del GPS de smartphones en los momentos antes, durante y después de realizar llamadas y enviar mensajes de texto. Los investigadores hipotetizaron que los movimientos sutiles (medidos con acelerómetros) covarían con el nivel de ansiedad social y el contexto social. Por ejemplo, las manos pueden temblar más cuando el estudiante está en un ambiente público, debido a que la probabilidad de evaluación negativa por otros es alta. Los autores denominan a estos movimientos como comportamientos dinámicos del usuario.

Para realizar el estudio, 52 estudiantes (20.5±3.8) fueron reclutados del grupo de participantes del departamento de psicología de la Universidad de Virginia. De cada participante se tomaron los datos del acelerómetro, el GPS y datos relacionados a las llamadas y a los mensajes de texto. Luego, modelaron los datos como un sistema lineal dinámico. Para ello, primero aplicaron el algoritmo Foursquare para obtener los puntos de interés a partir de los datos crudos del GPS de cada participante, y luego, los dividieron en sus locaciones semánticas (hogar, transporte, vida personal, lugar de comida). Respecto a los datos sociales (llamadas y mensajes de texto), los autores analizan por separado períodos de observación minutos antes y después de una llamada, para detectar por ejemplo, cómo varían los indicadores de ansiedad de un sujeto cuando debe realizar una llamada y no recibe una respuesta a la llamada realizada. Además, antes de que comenzase el experimento, cada estudiante fue evaluado de acuerdo a la Escala de Ansiedad de Interacción Social (SIAS, por sus siglas en inglés).

Los resultados obtenidos en el estudio mostraron una relación estadística significativa entre los datos de los sensores de smartphones y la ansiedad social de los estudiantes. Estos resultados sugieren que es posible integrar datos de acelerómetros y de GPS para identificar marcadores de comportamiento asociados con la ansiedad social en varias locaciones. Por lo anterior, este trabajo abre nuevas posibilidades para monitorear pasivamente marcas de comportamiento de ansiedad social a través de la integración de los datos del acelerómetro y los de GPS.

Kanjo et al. [(Kanjo et. al. 2019)](https://paperpile.com/c/FQGDmo/zeAi) adoptó un enfoque de Deep Learning para la clasificación de emociones a partir de datos de diferentes tipos de sensores portables. La clasificación de emociones a menudo requiere del modelado de tipos de datos de entrada de diferentes modalidades, como señales fisiológicas, datos ambientales, video y datos de movimiento y ubicación. Este problema ya había sido abordado desde la utilización de algoritmos del área de Machine Learning, pero aunque los procesos de ingeniería de características a menudo integrados en estos algoritmos son beneficiosos para la modelización de emociones, heredan algunas limitaciones críticas que pueden dificultar el desarrollo de modelos fiables y precisos.

En este trabajo, los investigadores adoptaron un enfoque de aprendizaje profundo para la clasificación de emociones a través de un proceso iterativo mediante la adición y eliminación de un gran número de señales de sensores de diferentes modalidades. El conjunto de datos fue recogido en un estudio del mundo real desde teléfonos inteligentes y dispositivos portátiles. Dicho conjunto de datos combina la interacción local de tres modalidades de sensores: en el cuerpo, el medio ambiente y la ubicación en un modelo global que representa la dinámica de la señal junto con las relaciones temporales de cada modalidad. El enfoque propuesto emplea una serie de algoritmos de aprendizaje profundo que incluyen un enfoque híbrido usando la Red Neuronal Convolucional y la Red Neural Recurrente de Memoria Corta a Largo Plazo (CNN-LSTM) en los datos crudos del sensor. Utilizando los datos crudos del sensor como entrada para las redes neuronales se consigue elimar las necesidades de extracción manual de característica. Los resultados muestran que la adopción de enfoques de aprendizaje profundo es efectiva en la clasificación de la emoción humana cuando se utiliza un gran número de sensores de entrada (precisión media 95% y F-Measure=95%) y los modelos híbridos superan a la red neuronal profunda tradicional totalmente conectada (precisión promedio 73% y F-Measure=73%). Además, los modelos híbridos superan los algoritmos Ensemble previamente desarrollados que utilizan la ingeniería de características para entrenar la precisión media del modelo 83% y F-Measure=82%)

Wu et al. [(Wu et al. 2018)](https://paperpile.com/c/FQGDmo/H0Qg) utilizaron redes de encuentro basadas en los registros de encuentros Bluetooth para monitorear el nivel de stress de los usuarios. Bluetooth es una red de corto alcance comúnmente disponible en los dispositivos móviles. Al mismo tiempo, la red Bluetooth puede ser utilizada como un sensor móvil ampliamente disponible que puede detectar el entorno social de un individuo basado en la proximidad física. Los autores señalan que en la bibliografía existe una falta de ingeniería sistemática de características, así como también evidencia de su poder predictivo en problemas de salud mental. En este trabajo se extrajeron características de los datos de encuentro de Bluetooth de los teléfonos inteligentes de 49 estudiantes a lo largo de 66 días, provenientes del dataset StudentLife. El análisis de correlación sugiere que los características generadas pueden explicar la varianza en el stress significativamente mejor que los modelos nulos.

En este trabajo los autores evaluaron la predictibilidad del nivel de stress que podía conseguirse utilizando las redes de encuentro de Bluetooth, más otros datos tales como la calidad del sueño y otros datos de sensores como GPS, acelerómetro, uso del teléfono, etc. Sus predictores pueden ser clasificados de acuerdo diferentes características: predecir el nivel de stress o el cambio del mismo, utilizar la técnica de evaluación Leave-One-Subject-Out (LOSO) o Leave-one-observation-out (LOOO), predecir el presente o el futuro. Los modelos utilizaron la técnica Random Forest en todos los casos. Los resultados indican que las características extraídas de las redes de encuentro representan una mejora en términos de predictibilidad del nivel de stress y del cambio de éste en el presente, pero no en el futuro. Estos resultados proveen evidencia del valor potencial de mejora al incorporar las redes de encuentro basadas en datos de redes Bluetooth a aplicaciones de monitoreo de stress.

Zia et al. [(Zia et al. 2016)](https://paperpile.com/c/FQGDmo/SRc7) destacan que utilizar los datos de dispositivos portátiles puede ayudar a que pacientes con Parkinson vivan de forma más segura e independientemente. Partiendo de esta idea, el objetivo de este trabajo consistió en determinar la viabilidad de utilizar redes neuronales para predecir la congelación de la marcha, que es un síntoma del Parkinson. Particularmente, los investigadores usaron una clase de red neuronal conocida como Redes Recurrentes en capas (LRN). El dataset utilizado para entrenar las redes fue uno público donado por el Repositorio de Aprendizaje Profundo de la Universidad de California y comprende un registro de los datos de movimiento (3 acelerómetros) de 3 sujetos, donde los momentos en los cuales se predice el Congelamiento de la Marcha están anotados como un timestamp. El trabajo no especifica las características obtenidas a partir de los datos del acelerómetro, por lo que se supone que utilizaron un modelo end-to-end, en el que al igual que en otros trabajos ya expuestos, son las primeras capas de las redes quienes se ocupan de aprender a generar las características. Los parámetros que fueron variando en las diferentes redes neuronales testeadas fueron la cantidad de capas, las cantidad de neuronas por capas, el tamaño de batch y el factor de down sampling -esto es, la cantidad de ejemplos de entrenamiento que se quitaron del dataset-.

Los resultados mostraron que las redes que mejor precisión y recall tuvieron son aquellas de una sola capa, baja cantidad de neuronas y alto down sampling. Los resultados para cada uno de los 3 sujetos evaluados fueron 47, 30 y 27 de recall y 42, 89 y 57 de precisión. Los autores propusieron, como trabajo futuro, utilizar otros tipos de datos para que se complementen con los datos de movimiento.

## Predicción de actividades

La predicción de actividades es un área de investigación que consiste en hipotizar información sobre las actividades en las que un sujeto va a estar involucrado en el futuro basándose en datos de sensores. No debe confundirse al área de predicción de actividades con el de reconocimiento de actividades. La diferencia entre ambas es que, en la segunda, se utilizan datos de sensores tanto del pasado como del presente y se los mapea con con un tipo de actividad que está ocurriendo en el momento actual. Mientras que en la primera, claramente, no se tiene ningún dato de sensor sobre la actividad que va a llevarse a cabo en el futuro y se quiere predecir. La predicción de actividades puede ser utilizada, por ejemplo, para anticipar las actividades de una persona en una casa, facilitando la automatización de dispositivos IOT, como calentando la casa antes de que la persona llegue del trabajo.

## Redes Neuronales y time-series

Los nuevos sensores remotos poseen gran resolución, tanto espectral como espacial, lo que ha beneficiado a las Series de Tiempo de Imágenes Satelitales (SITS, por sus siglas en inglés). Las series de imágenes son un componente clave para la sistemas de clasificación que apuntan a obtener mapas actualizados y precisos de la cobertura terrestre de las superficies de la Tierra. Pelletier et al. [(Pelletier, Webb, and Petitjean 2019)](https://paperpile.com/c/FQGDmo/qKiY) propone el uso de redes convolucionales temporales, es decir, aquellas que aplican la convolución en la dimensión temporal, para la

Dividir trabajos relacionados en: aquellos de activity prediction, aquellos de prediccion con redes neuronales (se podrian hacer 3 subsecciones de acuerdo a rnn tcn y cnn) y prediccion a partir de datos de smartphones

[https://www.google.com/url?q=https://arxiv.org/pdf/1811.10166.pdf&sa=D&ust=1557167443720000&usg=AFQjCNFHEi-ii6qbZWtSebRrGoY61l0Kqw](https://arxiv.org/pdf/1811.10166.pdf)

## Predicción del comportamiento sedentario

El comportamiento sedentario puede ser clasificado como un tipo de tarea de prediccion de actividades (deberia llegar a que es un tipo de cada una de las secciones de trabajo relacionado anteriormente). En esta subsección se presentan los trabajos relacionados al área en la que se basa esta esta tesis, que es, la predicción del comportamiento sedentario.

Cabe aclarar que los trabajos que buscan resolver el problema de predecir el comportamiento sedentario a futuro, son escasos y hacen suposiciones diversas a la hora de tratar de resolverlo.

Para realizar la búsqueda de trabajos relacionados a la predicción de comportamiento sedentario futuro se utilizaron las librerías IEEE Xplore, ACM, Wiley, ScienceDirect y Scopus utilizando la búsqueda. Hasta el momento, Q. He and E. Agu son los únicos autores que hemos hallado como trabajos aptos a partir de los resultados de dicha busqueda. A continuación se analizan todos los trabajos de los ya nombrados autores, quizas con mas detalle debido a la relacion cercana con el trabajo motivo de esta tesis.

Sólo A diferencia del enfoque propuesto en el presente trabajo, Q. He and E. Agu determinan el nivel de sedentarismo como el porcentaje de registros de actividad estacionarios que se tomaron de los datos de actividad física de cada usuario en cada hora, con respecto a los demás tipos de registros de actividad (corriendo, caminando, etc.).

Definir extensamente cada trabajo de He y Agu

Primero, propusieron un algoritmo de dominio de frecuencia para detectar patrones sedentarios recurrentes a partir de los datos de los usuarios. Este trabajo buscó ajustar funciones periódicas (seno, coseno) al porcentaje de registros de actividad estacionaria de cada hora y de cada usuario, tratando así de identificar los patrones subyacentes en su actividad física.

En el primer estudio llevado a cabo por Q. He y E. O. Agu (A Frequency Domain Algorithm to Identify Recurrent Sedentary Behaviors from Activity Time-Series Data) propusieron un algoritmo de la frecuencia para detectar patrones de comportamiento sedentario recurrentes.

En este trabajo, como en los demas, los autores utilizaron el dataset StudentLife para validar sus hipotesis. Con respecto al procesamiento de los datos, primero, discretizaron los datos de actividad fisica en buckets de distintas escalas de tiempo, ya que, en dicho dataset, cada registro esta acompañado de un timestamp. Luego, para cada bucket, computan lo que ellos llaman nivel de sedentarismo, que definen como el porcentaje del total de actividades físicas etiquetadas como estacionario pertenecientes a ese determinado bucket.

A partir del nivel de sedentarismo computado, los investigadores llevaron a cabo un análisis sobre el nivel de sedentarismo a lo largo de las horas del día y de los días de la semana, que les permitió observar en qué momentos del día los estudiantes eran más sedentarios y cuánto difieren entre sí los patrones de comportamiento sedentario de los diferentes estudiantes y a lo largo de la duración del estudio.

En base a esto los autores propusieron un modelo computacional capaz de capturar los patrones recurrentes a diferentes escalas de tiempo. Estos modelos utilizan el nivel de sedentarismo considerandolo como una señal periodica, donde cada elemento de dicha señal corresponde a un bucket de un determinado usuario. Para generar los modelos, los autores aplicaron una transformacion de Fourier sobre la señal periodica, descomponiendola en una suma de funciones coseno y tomando las k primeras funciones con el fin de minimizar el error cuadrado y eliminar el ruido. Los modelos son generados para cada uno de los usuarios y a diferentes escalas de tiempo, es decir, con diferentes tamaños de bucket.

Los investigadores obtuvieron resultados muy variados de acuerdo a los diferentes usuarios y escalas de tiempo utilizadas. Algunos modelos obtuvieron un error cuadrado muy bajo lo que significó que el usuario al cual corresponde el modelo presenta comportamientos sedentarios de forma recurrente y periódica. Sin embargo, otros modelos no lograron conseguir un error cuadrado bajo, lo que mostró que el usuario al cual corresponde el modelo no presenta comportamientos sedentarios de forma recurrente y periódica.

En un segundo estudio, Q. He y E. O. Agu (Smartphone Usage Contexts and Sensable Patterns as Predictors of future sedentary behaviour), resaltaron que los teléfonos inteligentes pueden monitorear los comportamientos sedentarios llevados a cabo por sus usuarios, asi como tambien pueden monitorear el contexto en que dichos comportamientos sedentarios ocurren. En este trabajo, los investigadores exploraron si es posible utilizar los datos de contexto para predecir el comportamiento sedentario futuro.

Para probar sus hipótesis, He y Agu utilizaron el dataset público StudentLife, que contiene datos de actividad física y datos del contexto de 49 estudiantes universitarios a lo largo de 10 semanas. El dataset StudentLife ha sido utilizado por todos los trabajos de investigación que se encuentran en esta sección. Ademas, es el dataset utilizado en esta tesis, por lo que su estructura y contenido sera detallado en la seccion ….

En el dataset StudentLife, todo datos, ya sea de actividad fisica o de contexto esta asociado a un timestamp. Dependiendo del tipo de dato, éstos fueron muestreados con cierta frecuencia. Con el fin de normalizar …, el paso siguiente que llevaron a cabo los investigadores fue el de discretizar el tiempo en buckets de una hora. Es decir, todos los datos de sensado de los usuarios fue asociado a un hora y dia especifico. En total, He y Agu obtuvieron como resultado 65,601 buckets.

Luego, a partir los datos de actividad física, los investigadores computan, para cada bucket, lo que ellos llaman nivel de sedentarismo, que definen como el porcentaje de actividades físicas etiquetadas como estacionario. Esta forma de cuantificar el comportamiento sedentario sera discutida mas adelante en la seccion …Además, discretizaron el nivel de comportamiento sedentario en 3 clases, cada una asociada a un rango específico: muy sedentario (99.99-100.00), sedentario (90.52 - 99.99) y menos sedentario (0 - 92.52). Dichos rangos no fueron determinados con ningún tipo de base teórica, sino que fueron así definidos para obtener la misma cantidad de registros de actividad para cada clase.

A continuación, los investigadores analizan los datos de contexto que provee el dataset utilizado y su relación con el nivel de sedentarismo computado. A partir de este análisis, los investigadores definen y calculan, en total, 22 variables de contexto asociadas a cada bucket, como el dia de la semana y o la actividad física que mas llevo a cabo un determinado usuario en una hora específica.

Por último, los investigadores intentaron predecir el nivel de sedentarismo futuro, es decir, si en la siguiente hora el usuario iba a ser muy sedentario, sedentario o menos sedentario, a partir de las variables de contexto que definieron. Para realizar esta tarea, y dato que la variable que se intenta predecir es categórica, los autores de este trabajo utilizaron los modelos Naive Bayes y Regresión Logística. Ademas, para validar los resultados utilizaron la técnica de validación cruzada. Naive Bayes alcanzó una precisión de 59.7% y recall de 64.2%, mientras que Regresion Logistica obtuvo una precisión de 60.8% y recall de 62.6%. Como ultimo experimento, los investigadores agregaron al conjunto de datos de entrenamiento el nivel de sedentarismo actual, mejorando ligeramente los resultados. Como conclusión de estos resultados los autores resaltaron que para predecir si el usuario iba a ser muy sedentario en la hora siguiente, consiguieron una precisión de 73.1% y recall de 87.7%.

En otro estudio Q. He y E. O. Agu (Towards sedentary lifestyle prevention An autoregressive model for predicting sedentary behaviors) destacaron que el tiempo, el ritmo diario y los habitos sedentarios del pasado son 3 determinantes del comportamiento sedentario. Los investigadores propusieron entonces un método para descubrir automáticamente los 3 determinantes analizando los registros de las series de tiempo de sensores de smarthphones. Dicho método analiza el historial de comportamiento de un usuario para encontrar patrones temporales que puedan ser utilizados para predecir el comportamiento sedentario futuro.

En este trabajo, los investigadores suponen que el comportamiento sedentario de los usuarios está correlacionado linealmente en el dominio temporal de tal forma que la probabilidad de que un usuario tenga un comportamiento sedentario en el siguiente período de tiempo depende de su comportamiento actual y pasado. Para hallar estos patrones, los autores utilizan Modelos Autoregresivos junto al Método de Máximo Entropía para estimar los coeficientes de dichos modelos.

Para evaluar los modelos propuesto Q. He y E. O. utilizaron el dataset StudentLife. Dicho dataset dispone de registros de series de tiempo sensados automaticamente, donde cada registro esta compuesto de un timestamp y un tipo de actividad (estacionario, caminando, corriendo). Como en los trabajos anteriores, discretizaron las series de tiempo en buckets de una determinada duración, calculando para cada una el porcentaje de registros etiquetados como estacionario. Luego, generaron en total 324 modelos variando el orden del Modelo Autorregresivo (cantidad de informacion del pasado que el modelo utiliza) y el tamaño de bucket (1, 2, 3 o 6 horas). Por ultimo, calcularon el Error Cuadrado de cada modelo y para cada usuario.

Los resultados mostraron que los modelos que tuvieron un Error Cuadrado fueron aquellos que utilizaban buckets de 6 horas. El MSE mas bajo obtenido fue sobre el usuario numero 23, con un modelo autorregresivo de orden 26 y tamaño de bucket de 6 horas Los investigadores hipotetizaron que esto de debio a que la influencia de factores y aleatorios. tienen menos influencia cuanto mas grande sea el tamaño de bucket.

En 2017, Q. He y E. O . publicaron un ultimo trabajo de investigacion en el cual plantearon un nuevo enfoque para poder predecir el comportamiento sedentario de una persona. (A Rhythm Analysis-Based Model to Predict Sedentary Behaviors). En este trabajo, los autores propusieron modelos que estan basados en el concepto de analisis de ritmo, propuesto de Lefebvre, que postula que muchos comportamientos humano siguen ritmoso naturales. Por lo tanto, en este trabajo los investigadores se concentraron en detectar los ritmos predominantes de comportamiento sedentario y en modelar los ritmos ciclicos y los ritmos lineales utilizando funciones ciclicas y funciones lineales, respectivamente. Como ejemplo de ritmo ciclico, los autores ejemplifican con una persona que se sienta en un sillon todos los dias a la misma hora. Como ejemplo de ritmo lineal, los autores ejemplifican con una persona que se acuesta exhausta luego de llevar a cabo un ejercicio de alto requerimiento fisico.

Por un lado, los autores de este trabajo denominan a los modelos que utlizan funciones ciclicas como libres de historia (history-free) (?, ya que una vez determinadas las funciones ciclicas no es necesario guardar los datos de gasto energetico del usuario para predecir su siguiente comportamiento sedenterio. Por otro, los autores de este trabajo denominan a los modelos que utlizan funciones lineales como ligados dependientes de historia (history-dependent)? ya que estos deben guardar los datos de gasto energetico del usuario aun cuando el modelo ha sido generado.

Para generar los modelos libres de historia los autores utilizan un procedimiento similar al que siguieron en (A Frequency Domain Algorithm to Identify Recurrent Sedentary Behaviors from Activity Time-Series Data), mientras que para generar los modelos dependientes de historia utilizan un procedimiento similar al que siguieron en (Towards sedentary lifestyle prevention An autoregressive model for predicting sedentary behaviors).

A continuacion, los autores propusieron modelos hibridos, que combinan los modelos libres de historia y los modelos dependientes de historia. Analiticamente, los modelos hibridos se obtienen como una suma ponderada de los dos tipos de modelos, donde los primero buscan calcular los patrones causados por ritmos cicliclos, mientras que los segundos buscan capturar los patrones dentro de esos ciclos.

En total He y Agu generaron 49248 modelos hibridos para los 49 participantes del dataset. Para cada uno de estos modelos, los autores debieron hallar los valores optimos de l (el orden de los Modelos Autorregresivos) y k (la cantidad de funciones periodicas utilizadas). Los resultados mostraron que el modelo de base, donde k vale 0 y l vale 0, fue el optimo en casi la mitad de los modelos generados, es decir, el que menos MSE obtuvo. Estos resultados contradicen su hipotesis inicial sobre que el comportamiento sedentario puede ser explicado a partir de los ritmos ciclicos y lineales de las personas. Los autores indicaron que este resultado era de esperar debido a las horas en que los usuarios duermen, en las cuales los modelos pueden simplemente predecir el promedio de los resultados de las horas anteriores.

## Marco Teórico

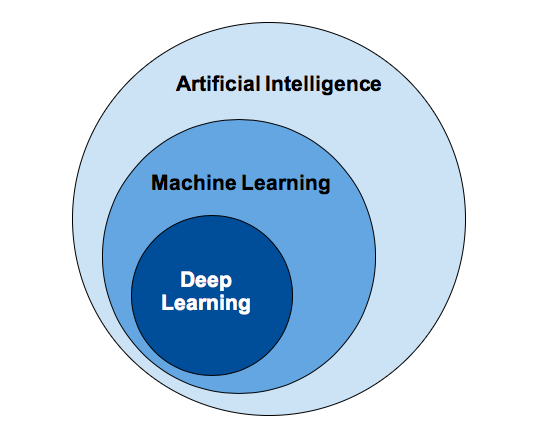
El objetivo principal de esta tesis es evaluar el uso de modelos Machine Learning y de Deep Learning para el problema de predicción del comportamiento sedentario futuro. Por lo tanto, es importante dar una base a los conceptos más importantes en ambas áreas. Es debido notar que tanto Deep Learning como Machine Learning pertenecen al campo de Inteligencia Artificial y que a su vez Deep Learning pertenece al campo de Machine Learning. Por lo tanto, a continuación se presentan los conceptos más importantes de Machine Learning, que servirán para comprender el proceso llevado a cabo para generar los modelos de Machine Learning. Luego, se presentan los conceptos más importantes de Deep Learning, que son una extensión de los presentados en la sección de Machine Learning.

Mitchell (1997) define que un programa de computadora aprende a partir de una experiencia E con respecto a una tarea T y un determinado desempeño medido por D, si su desempeño en la tarea T, mejora con la experiencia E. Esta definición puede aplicarse a la totalidad del área de la Inteligencia Artificial. La diferencia entre Inteligencia Artificial y Machine Learning es que en el primero el sistema de Inteligencia Artificial solo sigue reglas específicas donde el conocimiento es generado de una forma supervisada, por humanos. En el caso de Machine Learning, el algoritmo de aprendizaje debe encontrar patrones en los datos de entrada, de esta forma generando conocimiento.

Una característica importante de Machine Learning es que el desempeño de sus algoritmos depende fuertemente de la representación de los datos de entrada (hand-crafted). Es decir que dicha representación ha sido calculada a partir de los datos crudos, ya que los algoritmos de Machine Learning no pueden aprender de ellos o les es más difícil. En este contexto, cada unos de los datos de entrada calculados es denominado característica o feature.

En [Deep Learning BOok Introduction] los autores incluyen, además, el área Representation Learning, que es un subconjunto de Machine Learning pero a su vez un superconjunto de Deep Learning. En Representation Learning se utiliza Machine Learning no solo para aprender del mapeo de los datos de entrada a la salida (predicción), sino que también para descubrir la representación óptima de los datos de entrada. De esta forma se pueden obtener mejores desempeños, sobretodo para tareas complejas, para las cuales es difícil decidir cuál es el conjunto de características óptimo. Igualmente, existen casos en los cuales es igual de difícil aprender la representación óptima que resolver el problema original. Esta forma de aprendizaje de la representación óptima de los datos de entrada (por ejemplo, una foto, audio, datos de sensores) es llamada comúnmente end-to-end e implica que no es necesario realizar la búsqueda manual de la representación de los datos de entrada, sino que son hallados automáticamente.

Finalmente, Deep Learning resuelve el problema de Representation Learning, introduciendo representaciones que son expresadas en términos de otras representaciones más simples. De esta forma, el algoritmo de aprendizaje puede aprender conceptos más complejos a partir de conceptos más simples a través de diferentes niveles de representación. Un ejemplo de esto son las Redes Neuronales Convolucionales aplicadas a imágenes, donde se aprenden automáticamente filtros que transforman las imágenes originales. Diferentes niveles de filtros son aprendidos, de forma que a imágenes ya filtradas se les aplican nuevos filtros, de esta forma, llegando a encontrar la representación óptima de las imágenes para la tarea de aprendizaje que se quiere llevar a cabo. Por lo tanto, la representación aprendida por un mismo algoritmo va a diferir si la tarea en cuestión es, por ejemplo, la de encontrar objetos en la imagen o si esta es la de predecir si en la imagen es de noche o de día.



## Machine Learning

## Introduccion

Machine Learning puede ser definido de varias maneras, dependiendo del autor y del campo de estudio desde el cual se aproxime uno. En [Machine Learning a probabilistic approach] es definida como un conjunto de métodos que pueden detectar patrones en los datos de entrada automáticamente, pudiendo usar luego esos patrones descubiertos para realizar diferentes tipos de decisiones bajo incertidumbre, dependiendo de si la tarea que realiza el modelo es supervisada o no supervisada.

En (Libro Deep Learning), Machine Learning es definido como “una forma de estadística aplicada que hace gran énfasis en el uso de computadoras para estimar estadísticamente funciones complejas y un menor énfasis en probar intervalos de confianza en dichas funciones”.

En general, en el área de Machine Learning las tareas son descritas a partir de cómo el modelo de Machine Learning debería procesar un determinado caso o ejemplo. Un caso puede ser descrito como un conjunto de características que han sido cuantitativamente medidas a partir de cierto evento u objeto, que queremos que el modelo de Machine Learning procese. Dichas características pueden o no haber sido calculadas a partir de los datos originales. Además, puede clasificarse a los casos segun en que fase del algoritmo de aprendizaje serán utilizados en: casos de entrenamiento, casos de testeo o casos de validación.

Típicamente, cada caso se representa como un vector perteneciente a donde cada componente del vector representa una característica diferente. Ademas, los casos pueden ser representados en forma matricial u en otros espacios con mayores dimensiones. La representación de los datos va a diferir de acuerdo a la naturaleza de los casos y a la estructura de los datos de entrada que espera el algoritmo de aprendizaje. Por ejemplo, las características de una imagen son usualmente los valores de los píxeles de dicha imagen. Dependiendo de las imágenes, pueden estar representadas en 3 dimensiones (alto x ancho) o en 3 dimensiones (alto x ancho x canales) si la imagen tiene más de un canal, como es el caso de las imágenes satelitales. A su vez, el algoritmo de aprendizaje puede esperar como entrada un vector (como sería el caso de un Perceptron Multicapa) o alguna de las dos estructuras anteriormente nombradas. De esta forma, puede apreciarse que la estructura de los datos de entrada puede ser muy variada de acuerdo a la naturaleza de los datos, si los datos han sido preprocesados para generar nuevas características y del algoritmo de aprendizaje que se desea utilizar.

## Aprendizaje Supervisado y No Supervisado

Los diferentes algoritmos pueden ser clasificadas a partir de si su forma de aprender es supervisada o no.

En el caso del aprendizaje supervisado, los algoritmos utilizan datasets donde cada caso está asociado a una etiqueta . Dicha etiqueta es llamada variable objetivo. Las tareas de aprendizaje supervisado se basan en lograr predecir el valor de a partir de las características de . Durante la fase de entrenamiento, los algoritmos de este tipo pueden saber cuán acertadas son las predicciones de acuerdo a cuán alejadas están de la variable objetivo y así cambiar su forma de predecir para minimizar su error de predicción.

En este tipo de algoritmos se utilizan datasets que contienen diferentes características y casos de estudio y la tarea es tratar de aprender propiedades útiles acerca de la estructura del dataset. Ejemplos: encontrar grupos con ejemplos similares (k near neighbours) o aprender la distribucion de probabilidades del dataset (denoising, synthesis).

## Tipos de tareas:

* Clasificación: en este tipo de tarea un programa de computadora debe especificar cuáles de las categorías una determinada un determinado valor de entrada pertenece para resolver esta tarea el algoritmo de aprendizaje he preguntado producir una función . Si, para un determinado ejemplo, Quiere decir que el modelo asigna un ejemplo descrito por el vector X una categoría identificada cómo y. Existen también otras variantes de tareas de clasificación por ejemplo cuando la función En vez de dar como resultado una categoría específica si no que una distribución de probabilidades sobre las clases. Un ejemplo de tarea de clasificación en el reconocimiento de objetos dónde la entrada al algoritmo de aprendizaje es una imagen generalmente descripta como un conjunto de píxeles ir a salida es un código numérico identificando al objeto en la imagen.

Ejemplos de tareas de clasificación

* Clasificación con valores faltantes: la tarea de clasificación se vuelve más difícil si el programa de computadora no se le garantiza que cada características en el vector de entrada va a estar siempre presente. en la categoría anterior el algoritmo de aprendizaje solo tiene que definir una sola función para mapear un vector de entrada a una salida de tipo categorica. cuando algunos de los valores de entrada faltan el algoritmo de aprendizaje debe aprender un conjunto de funciones dónde cada función corresponde a clasificar al lector de entrada X a partir de un subconjunto diferente de sus características faltantes . una forma de definir eficientemente este gran conjunto de funciones que deben ser aprendidas es aprender la distribución de probabilidad sobre las variables de entrada y luego resolver la tarea de clasificación completando las variables faltantes a partir de la distribución de probabilidad aprendida.

* Regresión: Este tipo de tarea programa de computadora debe predecir un valor numérico dado un cierto vector de entrada punto para resolver esta tarea el algoritmo de aprendizaje tiene que aprender una función . este tipo de tarea es similar a las tareas de clasificación, excepto por el hecho de qué formato de salida es diferente. que decir en el caso de una tarea de clasificación se tiene un número infinito de posibles valores de salida mientras que en una tarea de regresión el número de valores posibles de salidas infinito. un ejemplo de tarea de regresión es la de predecir el valor hay que debe venderse una casa, dadas las características de dicha casa cómo entrada para el algoritmo de aprendizaje.
* Transcription: En este tipo de tarea el sistema de machine learning observar una representación relativamente desestructurada de algún tipo de dato y transcribir la información formato textual. por ejemplo en reconocimiento óptico de carácter el programa de computadora una fotografía qué contiene una imagen contexto y debe devolver el texto en la forma de una secuencia de caracteres. a forma de otro ejemplo Google Street utiliza técnicas de Deep learning para procesar la dirección es de los números de las casas utilizando transcripción.
* traducción de máquina : en una tarea de traducción de máquina la entrada input consiste en una secuencia de símbolos en un determinado lenguaje y el programa de computadora de convertirlos en una secuencia de símbolos de otro lenguaje. Este tipo de tarea es comúnmente aplicada a lenguajes naturales como por ejemplo traducir del español al portugués.
* salida estructura las tareas de salida: estructura involucran cualquier tarea donde la salida es un vector con importantes relaciones entre los diferentes elementos de dicho vector. está categoría subsume a otras como transcripción y traducción también otras. un ejemplo es el de la función de un parser donde la entrada un texto en lenguaje natural y la salida es un árbol que escribe la estructura gramatical de dicho texto etiquetando cada nodo del árbol como verbos sustantivos adverbios etcétera.
* detección de anomalías: este tipo de tarea el programa de computadora analiza un conjunto de eventos objetos y reporta algunos de esos como inusuales o atípicos. un ejemplo de tarea de detección de anomalías es la detección de fraudes de tarjetas de crédito. modelando los hábitos de compra una compañía de tarjeta de crédito puede detectar el mal uso de dichas tarjetas. si un ladrón roba una tarjeta de crédito las compras el ladrón vendrán de una distribución de probabilidad diferente .
* síntesis y sampleo puntos en este tipo de tarea el algoritmo de machine learning de generar nuevos ejemplos que son similares a aquellos del conjunto de datos entrenamiento.
* Denoising: En este tipo de tarea algoritmo de machine learning tiene como entrada un ejemplo corrompido obtenido a partir de un proceso de corrupción no conocido aplicado a un ejemplo limpio. el algoritmo debe entonces prefiero ir el ejemplo limpio a partir del ejemplo corrompido.
* estimación de densidad estimación de función de probabilidad de masa: el problema de estimación de densidad el algoritmo de machine learning debe aprender una función qué puede ser interpretada cómo es la función de densidad de probabilidad ( si x es continuo) como la función de probabilidad de masa ( si x se discreta). En general todos los tipos de tareas descriptos arriba requieren que el algoritmo de aprendizaje al menos capturen la estructura de la distribución de probabilidades. este tipo de tarea es utilizada para completar valores faltantes en los datos

## Evaluación de los modelos [Deep Learning with Pyton, hoja 100]

Existen diferentes métodos para evaluar el desempeño de un modelo de Machine Learning. El uso de tal o cual modelo depende de la naturaleza de los datos y la cantidad de ellos.

La forma mas simple de evaluar un modelo es dividirlo en un dataset de entrenamiento y otro de testeo. En este caso, se entrena a el modelo utilizando como datos de entrada el dataset de entrenamiento y luego se evalua al modelo a partir de los datos de testeo. Es importante notar que los datos de testeo deben ser datos a partir de los cuales el modelo no fue entrenado, por lo que estos dos conjuntos deben ser siempre disjuntos. De esta forma, se evalúa la capacidad de **generalizar** del modelo.

En el proceso de entrenamiento se buscan los valores óptimos de los parámetros que permitan el mejor desempeño del modelo en el conjunto de entrenamiento, a partir de un método de optimización, como el descenso por el gradiente. Ahora bien, como se buscan los valores óptimos de los hiper-parametros? Bajo este esquema, lo más común es entrenar y testear al modelo a partir de diferentes combinaciones de hiper-parametros, seleccionando al final la combinación a partir de la cual el modelo ha obtenido el mejor desempeño en el conjunto de testeo. Estas combinaciones pueden crearse de manera manual o mediante algún método de optimización, como la Optimización de Bayes.

Puede observarse ahora que, de la misma forma que se optimizaron los parametros del modelo para obtener un buen desempeño en el conjunto de entrenamiento, los hiper-parametros tambien fueron optimizados para alcanzar un buen desempeño en el conjunto de testeo. Por lo tanto, existe la necesidad de un tercer conjunto, que permita verificar el poder de generalización del modelo que mayor desempeño alcanzó en el conjunto de testeo. Este último conjunto es llamado dataset de validación.

Cuando los datos disponibles son escasos, el método anterior no debe ser utilizado, ya que las diferentes particiones dejarán de ser estadísticamente representativas de la fuente original de los datos (sobretodo los conjuntos de testeo y validación, que, en general, representan un porcentaje mínimo del dataset). En estos casos, la técnica de validación *k-fold* debe ser utilizada. En el caso más simple, esta técnica consiste en dividir el *dataset* en particiones. Para cada partición , se entrena el modelo en las particiones restantes y se lo evalua en la particion . La evaluación final del modelo es igual a la media del desempeño en cada una de las iteraciones.

En el caso de que se está prediciendo el futuro a partir del pasado, como es el caso de las *time-series,* ciertas consideraciones especiales deben ser tomadas en cuenta. Para empezar, los datos no deben ser aleatoriamente reordenados. Y, más importante, sin importar la técnica de validación, los datos de testeo deben ser siempre temporalmente posteriores a los datos de entrenamiento. Existen técnicas de validación específicas para las series de tiempo que serán explicadas más adelante.

## Generalizacion, underfitting y overfitting

El objetivo principal en Machine Learning es que el modelo entrenado para llevar a cabo una tarea determinada tenga un buen desempeño en datos de entrada nuevos. Esto es, en casos que no habían sido visto antes por el modelo, y no solamente en los casos que fueron utilizados para entrenar a dicho modelo. La habilidad de un modelo para desempeñarse bien en datos que no habían sido observados es llamada generalización.

Como se explica en la sección anterior, al entrenar un algoritmo de Machine Learning, se utiliza un conjunto de entrenamiento. A partir del cual se intenta minimizar el error, lo que puede considerarse un problema de optimización. A partir de esto, se calcula el error de generalización o de testeo. El error de testeo es el error esperado de un modelo al predecir a partir de un caso no observado, que es calculado a partir del conjunto de testeo. Los parámetros del modelo (aquellos que se ajustan en el proceso de optimización) son ajustados de acuerdo al conjunto de entrenamiento, por lo que se espera que el error de testeo sea siempre mayor al error de entrenamiento.

Por lo tanto, se busca lograr dos objetivos claros al momento de validar un algoritmo de Machine Learning:

* Lograr que el error de entrenamiento sea bajo, y
* Lograr que el error de testeo sea lo más similar posible al error de entrenamiento.

Estos dos objetivos se corresponden con dos desafíos importantes a superar en Machine Learning: underfitting y overfitting. Underfitting se da cuando el modelo no logra obtener un error de entrenamiento lo suficientemente bajo, mientras que overfitting ocurre cuando la diferencia entre el error de entrenamiento y el de testeo es demasiado grande.

La tarea a la hora de entrenar un modelo es encontrar un punto medio de tal modo de que no se produzca ni overfitting ni underfitting. Esto último se logra aumentando o disminuyendo la capacidad del modelo. La capacidad se refiere a la complejidad del espacio de hipótesis, esto es, todas las funciones que que el algoritmo de aprendizaje puede llegar a seleccionar como posible solución hipótesis.

Por ejemplo, el espacio de hipótesis de la regresión lineal es el conjunto de todas las posibles funciones lineales de la forma

.

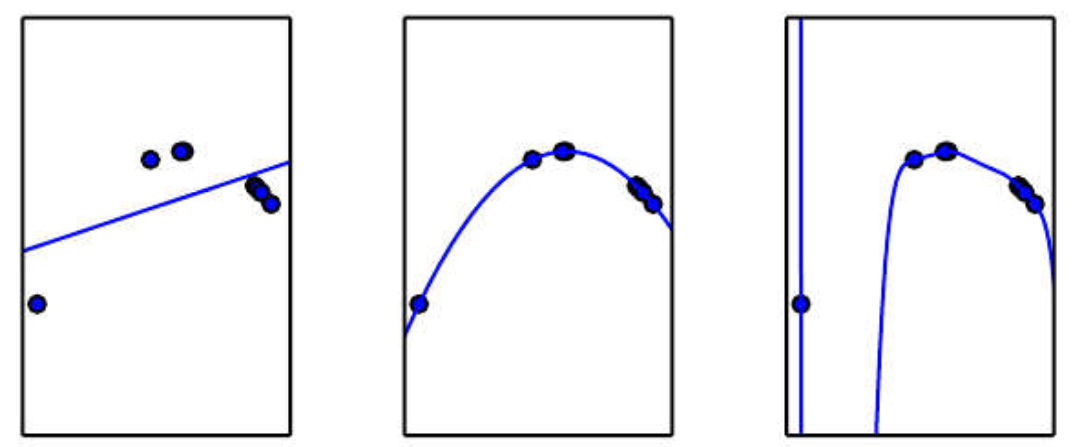
La capacidad de un algoritmo de este tipo puede aumentarse generalizando el espacio de hipótesis para incluir polinomios de grado de la forma

.

El desempeño de los modelos de Machine Learning va a ser mejor cuanto mejor se adapte su capacidad a la complejidad de la tarea a llevar a cabo. De esta forma, los modelos con una baja capacidad son incapaces de resolver tareas complejas, por lo que va a producirse underfitting. Mientras que los modelos con alta capacidad pueden resolver tareas complejas, pero sí su capacidad es más alta que la necesaria para resolver una determinada tarea, pueden cometer overfitting, ya que la función aprendida no tendrá un buen desempeño en los casos de testeo.

En la Fig X pueden verse tres diferentes modelos entrenados a partir de un conjunto de datos. Dicho conjunto de datos fue generado evaluando valores aleatorios de x en una función cuadrática. En este caso, la tarea de los modelos es la de predecir el valor de para cada valor de . Los modelos fueron ajustados a una función lineal, una función cuadrática y un polinomio de grado 9, respectivamente. Cada modelo ha debido ajustar un número diferente de parámetros, uno por cada grado del polinomio. Es decir, el primer modelo debió ajustar 2 parámetros (el peso asociado a la variable de grado 1 y aquel asociado a la variable de grado 0, también llamado bias), el segundo 3 y el ultimo 9. Por lo tanto, la capacidad de los modelos aumenta de acuerdo al grado del polinomio, ya que el conjunto de hipótesis de un polinomio de grado incluye al de uno de grado , si . Puede observarse entonces que:

* El modelo de la izquierda produce underfitting, ya que debido a su baja capacidad no puede obtener un bajo error de entrenamiento.
* El modelo de de la derecha produce overfitting, ya que tendrá un error alto en los casos de testeo por no haber podido aprender la función verdadera a partir de la cual el conjunto de datos fue generado.
* Por último, el modelo del centro la capacidad es la adecuada para la tarea a realizar, por lo que el modelo a logrado generalizar a partir de los casos de entrenamiento.



Funcion lineal, Polinomio grado 2, grado 9.

## Regularizacion (debajo de funcion de costo/perdida)

En la seccion anterior, se ha presentado un trade off entre overfitting y underfitting a partir de aumentar o disminuir la capacidad de un algoritmo de aprendizaje al cambiar el modelo del cual aprende (la función). Este proceso puede llevarse a cabo de forma mas general, ya que es posible configurar a un algoritmo para preferir algunas funciones pertenecientes a su espacio de hipótesis por sobre otras de acuerdo a cierto criterio utilizando una técnica de regularización. Es decir, las funciones menos preferidas van a ser elegidas sólo si se desempeñan mucho mejor que las preferidas en el conjunto de entrenamiento. Puede interpretarse a la regularización como una forma general de variar la capacidad de un algoritmo de aprendizaje, ya que quitar a una función del espacio de hipótesis puede interpretarse como una preferencia infinita de las funciones que aún pertenecen al espacio de hipótesis por sobre la función excluida. Por lo tanto, las técnicas de regularización tienen como objetivo disminuir la capacidad de un algoritmo de aprendizaje sin excluir explícitamente ninguna función del espacio de hipótesis, sino que dejando que el algoritmo defina cuales son las funciones más apropiadas a aprender en el proceso de entrenamiento.

Se define a la regularización como cualquier cambio que se se le haga al algoritmo de aprendizaje para reducir el error de testeo o generalización, pero no su error de entrenamiento. Dicho de otra manera, se busca que el modelo aprenda la función más simple posible que consiga un bajo error de entrenamiento, de tal forma que pueda aun generalizar, es decir, tener un bajo error de testeo.

A modo de ejemplo, supóngase que se generan conjuntos de entrenamiento y testeo a partir de una función cuadrática. A continuación, se entrena un modelo cuya capacidad son todas las funciones polinómicas. Si no se utilizase ninguna técnica de regularización, el algoritmo aprendería una función utilizando un polinomio de grado (es decir, no tendrá ninguna preferencia por aprender polinomio de grado 2). De esta forma, el error de testeo será elevado, ya que el modelo predecirá el conjunto de testeo a partir del polinomio aprendido y no específicamente a partir de la función cuadrática. En este caso, una técnica de regularización podría penalizar al algoritmo por cuanto más grande sea el grado del polinomio. Debido a esto, el algoritmo deberá minimizar, no solo el error de entrenamiento, sino que también el grado del polinomio. Por lo tanto, el algoritmo terminará aprendiendo la función cuadrática. Nótese que el algoritmo no deberia aprender un polinomio de grado 1 (función lineal) ya que de esta forma aumentaría el error de testeo (suponiendo que el conjunto de entrenamiento es lo suficientemente grande como para no parecerse a una función lineal).

A continuación se presentan las técnicas de regularización mas comunes que pueden ser utilizadas tanto en algoritmos de Machine Learning como de Deep Learning. Las técnicas de regularización que son aplicadas específicamente para algoritmos de Deep Learning serán detalladas en su propia sección.

### Penalización de la norma de los parámetros

Este tipo de regularización es el más comunacho de todos y consiste en añadir una penalidad de la norma de los parámetros a la función objetivo , de tal forma que el algoritmo de optimización deba minimizar, no solo el error, sino que también la norma de los parámetros. Si llamamos a esta nueva función objetivo, entonces:

Donde es un hiper parámetro que modifica la contribución del término de penalización a la función objetivo. Si es 0 entonces no hay regularización y cuanto mayor es mayor es la regularización aplicada. Existen diferentes funciones de la norma de los parámetros que pueden ser usadas como el término en la función objetivo regularizada. A continuación se presentan varios de ellos:

* Regularizacion: Esta técnica de regularización es también conocida como wight decay o ridge regression. La regularización acerca el vector de los parámetros al origen añadiendo el siguiente término de regularización

donde w es el vector de los parámetros.

* Regularizacion : Esta técnica de regularización tiende a que algunos parámetros tomen el valor de 0, por lo que puede ser usada como para seleccionar las características más importantes para el modelo (feature selection). Está definida por el término

,

## Modelos lineales y no Lineales

Logistic regression is a linear classifier since the input of the predicted log-odds is a linear

function of . The advantage of linear models is that they can be fit efficiently and reliably,

but the capacity of the model is limited to linear functions, so the model cannot understand

the interaction between any two input variables.

## Clasificación y Regresión (definicion mas precisa)

En esta tesis, defino a la prediccion del comportamiento sedentario futuro (PCSF) como el problema de predecir si un usuario va a superar, o no, 1.5 METS en el futuro cercano. En este punto, podemos divergir el camino de investigacion en dos. En el primero, se pueden tomar los datos de la actividad fisica, calcular el nivel de MET y generar modelos que puedan predecir el valor de MET calculado en el futuro cercano. En este primer caso, la variable a predecir es un valor continuo. En el segundo camino, se puede utilizar el valor de MET calculado para generar dos clases, una que indique comportamiento sedentario y otra que no. En este caso, la variable a predecir es discreta y puede tomar solo dos valores diferentes.

Estos dos tipos de caminos distinguidos representan formas de atacar el problema con algunas diferencias, ya que cada una tiene ventajas y desventajas. Sobretodo, los modelos utilizados para cada una de ellos va a ser diferentes. Es decir, tanto la forma en que cada modelos se entrena como evaluación de su desempeño va a diferir. En el primer camino nombrado, estamos en la presencia de un problema de regresión, donde se busca predecir una variable continua. En el segundo, el problema es de clasificación, donde se busca predecir una variable discreta. A continuación, paso a describir las características, similitudes y diferencias entre estos dos tipos de modelos.

## Métricas de evaluación

Para evaluar las habilidades de un algoritmo de Machine Learning debe utilizarse una métrica cuantitativa de su desempeño. Las métricas utilizadas poseen diferentes cualidades y existen diferentes de ellas para cada tipo de tarea. Además, cada una de las métricas existentes se desempeña mejor o peor dependiendo de las características específicas del problema. A continuación, se detallan y discuten algunas de las métricas de evaluación más utilizadas para los las tareas de regresión y clasificación.

## Métricas de regresión

### Mean Squared Error:<https://www.dataquest.io/blog/understanding-regression-error-metrics/>

## Métricas de clasificación

* Matriz de confusión:

La matriz de confusión es una de las formas más intuitivas para analizar cómo está funcionando un algoritmo de clasificación. La matriz de confusión Es una matriz cuadrada dónde se corresponde con la cantidad de clases posibles. Las columnas de la matriz de confusion representan a los valores reales de las clases, mientras que las filas representan los valoers predictos. En cada celda de la matriz de hallan entonces la cantidad de casos clasificados como cuando su clase era . Para una tarea de clasificación binaria, podemos observar que la matriz de confusión será de y contará con cuatro celdas. Estas cuatro celdas representan los 4 tipos en los que pueden clasificarse las predicciones de un algoritmo de clasificación. En el caso de tareas de clasificación con más de dos clases, el problema puede descomponerse en tareas de clasificación binaria. Cabe aclarar que en un problema de clasificación binaria las dos clases son comúnmente denotadas de diferentes formas, como verdadero y falso, positivo y negativo, 1 y 0, etc. A continuación se explican los valores con los que se completan las matrices de confusión. En este caso se usa como valores de las clases Positivo y Negativo.

Verdaderos positivos (TP): los casos en los que la clase actual (la correcta) es 1 y se corresponde con la clase predicha (1). El término verdadero se debe a que el clasificador ha hecho una predicción correcta.

Verdaderos negativos (TN): los casos en los que la clase actual (la correcta) es 0 y se corresponde con la clase predicha (0).

Falsos positivos (FP): los casos en los que la clase actual (la correcta) es 0 y no se corresponde con la clase predicha (1). El término falso se debe a que el clasificador ha hecho una predicción incorrecta.

Falsos negativos (FN): los casos en los que la clase actual (la correcta) es 1 y no se corresponde con la clase predicha (0).

Ahora que el concepto de matriz de confusión ha sido presentado, vamos a ver que cada una de las métricas que serán presentadas a continuación son calculadas a partir de los 4 tipos de resultados de clasificación.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | |
|  |  |  |  |
|  |  |  |  |
|  |  |  |

### Accuracy:

Accuracy y Precision son dos métricas que llevan el mismo nombre si son traducidas al español, porque lo se mantienen sus nombre en inglés. Se denomina Accuracy al número de predicciones correctas hechas por el modelos sobre la cantidad de predicciones hechas.

Esta métrica no debe ser utilizada cuando las clases están desbalanceadas, es decir, que la cantidad de casos de una clase es mucho mayor a la cantidad de casos de la otra. Esto es así porque la ecuación de esta métrica es sensible al desbalanceo de clases. Supongamos, por ejemplo, que se aumenta en un factor el número de casos de la clase negativa, mientras que el desempeño del clasificador permanece igual. Reescribiendo la ecuación, obtenemos . Por ejemplo, en la tarea de predecir si una persona tiene cáncer o no, supongamos que en el dataset con el que se testea al algoritmo el 99% de las persona no tiene cáncer. Si el algoritmo de clasificación simplemente preside siempre que la persona no tiene cáncer, obtendría un Accuracy de 0.99. En estos casos, es preferible optar por otros tipos de métrica. Como definición mas general, todas las metricas que usan solo una columna de la matriz de confusion son aptas para evaluar el desempeño de un clasificador con un *dataset* desbalanceado, lo cual no quiere decir que no existan metricas que utilicen las dos columnas de la matriz de confuncion que sean aptas para lo anterior.

### Tasa de error:

esta metrica es el complemento de la metrica anterior, es decir, ambas suman 1. Esta metrica es, tambien, sensible al desbalance entre la clases.

### Precision o Valor de Prediccion Positiva:

proporcion de casos clasificados correctamente como positivos sobre el total de casos clasificados como positivos. Puede observarse que la precision no depende de ninguna manera de los casos negativos. Esta metrica es comunmente utilizada cuando se busca minimizar el numero de falsos positivos.

### Recall o Sensitividad o Tasa de Verdaderos Positivos(TPR):

es la proporcion de casos positivos que fueron correctamente clasificados. Esta metrica es comunmente utilizada cuando se busca minimizar el numero de falsos negativos.

### Rollout o Tasa de Falsos Negativos (FPR):

### Especificidad o Tasa de Verdaderos Negativos (TNR):

esta metrica evalua lo contrario al Recall. La especificidad es la proporcion de casos negativos que fueron correctamente clasificados.

### Rollout o Tasa de Falsos Positivos (FPR):

Tanto la Tasa de Verdaderos Positivos como la Tasa de Verdaderos Negativos son metricas que no son sensibles a datasets desbalanceados, ya que estan computadas a partir de una sola columna de la matriz de confusion. Por lo que, si realizamos el mismo calculo que realizamos con la metrica Accuracy, el factor se cancela.

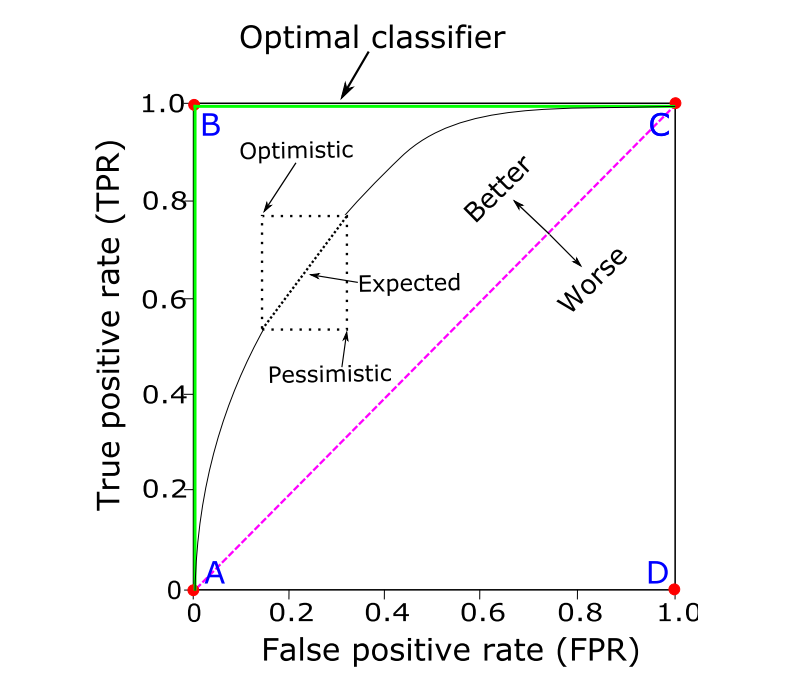
### Metrica F1 o F1 Score:

esta metrica combina a las metricas Precision y Recall, entiendiendo que estas dos ultimas representan aspectos diferentes del desempeño de un clasificador. F1 se obtiene calculando la media armonica entre Recall y Precision. El valor de la metrica F1 es similar a la media aritmetica cuando Precision y Recall son similares, mientras que cuando un valor es mucho mas alto que el otro, toma un valor cercano al mas bajo.

Por otro lado, como esta metrica utiliza a la metrica Precision, es sensible a datasets desbalanceados, por lo que no es recomendada su utilizacion en esos casos.

### ROC:

esta metrica, a diferencia de todas las anteriores, no da como resultado un numero, sino que un grafico. Dicho grafico tiene dos dimensiones, donde el eje representa la Tasa de Falsos Positivos (FPR) y el eje representa la Tasa de Verdaderos Positivos (TPR, o Recall).



Si se tiene un clasificador cuya salida es un valor discreto (0 o 1) sólo podra anotarse un punto en el grafico ROC. En cambio, en el caso de clasificadores cuya salidad sea un valor continuo y no un valor discreto, puede generarse una curva completa. El calculo de la curva se obtiene ordenando de menor a mayor el score (confianza) generado por el clasificador para cada caso de testeo. A partir de eso, se generan tantos thresholds como scores diferentes haiga. Cada threshold va a definir a partir de que valor un caso es considerado como positivo. Por ejemplo, para el threshold 0.7, todos los casos para los cuales el clasificador haya dado un score mayor a 0.7 seran considerados como positivos, mientras que el resto serán negativos. A continuacion, a partir de cada threshold se calculan TPR y FPR para obtener un punto en el gráfico ROC.

It is worth mentioning that the comparison between different classifiers using ROC is valid only when (1) there is only single dataset, (2) there are multiple datasets with the same data size and the same positive:negative ratio.

### AUC:

Area under the ROC Curve: Utilizando ROC es dificil de comparar dos clasificadores, ya que puede suceder que ambos clasificadores tengan un mejor valor en la curva en diferentes partes de esta. Para resumir el desempeño de un clasificador a partir de ROC se calcula el area bajor la curva, que consiste en una suma de trapezoides[extender?] y da como resultado un escalar.

### Precission - Recall curve:

similar a ROC

## Reproductibilidad de resultados

## Funcion de hipotesis

## Funcion de perdida

## Función de optimización

## Tecnicas de Regularizacion

## Deep Learning

Como fue notado brevemente en la introducción, el Deep Learning se diferencia de Machine Learning en que el primero utiliza sucesivas representaciones, lo que permite conseguir representaciones mas complejas. Igualmente, como es en el caso de Machine Learning, todos los algoritmos de aprendizaje tienen por objetivo resolver una tarea específica al aproximar un función .

Todos los algoritmos utilizados en esta tesis son casos de redes neuronales. Cada uno de los algoritmos tiene características que los diferencian de los demás y que los hacen más o menos propensos a ser usados en ciertos tipos de tarea. A su vez, estas diferencias implican la existencia de ventajas y desventajas a la hora de utilizarlos, y no solo teniendo en cuenta los tipos de tareas en los cuales se desempeñan mejor, sino que también entran en juego otros factores como el tiempo de entrenamiento necesario, el numero de parametros a entrenar, el numero de hiper-parametros a ajustar, el tamaño del modelo, etc.

La palabra red de “Redes Neuronales” se debe a la estructura de grafo a partir de la cual estos algoritmos pueden ser representados,

Durante el entrenamiento de una red neuronal, se intenta que la función sea lo más parecida posible a la función subyacente que da origen a los datos de entrada, . Como en el caso de Machine Learning, cada caso de entrenamiento está acompañado por una etiqueta o *label .* En Deep Learning, se busca que el resultado de la *output layer* sea lo más parecido posible a .

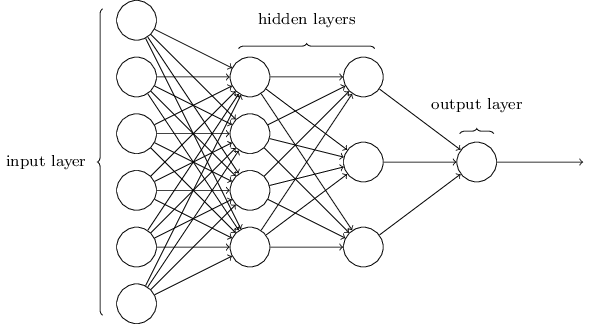
## Modelos de DL utilizados

A continuación, se presentan los diferentes modelos utilizados, todos ellos pertenecientes al área de Aprendizaje Profundo o *Deep Learning.* Para cada uno de ellos, se realiza una breve introducción y se desarrollan sus características, ventajas y desventajas de su utilización. Además, se discute el uso de cada tipo de modelo en el marco de problemas donde los datos de entrada poseen una organización secuencial.

## Perceptrón Multicapa

El Perceptrón Multicapa o Multilayer Perceptron - MLP, por sus siglas en ingles- es la arquitectura más sencilla dentro del campo de Deep Learning y puede aprender múltiples niveles de representación de los datos de entrada para modelar relaciones complejas entre dichos datos. De esta forma, las características de alto nivel están definidos en términos de características de bajo nivel. En este sentido, se toma a cada capa como un nivel de representación que utiliza la representación de la capa anterior. Por lo tanto, los MLP se componen de una secuencia de capas, donde cada capa modela una función que utiliza como entrada la salida de la capa/función anterior. De esta forma, puede representarse a los MLP como una composición de funciones (Goodfellow, Bengio, and Courville 2016).

En las MLP, cada capa está compuesta por un cierto número de unidades computacionales o neuronas. A su vez, cada neurona posee conexiones directas con todas las neuronas de la capa anterior. Cuando existen conexiones entre todas las neuronas de la capa y la capa , la capa es llamada capa densa. Cada neurona lleva a cabo cierto cálculo y produce como salida un valor que es luego emitido a hacia todas sus conexiones salientes. Cada conexión tiene un peso que corresponde a cuán fuertemente están conectadas dos neuronas, Estos pesos son los parametros de las MLP, y más particularmente, de las capas densas, y deben ser ajustados por el algoritmo de aprendizaje. Típicamente, la computación que lleva a cabo cada neurona está separada en dos etapas: primero realiza la suma ponderada de todas sus conexiones entrantes y luego aplica una función de activación.



Como ya se ha mencionado, la función aplicada por cada capa puede ser vista como una representación diferente de la salida de la capa anterior. Es interesante analizar esto gráficamente a través de elementos del álgebra lineal. Para ello, es necesario mostrar qué cálculos lleva a cabo cada capa densa. Sea:

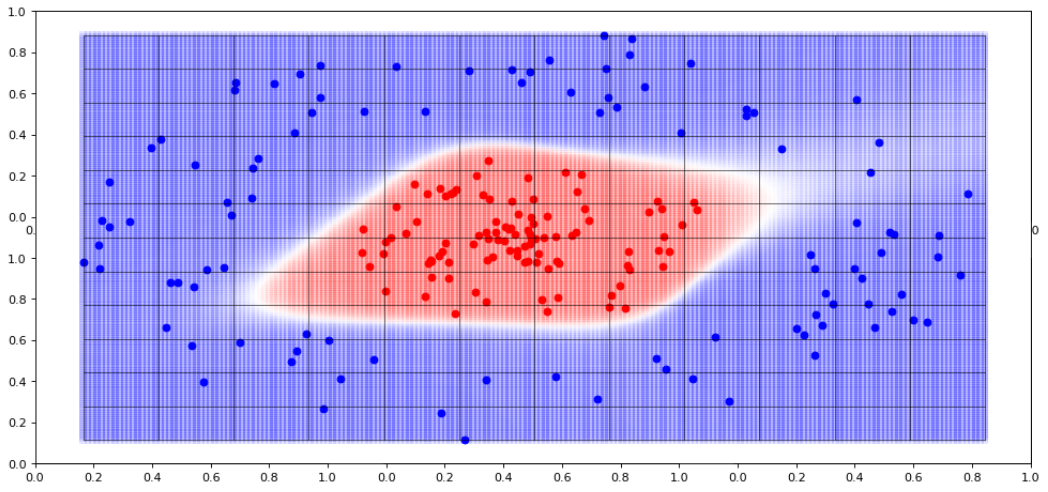
* la dimensión de entrada y la cantidad de neuronas de la capa densa
* el vector de salida la capa , perteneciente a .
* la matriz que contiene todos los pesos entre la capa e , perteneciente a .
* b el *bias,* perteneciente a .
* A una función de activación no lineal.

Entonces, la salida de la capa es la funciùon :

)

Donde puede ser interpretada como una matriz de transformación lineal que mapea un vector de a , es un desplazamiento del vector resultante de la transformación lineal, y es aplicada a cada componente del vector resultado. Por lo tanto, .

A modo de visualizar lo explicado anteriormente, implemente un Jupyter Notebook[[3]](#footnote-2) en la cual se entrena a un MLP simple cuya tarea es la de clasificar un dataset artificial formado por 2 clases de puntos concéntricos pertenecientes a . Una vez que el modelo ha sido entrenado, puede mostrarse la predicción del predictor para cada punto del espacio, junto a su confianza, como muestra la figura. Pero lo interesante es que pueden obtenerse las salidas de las capas intermedias y verse, paso a paso, cómo se van llevando a cabo las transformaciones y cuales son las representaciones de cada capa. En la ***FIGURA***, la primera fila muestra las transformaciones lineales de los vectores generados por la capa anterior más un desplazamiento, mientras que las columnas muestran el resultado de la aplicación de la función de activación (en este caso la función es sigmoide). Debe notarse que como las transformaciones aplicadas por cada capa se muestran de forma acumulativa, los espacios no pertenecen a la salida de una transformación lineal. Por el tipo de dataset, las capas internas de la red entrenada tiene 3 neuronas y no 2. Por lo tanto, todas las representaciones que se muestran son parciales, es decir, se grafican 2 de las 3 dimensiones. Sin importar cual sea el dataset, la MLP intentará hallar un hiperplano que separe ambas clases[[4]](#footnote-3). Como en este caso no existe una recta que separe ambas clases, es necesario proyectar el dataset a una dimensión mayor, en cuyo caso la red termina aprendiendo un plano que divide ambas clases. En las 2 dimensiones graficadas, se puede observar que las dos clases son linealmente separables.





Una de las desventajas de utilizar este tipo de redes para el modelado de secuencias es que ignora completamente la topología de la entrada, es decir, timesteps y características se representan en la misma dimensión, por lo que no es posible “obligar” a la red a que tenga en cuenta estas representaciones. Dicho de otra manera, las variables de entrada podrían ser presentadas en cualquier orden sin afectar el resultado del entrenamiento [15]. En este trabajo se consideró dicha arquitectura con el fin de evaluar si utilizar arquitecturas que tengan en cuenta la estructura secuencial de los datos, representa o no una mejora en la precisión de los modelos.

Another difference between DNNs and linear models is that DNNs have some hyperparameters that have to be tuned in order to maximize the performance of the model. Some of the decisions that have to be made are: how many layers the network should contain, how these layers should be connected to each other, how many units should exist in each layer, which activation function should use each unit, which optimizer should be used, how many epochs should be used for training, which regularization technique should be used, etc.

## Redes Neuronales Recurrentes

Las Redes Neuronales Recurrentes -o RNN por sus siglas en inglés- son modelos donde la entrada de la red son secuencias. Las unidades de las RNN mantienen un estado interno que es propagado a través del tiempo. Dicho estado interno funciona como la memoria de las unidades y puede actuar como una representación de todo aquello que la red haya visto en la secuencia de entrada hasta el momento. En síntesis, las RNN tiene como entrada un vector que representa una secuencia (por ejemplo: una oración) y las neuronas o unidades procesan cada elemento de dicho vector uno a la vez e iterativamente [Elman, 1990]. Las RNN son representadas computacionalmente mediante grafos cíclicos, es decir que poseen ciclos. Dichos ciclos representan la influencia del valor actual de una variable en su valor en un momento futuro.

Las RNN han ganado mucha popularidad por su buen desempeño en tareas relacionadas al modelado del lenguaje [Hermans, 2013] y traducción de máquina [Bahdanau, 2015]. Desgraciadamente, las arquitecturas más simples de RNN son difíciles de entrenar [Pascanu, 2013], ya que presentan problemas como *vanishing gradient* y *exploding gradient*, por lo que en la práctica se utilizan arquitecturas más complejas que logran sortear en cierta medida dichos problemas. Estas variantes utilizadas son las LSTM [Hochreiter, 1997] y las GRU [Kyunghyun, 2014] que añaden mayor cantidad de información a cada neurona en comparación con las RNN más simples, como las compuertas de olvido.

Más allá del tipo de neurona que se use en una RNN, el mecanismo es similar a todas ellas. Como ya se ha dicho, la entrada de cada neurona es una secuencia que es procesada iterativamente, es decir, un elemento de la secuencia a la vez. Cada neurona tiene un estado interno que va cambiando a medida que la secuencia es procesada (más o menos complejo dependiendo del tipo de neurona recurrente), emulando la memoria de la red.

Es importante señalar aquí una diferencia clara entre las RNN y el tipo de arquitectura explicada en la subsección anterior, las MLP. Como ya ha sido explicado, las unidades de las RNN reciben y procesan la información de manera iterativa. De esta manera, el orden en que la secuencia de entrada es procesada es importante y debe respetar su orden natural (por ejemplo, el del tiempo, o de derecha a izquierda en el caso del procesamiento de texto) ya que la red está diseñada para sacar provecho de dicho orden. Si se compara este punto con las MLP, puede verse que, en ellas, cada neurona recibe toda la información “al mismo tiempo”, impidiendo que pueda tomarse en cuenta el orden secuencial. De esto se desprende otra importante diferencia: las RNN comparten los mismo parametros para cada componente de la secuencia, a diferencia de las MLP que poseen parametros diferentes para cada parte de ella [deep learning book, rnn].

## Redes Neuronales Convolucionales

La Redes Neuronales Convolucionales -CNN, por sus siglas en ingles- son muy comunes en el procesamiento de imágenes y presentan varias diferencias con respecto a las MLP. Por un lado, las CNN toman como entrada datos claramente estructurados, en los que las variables cercanas están fuertemente correlacionadas (por ejemplo, imágenes). Por otro lado, este tipo de redes reduce en gran medida el número de parámetros a entrenar, ya que éstos son compartidos por todas las neuronas de una determinada capa. Estos parámetros son llamados filtros, y a partir de aplicar la convolución a cada sector de los datos de entrada (sean de la entrada de la red o de una capa intermedia), extraen características locales que luego son combinadas sucesivamente generando características globales [15]. Por lo tanto, dichos filtros pueden ser interpretados como características que crecen en complejidad con cada capa de la red. Comúnmente, entre cada capa de convolución se aplica un método llamado *pooling* que reduce la cantidad de neuronas y, por lo tanto, la resolución de los datos procesados, obteniendo invariancia ante la translación y mejorando la capacidad de generalización de la red.

Las CNN han sido utilizadas con éxito en tareas de predicción de series de tiempo en diversas investigaciones [16, 17, 18]. Más recientemente, este tipo de redes, han mostrado un rendimiento del estado del arte en tareas que eran dominadas por las Redes Neuronales Recurrentes (RNN), como las síntesis de audio, modelado de lenguaje a nivel de palabra, traducción de máquina [19, 20, 21]. A partir de estos trabajos, se han implementado diferentes tipos de CNN para el modelado de secuencia, y, en especial, para la predicción en series de tiempo.

A related idea is the use of convolution across a 1-D temporal sequence. This convolutional approach is the basis for time-delay neural networks (Lang and Hinton, 1988; Waibel et al., 1989; Lang et al., 1990). The convolution operation allows a network to share parameters across time but is shallow. The output of convolution is a sequence where each member of the output is a function of a small number of neighboring members of the input. The idea of parameter sharing manifests in the application of the same convolution kernel at each time step. Recurrent networks share parameters in a diﬀerent way. Each member of the output is a function of the previous members of the output. Each member of the output is produced using the same update rule applied to the previous outputs. This recurrent formulation results in the sharing of parameters through a very deep computational graph.

## Redes Temporales Convolucionales

Las redes temporales convolucionales -o TCN, por sus siglas en inglés-, son un tipo de CNN con ciertas características que la hacen especialmente aplicables a tareas donde la entrada del modelo es una secuencia. En Bai et al. 2018 [Bai, 2018], los investigadores realizan una evaluación sistemática de arquitecturas convolucionales (TCN) y recurrentes. Específicamente, se compara ambos tipos de modelos en tareas que han sido históricamente utilizadas como *benchmark* para comparar diferentes arquitecturas RNN. Los resultados muestran que la común asociación entre el modelado de secuencias y las RNN debe ser reconsiderada, ya que fueron las TCN las arquitecturas que alcanzaron el mayor desempeño en los *benchmarks*. Las TCN ya habían sido utilizadas en otros trabajos [Van den Oord 2016 y Lea 2017] con ciertas variaciones. En [Bai et al. 2018] se implementa una versión de una TCN generica. En dicha versión, se busca que las TCN puedan aprovechar las ventajas que presentan las CNN más simples, pero siendo adaptadas para el modelado de secuencias y la predicción de series de tiempo. Estos cambios introducen ciertas diferencias entre las TCN genéricas de Bai et al y las CNN tradicionales. Las diferencias más importantes son:

• las TCN no utilizan capas de pooling, de tal forma que la dimensión de entrada es igual a la de

salida,

• las TCN siempre aplican la convolución de una sola dimensión (la del tiempo).

• Las convoluciones llevadas a cabo por las TCN son causales, es decir, las convoluciones nunca consideran información del futuro,

• Utilizan dilataciones para aumentar el campo receptivo exponencialmente y así

poder cubrir largas dependencias en los datos secuenciales.

Las TCN propuestas en [Bai et al] están compuestas por capas convolucionales densas, esto es, todas las capas de la red son convolucionales, con excepción de las capas de activación y aquellas utilizadas para regularización. Las TCN están organizadas en bloques residuales. Cada bloque residual está compuesto por dos bloques formados por: una capa convolucional causal con tamaño de kernel y una dilatación , una capa de activación Relu, una capa de regularización - la técnica de regularización por normalización utilizada varía entre *Batch Normalization, Layer Normalization y Weight Normalization* según la implementación, aunque en [Bai et al] se propone la última-*,* yun *Dropout* espacial. La diferencia entre una capa de Dropout normal y una espacial es que la última le da el valor de 0 a mapeos de característica enteras, es decir, al resultado completo de la aplicación de un filtro sobre la entrada. Luego la salida de cada bloque utiliza conexiones residuales en las que la entrada y la salida se suman. Esta suma puede no ser posible si la entrada y la salida del bloque residual tienen dimensiones diferentes, en cuyo caso, se le aplica al vector de entrada una convolución de dimensión 1x1 para hacer coincidir las dimensiones de la entrada y la salida. Esta inconsistencia entre las dimensiones de entrada y la salida de los bloques residuales solo se da en la primera entrada, es decir, en el input de la red, ya que, todas capas internas tienen el mismo tamaño y en todas ellas se aplica zero *padding*.

Un concepto muy importante para las TCN es el de campo receptivo, o *receptive field,* que indica la cantidad de componentes de la entrada a los que el filtro de una capa convolucional arbitraria tiene acceso. Esto es, el campo receptivo indica de cuantos componentes de la entrada un filtro está utilizando información al llevar a cabo la convolución. A diferencia de las CNN comunes, donde el campo receptivo aumenta linealmente con la cantidad de capas convolucionales - aunque esto puede variar a partir del uso de técnicas como *stride* o *max-pooling-* [deep learning book, capitulo cnn] el uso de las dilataciones en las TCN permite que el campo receptivo aumente exponencialmente con cada bloque residual. Esto se da gracias a que las dilataciones se determinan a partir de potencias de 2 (por ejemplo: 1,2,4,8, etc.). Esto permite que las TCN puedan tener una historia larga con pocas capas computacionales, pudiendo reducir la cantidad de parametros necesario a ajustar.

En las TCN, el campo receptivo es un hiper-parámetro a ajustar. Este hiper-parámetro no se ajusta directamente, sino que varía a través de la cantidad de dilataciones y el tamaño del filtro. Es importante que el campo receptivo sea lo suficientemente largo como para cubrir toda la secuencia de entrada [Bai et al].

Diferencias entre tcn y rnn:

Ventajas: paralelismo, campo receptivo flexible, gradientes estables, menos memoria para entrenar, variable length input,

Deseventajas: almacenamiento de datos mientras se evalua, cambio de parmetros si se cambia de dominio.

## Combinacion entre ellos

## Hiper-Parametros

### Diferencia entre parametros e hiper-parametros

### Numero de capas

### Número de neuronas

### Funcion de acticacion

### Numero de epocas

### Tamaño de bache

### Tuning de hiper-parametros

.

Teniendo en cuenta la definicion de PCSF definida anteriormente, analizaremos a continuacion el enfoque de clasificacion. Recapitulando, si consideramos que un nivel de MET menor a 1.5 corresponde a un comportamiento sedetanrio y un MET mayor a 1.5 corresponde a un comportamiendo no sedentario, estamos en la presencia de un problema de clasificacion.

## Predicción de Comportamiento Sedentario Futuro

El problema que nos proponemos resolver es el de predecir el gasto de energía que tendrá una persona en el futuro en base a datos recolectados de sensores de su smartphone. El gasto energético es medido en MET, que es la medida estándar en la comunidad científica que estudia la salud en relación con la actividad física. En el ámbito de la salud, se ha llegado a un acuerdo entre los investigadores en determinar cómo actividad sedentaria a toda aquella actividad cuyo MET asociado sea menor o igual a 1,5.

En el contexto de este trabajo, se define la predicción del comportamiento sedentario futuro (FSBP) como la tarea de decidir si la actividad física de un usuario superará o no, en promedio, 1,5 MET en un futuro próximo.

En el marco del aprendizaje de máquina, al ser el MET un valor continuo, el problema a resolver es una tarea de regresión. Dentro de los algoritmos de aprendizaje profundo, las Redes Neuronales Recurrentes (RNN), las Redes Neuronales Convolucionales (CNN) y las Redes Temporales Convolucionales (TCN) son especialmente aptas para el problema a tratar porque los datos registrados de la actividad del usuario están organizados en series de tiempo en las que la secuencialidad es importante.

En este trabajo, se buscará predecir el valor de MET para un usuario únicamente en la hora siguiente. Es decir, dados los datos de las horas se utilizarán Redes Neuronales para hallar el gasto energético en el tiempo. Nuestro objetivo principal es evaluar la factibilidad de utilizar redes neuronales capaces de aprender, a partir de una secuencia de datos de la actividad de un usuario, patrones de comportamiento que sean útiles para el problema de FSBP. Además, nos interesa evaluar la performance de las TCN en el contexto del problema a abordar, ya que, para tareas de modelado de secuencias, han demostrado superar a las RNN que han sido comúnmente utilizadas como punto de referencia en una gran variedad de dominios [36].

Las TCN son un tipo de CNN [32] que aplican la convolución de una dimensión, son causales (es decir, solo consideran información del pasado), utilizan conexiones residuales y dilataciones para aumentar el campo receptivo. Las TCN demostraron superar ampliamente a las clásicas RNN basadas en GRU o LSTM en los problemas típicos como el problema de sumar o de copiar [36]. Además, las TCN han demostrado tener más memoria que redes recurrentes de la misma capacidad. Por otro lado, y no menos importante, las TCN y las CNN en general, poseen la característica de ser altamente paralelizables, en contraste con las RNN, por los que el proceso de la búsqueda óptima del valor de los hiper-parámetros se realiza de una forma más rápida.

Uno de los hiper-parámetros que varía en las distintas arquitecturas propuestas es la cantidad de time-lags utilizados. Los time-lags son la cantidad de información que se le da a la arquitectura sobre el pasado. Es decir, si se quiere predecir el gasto de energía de un usuario en un tiempo , y la cantidad de time-lags es 3, la entrada de la red neuronal estará formada por las características (features) de los tiempos , y .

En general, los hiper-parámetros a configurar son: tipo de capas utilizadas, cantidad de capas, cantidad de unidades por capa, técnica de regularización a utilizar, time-lags a utilizar, algoritmo de optimización y la función de pérdida. Además, las CNN poseen ciertos parámetros que no poseen las demás arquitecturas, como la cantidad de filtros por capa o el tamaño del núcleo (kernel). Más aún, las TCN poseen hiper-parámetros que no poseen las CNN: como la cantidad de bloques residuales o la lista de dilataciones. Estos últimos hiper-parámetros se tunean de una forma menos aleatoria que los demás, ya que lo que se intenta calcular en este caso es que el campo receptivo sea igual o mayor a la cantidad de time-lags.

En resumen, se evaluarán 4 arquitecturas de redes neuronales: la arquitectura 1 es una RNN, la arquitectura 2 es una CNN, la arquitectura 3 es una TCN y la arquitectura 4 es una NN. La arquitectura 4 utiliza únicamente información de la hora anterior para predecir el valor del MET de la hora siguiente y fue agregada para evaluar la utilidad de utilizar un enfoque de modelado de secuencias para FSBP.

A continuación, la Tabla 1 muestra una descripción de las arquitecturas propuestas. En todos los casos de utiliza el Error Cuadrático Medio (MSE) como función de pérdida, ADAM como algoritmo de optimización, siendo 128 la cantidad de épocas y 64 el tamaño del lote.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Arq. 1 | Arq. 2 | Arq. 3 | Arq. 4 |
| Tipo de capas | LSTM x 2 - FC x 1 | No aplica | Conv1D x 1 - Flatten x 1 - FC x 1 | Dense x 3 |
| Neuronas por capa | 64-32-1 | No aplica | 32 - No aplica - 1 | 64 - 32 - 1 |
| Técnica de regularización | Dropout | Dropout | Dropout y Batch Normalization | Dropout |
| Número de bloques residuales | No aplica | 1 | No aplica | No aplica |
| Lista de dilataciones | No aplica | 1-2-4 | No aplica | No aplica |
| Tamaño del kernel | No aplica | 2 | 2 | No aplica |
| Número de filtros | No aplica | 6 | 32 | No aplica |

FC = Fully Connected

Tabla 1. Descripción de las arquitecturas propuestas

## Evaluación Experimental

En esta Sección, primero se presenta el dataset utilizado para evaluar las diferentes arquitecturas propuestas para la tarea de FSBP. Luego, se describe el pre-procesamiento realizado y las características extraídas del dataset. A continuación, se exponen los usuarios del dataset seleccionados como casos de estudio. Finalmente, se muestran y se discuten los resultados obtenidos. Para la implementación de las arquitecturas se utilizó la librería Keras1 de Python 3. Los scripts implementados para el procesamiento del dataset y el entrenamiento de los modelos se dejan disponibles públicamente2.

## Descripción del Dataset

## Introduccion

Para validar la eficiencia de las arquitecturas propuestas fue utilizado el dataset StudentLife [2014, Wang]. Este dataset fue recolectado a partir de la aplicación de sensado continuo StudentLife. Los participantes en este estudio fueron 48 estudiantes a lo largo de 10 semanas en la primavera de 2013.

Más específicamente, el dataset se recogió de 30 estudiantes de pregrado y 18 graduados. De todo el grupo de estudiantes, 38 eran varones y 10 mujeres. Del grupo de estudiantes de pregrado, dos eran de primer año, 14 de segundo año, 6 de tercer año y ocho de cuarto año. También hubo 13 estudiantes de maestría de primer año y 1 de segundo año, y 3 estudiantes de doctorado. Los participantes fueron racialmente diversos, con 23 caucásicos, 23 asiáticos y 2 afroamericanos.

## Tipos de datos del dataset

Los tipos de datos disponibles incluidos en el dataset StudentLife son de los siguientes tipos:

* App\_usage: contiene información sobre el uso de las aplicaciones de los smartphones.
* Calendar: contiene información sobre el calendario de los usuarios. Es decir, cada registro indica una fecha en la que el usuario tiene un evento en el calendario -no especifica el tipo de evento-.
* Call-log: contiene información sobre las llamadas producidas desde el teléfono del usuario, asi como su duración.
* Dinning: contiene información sobre las compras alimenticias dentro del campus universitario de la Universidad de Dartmouth.
* Education: contiene información diversa de los diferentes cursos a los cuales los estudiantes asistieron a lo largo del estudio. Esto incluye los cursos a los cuales cada estudiante estaba inscripto, información detallada de cada curso, las fechas límites de cada estudiante, asi como también el puntaje obtenido en cada asignatura.
* EMA (Ecological momentary assessments): contiene información sobre diversas y pequeñas encuestas que han sido respondidas por los usuarios a lo largo de las 10 semanas que duró el estudio. Estas encuestas buscan evaluar diferentes cuestiones de orden psicológicos, como el estado anímico, el nivel de estrés y el comportamiento en general. En promedio cada usuario respondió 8 EMAs por dia.
* Sensing: contiene información recolectada a partir de los sensores de los smartphones portados por los usuarios. Este tipo de datos ha sido la principal fuente de datos para entrenar los modelos desarrollados en esta tesis, por lo que tiene su sección propia.
* SMS: similar al tipo de datos Call-log, pero conteniendo información sobre los mensajes de texto enviados y recibidos.
* Survey: contiene 8 diferentes encuestas que fueron realizadas antes y después de comenzar el estudio. Las encuestas son: BigFive, Flourishing Scale, Loneliness Scale, Panas, Perceived Stress Scale, PHQ-9, PSQI y VR-12.

## Datos no disponibles

La gran mayoria del total de los datos recolectados durante el estudio StudentLife no se encuentran disponibles en el dataset disponible publicamente, debido a dos razones. La primera razon es que algunos datos no fueron incluidos en el dataset publicado debido a cuestiones de privacidad, como los mensajes de los SMS, los datos de audio o los nombres de los usuarios. Otros datos que no estan disponibles fueron utilizados solo para procesar otros datos que si fueron incluidos en el dataset publicado. Por ejemplo, los datos de audio recolectados no estan disponibles, pero fueron utilizados para Por ejemplo, el total de los datos recolectados a partir de los diferentes sensores del celular es de 52.6 GB, mientras que en el dataset disponible publicamente solo son 2.6 GB aproximadamente.

## Datos del acelerómetro

Los únicos datos sin procesar que se encuentran disponibles son los del acelerómetro, pero no han podido ser utilizados por problemas técnicos. Los datos del acelerómetro para cada usuario fueron puestos a disposición de tal forma que cada usuario tiene asociado un archivo de base de datos de tipo SQLite. Cada base de datos está compuesta por solo una tabla con dos columnas. La primera columna es llamada timestamp y representa el momento en el que se registró la información del acelerómetro. La segunda columna es llamada feat\_data, que debiera contener la información del registro del acelerómetro. El problema hallado es que esta última columna es de tipo BLOB, que es utilizado para almacenar cadenas de longitud variable de bytes binarios. No logré encontrar la forma correcta de extraer los datos del acelerómetro de tipo BLOB, por lo que dichos datos no han sido utilizados. Es importante señalar que los datos del acelerómetro podrían haber representado una importante mejoría para los modelos implementados, ya que podrían haberles aportado información muy importante acerca de la actividad física del usuario. Esto es así debido a que los datos de la actividad proveídos por el dataset StudentLife tienen una granularidad muy pobre (solo 3 tipos).

## Datos de sensado

Cada uno de estos tipos de datos de sensado son provistos para cada usuario en formato csv (comma separated values, valores separados por una coma). Por lo tanto, a cada tipo de dato le corresponden 48 archivos csv. Cada uno de estos datos está acompañado de un timestamp que indica el horario y la fecha del registro. Estos timestamp tienen el formato de Unix, por lo que se representan mediante un número entero positivo que indica la cantidad de segundos pasados desde el primero de enero de 1970. Por ejemplo el timestamp 1364357009 equivale a 03/27/2013 4:03 AM.

Por lo tanto, cada registro se compone de un timestamp y diversos datos asociados al tipo de dato, pudiendo ser uno solo (como en el caso de activity) o varios, como en el caso de GPS. Aunque este es el caso para la mayoria de tipos de dato de sensado continuo, algunos presentan otro formato, en el cual existen dos timestamp que indican el momento de inicio y finalización de un evento. Al detallar cada tipo de dato de sensado, se clasificará a estos dos tipos de datos como Tipo de Sensado Discreto (TSD) y Tipo de Sensado por Intervalo (TSI).

Los investigadores utilizaron la aplicación BeWell [Lane et al] para proveer un framework para el sensado automático en el estudio StudentLife. Los investigadores aclaran que el estudio StudentLife fue llevado a cabo antes que Google anunciara su servicio de reconocimiento de actividad para teléfonos Android, que fue el utilizado en el estudio. Debido a esto, se utilizó el motor de sensado Jigsaw [2010, J. Lu], desarrollado por los investigadores en un trabajo previo, que permite recolectar y clasificar datos del acelerómetro, el GPS y audio de forma eficiente.

A continuacion, se listan los diferentes datos de sensado continuo incluidos en el dataset StudentLife.

* Tipo de Sensado Discreto
  + Activity: contiene registros de la actividad física del usuario. Este tipo de dato ha sido recolectado aproximadamente cada 2-3 segundos, dependiendo de la frecuencia de sampleo del acelerómetro del smartphone. El clasificador de actividad del motor de sensado Jigsaw alcanzó un 94% de precisión. Los registros de actividad están clasificados en 4 tipos diferentes y se representan mediante un código en los archivos .csv. Por ejemplo, un registro de actividad física es el siguiente [1364356899, 0], que significa que los datos del acelerómetro que fueron recolectados el 27/03/2013 a las 4:01:39 fueron inferidos como estacionario por el clasificador de actividad de Jigsaw. A continuación, se listan los tipos de actividad física junto con el código con el cual están representados en el dataset:
    - Stationary (Cód. 0): el usuario se encuentra en estado estacionario.
    - Walking (Cód. 1): el usuario se encuentra caminando;
    - Running (Cód. 2): el usuario se encuentra corriendo;
    - Unknown (Cód. 3): se desconoce el estado de actividad del usuario (en un intercambio de mails con Rui Wang, el investigador explicó que estos registros se encuentran principalmente al principio del estudio StudentLife, por lo que no deberían representar un inconveniente para realizar experimentos utilizando el dataset).
  + Audio: Contiene registros de audio del usuario. El clasificador de audio de Jigsaw funciona realizando inferencias de audio durante 1 minuto y luego se pausa por 3 minutos, a no ser que el clasificador de conversación detecte que hay una conversación en curso, en cuyo caso el clasificador de audio no se detiene. Este tipo de dato ha sido recolectado aproximadamente cada 2-3 segundos. A continuación, se listan los tipos de audio junto con el código con el cual están representados en el dataset:
    - Silence (Cod. 0): el clasificador de audio infiere silencio.
    - Voice (Cod. 1): el clasificador de audio infiere voz (posiblemente una conversacion).
    - Noise (Cod. 2): el clasificador de audio infiere ruido.
    - Unknown (Cod. 3): el clasificador de audio no reconoce el tipo de audio.
  + Bluetooth: contiene informacion sobre escaneos del dispositivo de bluetooth perteneciente al smartphone de cada usuario. En este caso, y a diferencia de otros tipos TSD, varios registros pueden compartir el mismo timestamp. Esto es normal, ya que cada escaneo reporta informacion sobre las redes bluetooth cercanas halladas, que pueden ser mas de una. Para cada red bluetooth escaneada, se obtiene la direccion MAC y la intensidad de la señal. La pagina reporta que la frecuencia de muestreo para este tipo de sensor es de 1 cada 10 minutos.
  + GPS: contiene informacion sobre el GPS. Dicha informacion incluye la proveniencia de la señal (GPS o Network), la latitud, la longitud y la altitud. La pagina reporta que la frecuencia de muestreo para este tipo de sensor es de 1 cada 10 minutos.
  + Wifi : Similar al tipo de dato Bluetooth, estos registros arrojan informacion sobre las redes de tipo WIFI escaneadas, como la direccion MAC del router, la frecuencia del canal por donde se realiza la conexion y la intensidad de se la señal. Al igual que en Bluetooth, pueden existir varios timestamp identicos.
  + Wifi location: los autores del dataset StudentLife dividieron el Campus de la ciudad de Dartmouth en diferentes clusters. En los registros de este tipo de sensor, podemos hayar una ubicacion aproximada de un usuario en un determinado momento a partir de la red WIFI a la que esta conectado. Los clusters se clasifican en In[x] (dentro del edificio x) y Near[x] (dentro del edificio x).

* Tipos de sensado por intervalo
  + Conversation: intervalo de tiempo en el que el usuario llevo a cabo una conversacion.
  + Phone charge: intervalo de tiempo en el que el smartphone se encuentra siendo cargado.
  + Phone lock: intervalo de tiempo en el que el smartphone se encuentra bloqueado.
  + Light: intervalo de tiempo en el que el smartphone se encuentra en la oscuridad.

## Datasets descartados

## Pre-procesamiento del dataset

A continuacion se describe el proceso llevado a cabo para procesar el dataset StudentLife con la finalidad de ser utilizado por los diferentes modelos planteados para resolver el problema de PCSF. Primero, se realiza un analisis de los datos para conocer las caracteristicas del dataset y definir que tipos de datos van a ser utilizados.

Como se ha dicho con anterioridad, cada registro del dataset StudentLife posee un timestamp asociado. Por lo tanto, todas las características que se generaron corresponden a una combinación de usuario/hora en particular y son algún cómputo que resume algún aspecto de los datos de algún usuario/hora particular. Por ejemplo, un intervalo específico puede corresponder al usuario 10 y la hora 2013-04-24 19: 00–20: 00.

## Análisis de datos

## Tamaño de buckets

Como ya ha sido mostrado, los tipos de datos que provee el dataset StudentLife son un conjunto heterogéneo, es decir, existen muchos tipos diferentes de datos. Por lo tanto, si se quiere utilizar este dataset para generar modelos de Machine Learning con el fin de llevar a cabo la tarea de PCSF, debe llevarse a cabo un proceso previo que permita que dicho dataset sea apto para los diferentes modelos. Para esto, a partir del dataset StudentLife debe generarse un nuevo dataset que esté compuesto por casos, donde cada caso está asociado a un valor de la variable objetivo. Cada caso de estudio estará compuesto por características o features que serán utilizadas para predecir la variable objetivo. Por su parte, la variable objetivo representará el nivel de sedentarismo (variando de acuerdo a si la tarea es de regresión o de clasificación).

El problema más grande que trae la gran heterogeneidad de los datos es su disponibilidad. Podemos distinguir 3 tipos de datos presentes en el dataset según su disponibilidad:

* Las encuestas que se hicieron antes y después del estudio StudentLife,
* los datos EMA y,
* los datos de sensores y derivados de ellos.

De estos 3 tipos, solo en el tercer caso existen datos presentes uniformemente a lo largo de todo el estudio -incluyendo los datos de actividad física-, lo que permite que permite generar una gran cantidad de casos o ejemplo de entrenamiento para los modelos, los que es crucial para lograr entrenar modelos de Machine Learning que alcancen un buen desempeño. Por esta razón, los datasets para entrenar y testear los modelos serán generados a partir de los datos de sensores y derivados de ellos. Los otros dos tipos de datos no serán utilizados debido a su baja disponibilidad. Es decir, si fuesen utilizados solo podrían generarse casos de entrenamiento para aquellos momentos en los que hay datos disponibles, que son muy pocos en comparación a los datos de sensado.

Ahora bien, dentro de los datos de sensores, puede observarse también una disponibilidad variable. Es decir, cada tipo de datos de sensor tiene una cantidad promedio de registros por unidad de tiempo diferente (esta cantidad de registros está dada en la descripción del dataset, pero aun así es calculada junto con otros estadísticos en la siguiente sección a modo de comprobación). Por esta razón, para generar casos es necesario procesar los datos de sensores de tal forma que cada caso pueda incluir información de todos los sensores. Esto último se logra discretizando el tiempo en intervalos, llamados buckets. La longitud de estos intervalos debe definirse teniendo en cuenta la frecuencia que tienen cada uno de los datos. Es decir, cada bucket debe contener por lo menos un registro de cada tipo de dato que se quiere usar, ya que los modelos de Machine Learning no aceptan características sin valor. Así mismo, es válido discutir un número mínimo de registros necesarios para que un tipo de dato en un intervalo de tiempo específico sea válido. Es decir, es necesario responder a la pregunta: a partir de qué cantidad de registros se considera válido un tipo específico de dato? A modo de ejemplo, en validó un bucket de 1 hora de longitud donde se cuenta con solo un registro de actividad física? Puede considerarse que ese registro representa a todo el intervalo (teniendo en cuenta que según la descripción del dataset los registros de actividad física se han registrado cada 3 segundos, es decir, que se esperan 1200)? Si no, cuál debería ser el número mínimo? Es conveniente descartar estos buckets sabiendo que reducen el número de datos de entrenamiento y, en consecuencia, el desempeño de los modelos?

Una vez que el dataset se discretiza en buckets, diferentes características pueden ser calculadas a partir de los registros de cada tipo de dato incluido en cada uno de los buckets. Por ejemplo, para un Tipo de Sensor Discreto en el que se cuentan con registros en un intervalo de tiempo , correspondiente al tamaño de bucket, se generan características a partir de la información disponible. Dichas características pueden ser estadísticos que expliquen los datos disponibles u otros cálculos ad-hoc. Es válido tanto que varios tipos de datos puedan ser combinados para generar una sola característica, así como que varias características sean generadas a partir de un solo tipo de dato.

De acuerdo a las explicaciones anteriores, el tamaño de bucket va a definir cuantos registros estén presente en cada uno de ellos asi como tambien la cantidad de bucket resultantes. De esta manera, se da la siguiente relación: cuanto mayor sea el tamaño de bucket, mayor será la cantidad de datos disponibles en cada uno de ellos, a la vez que la agrupación del dataset en buckets dará como resultado una menor cantidad total de buckets. Esta relación sirve como punto de partida para analizar y definir cuál debería ser el tamaño de bucket. Por un lado, que haya más buckets en total es beneficioso para los modelos ya que tienen un dataset mas grande del cual aprender y la granularidad a la hora de predecir el comportamiento sedentario es menor, por lo que un tamaño de bucket pequeño es deseado. Por otro lado, hay tipos de datos cuya frecuencia de muestreo es muy baja, por lo que si el tamaño de bucket es demasiado pequeño, puede que no haya datos disponibles para ese tipo de sensor.

A partir del análisis anterior, se decidió realizar los experimentos utilizando dos tamaños de buckets diferentes: de 1 hora y de 30 minutos.

Por un lado, se toma el tamaño de bucket de 1 hora debido a que es el utilizado en todos los trabajos relacionados a la tarea de PCSF que usan el dataset StudentLife [citar todos los trabajos]. Gracia a utilizar este tamaño de bucket es posible comparar diferentes cuestiones con esos trabajos, como la cantidad de buckets resultantes del procesamiento, a pesar de que estos trabajos difieren en el enfoque propuesto en esta tesis en relación a cómo medir el comportamiento sedentario. Por otro lado, los intervalos de 1 hora son suficientemente grandes como para incluir en promedio siempre más de un registro por bucket para generar las características.

Con respecto a la división entre buckets de una hora, fue incluida para comparar el impacto en el desempeño de los algoritmos de aprendizaje al doblar aproximadamente la cantidad de casos generados a partir del dataset original, con la desventaja de que habrá más buckets que deberán ser descartados por no presentar registros de alguno de los tipos de datos. Como consecuencia, diferentes modelos que seran entrenados a partir de la utilizacion de varias buckets consecutivos (y no solo uno) se veran afectados porque algunos buckets deberan ser descartados.

## Disponibilidad de los datos

La frecuencia de cada uno de los tipos de datos de sensado continuo fue listada en la sección [Datos de sensado]. Como ya ha sido discutido, para entrenar modelos de Machine Learning, es preferible utilizar los tipos de datos para los cuales haya más información disponible. Es por esta razón que se seleccionan del dataset StudentLife principalmente los tipos de datos de sensado continuo, y se descartan aquellos relacionados a test psicológicos o encuestas EMA.

En esta sección se presenta un análisis sobre la disponibilidad de cada tipo de datos de sensado continuo con el objetivo de verificar la información sobre la frecuencia de cada tipo de dato listada en la descripción del dataset. El análisis de la disponibilidad de cada tipo de dato difiere según si el tipo es TSD o TSI. Para realizar un cálculo de la disponibilidad de cada tipo de dato se discretiza el tiempo en dos tipos de buckets: de 1 hora y de 30 minutos.

Para los TSD, se calcula:

* la cantidad de registros disponibles,
* la cantidad de buckets totales resultantes luego de la discretizacion,
* los buckets para los cuales se tiene información - que serán siempre menores o iguales a los buckets totales-y,
* los siguientes estadísticos: media, desviación estándar, mínimo, máximo de registros por buckets.

Los resultados pueden verse en la tabla X. En el caso especifico del sensor Activity, existen muchos valores nulos, por lo que se incluye una fila con los resultados de eliminar los registros clasificados como unknown.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Cantidad total de registros | Cantidad de buckets para los cuales hay datos | | Cantidad de buckets | | Media y desviación estándar de registros por bucket | | Maximo y minimo de registros por bucket | |
| 1h | 30min | 1h | 30min | 1h | 30min | 1h | 30min |
| Activity | 22.842.191 | 62.807 | 125.009 | 77665 | 155330 | 363.68/193.29 | 182.72/100.09 | 1 / 4820 | 1 / 3538 |
| Activity (sin unknowns) | 22.233.593 | 62.767 | 124.887 | 77665 | 155330 | 354.22/186.45 | 178.02/96.42 | 1 / 4367 | 1 / 3537 |
| Audio | 99.298.223 | 62.792 | 124.968 | 77665 | 155330 | 1581.38/973.7 | 794.58/514.07 | 5 / 45.294 | 1 / 22692 |
| GPS | 202.877 | 59.223 | 115.896 | 77665 | 155330 | 3.42/1.3 | 1.75/0.77 | 1 / 16 | 1 / 9 |
| WIFI | 19.244.309 | 61.750 | 122.526 | 77665 | 155330 | 311.64/522.3 | 157.06/273.09 | 1 / 6144 | 1 / 3500 |
| WIFI location | 1.893.838 | 50.747 | 99.407 | 77665 | 155330 | 37.31/59.74 | 19.05/30.63 | 1 / 357 | 1 / 179 |
| Bluetooth | 1.288.526 | 39.693 | 70.075 | 77665 | 155330 | 32.46/146.35 | 18.38/79.55 | 1 / 4987 | 1 / 2676 |
| Calendar | 1.687 | 563 | 563 | 65660 | 131292 | 2.99/2.31 | 2.99/2.31 | 1 / 20 | 1 / 20 |

Para comenzar el análisis, vamos a empezar por los TSD. Como puede verse en la Tabla X la cantidad de registros disponibles varia mucho entre los diferentes TSD, yendo desde poco mas de 200 mil registros disponibles para el GPS, hasta casi 100 millones en el caso de los datos de audio. A pesar de esto, la cantidad de buckets para los cuales hay informacion no se ve tan afectada, tanto en el caso de buckets de una hora como para los de 30 minutos. En el primer caso la cantidad minima de buckets corresponde al sensor Bluetooth, con 39.693 buckets, mientras que la maxima al sensor Activity, con 62.807 buckets. Sin embargo, para la gran mayoria de los sensores de este tipo la cantidad de buckets ronda los 60.000. El mismo resultado se da en el caso de buckets de 30 minutos, con la diferencia de que las cantidades son de aproximadamente el doble en cada caso.

La columna “Cantidad de buckets” se calcula como la cantidad de buckets totales existentes desde el primer registro al último. Puede observarse que son iguales para todos los TSD y en ambos tamaños de buckets. La fecha del primer registro data del 2013-03-27 a las 05:00:00, mientras que la ultima del 2013-06-01 a las 05:59:59.

Las últimas dos columnas muestran diferentes estadísticos que permiten comprender mejor cómo se distribuye la totalidad de los registros en cada bucket. De la media y la desviación estándar podemos observar que en general la cantidad de registros por buckets difiere mucho, tanto que en algunas casos la desviación estándar es mayor que la media, de lo que se puede deducir que hay periodos de tiempo en los que la recolección de datos es muy alta y otra en las que fue casi nula. Además, nos muestra que en ningún caso la frecuencia de muestreo es igual a la reportada en la descripción del dataset para cada tipo de dato de sensado continuo. Por ejemplo, la frecuencia reportada para el sensor Activity es de 3 segundos, por lo que se esperarían 1200 registros por hora, mientras que la media de registros por buckets es de 360. Además, si se analizan los máximos y mínimos de cada sensor se pueden observar más inconsistencias, ya que todos los sensores poseen por lo menos un bucket para el cual se dispone del número mínimo de registros, es decir uno. Mientras que el número máximo representa picos, también en todos los sensores, de manera extraña ya que en muchos casos se alejan mucho de la media, siendo el caso más extremo para el sensor Audio, para el cual existe un bucket de una hora con 45.294 registros. Este análisis no representa en sí un problema a la hora de procesar el dataset StudentLife, pero si da la pauta de que el dataset posee inconsistencias, por lo que puede que lo resultados obtenidos no sean del todo confiables.

Con respecto a la importante diferencia entre la cantidad de buckets con información disponible para cada sensor, debe realizarse un análisis más detallado. Si bien para que un bucket sea apto para ser utilizado en el dataset de casos debe poseer información sobre todos los sensores que sean utilizados para generar las características, esto no quiere que la falta de registros en un bucket equivalga a la falta de información. Esto es, en algunos casos, puede interpretarse la ausencia de datos como información útil que pueda ser computable. Por ejemplo, en el caso del sensor Bluetooth, que contiene registros correspondientes a las redes Bluetooth cercanas, la falta de registros en un bucket determinado puede interpretarse como que no se han detectado redes Bluetooth cercanas. Por lo tanto, si una característica computada para cada bucket es el número de redes Bluetooth detectadas, los buckets para los cuales no se tengan registros Bluetooth tendran un numero 0 de redes Bluetooth detectadas y no deberán ser descartados del dataset.

En el caso de los sensores que sean usados y no sea posible interpretar los buckets para los cuales no se tengan registros disponibles como información útil, se los deberá descartar. Por lo tanto, en estos casos es importante analizar si es mejor no utilizar los datos de ese sensor en pos de tener más buckets disponibles o viceversa. Como la cantidad de buckets en todos los casos es similar, este decisión desembocará siempre en utilizar el sensor con buckets vacíos en lugar de descartar. Otro punto importante a aclarar es que el número total de casos resultantes no será el número de buckets del sensor con menos buckets disponibles con información, sino que será la cardinalidad de la unión de todos los buckets con información útil disponible para cada uno de los sensores utilizados. Hago esta aclaración ya que los buckets sin información de un sensor pueden no coincidir con los buckets sin información de otro sensor.

Para los TSI, se calculan:

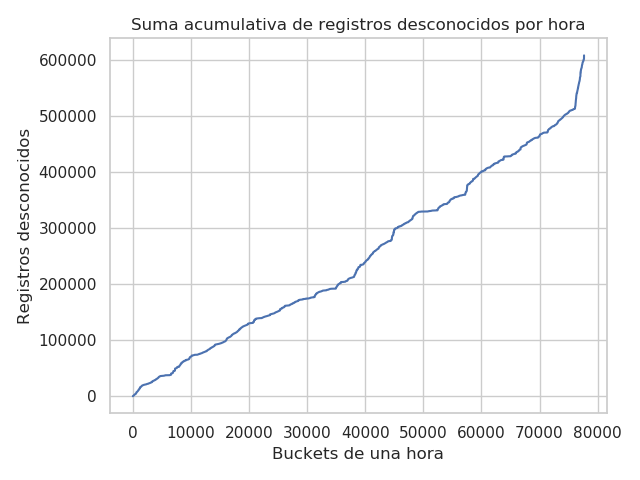
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Cantidad total de registros | Promedio y desviación estándar del tamaño de los intervalos | | Maximo y minimo del tamaño de los intervalos | | Porcentaje de buckets afectados | | Promedio y desviación estándar de la fracción afectada por bucket | |
| 1h | 30min | 1h | 30min |
| Dark | 7.269 |  | |  | |  |  |  |  |
| PhoneLock | 9.275 |  | |  | |  |  |  |  |
| PhoneCharge | 3.318 |  | |  | |  |  |  |  |
| Conversation | 79.023 |  | |  | |  |  |  |  |

## Registros desconocidos

Existen dos razones por las cuales es importante realizar un analisis especificamente sobre los datos de los datos del sensor Activity. Por un lado, este es el unico sensor que presenta registros clasificados como desconocidos, por lo que es importante llevar a cabo un analisis que permita estimar las consecuencias de la presencia de estos registros faltantes en el dataset StudentLife. Por otro, los datos de este sensor son los unicos utilizados para generar la variable objectivo, es decir, la variable a predecir. El nivel de sedentarismo del usuario en un bucket dado se calcula a partir de todos los registros de actividad fisica disponibles en ese bucket. Claro esta, los registros de actividad fisica clasificados como unknown no pueden ser utilizados ya que se desconoce cual es la actividad fisica que el usuario estaba llevando a cabo en ese momento.

En esta seccion se analizan los registros clasificados como unknown para entender su distribucion a lo largo del estudio StudentLife, asi como tambien como varia su distribucion para cada uno de los usuarios. Esto permitira evaluar si es necesario tomar medidas con respecto al manejo de estos registros desconocidos y que problemas puede causar.

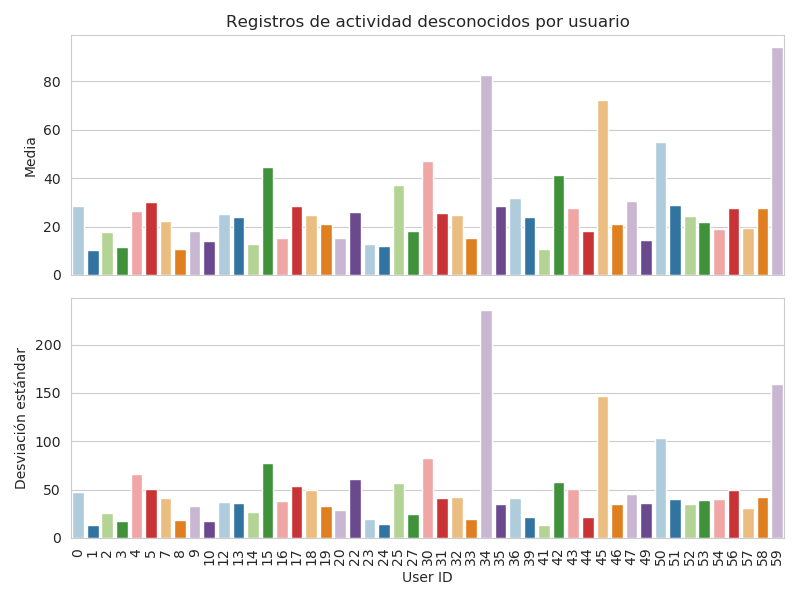
Los intervalos para los cuales no se tenía información sobre la actividad física son codificados por el numero 3. Sabiendo esto, resulta sencillo generar graficos que muestren claramente la distribucion de dichos registros. Cabe aclarar que todos los graficos que se muestran en esta seccion han sido generados a partir de los buckets de una 1 hora. Es facil deducir que no es necesario realizar el mismo analisis para ambos tipos de buckets ya que arrojarian resultados similares.

En la figura X se muestra la suma acumulativa de registros desconocidos para todas las horas disponibles. En dicha figura, el eje representa a cada bucket que es posible generar a partir de los datos del sensor Sensing, sin importar a que usuario pertenezcan. Mientras que el eje representa la suma cumulativa de registros clasificados como unknown, es decir que el valor de es la suma desde el primer bucket hasta el bucket . 

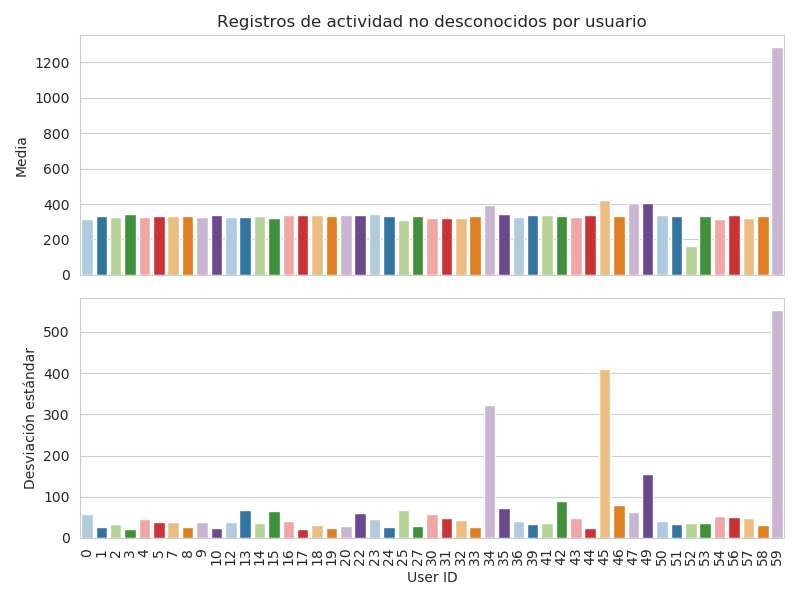
En la figura, puede observarse que la cantidad de registros crece de forma aproximadamente lineal con respecto a la cantidad de buckets a lo largo de todo el dataset. Esto es una buena señal, ya que significa que a priori los buckets tendrían una cantidad similar de registros a partir de los cuales generar las características de actividad y que los registros unknown no están agrupados en una sección específica del dataset. Esto quiere decir que, al momento de calcular el nivel de sedentarismo, cada bucket tendrá un soporte similar para calcular dicho nivel.

Al final del eje , se puede observar un pico en la cantidad acumulada de registros unknown. Estos registros pertenecen al último usuario, que presenta algunas diferencias a los demás usuarios, que podrán ser mejor visualizadas en el siguiente gráfico.

En la Figura Y, se muestran dos gráficos de barras. Estos gráficos de barras muestran la media y la desviación estándar de la cantidad de registros desconocidos para cada hora de un determinado usuario. Es decir, si para un usuario se disponen 1000 horas, se calcula la media y la desviación estándar de la cantidad de registros desconocidos disponibles para cada hora.



Varias observaciones pueden hacerse a partir de estas figuras. Con respecto a los valores faltantes, puede verse que la mayor cantidad de los usuarios presenta entre 20 y 60 registros desconocidos por hora, lo que explica el carácter lineal de la función graficada en la Figura X. Sin embargo, existen ciertas excepciones, como los usuarios 34, 45 y 59, donde se puede observar una media mayor de registros faltantes, asimismo como una desviación estándar mayor. El usuario 59 es el que mayor cantidad de registros desconocidos posee, lo que explica el pico observado al final de la figura X.

Por su parte, la figura Z muestra los mismo estadísticos, pero en lugar de ser estos generados a partir de los registros desconocidos, son generados a partir de los registros no desconocidos, es decir, a partir de los cuales se tiene información certera. En el primer gráfico, donde se muestra la media, puede verse que todos los usuarios presentan casi la misma cantidad de registros por bucket (aproximadamente 350). Sin embargo, los mismos usuarios que poseen muchos registros desconocidos (34, 54, 59) también presentan una alta desviación estándar en la cantidad de registros no desconocidos por hora, hechos que podrían estar relacionados. Es posible observar, también, que el usuario 59 posee aproximadamente el cuádruple de registros por buckets. Esta diferencia es extraña ya que no he hallado una referencia a ella en el informe del estudio StudentLife [CITAR PAPER] ni en la descripción del dataset[[5]](#footnote-4). Más adelante se discutirá la posibilidad de tratar a estos usuarios que presentan una representación diferente en el dataset como outliers (definir outliers). Al tratarlos como outliers puede evaluarse si al eliminarlos del dataset de entrenamiento y testeo podría llegar a conseguirse un mejor desempeño en los modelos impersonales.  


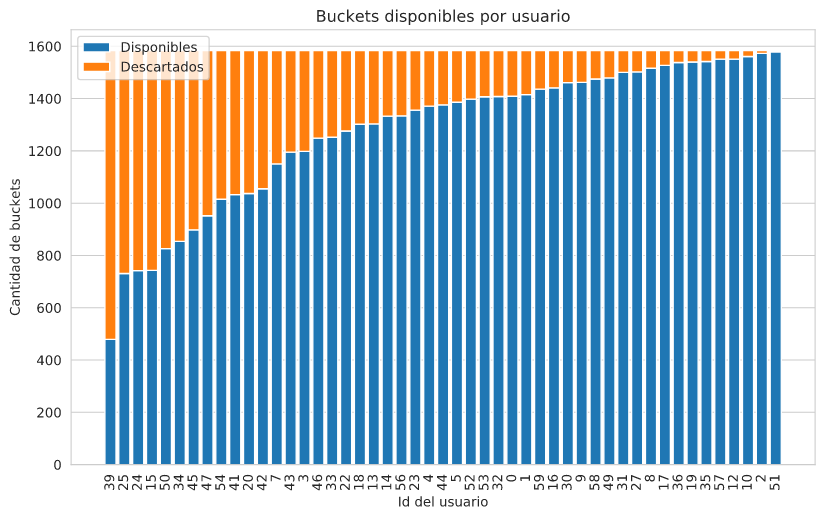
## Depuracion de buckets

Una vez que el dataset StudentLife ha sido discretizado, el siguiente paso es eliminar los buckets para los cuales no se posee informacion sobre alguna de las variables. Es preciso notar que solo se eliminan los buckets sin ningún registro que no pueden ser interpretadas por ninguna característica. Por ejemplo, para los buckets en los que no hubo registros de audio, puede interpretarse que hubo silencio, por lo que las características diseñadas pueden aprovechar esta interpretabilidad y tener un valor para esos buckets. En este caso, el valor para AudioMajor será 0 o “en silencio”.

En la figura X se muestra la cantidad de buckets resultantes disponibles para cada usuario y la cantidad de buckets descartados por la falta de registros para alguna variable, luego de depurar el dataset preprocesado. Como puede observarse, existe una gran diversidad con respecto a la cantidad de buckets para cada usuario. El usuario con menor cantidad de buckets disponibles es el 39, con 479 buckets, mientras que el usuario con más buckets es el 51, con 1579.

Como ya ha sido discutido en el marco teorico, la cantidad de casos de entrenamiento para un algoritmo de aprendizaje puede determinar que dicho algoritmo alcance un buen desempeño o no. Es necesario recordar en este momento que se comparan dos tipos de modelos de acuerdo a de donde provengan los datos de entrenamiento: los modelos personales y los inpersonales.

Con respecto la division anterior, es posible hipotetizar que para los usuarios con muy pocos buckets disponibles, como es el caso del usuairo 39, los modelos impersonales obtendran un mayor desempeño que los personales. La explicacion de la hipotesis anterior se da porque puede que la baja cantidad de buckets no llegue a darle la informacion al algoritmo de aprendizaje sobre las idiosincracias de estos usuarios en particular, que es la ventaja que presentan los modelos personales por sobre los impersonales.



## Nivel de MET

Los registros de actividad de cada usuario están clasificados en el dataset como estacionario, caminando o corriendo. A partir de estos registros se calcula el valor del gasto energético para cada intervalo. Tal como fue explicado anteriormente, el gasto energético es comúnmente medido de términos de Metabolic Equivalent of Tasks (MET), o Equivalente Metabólico de Tareas.

A cada tipo de actividad disponible en el dataset se le asigna un valor estático aproximado de MET de acuerdo al Compendio de Actividades Físicas [(Ainsworth et al. 2011)](https://paperpile.com/c/NW9nca/6z7v)

[[6]](#footnote-5). El compendio de Actividades Físicas posee 21 categorías, donde cada una de ellas está conformada por tipos de actividades. A su vez, cada actividad está acompañada por el valor de MET asociado a esa actividad. Para cada una de las actividades presentes en el Compendio existe evidencia publicada que soporta la asociación entre dicha actividad y su valor de MET.

Una vez analizado el compendio de actividades físicas,, se seleccionaron los tipos de actividades más similares a aquellas presentes en el dataset StudentLife. A las actividades registradas como estacionarias, se les otorgó un valor de MET de 1.3, el cual corresponde a actividades como “sentado en calma”, “acostado en calma”, “no haciendo nada”, “acostado despierto en la cama”, “escuchando musica”. A las actividades registradas como “caminando”, se les otorgó un valor de MET de 5.0, el cual corresponde a actividades como “cargando aproximadamente 6 kilos (por ejemplo, una mochila), en terreno llano o bajando escaleras”. A las actividades registradas como “corriendo” se les otorgó un valor de MET de 8.3, el cual corresponde a actividades como “corriendo, 5mph (12 min/milla”).

Una vez definida a qué valores de MET sera asociacada cada tipo de actividad, se transforma la información disponible sobre la actividad (variable categórica ordinal) en una variable numérica. Ahora bien, es el momento de definir como sera resumida la informacion de la actividad fisica - expresada en METs- para cada bucket, de tal forma que el resultado sea un solo numero. Es decir, como todos los datos disponibles son agrupados por buckets debe calcularse un valor de MET representativo para cada uno de ellos. Para ello, se define que el nivel de MET de un bucket está dado por la media del valor de MET de cada uno de los registros de actividad perteneciente a dicho bucket. Por lo tanto el nivel de MET de un bucket b está dado por la fórmula:

MET(b) = ,

Donde es el valor de MET de , el registro de actividad número y la cantidad de registros de actividad disponibles para el bucket. Por ejemplo, si un bucket contiene 100 registros de tipo “estacionario”, 10 registros de tipo “caminando” y 10 registros de tipo “corriendo”, el nivel de MET ese bucket se calcula como:

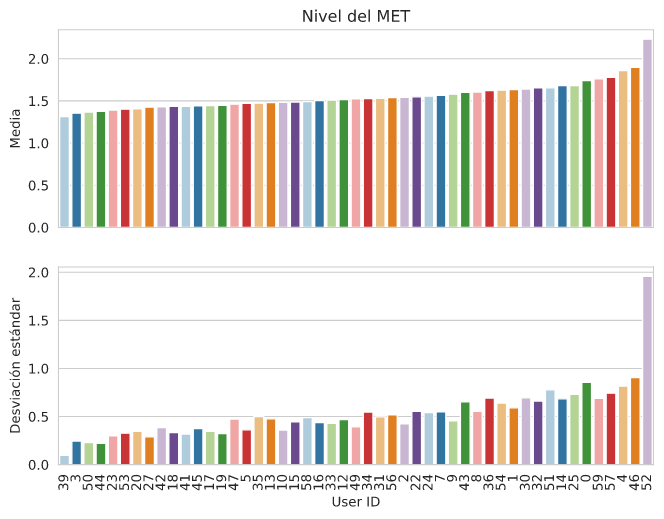
Una vez obtenido el nivel de MET puede ser necesario un paso más para obtener la variable objetivo. Esto es, en el caso de que el problema sea tomado como de clasificación. En ese caso, debe calcularse si cada hora como sedentaria o no. Como ya ha sido explicado con anterioridad, un comportamiento es considerado como sedentario si supera el valor de 1.5 METs. La anterior afirmación es adaptada en esta tesis de la siguiente forma: el comportamiento llevado a cabo en un bucket es considerado como sedentario si el nivel de MET para dicho bucket es superior a 1.5. De esta forma, todos los buckets cuyo nivel de MET sea superior a 1.5 serán clasificados como sedentarios, obteniendo así, finalmente, la variable objetivo en el caso de que la tarea sea de clasificación. En el ejemplo anterior, el bucket cuyo nivel de MET resultó ser de 2.2 es clasificado como no sedentario, ya que es mayor a 1.5.

Es necesario recordar en este punto que la baja granularidad en el tipo de actividad física especificadas en el dataset StudentLife representa una limitación, ya que el tipo de actividad del Compendio de Actividades Físicas asociadas a cada una de las actividades físicas especificadas en el dataset StudentLife puede no ser certero y distar de la verdadera actividad que el usuario estaba llevando a cabo al momento de ser registrada. Ademas, los valores de MET asociados a cada activiad en el Compendio no estan destinados a ser utilizados de manera individual, ya que segun se explica en la pagina del Compendio “Los valores en el Compendio no estiman el gasto de energia de la actividad fisica en una forma en la cual se tengan en cuenta diferencias en masa coporal, adiposidad, edad, sexo, eficiencia en el movimiento, condiciones geograficas y ambientales en las cuales las actividades son llevadas a cabo. Por lo tanto, las diferencias entre el gasto energético para la misma actividad puede ser grande y el gasto energético real para un individuo puede o no ser cercano al promedio establecido de MET presentado en el Compendio”.

## Análisis del nivel de MET

En esta sección se realiza un análisis de la actividad física, con el fin de comprender cómo se distribuye esta a lo largo de todo el dataset StudentLife. Este análisis permitirá conocer cuales son las actividades que más se llevan a cabo y en qué medida. Ademas, a traves de los gráficos podrán observarse los patrones de comportamiento de los diferentes usuarios.

La figura X muestra, de manera similar a las figuras mostradas en la sección anterior, dos gráficos de barras en los que se grafica el promedio y la desviación estándar del nivel de MET. El orden en el cual estan ordenados estos gráficos es de acuerdo al promedio del nivel MET. Ambos estadísticos son computados a partir del nivel de MET de todos los buckets disponibles para cada usuario. Como puede observarse, el promedio de MET de cada usuario ronda los 1.5, es decir, el nivel de MET que delimita lo que se considera un comportamiento sedentario o no.



Si se observa el gráfico de barras que muestra la desviación estándar, podemos ver que crece aproximadamente proporcionalmente a medida que crece el promedio de MET. Esto puede deberse a que estos usuarios presentan más buckets en los cuales llevaron a cabo actividades físicas. Por lo tanto, esto explicaría la diferencia en la desviación estándar entre los diferentes usuarios, donde los usuarios que practican menos actividad física son aquellos que tienen el menor valor para este estadístico.

También, puede interpretarse este crecimiento en la desviación estándar proporcional a la media del gasto energético con respecto cuan rutinario es cada usuario. Por lo tanto, estos gráficos parecen mostrar que cuanto más activo es un usuario, menos rutinario es. Esta hipótesis puede comprenderse mejor si se muestra cuán activo es un usuario a lo largos de las horas del día y los días de la semana. Para ello, se generaron mapas de calor con el promedio y la desviación estándar del nivel de MET para todos los usuario de acuerdo al día de la semana y las horas del dia. Para llevar a cabo esto, se agruparon los buckets por dia de la semana (eje y) y la hora del dia (eje x). Los mapas de calor que muestran el promedio del nivel de comportamiento sedentario ya habían sido usados en otros trabajos pero sin tener en cuenta el nivel de MET. En esos trabajos relacionados [(He and Agu 2016a)](https://paperpile.com/c/NW9nca/LNSK); [(He and Agu 2016b)](https://paperpile.com/c/NW9nca/ja8F); [(He and Agu 2016c)](https://paperpile.com/c/NW9nca/iT3M) los mapas de calor fueron generados a partir del porcentaje de actividades clasificadas como estacionarias en cada bucket de 1 hora. Los mapas de calor generados a partir de la desviación estándar de cada bucket de 1 hora no han sido aún analizados en otros trabajos relacionados. Además, opino que los dos tipos de mapas de calor se complementan entre sí porque, juntos, me permiten generar hipótesis acerca de las rutinas de los usuario en relación a su comportamiento sedentario y, así, entender la importancia de las características de tiempo al momento de generar los modelos predictivos.

En la siguiente Figura, se muestra un mapa de calor en el cual

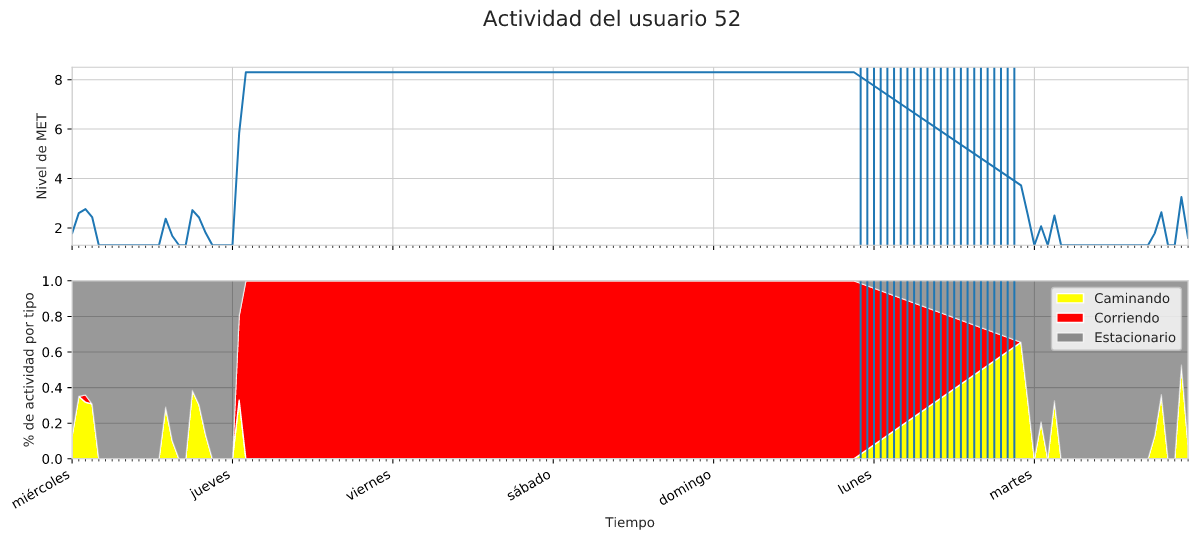
Fig. 1 y Fig. 2 muestran como ejemplo los mapas de calor del promedio y la desviación estándar del usuario 46. Puede observarse que las horas del día con el mayor comportamiento sedentario (eso es, niveles de MET menores a 1.5) se corresponden, también, con una desviación estándar baja. Esta misma observación puede ser realizada para la mayoría de los usuarios del estudio y es consistente con los que ya se analizó en la Fig. X. Una baja desviación estándar sugiere que el comportamiento sedentario observado en estos mapas de calor corresponde a un comportamiento de rutina, mientras que el comportamiento no sedentario no forma parte de la rutina de los usuarios debido a la alta desviación estándar.

Se realizó el cómputo la correlación de Pearson entre el medio y la desviación estándar para todos los usuarios y para todas las posibles combinaciones hora/dia de la semana. Luego, se calculó el promedio de la correlación y resultó ser 0.87, lo que sirve de soporte a la observación sobre que el comportamiento sedentario está correlacionado positivamente con el comportamiento rutinario. Dado que los usuario participaron que participaron en los experimento son estudiantes, esta correlación puede ser explicada por el hecho de que dos posibles comportamientos rutinarios que tienen un bajo nivel de gasto energético son las horas de sueño y los horarios de lo cursos. La información sobre las clases a las cuales los usuarios asistieron forman parte del dataset, por lo que se verificó que los horarios de cursada de los estudiantes se corresponden con los horarios de bajo gasto energético. En resumen, los usuarios tienden a ser más sedentarios en actividades rutinarias que en actividades no rutinarias. Esta observación deja lugar a la hipótesis de que el comportamiento sedentario es más predecible que el comportamiento no sedentario en términos de variables de tiempo.

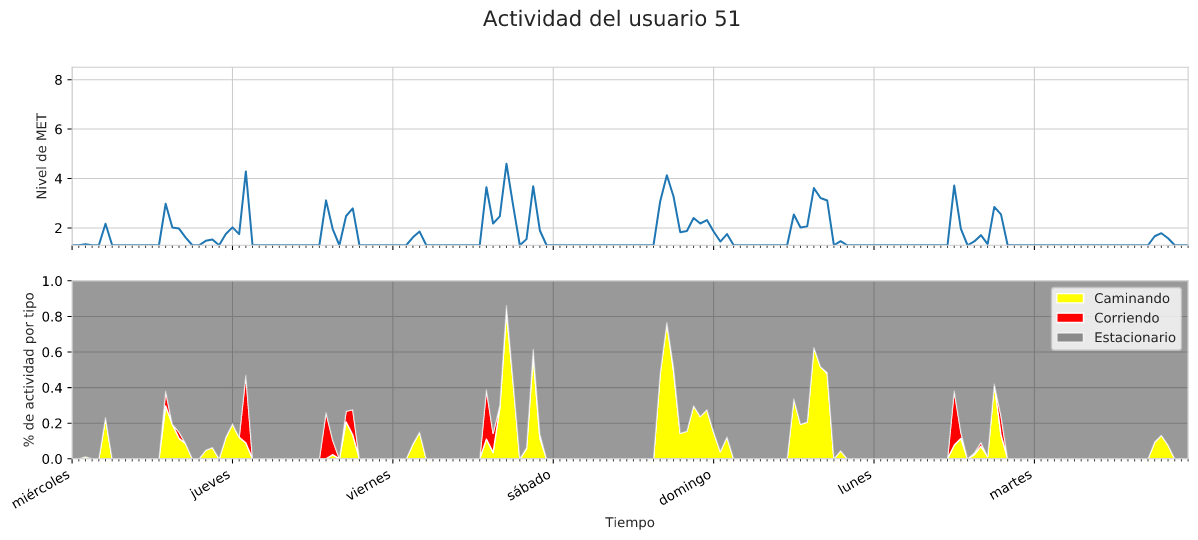
## Inconsistencias encontradas

Puede verse que el usuario 52 se diferencia de los demás en ambos gráficos, ya que posee un promedio de más de 2 METs y una desviación estándar mucho más elevada que la de los demás, cercana a 2 METs. Esto quiere decir que, en promedio, el gasto energético de cada hora de este usuario se aleja en 2 METS de la media. Por lo tanto, puede deducirse que este usuario tiene muchas horas en las que presenta un alto gasto energético. Si se analizan gráficos en los que se muestre que actividad llevó a cabo este usuario a lo largo del tiempo pueden apreciarse inconsistencias. Por ejemplo, en la figura X, se muestra el porcentaje de cada actividad (como suma acumulativa) en el periodo de 2013-05-22 al 2013-05-28. Como puede observarse que, según los datos disponibles para este usuario en ese periodo de tiempo, el usuario 52 estuvo corriendo por 4 dias, sin parar, lo cual es imposible y se deba posiblemente a un error en la aplicación de senseo Jigsaw. Además, en la figura, se muestran líneas verticales para cada hora en la que no se posee información de actividad del usuario. Como puede observarse, a continuación de la inconsistencia anterior no se dispone de ningún bucket de el dia lunes.

Las inconsistencias detalladas en el párrafo anterior se repiten en diferentes porciones de los datos disponibles del usuario 52. Por esta razón, se decidió eliminarlo y no utilizarlo para entrenar los modelos propuestos. Esta decisión se toma en base a que los algoritmos que utilicen los datos de este usuario para realizar predicciones están aprendiendo a partir de una distribución que no corresponde ni al del usuario 52, ni al de ningún otro usuario, por lo que empeora el desempeño de los modelos.



A modo de comparación, en la figura Y se muestra un ejemplo de la actividad llevada a cabo en una semana por el usuario 51. Puede observarse claramente que hay muchos espacios temporales donde el usuario se encuentra totalmente estacionario, lo que puede deberse tanto a horas de sueño como horas de ocio o asistencia a cursos. De hecho, en la figura se observa que las primeras horas del dia son, en general, completamente estacionarias, con la excepción del sábado donde el usuario pudo haber llevado a cabo alguna actividad social. Este tipo de patrón es el que está presente en la mayoría de los usuarios.



## Pipeline del preprocesamiento

## Seleccion de caracteristicas

En esta sección, se definen las características que fueron generadas a partir del dataset StudentLife. Cada una de las características fue seleccionada por alguna/s de la/s siguiente/s razón/es.:

* Características generadas en base a investigaciones previas [He and Agu 2016b; Cook and Krishnan 2015 Activity Learning Book cap 7)](https://paperpile.com/c/NW9nca/ja8F+Vm8X)
* Características generadas en base al análisis realizado en la sección anterior.
* Características que no incluyen información directamente ligada a la vida estudiantil, con la finalidad de que los resultados puedan ser más universales y no únicamente ligados a los estudiantes.

A su vez, se definen dos tipos de características: aquellas que fueron generadas a partir de los sensores de tipo TSD (Tipo de sensado discreto) y aquellas que fueron generadas a partir de sensores TSI (Tipo de sensado por intervalo). Estos dos tipos de características fueron llamadas Características Discretas y Características por Intervalo, respectivamente.

Las características son nombradas de acuerdo a la convención propuesta en PEP 8 (Python Enhancement Proposal)[[7]](#footnote-6).

## Características Discretas

### Características obtenidas a partir del GPS [Saeb 2016, cualquiera de los dos]:

* location\_variance: utilizado para medir la variabilidad en la ubicación GPS de un participante en cada bucket.
* location\_mean: similar a la característica anterior, pero con respecto al promedio.
* Speed\_mean: promedio de la velocidad instantánea, cómo se computa en Saeb 2016.
* Speed\_variance: varianza de la velocidad instantánea, cómo se computa en Saeb 2016.
* total\_distance: desplazamiento geográfico total.

Para el cálculo de las últimas 3 característica se tomaron un serie de decisiones importantes.

En todos los casos en los que es necesario calcular las diferencias entre dos registros consecutivos (por ejemplo, entre y ) , hay casos en que el registro anterior al actual no existe. Esto se da en los casos en que el registro es el primero de un usuario y cuando es el primero del bucket. En el caso en el que el registro es el primero del usuario, se tomó la decisión de eliminar el registro, ya que no hay forma de realizar los cálculos. Para el caso en el que el registro es el primero del bucket, se decidió encontrar una manera de no descartar el registro en cuestión, ya que es muy baja la cantidad de registros disponibles para el sensor GPS. La solución hallada que logra aprovechar estos registros es la de utilizar el último registro del bucket anterior, lo que por un lado puede parecer erróneo, ya que se esta mezclando la información de dos buckets diferentes, pero por otro lado estos dos registros (el primero del bucket y el último del bucket ) están estrechamente relacionados ya que se da en muchos casos en los que se aporta información complementaria al bucket en cuestión. Tomando esta última decisión, se descartan 49 registros (uno por usuario), en lugar de 59.223 (una por cada bucket disponible para el sensor de GPS, como se muestra en la table x).

Segundo, para el calculo de speed\_variance, cuando se calcula la varianza:

si el bucket solo contiene solo un registro y la velocidad espontánea para ese registro es 0 (posiblemente porque el usuario permanece siempre en el mismo sitio) se da que , que da como resultado indeterminado. La forma de expresar computacionalmente este valor es NaN (not a number, no es un número). Obviamente, estos valores no pueden ser almacenados en el dataset generado, por lo que se tomó la decisión de reemplazar estos valores nan por (cero).

Por último, como sucede con muchos de los sensores, hay bucket para los cuales no hay ningún registro disponible. Como ya ha sido discutido, en algunos casos es posible interpretar la ausencia de datos de manera que esos buckets no deben ser descartados. Segun la descripcion del motor de sensado Jigsaw [2010, J. Lu], con el objetivo de utilizar la menor porción de batería, al mismo tiempo que minimizando el error del GPS, este sensor puede permanecer inactivo hasta que se detecte que el usuario está en movimiento a partir del acelerómetro. Por lo tanto, puede interpretarse que para los buckets para los cuales no existen registros del sensor GPS, el usuario permanece en el mismo sitio que la hora anterior. De esta forma, las características generadas a partir del sensor GPS para los cuales no existen registros disponibles se completa de la siguiente manera:

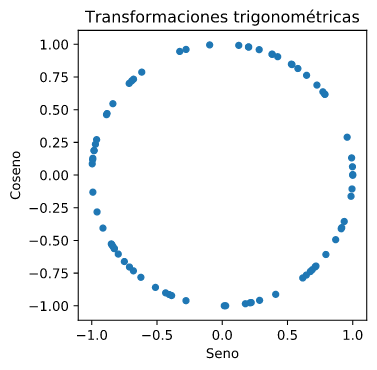
* location\_variance: se completa con 0;
* location\_mean: se completa con el valor de esta característica para la hora anterior;
* speed\_mean: se completa con 0;
* speed\_variance: se completa con 0;
* total\_distance: se completa con 0.

### Características basadas en el tiempo:

* second\_sine: transformación de seno de la cantidad de segundos pasados a partir del comienzo del dia.
* second\_cosine: transformación de coseno de la cantidad de segundos pasados a partir del comienzo del dia.
* weekday\_sin: transformación de seno del día de la semana;
* weekday\_cos: transformación de coseno del día de la semana;

Estas últimas 4 características tienen un claro objetivo, y es el de darle la posibilidad de comprender a los algoritmos de machine learning la natuleza ciclica de las variables relacionadas al tiempo. Sin realizar esta tranformacion, los modelos de machine learning no tienen la informacion necesaria para comprender que entre el primer valor posible y el ultimo existe la misma distancia que entre otros dos valores consecutivos. Es decir, en el caso de los dias de la semana, la diferencia entre martes y miercoles, es la misma que entre domingo y lunes. El problema subyace en que en informatica, los dias de la semana se expresan numericamente (y los numeros no son cicliclos), donde 0 representa al domingo y 6 al sabado (esto puede variar dependiendo de las implementaciones).

La transformacion llevada a cabo consiste en aplicar por un lado el seno, y por otro el coseno, a la variable ciclica estadanzarizada (con valores entre 0 y 1) multiplicada por 2 radianes (2). Es decir, primero se transforma a la variable categorica como un angulo entre 0 y 2, y luego se le aplican las dos funciones trigonometricas. De esta forma, danto observando el seno como el coseno, el valor del primer valor es similar al ultimo valor, ya que ambas funciones trigonometricas son periodicas. Aunque con una de las dos funciones alcanzaria para plasmar el caracter ciclico de las variables de tiempo, se da el fenomeno de que hay diferentes partes del dia que no se diferenciarian entre si. Por ejemplo, con respecto al tiempo pasado desde el comienzo del dia, si se utiliza el seno, las 20 horas tienen el mismo valor que las 0 horas. El mismo fenomeno se da para el coseno y las demas variables, como los dias de la semana. Por lo tanto, es importante usar ambas transformaaciones para completar la informacion y permintir a los modelos diferencias todos los momentos del dias, a su vez que permitir deducir que las variables son periodicas. En la Figura X se muestra un grafico de dos dimensiones, donde el eje X representa el valor de transformar la cantidad de segundos pasados desde el comienzo del dia a partir del coseno, mientras que el eje y muestra la misma transformacion pero a apartir del seno. Como puede observarse los puntos toman la forma del circulo o de un reloj, denotando asi su caracter periodico.



al combinar estas dos variables, se logra que se plasme en el dataset el carácter cíclico de estas variables de tiempo.

* past\_minutes: el número de minutos transcurridos desde el comienzo del día;
* remaining\_minutes: el número de minutos que quedan para terminar el día;

Como se mostró en el análisis del dataset, las características de tiempos pueden llegar a tener un papel muy importante a la hora de predecir el comportamiento sedentario de un usuario, debido a que, por lo general, estos siguen patrones temporales. Estos patrones ya fueron explorados en otros trabajos relacionados y es posible que la redes neuronales puedan sacar provecho de eso.

### Características de actividad física:

* stationary\_count: la cantidad de instancias de actividad física clasificadas como ‘estacionario/a’ en cada bucket;
* walking\_count: la cantidad de instancias de actividad física clasificadas como ‘caminando’ en cada bucket;
* running\_count: la cantidad de instancias de actividad física clasificadas como de ‘corriendo’ en cada bucket;
* total\_activity\_count: la cantidad de instancias en cada bucket;
* activity\_major: el tipo de actividad física con más instancias en cada intervalo;

### Características de audio:

* silence\_count: la cantidad de instancias de audio clasificadas como ‘silencio’ en cada intervalo;
* voice\_count: la cantidad de instancias de audio clasificadas como ‘voz’ en cada intervalo;
* noise\_count: la cantidad de instancias de audio clasificadas como ‘ruido’ en cada intervalo;
* total\_audio\_count: la cantidad de instancias de actividad en cada bucket.
* number\_of\_conversations: el número de conversaciones que ese estudiante tuvo en cada intervalo;

### Otras características

* wifi\_changes: el número de cambio de conexiones wifi en cada intervalo;

## Características por intervalo

* is\_charging: si el *smartphone* se estaba cargando;
* charging\_ratio: representa la proporción del *bucket* en la cual el *smartphone* se estaba cargando.

Para realizar este cálculo, se divide, iterativamente, cada intervalo que supere el tamaño de *bucket* en nuevos intervalos más pequeños, hasta que todos los intervalos tengan un tamaño menor o igual al tamaño de *bucket.* Una vez hecho esto, para cada intervalo se calcula la proporción que ocupa del *bucket* al que pertenece. Luego, se dividen todos los intervalos en grupos de acuerdo al *id* del usuario y al *bucket* al que pertenecen (por ejemplo: si los *buckets* son de una hora y el intervalo comienza en 14:34:20, se utiliza la función piso para obtener el *bucket* correspondiente, que da como resultado 14:00:00). Finalmente, se suman las proporciones de cada intervalo perteneciente a cada grupo en particular.

Al calcular esta categoría se descubrieron nuevas inconsistencias en los datos (sobretodo en el caso del usuario 49), ya que el proceso de realizar los cálculos existían *buckets* para los cuales la proporción en la cual el *smartphone* se estaba cargando era mayor a 1. Realizando un análisis se descubrió que había intervalos de tiempo superpuestos. Por ejemplo, para el usuario 49 existe un intervalo que va desde 00:34 hasta 08:53, mientras que otro intervalo va desde 00:13 hasta 03:22.

En la descripción del *dataset StudentLife* se afirma que todos los intervalos tienen una duración mayor a 1 hora. Si se analiza el caso de los *buckets* de 1 hora, al momento de la agrupación debería existir solo un intervalo por grupo. Sin embargo, se hallaron en total 355 grupos con más de un intervalo, lo que quiere decir que existen 355 *buckets* de una hora que presentan inconsistencias debido a la superposición de intervalos. La cantidad de *buckets* que superan la proporción de 1 son 45, por lo que la mayoría de *buckets* con inconsistencias no superan la proporción total aunque los valores calculados resultan ser igualmente erróneos.

No es posible decidir cuál de los intervalos superpuestos es el correcto o no, por lo que la única forma de solucionar este problema sería eliminando los intervalos inconsistentes. Esta accion implicaría eliminar los *buckets* involucrados, lo cual reduciría el tamaño del *dataset*. Por lo tanto, se decidió no descartar estos *buckets* inconsistentes, y en lugar de ello, reemplazar por 1 (ocupación total del *bucket*) aquellos *buckets* cuya proporción era superior a 1, y dejando igual los demás *buckets* inconsistentes.

Esta decisión puede no ser la mejor ya que realizando lo anterior no se soluciona el problema, pero creo que el problema sería aún mayor ya que si se eliminan los *buckets* inconsistentes se reduciría mucho el tamaño del *dataset*. Además, si el dataset es lo suficientemente grande, puede llegar a detectar el ruido que representan estas inconsistencias.

Para el caso de las demás características generadas a partir de sensores TSI se da la misma inconsistencia, en mayor o menor grado. El proceso que se utiliza es el mismo para todas ellas.

Para complementar esta característica de proporción se agrega también una característica booleana que indica si en un *bucket* determinado se halló uno o más intervalos.

* is\_locked: si el smartphone estaba bloqueado;
* locked\_ratio: representa la proporción del *bucket* en la cual el *smartphone* se estaba bloqueado.
* is\_in\_dark: si el smartphone estaba en la oscuridad;
* dark\_ratio: representa la proporción del *bucket* en la cual el *smartphone* se estaba en un ambiente oscuro (por ejemplo, en un bolsillo).
* Is\_in\_conversation: si el usuario mantuvo o no una conversasion.
* Conversation\_ratio: representa la porporcion del *bucket* en la cual el usuario se encontraba manateniendo una conversacion.
* Conversation\_number: cantidad de conversaciones diferentes manatenidas en el *bucket.*

### Variables objetivo

* sLevel: el valor de MET para el *bucket*.
* sClass: indica si en ese *bucket* el nivel de MET supera o no el valor de 1.5.

Estas variables son las posibles variables objetivos. Es decir, son las variables que los modelos de Machine Learning trataran de predecir a partir de todas las demás características. Las variables anteriores nunca están presentes ambas en el mismo dataset, sino que se calcula una o la otra dependiendo de si el tipo de modelo es de regresión (en cuyo caso se calcula solo *sLevel)* o clasificación (en cuyo caso se calcula solo *sClass).*

A pesar de ser variables objetivo, también son usadas como características. Es decir, como la tarea de predicción que se busca llevar a cabo en esta tesis es la de predecir el comportamiento sedentario futuro, el nivel de sedentarismo (sea *sLevel* o *sClass*) del momento actual y pasado es ya conocido, por lo que puede ser usado por los modelos para aprender.

## Tratamiento de variables categóricas

Como las redes neuronales no pueden recibir como datos de entrada variables de tipo categórica (sobretodo si estas no están representadas en forma numérica), es necesario utilizar un método para convertirlas o codificarlas en un tipo de variable que sea aceptable por el modelo. Además, no solo debe ser aceptable por el modelo, sino que tiene que representar de alguna manera la mayor información posible de tal manera que el modelo pueda utilizarla para realizar las predicciones. Una vez generadas las características, el dataset resultante posee dos características de tipo categóricas: day\_of\_week y activity\_major. Es importante notar que dichas variables son nominales y no ordinales, ya que no puede intuir claramente un orden entre ellas. Si estas variables fuesen nominales sería más fácil convertirlas, ya que compartían una característica con el resto de las variables numéricas, un orden entre ellas.

Hay muchos métodos que permiten codificar variables categóricas. Algunos de los más comunes son: Label Encoding, One-Hot Encoding, Dummy Encoding, Target Encoding, etc. La utilización de uno por sobre otro depende del dataset que se esté utilizando. A continuacion se describe brevemente cada uno de estos metodos.

* One-Hot Encoding: consiste en generar tantas nuevas variables como cantidad de categorías diferentes tenga la variable categórica en cuestión. Si la variable categorica tiene valores distintos, las nuevas variables serán vectores pertenecientes donde el componente , con , del vector tomará el valor de 1 si la variable categórica tiene el valor para el caso . Se recomienda no usar One-Hot Encoding para casos donde la variable categórica tenga muchas categorías diferentes (ejemplo, >20) ya que se incurre en un problema llamado *curse of dimensionallity*. Es claro observar que cuantas más categorías haya cada variable generada por este método aportará menos información.
* Dummy Encoding: Muy similar a One-Hot Encoding, con la diferencia de que se generan nuevas variables, en lugar de , nuevas variables en el proceso de codificación. La presencia de la última categoría se representa dejando en 0 el valor de todas las otras variables. Este método de codificación de variables categóricas tiene la ventaja de generar menos nuevas variables, aunque no es recomendable para métodos de aprendizaje basados en árboles de decisión, ya que estos no tienen forma de deducir la existencia de la última variable.
* Label Encoding: En este método, quizás el más simple, se reemplaza cada categoría por un valor numérico. Por ejemplo, cada categoría puede ser reemplazada por un número natural, o cualquier otro a elección. Este método de codificación es recomendable siempre y cuando la variable categórica sea ordinal. La ventaja de este método de codificación es que no aumenta el número de variables en el dataset,
* Target Encoding: En síntesis, este método consiste en reemplazar la ocurrencia de cada categoría por el valor promedio de la variable objetivo. Por ejemplo, si el tarea es de clasificación binaria, el valor por el cual se reemplace cada categoría de una variable categórica será un número real entre 0 y 1. Este método tiene varias implementaciones diferentes que buscan mejorarlo en términos de generalización, es decir, de reducir el *overfitting.* No es recomendable utilizar este metodo en el caso de que los datos de entrada de los modelos sean series de tiempo, ya que la la naturaleza de este tipo de datos tiende a cambiar mucho entre los datos de entrenamiento y de testeo.

Finalmente, se decidió aplicar Dummy Encoding sobre las 2 variables categóricas presentes en el dataset procesado. Con esta transformación, se añadieron 6 nuevas variables. Como resultado de esta transformación, se obtuvieron en total con 36 características. Esta es la última transformación que se les hace a cada *bucket* en particular. Los próximos pasos del comportamiento son de otra naturaleza y se basan en preparar al dataset hasta aquí generado para que pueda ser utilizado por los modelos predictivo.

## Generación de datasets con retrasos

El siguiente paso del procesamiento consiste en generar los time-lags. En esta etapa del procesamiento se prepara el dataset para que pueda ser utilizado para entrenar y testear los modelos. Dos parámetros intervienen en este paso: la cantidad de *lags* y la cantidad de períodos. Cuando se realicen los experimentos, diferentes valores para estos parámetros serán probados en pos de decidir cuál de ellos es mejor para cada tipo de modelo.

Para llevar a cabo esta etapa del procesamiento, se toma cada *bucket* como si representara el momento presente. A continuación, se prepara la lista de *buckets* que serán utilizados, a partir de su ubicación temporal relativa al *bucket* que se ha tomado como representativo del momento presente. Para preparar dicha lista, se tienen en cuenta la cantidad de *lags* y de periodos recibidos como parámetro. Esta lista va a definir de qué información se valdrán los modelos para predecir el nivel de MET del momento presente.

A continuacion, se concatenan los *buckets* seleccionados al *buckets* actual. Ademas, se eliminan todas las características del *bucket* actual salvo la variable objetivo. En este punto, se tiene una matriz de dimensiones .

## Separacion entre X e y

En este punto, ya es posible dividir al dataset en las características y la variable objetivo. Respetando la convención en el área de Machine Learning, se llama al conjunto resultante de características e al vector cuyos componentes representan el valor de la función objetivo para cada caso, es decir, el nivel de MET o la clase determina a partir de éste. A partir de este momento, se utilizarán indistintamente los termino *dataset* y para referirse al conjunto de características. Como se discute a continuación, las forma de representar al dataset puede variar según el modelo que utilizara dicho *dataset.*

## Sepearacion entre entrenamiento y testeo

Splits basados en series de tiempo

## Dimensiones de X

La forma de representar el dataset que resulta de esta etapa depende del modelo que lo usará para entrenarse. Esto es, algunos modelos esperan matrices mientras que otros esperan tensores. Un tensor es la extensión del concepto de matriz a un espacio de dimensiones. Por diversas características, esta representación de un alto número de dimensiones puede beneficiar a algunos tipos de modelos, ya que estos pueden aprovechar mejor la estructura espacial de los datos de entrada.

En el caso de las redes Perceptron Multicapa, se espera como entrada una matriz, donde las filas representan los casos y las columnas las características. De esta forma, este tipo de red espera una matriz de dimensión . Como puede observarse, la cantidad de características crece proporcionalmente a la cantidad de *lags.*

En el caso de los modelos que esperan tensores, serán, en todos los casos de espacios vectoriales de 3 dimensiones. Estas 3 dimensiones representan: los casos, las características y los *lags* de dichas características. Por lo tanto las dimensiones de estos tensores serán de . Con esta organización de los datos, los modelos pueden sacar provecho de la característica temporal de dichos datos. Esto es así porque gracias a esta disposición los datos de una misma característica son contiguos entre dos *lags* contiguos, hecho que no se da en la representación de matrices. De esta forma, es más fácil para los modelos - aquellos que puedan utilizar datos organizados de acuerdo a esta disposición- aprender características relacionadas a la dimensión temporal. Por ejemplo, la relación entre las características relacionadas a la actividad en dos o más *lags* consecutivos.

## Normalización de los datasets generados

El último paso del pre-procesamiento consistió en normalizar los datos. Para este paso se utilizaron normalizadores provistos por la librería Scikit-learn5. El proceso de normalización se basa en que el normalizador aprenda la transformación a realizar a partir de los datos de entrenamiento y, luego, con la transformación aprendida, normalizar los *datasets* de entrenamiento y testeo. Para evitar la fuga de datos (*data leakage*) es que se aprende la transformación de los datos sólo a partir del conjunto de entrenamiento. Lo que aprende el normalizador depende de que tipo sea. En el caso de *StandardScaler[[8]](#footnote-7)* el normalizador aprende la media y la desviación estándar de cada característica, mientras que en el caso de *MinMaxScaler[[9]](#footnote-8)* el normalizador aprender el máximo y el mínimo de cada columna. El fenómeno *data-leakage* debe ser siempre tenido en cuenta para evaluar la capacidad de generalización de los predictores.

## Modelos propuestos

## Validación de los modelos

Como ya fue mostrado en el marco teórico, existen muchos métodos utilizados para validar modelos de Machine Learning. El objetivo de la validación es verificar el desempeño del modelo. Para validar los modelos debe discutirse cuál de los métodos existentes es el más adecuado para ser utilizado en el problema que motiva esta tesis. Como los datos que serán utilizados para entrenar y testear los modelos propuestos son de tipo *timeseries,* no todos los metodos de validacion son aplicables, ya que deben cumplir con ciertas características.

En la gran mayoría de dichos métodos se debe dar por supuesto que las variables del *dataset* son independientes e idénticamente distribuidas, lo cual no puede darse por supuesto en el caso de *time-series,* ya que claramente no pueden intercambiarse casos entre sí sin modificar la distribución del *dataset.*

Además, el método de validación a utilizar debe cumplir con la característica de adaptarse a la forma real en la cual los datos son registrados y puestos a disposición de los modelos. Es decir, todos los datos que use un modelo para entrenarse deben ser datos registrados en el pasado.

Por último, el método a utilizar debe adaptarse para los diferentes experimentos propuestos en esta tesis. Sobre todo, debe ser flexible a la categorización entre modelos personales y no personales. Esto es necesario ya que los datos utilizados por los modelos para ser entrenados y testeados pueden o no pertenecer al mismo dataset.

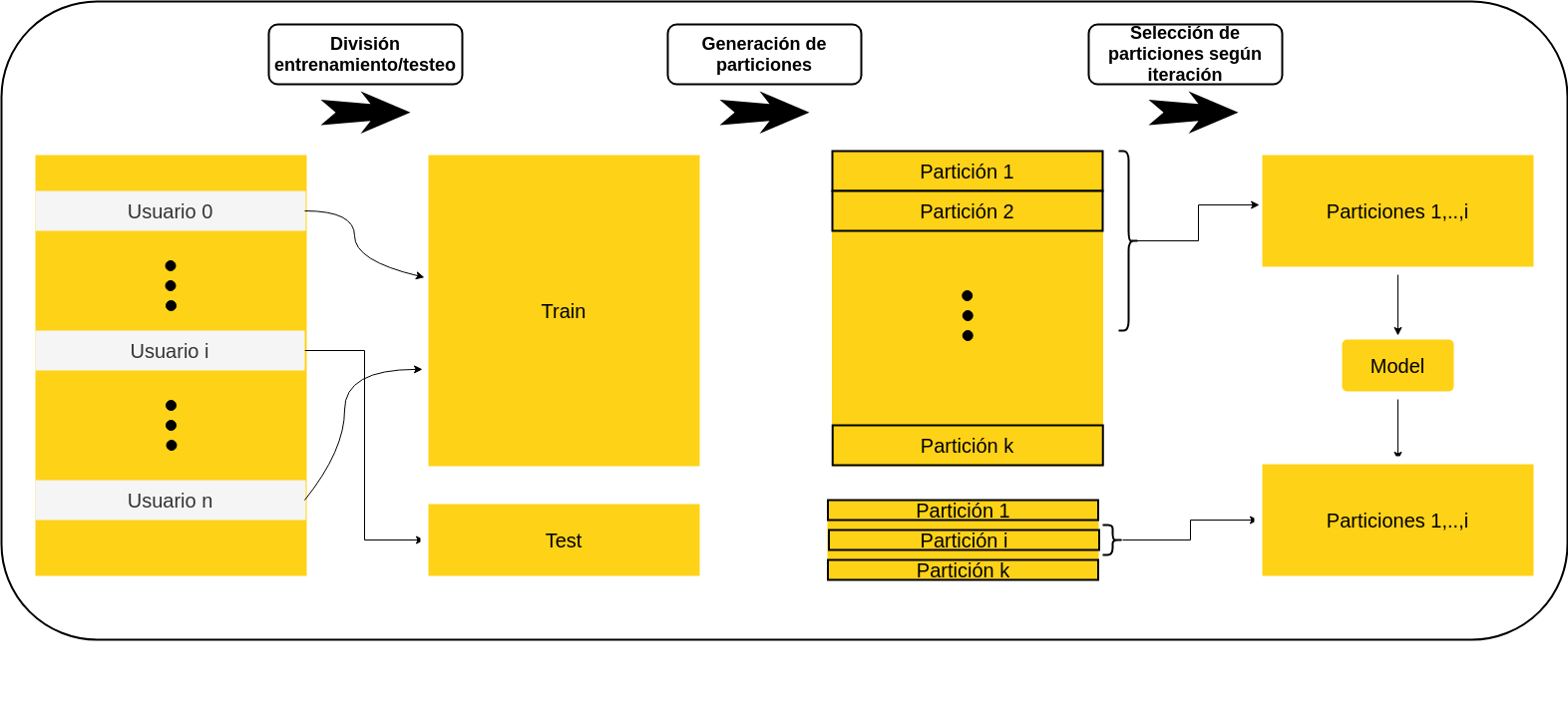
De acuerdo a las limitaciones anteriores, se propuso un método de validación que es especialmente aplicable a *datasets* de *timeseries*, basado en aquel disponible en la libreria Scikit-Learn, llamado *TimeSeriesSplit*[[10]](#footnote-9)*. TimeSeriesSplit* consiste en dividir el *dataset* en *k* particiones, respetando el orden temporal, donde cada partición tiene la misma cantidad de casos. Luego, se llevan a cabo iteraciones en las cuales se re-entrena el mismo algoritmo (pero no el mismo modelo), con la particularidad de que en cada iteración el conjunto de casos de entrenamientos es un superconjunto del conjunto de casos de la iteración anterior. Más precisamente, el conjunto de casos de entrenamiento de la iteración es igual a la unión entre los conjuntos de entrenamiento y testeo de la iteración En la iteración se utilizan las primeras particiones para entrenar al modelo y la partición para testearlo. El resultado final de este método de validación es la media del desempeño alcanzado por el modelo sobre el *dataset* de testeo en cada iteración. En algunos casos, la desviación estándar puede ser un estadístico útil.

El método descrito en el párrafo anterior no puede ser utilizado en esta tesis ya que da por supuesto que los casos son observaciones en intervalos fijos de tiempo, lo cual no es aplicable a los *datasets* obtenidos a partir del *pipeline* de preprocesamiento, donde en dos intervalos de tiempo de igual longitud puede contener un cantidad diferente de casos. Además, el método *TimeSeriesSplit* no permite recibir dos *datasets (*uno de entrenamiento y otro de testeo) a partir de los cuales se podrían modelar los modelos personales e impersonales.

Por lo tanto, se propone un nuevo método basado en *TimeSeriesSplit* que resuelve los problemas por los cuales este último no es aplicable en el problema de la predicción del comportamiento sedentario futuro utilizando el *dataset StudentLife*. El método de evaluación propuesto difiere en lo siguiente de método *TimeSeriesSplit:*

* El criterio para generar las particiones es que estas deben representar el mismo intervalo de tiempo y no la misma cantidad de casos. Para esto, se calculan las fechas mínimas y máximas del *dataset.* Luego, se calcula la longitud del intervalo temporal comprendido entre las dos fechas calculadas (por ejemplo 65 días y 4 horas) y se lo divide por , obteniendo así la longitud de tiempo que aborda cada partición. Una vez obtenidas estas particiones, se prosigue de la misma manera que en *TimesSeriesSplit.*
* Para permitir que el método se adapte a los modelos personales e impersonales, éste recibe como parámetro dos *datasets* diferentes, uno para el entrenamiento y otro de testeo. De esta forma, en el caso de los modelos personales, los *datasets* de entrenamiento y de testeo pertenecen al mismo usuario, mientras que en el caso de los modelos impersonales el *dataset* de entrenamiento contiene los datos de todos los usuarios salvo el del usuario objetivo mientras que el *dataset* de testeo contiene los datos del usuario objetivo. Por último, en el caso de los modelos híbridos el *dataset* de entrenamiento representa el dataset entero (conteniendo los datos de todos los usuarios, incluidos los del usuario objetivo), mientras que el *dataset* de testeo contiene los datos del usuario objetivo. Debe prestarse atención, ya que aunque en los modelos personales e híbridos los datasets de entrenamiento y testeo comparten datos (es decir, su intersección es diferente del conjunto vacío), al momento de entrenar el modelo, los conjuntos de entrenamiento y testeo si son disjuntos, ya que pertenecen a dos particiones temporales diferentes.

En la figura X se muestra un ejemplo en el cual se puede observar la diferencia en cuanto al funcionamiento de este método para modelos personales e impersonales. Como puede observarse, bla bla bla.



## Tunning de los modelos

Como ya ha sido explicado, a la hora de entrenar un modelos de Deep Learning existen diferentes hiper-parametros que deben ser ajustados para lograr que el modelo alcance el mayor desempeño posible. Para esto, se deben probar diferentes valores para cada hiper-parámetro, comparando el desempeño obtenido entrenando el modelo con cada uno de ellos. Existen diferentes formas de realizar la prueba de los hiper-parametros. Para realizar el *tunning* o “busqueda de los hiper-parametros óptimos” en los modelos que serán comparados en la tarea de la predicción de comportamiento sedentario futuro a partir del *dataset StudentLife* se utilizo la Optimizacion de Bayes [Agnihotri & Batra, "Exploring Bayesian Optimization", Distill, 2020.].

## Optimizacion de Bayes

La optimización de Bayes es utilizada para buscar el máximo global en funciones analizadas como “cajas negras”, es decir aquellas que no tienen una forma especial y en las que no es posible utilizar la optimización por el gradiente. Los Redes Neuronales Profundas son un caso claro de este tipo de funciones, en las que por muchos años la mejor forma de tunear un modelo de Deep Learning fue el de utilizar una búsqueda aleatoria o simplemente evaluar todos los hiper-parametros posibles. El problema con este enfoque es que si el tiempo que se tarda en entrenar un modelo es grande es inviable realizar una búsqueda exhaustiva de los mejores hiper-parametros.

La Optimización de Bayes puede solucionar este problema decidiendo de forma inteligente qué valores de los hiperparametros evaluar en la siguiente iteración, encontrando un punto medio entre los siguientes dos objetivos:

* Obtener información sobre la función con el fin de tener más información a la hora de realizar las siguientes evaluaciones. Esto se logra evaluando en los puntos para los cuales la incertidumbre es mayor (en el caso de usar un Proceso Gausiano como modelo sustituto, donde la varianza sea la mayor).
* Encontrar el máximo de la función basándose en la información ya conocida por el modelo sustituto.

.

Mas formalmente, la Optimización de Bayes consiste en hallar correspondiente al maximo o minimo global de una función . La Optimización de Bayes tiene las siguientes restricciones:

* El conjunto factible de es simple.
* es continuo pero no es fácil de optimizar.
* La evaluación de no da información sobre el gradiente.
* es costosa de evaluar.
* es ruidosa.

A partir de estas restricciones, la Optimización de Bayes sigue los siguientes pasos:

1. Se elige un modelo sustituto para modelar la verdadera función y definir sus probabilidades a priori.
2. A partir de las evaluaciones de , se utiliza la regla de Bayes (de ahí el nombre del proceso de optimización) para obtener probabilidades a posteriori.
3. Se utiliza una función de adquisición , que es una función de la probabilidad a posteriori, para decidir cuál será la evaluación de . se selecciona de tal forma que .
4. Añadir la nueva observación al conjunto de observaciones e ir al paso 2. Repetir hasta que haya convergencia o hasta que se acabe el presupuesto.

En el caso donde la función representa un modelo de Deep Learning, cada evaluación se basa en obtener el desempeño del modelo para un conjunto de hiper-parametros determinado, y en base al valor obtenido, actualizar el modelo sustituto para luego seleccionar otro valor para los hiper-parametros basándose en la función de adquisición.

En esta tesis, se utiliza un Proceso Gaussiano (Görtler, et al., "A Visual Exploration of Gaussian Processes", Distill, 2019.) como modelo sustituto. Además, se decidió utilizar la función de adquisición llamada Probabilidad de Mejora. Se utiliza la implementacion de la Optimización de Bayes proveída por la librería *Scikit-Optimize*[[11]](#footnote-10). Esta librería es un proyecto paralelo al de la librería *Scikit-Learn* especializada en la optimización de modelos secuenciales.

## Selección de datasets modelo

Lo ideal a la hora de realizar los experimentos sería realizar el proceso de *tunning* para cada *dataset* diferente, lo cual resulta inviable debido a la alta cantidad de *datasets* que serán evaluados. Por esta razón, se decidió tomar ciertos aspectos de los experimentos como representantes de los demás. Las decisiones que se tomaron son las siguientes:

* Todos los *datasets* utilizados en el proceso de tunning comparten las siguientes características: granularidad de 1 hora, 4 *lags*, periodos de valor 1.
* Se explora exhaustivamente tanto modelos personales como impersonales, debido a que son la categoría que mayor diferencia implica en los *datasets*.
* Se realiza el proceso de ajuste de hiperparametros de acuerdo a cada tipo de arquitectura propuesta. Esto es: MLP, CNN, TCN y RNN.

Por lo tanto, la cantidad de arquitecturas totales resultantes del proceso de tunning será de

## Selección de usuarios modelo

utilizar ciertos usuarios como representantes de grupos para los cuales se realiza el proceso de tunning. Se toman 2 usuarios como representantes de los demás usuarios. Para decidir cuáles serán los usuarios representantes se utiliza el método de Machine Learning no supervisado K-Means, que se encarga de encontrar centroides a partir de ciertas características computadas de cada usuario. Dichas características son: promedio de MET, desviación estándar de MET, cantidad de buckets de 1 hora disponibles.

Si bien el dataset StudentLife cuenta con gran cantidad de usuarios, para los objetivos de nuestro estudio se analizó el comportamiento de tres de ellos. La razón de esto, es que cada usuario posee su propia rutina y características personales, que no tienen porque parecerse a la de los demás usuarios. Por otro lado, algunos usuarios poseen menos datos que otros.

Los usuarios fueron seleccionados de forma tal de representar patrones de gasto energético diferentes. Los usuarios seleccionados fueron: usuario 50 (bajo MET), usuario 31 (MET medio), y usuario 4 (MET alto). Los usuarios seleccionados poseen, además, un diferente grado rutinario en su actividad diaria. Lo anterior puede corroborarse con el cálculo de la correlación entre los gastos energéticos de las mismas horas de los diferentes días de la semana y su desviación estándar. En Fig. 1 y Fig. 2 se muestran el MET promedio y la desviación estándar, respectivamente, a lo largo de los días de la semana y las horas del día de los 3 usuarios seleccionados.

## Selección de hiper parámetros para los modelos

## Bayes Optimization

## Modelos de regresion

## Validación de los modelos

## Modelos de regresion

## Metricas

La métrica utilizada para evaluar el desempeño de las redes diseñadas es el Error Cuadrático Medio o MSE (Mean Squared Error). El MSE es una medida de desempeño cuantitativa utilizada comúnmente para medir el error que hay entre el valor real y el valor estimado. En este contexto MSE es igual a la sumatoria de los errores cuadráticos. En comparación con el Error Medio Absoluto o MAE, MSE amplifica y penaliza con mayor fuerza aquellos errores de mayor magnitud. La fórmula de cálculo del MSE se muestra a continuación:

Donde:

* es el resultado en el tiempo t.
* es el pronóstico de valor en el tiempo t.

diferencias MAE y MSE

-MAE es robusto frente outliers

-MSE penaliza a los errores mas grandes

-el MAE es linealmente dependiente de la diferencia entre el valor predicto y el valor real

-en MSE el error crece cuadráticamente.

-MAE expresa el error en una unidad del tipo de la variable a predicta

-MSE no lo hace, RMSE si.

-MSE y RMSE le da mayor peso a los errores mas grandes.

MSE o RMSE deben ser usados cuando los errores grandes son particularmente indeseables.

Creo que en nuestro problema fallar en los outliers no lastima tanto, pq aca lo importante es saber cuando es mayor que 1.5 el MET, no cambia tanto si es 2 o 8.

En otras palabras, compara un valor predicho y un valor observado o conocido.

## Modelos de clasificación

## Metricas

Para los modelos de clasificacion decidio utilizarse la metrica AUC para evaluar y comparar el desempeño de cada modelo

## Diseño de los experimentos

A partir de los modelos propuestos se diseñaron diferentes tipos de experimentos. Los experimentos pueden ser clasificados de acuerdo a, por un lado, el modelo utilizado en el, y, por otro, a los datos usados para entrenarlo y testearlo. En la primera categoria se posee a los modelos ya propuestos junto con sus respectivos hiper-parametros hayados ene la etapa de tuning. Por su parte, la segunda categoria, puede ser dividida, a su vez, en diversas sub-categorias. Para cada sub-categoría, se propone un conjunto de valores posibles para los cuales realizar la experimentacion (ver tabla X).

* Clasificación de los experimentos
  + Por el algoritmo de aprendizaje:
    - algos propuestos…
  + Por el tipo de modelo:
    - Personal:
    - Impersonal
  + Por el tipo de tarea a realizar:
    - Regresion
    - Clasificacion
  + Por el dataset:
    - Cantidad de lags: define la cantidad de buckets a las cuales el algorimo tiene acceso para cada caso.
    - Períodos: define el espacio temporal entre cada bucket. La cantidad de buckets que hay entre dos buckets segun un perido es igual a . Por ejemplo, si el periodo es 2 se seleccionan
    - Inclusión o no de horas de sueño: define si se incluyen o no los buckets en los que se estima que el usuario está durmiendo.
    - Granularidad del dataset: define el tamaño temporal de los buckets.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Categoria | Algoritmo de aprendizaje | Tipo de modelo | Tarea a realizar | Caracteristicas del dataset |
| Posibles valores | Mlp, rnn, cnn, tcn | Personal o Impersonal | Regresion | Ver tabla X |

|  |  |  |  |
| --- | --- | --- | --- |
| Categoria | Cantidad de lags | Cantidad de períodos | Granularidad del dataset |
| Posibles valores | 1,2,4,8 | 1,2,4 | 30 minutos, 1 hora, 2 horas |

Una vez descritas las diferentes formas de clasificar los datasets generados, puede pasarse a calcular la cantidad de modelos a entrenar. La cantidad de modelos a entrenar queda definida por la multiplicación de la cardinalidad de los conjuntos de posibles valores de cada categoria (o sub-categoria, si existe) en las cuales pueden clasificarse los experimentos. De esta forma, la cantidad de experimento a realizar es:

## Resultados

La Tabla 2 muestra los valores del Error Absoluto Medio (MAE) conseguidos para cada arquitectura propuesta. En todos los casos, se utilizaron ⅔ de los datos (aproximadamente las primeras 6 semanas) para el entrenamiento de los modelos y ⅓ (aproximadamente las últimas dos semanas) para testeo.

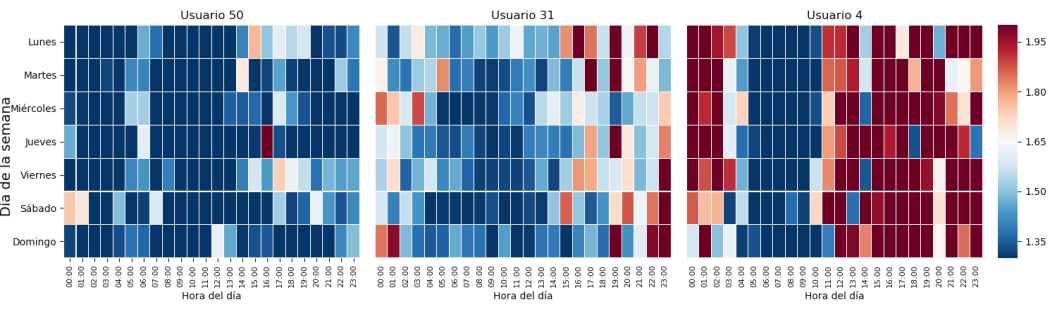


Fig. 1. MET promedio de los tres casos de estudio a lo largo de los días de la semana

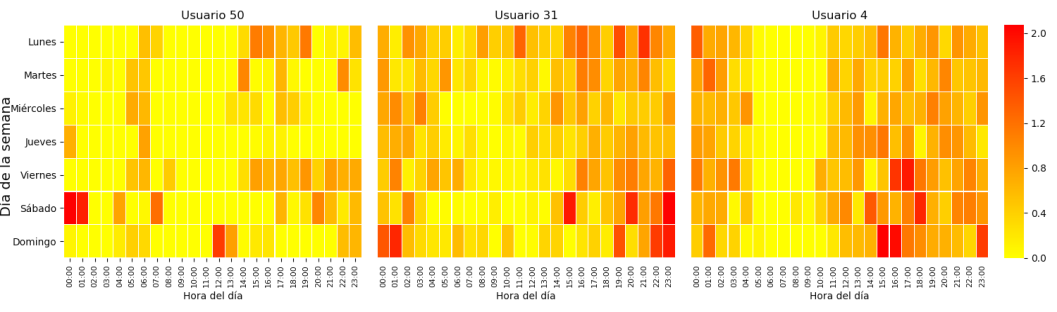


Fig. 1. Desviación estándar del MET de los tres casos de estudio a lo largo de los días de la semana

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| MAE Test | Lags | Arq. 1 | Arq. 2 | Arq. 3 | Arq. 4 |
| Usuario 50 | 2 | 0.102 | 0.097 | 0.086 | 0.109 |
| 4 | 0.094 | 0.097 | 0.113 | 0.108 |
| 8 | 0.140 | 0.090 | 0.123 | 0.145 |
| Usuario 31 | 2 | 0.306 | 0.325 | 0.282 | 0.285 |
| 4 | 0.313 | 0.343 | 0.296 | 0.318 |
| 8 | 0.359 | 0.331 | 0.334 | 0.339 |
| Usuario 4 | 2 | 0.528 | 0.537 | 0.479 | 0.529 |
| 4 | 0.608 | 0.511 | 0.496 | 0.550 |
| 8 | 0.682 | 0.530 | 0.546 | 0.573 |

Table 1. Valores de MAE conseguidos para cada arquitectura propuesta

## Discusión

El usuario que presentó mayor desafío para los predictores resultó ser el número 50, ya que fue el que contaba con más casos de comportamiento sedentario. Por lo tanto, resultó difícil que las arquitecturas propuestas aprendieran, a partir de las características del pasado, cuándo iba a tener un comportamiento no sedentario en el futuro. El modelo correspondiente a la arquitectura 3, con 2 lags, fue el que mejor desempeño presentó, con un MAE de 0.086. La Fig. 3 muestra las predicciones de prueba para esta arquitectura.

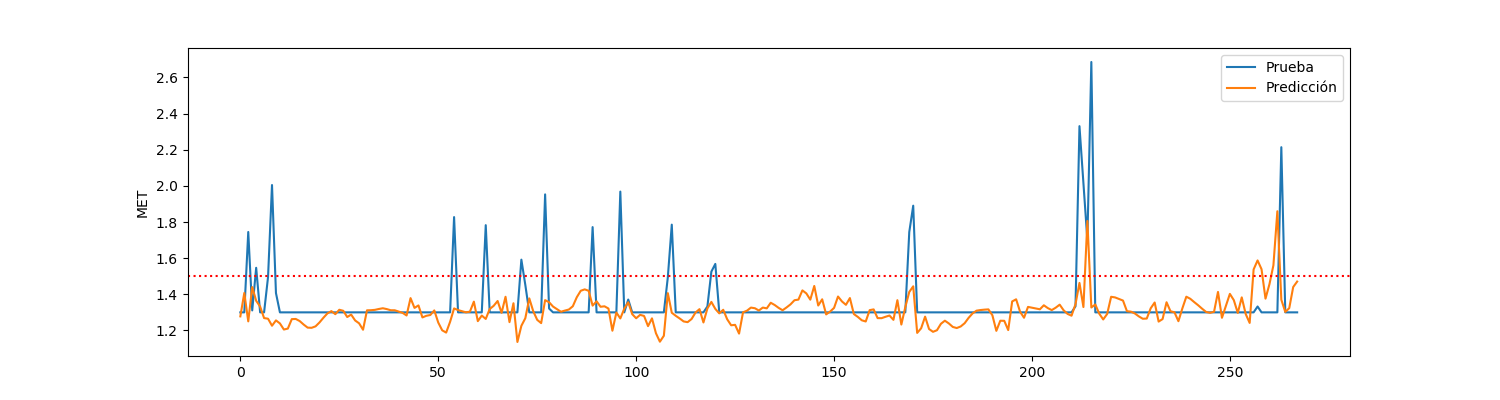


Fig. 1. Predicciones con datos de prueba del usuario 50 con la arquitectura 3 y 2 lags

En el caso del usuario 31, se puede observar un comportamiento más rutinario. Teniendo en cuenta que en el pre-procesamiento se le dio especial atención a las características de tiempo, es entendible que los regresores se desempeñen mejor en estos casos. La arquitectura 3, con 2 lags, fue la que mejor performance tuvo en este caso, con un MAE de prueba de 0.282. La Fig. 4 muestra las predicciones de prueba para esta arquitectura.

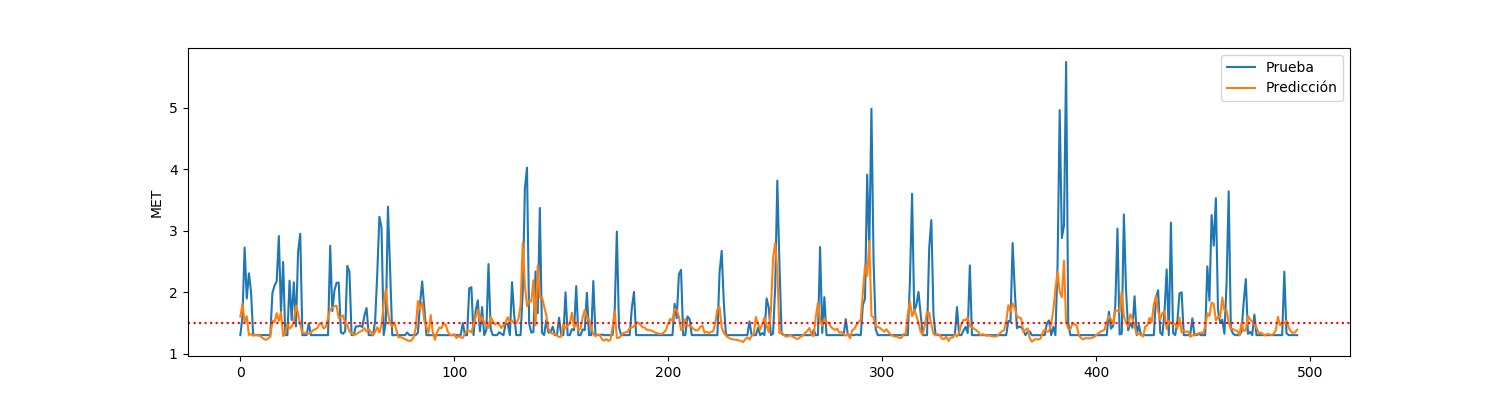


Fig. 1. Predicciones con datos de prueba del usuario 31 con la arquitectura 3

Por último, los modelos sobre el usuario 4 fueron los que mejor desempeño tuvieron, a pesar de tener un mayor MAE que los modelos sobre los otros usuarios. Esto se debe a que este usuario posee muchos más casos de comportamiento no sedentario, por lo que para que el regresor se desempeñe bien debe aprender los patrones de comportamiento del usuario a partir de las características provistas. En el caso del usuario 50, el predictor podría solo aprender a predecir un valor cercano a 1.3 METs para obtener un bajo MAE. La arquitectura que mejor se desempeñó en este caso fue, nuevamente, la número 3, con 2 lags, con un MAE de 0.479. La Fig. 5 muestra las predicciones de entrenamiento y testeo para esta arquitectura.

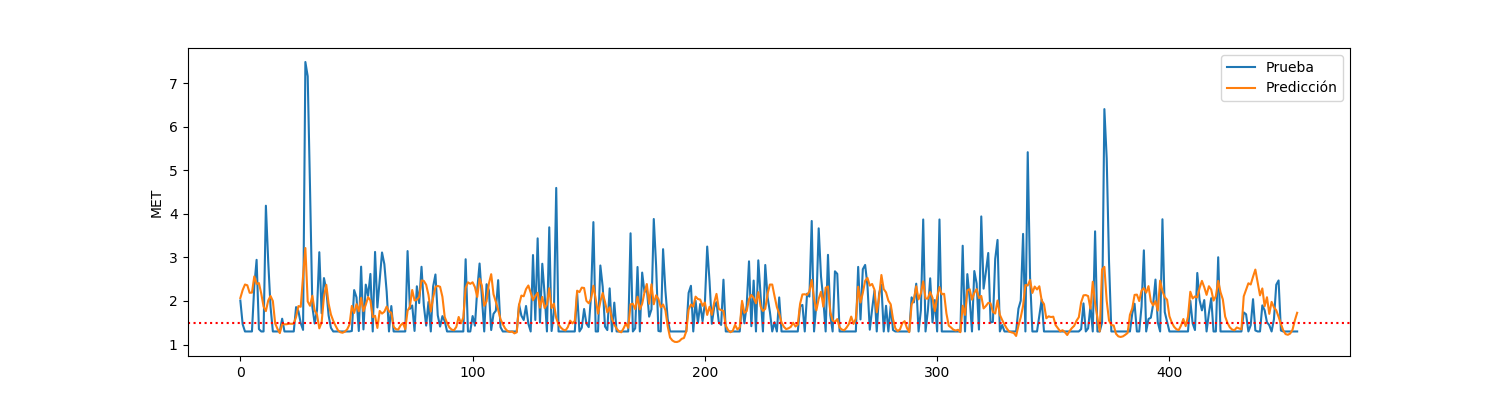


Fig. 1. Predicciones con datos de prueba del usuario 31 con la arquitectura 4

El hecho de que, para todos los modelos, las CNN tuvieron una mejor performance que las RNN, confirma nuestra hipótesis de que las CNN son más adecuadas para problemas de predicción que deban aprender de una secuencia de datos.

## Conclusiones

En este trabajo nos propusimos comparar diferentes tipos de redes neuronales para evaluar su performance en el problema de la predicción del comportamiento sedentario. Este trabajo es el primero en evaluar el rendimiento de redes neuronales que intenten sacar provecho al carácter secuencial del problema tratado. Se presentaron usuarios con diferentes niveles de gasto energético, lo que hace que muy difícil diseñar una arquitectura que tenga una buena performance para todos ellos.

A pesar de haber utilizado un alto grado de regularización, las NN más extensas son propensas a producir overfitting. En consecuencia, tendrán un mal rendimiento en la etapa de testing. Este es un trade-off que fue considerado fue el nivel de la complejidad de las redes. Al momento de diseñar cada red, se buscó que tuvieran, cada una, una performance aceptable para cada uno de los usuarios seleccionados. Por lo tanto, es importante que la complejidad de cada red sea la menor posible para que posean la capacidad necesaria de generalización.

Por otra parte, como solo se toman los datos de un usuario en particular para entrenar las redes, la cantidad de datos es muy baja y no permite llegar a entrenar apropiadamente a redes más complejas y con mayor cantidad de parámetros. Este fue un trade-off al momento de diseñar las arquitecturas propuestas y podría llegar a ser resuelto aumentando la granularidad en el pre-procesamiento y tomar, por ejemplo, buckets de tiempo de 30 minutos o 15 minutos, resultando así en más casos de entrenamiento.

Otro de los trade-off que tuvimos que enfrentar fue la cantidad de time-lags a tomar. Es lógico pensar que cuanta más información del pasado se le dé a la arquitectura, más información tendrá y así podrá hacer mejores predicciones. Esto no es cierto por dos razones. La primera de ellas es que el MET que tendrá un usuario en el tiempo t+1 puede no estar relacionado con las features que poseemos del mismo usuario en el tiempo t-n, siento n un entero arbitrario. Esto, al tener pocos casos de entrenamiento, puede generar ruido en el modelo. La segunda razón es que, como en muchos casos no se poseen datos sobre todas las horas que duró el experimento, cuantos más time-lags se tengan, mayor es la probabilidad de que un caso de entrenamiento sea descartado. Por ejemplo, si la cantidad de time-lags es n y si se tienen los datos del usuario x desde el tiempo t al t-n+1 pero no del tiempo t-n, el caso de entrenamiento se desecha.

Un punto importante a destacar es que, como los estudiantes mantuvieron, por lo general, un comportamiento totalmente sedentario, los clasificadores que intentaban siempre predecir valores cercanos a cero MET fueron descartados, ya que conseguían un MSE bajo pero no eran útiles. Este problema fue tratado en el trabajo realizado por M. Santillán Cooper y M. Armentano con las técnicas de over-sampling y asignación de peso a las clases.

Con respecto a las hipótesis planteadas al comienzo del informe, todas ellas pudieron ser evaluadas y comprobadas. Se lograron diseñar arquitecturas con un mejor desempeño que aquellas arquitecturas que no utilizan información del pasado. Además, se comprobó que las CNN pueden llegar a desempeñarse de una mejor manera que las RNN.

## Bibliografía

1. https://www.sedentarybehaviour.org/ [↑](#footnote-ref-0)
2. http://www.pewresearch.org/fact-tank/2018/09/28/internet-social-media-use-and-device-ownershipin-

   u-s-have-plateaued-after-years-of-growth/ [↑](#footnote-ref-1)
3. https://github.com/martinscooper/Making-it-linearly-separable [↑](#footnote-ref-2)
4. https://colah.github.io/posts/2014-03-NN-Manifolds-Topology/ [↑](#footnote-ref-3)
5. <https://studentlife.cs.dartmouth.edu/dataset.html> [↑](#footnote-ref-4)
6. <https://sites.google.com/site/compendiumofphysicalactivities/home> [↑](#footnote-ref-5)
7. <https://www.python.org/dev/peps/pep-0008/#function-and-variable-names> [↑](#footnote-ref-6)
8. <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html> [↑](#footnote-ref-7)
9. <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html> [↑](#footnote-ref-8)
10. <https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.TimeSeriesSplit.html> [↑](#footnote-ref-9)
11. <https://scikit-optimize.github.io/stable/> [↑](#footnote-ref-10)