T

E

S

I

S

Alumno: Santillán Cooper, Martín

Director: Armentano, Marcelo

Co-Directora: Tomassel, Antonela

[Introducción](#_ziax18pwx0k)

[Trabajos relacionados](#_ziax18pwx0k)

[Marco Teorico](#_ziax18pwx0k)

[Machine Learning](#_96a4zfw1rsd0)

[Introduccion](#_ziax18pwx0k)

[Tipos de tareas:](#_ziax18pwx0k)

[Fases de un predictor](#_ziax18pwx0k)

[Generalizacion, underfitting y overfitting](#_ziax18pwx0k)

[Modelos lineales y no Lineales](#_ziax18pwx0k)

[Aprendizaje supervisado y no supervisado](#_ziax18pwx0k)

[Supervisado](#_ziax18pwx0k)

[No supervisado](#_ziax18pwx0k)

[Clasificación y Regresion (definicion mas precisa)](#_ziax18pwx0k)

[Personales e Impersonales](#_ziax18pwx0k)

[Descenso por el gradiente](#_ziax18pwx0k)

[Métricas de evaluación](#_ziax18pwx0k)

[Metricas de clasificacion](#_hffmyfd8n6kz)

[Accuracy:](#_hffmyfd8n6kz)

[Tasa de error:](#_hffmyfd8n6kz)

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

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

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

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

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

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

[ROC:](#_hffmyfd8n6kz)

[AUC:](#_hffmyfd8n6kz)

[Precission - Recall curve:](#_hffmyfd8n6kz)

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

[Reproductibilidad de resultados](#_96a4zfw1rsd0)

[Funcion de hipotesis](#_96a4zfw1rsd0)

[Funcion de perdida](#_96a4zfw1rsd0)

[Función de optimización](#_96a4zfw1rsd0)

[Tecnicas de Regularizacion](#_96a4zfw1rsd0)

[Deep Learning](#_ziax18pwx0k)

[Modelos de DL usados](#_96a4zfw1rsd0)

[DNN](#_ziax18pwx0k)

[RNN](#_ziax18pwx0k)

[CNN](#_ziax18pwx0k)

[Combinacion entre ellos](#_ziax18pwx0k)

[Hiper-Parametros](#_96a4zfw1rsd0)

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

[Numero de capas](#_ziax18pwx0k)

[Número de neuronas](#_ziax18pwx0k)

[Funcion de acticacion](#_ziax18pwx0k)

[Numero de epocas](#_ziax18pwx0k)

[Tamaño de bache](#_ziax18pwx0k)

[Tuning de hiper-parametros](#_ziax18pwx0k)

[Descripción del problema](#_3v8ermb1rvk3)

[Descripción del dataset](#_3znysh7)

[Pre-procesamiento del dataset](#_2et92p0)

[Arquitecturas propuestas](#_tyjcwt)

[Métrica de evaluación seleccionada](#_3dy6vkm)

[Resultados](#_1t3h5sf)

[Discusión](#_4d34og8)

[Conclusiones](#_bu3zadqsn4l9)

[Bibliografía](#_bwxenck50ema)

# 

# 

# 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 basandose en datos de sensores. No debe confundirse al area de prediccion 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 aceurto 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-defendent)? 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 hayar 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 Teorico

## Machine Learning

ESLII 2.6

Our goal is to find a useful approximation fˆ(x) to the function f(x) that

underlies the predictive relationship between the inputs and outputs.

The additive error model is a useful approximation to the truth. For

most systems the input–output pairs (X, Y ) will not have a deterministic

relationship Y = f(X). Generally there will be other unmeasured variables

that also contribute to Y , including measurement error. The additive model

assumes that we can capture all these departures from a deterministic relationship via the error ε.

Y = f(X) + ε

### Introduccion

Aunque esta tesis se basó, sobretodo, en el uso de modelos de Deep Learning para el problema de prediccion del comportamiento sedentario futuro, es importante dar una base a los conceptos mas importantes en el área de Machine Learning, ya que Deep Learning es un campo de estudio que se encuentra dentro del de Machine Learning. Es decir, para comprender de manera correcta, es necesario tener un solido entendimiento de los principios basico de Machine Learning. Machine Learning es definido en (Libro Deep Learning) como “una forma estadistica aplicada que hace gran enfasis en el uso de computadoras para estimar estadistiamente funciones complejas y un menor enfasis en probar intervalos de confianza en dichas funciones”.

Es comun que se nombren a los algoritmos de machine learning como Algoritmos de Aprendizaje. 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. Ahora bien, esta definicion es mas que nada general y puede aplicarse a la totalidad del area de la Inteligencia Artificial. En general, en el area de Machine Learning las tareas son descriptas a partir de como el modelo de Machine Learning deberia procesar un determinado ejemplo o caso. Un ejemplo puede ser descripto como un conjunto de caracteristicas que han sido cuantitativamente medidas a partir de cierto evento u objeto, que queremos que el modelo de Machine Learning procese. Tipicamente, cada ejemplo se representa como un vector perteneciente a donde cada componente del vector es una caracteristica diferente. Por ejemplo por ejemplo borrar las características de una imagen son usualmente los valores de los píxeles de dicha imagen muchos tipos de tareas pueden ser resueltas utilizando machine learning incluya las siguientes.

### 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 descripto 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 clasificacion

* 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.

* Regresion: 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

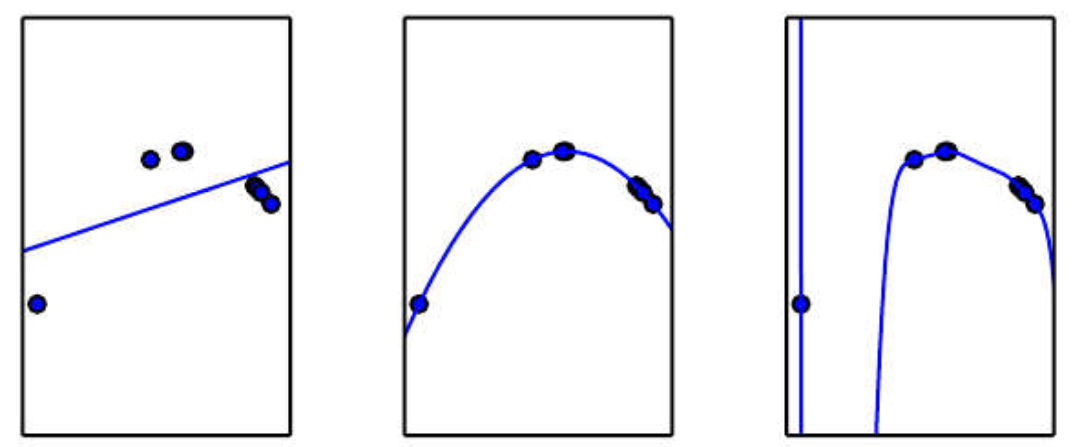
### Fases de un predictor

Existen tres importantes en el ciclo de vida de un predictor: la fase de entrenamiento, la fase de validacion y la fase de testeo. En la fase de entrenamiento, el algoritmo de aprendizaje ajusta los parametros del modelo a partir del conjunto de datos de entrenamiento. En esta fase, el error de entrenamiento es computado para medir cuan bien el modelo se ajusta a los datos de entrenamiento. La fase de testeo es utilizada generalmente para comparar diferentes versiones de un algoritmo de aprendizaje, esto es, con diferentes hiper-parametros. Esto permite evaluar al modelo con datos con los que no ha sido entrenado a partir del error de testeo. Por lo general, el error de testeo es mayor que el error de entrenamiento. Por ultimo, y en ciertas ocaciones, se utiliza la fase de validacion, en la cual se valida el modelo entrenado utilizando los hiper-parametros que obtuvieron un mejor desempeño en la fase de validacion. Esta tercera fase permite evaluar la capacidad de generalizacion del modelo. Cabe aclarar que los conjuntos de datos de las 3 fases, entrenamiento, testeo y validacion, son conjuntos disjuntos. A veces, la fase de validacion no es utilizada, ya que esta implica reservar una parte del conjunto de datos disponibles que el algoritmo de aprendizaje no puede utilizar para entrenarse. Esto tiene un gran impacto sobretodo cuando el conjunto de datos es pequeño. Cuando el conjunto de datos es muy grande - millones de entradas-, el conjunto de datos de validacion puede representar un porcion muy pequeña. Por ejemplo, el de los datos, lo cual, obviamente, no tiene un impacto signicativo en la cantidad de casos disponibles para entrenar al modelo.

### 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 entradas nuevos. Esto es, en casos que no habian sido visto antes por el modelo, y no solamente en los casos que fueron utilizados para entrenar el modelo. La habilidad de un modelo para desempeñarse bien en datos que no habian sido observados es llabada generalizacion.

Como se explica en la seccion anterior, al entrenar un algoritmo de machine learning, se utiliza un conjunto de entrenamiento. A partir del cual se puede se intenta minimizar el error, lo que puede considerarse un problema de optimizacion. A partir de esto, se calcula el error de generalizacion 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 parametros del modelo (aquellos que se ajustan en el proceso de optimizacion) son ajustados de acuerdo al conjunto de entrenamieno, 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 determinar si un algoritmo de machine learning: hacer pequeño el error de entrenamiento y hacer que el arror de testeo sea lo mas similar posible al error de entrenamiento. Estos dos objetivos se corresponden con los dos desafios mas importantes 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 en que no haya ni overfitting ni underfitting. Esto ultimo se logra aumentando o disminuyendo la capacidad del modelo. La capacidad es se refiere a la complejidad del espacio de hipotesis, esto es, todas las funciones que que el algoritmo de aprendizaje puede llegar a seleccionar como posible solucion. Por ejemplo, el espacio de hipotesis de la regresion lineal es el conjunto de todas las posibles funciones lineales de acuerdo a las entradas del modelo. La capacidad de un algoritmo de regresion lineal puede aumentarse generalizando el espacio de hipotesis para incluir polinomios de un determinado grado. 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 (underfitting). Mientras que los modelos con alta capacidad pueden resolver tareas complejas, pero si su capacidad es mas alta que la necesaria para resolver una determinada tarea, pueden cometer overfitting.



Funcion lineal, Polinomio grado 2, grado 9.

### Modelos lineales y no Lineales

### Aprendizaje supervisado y no supervisado

### Supervisado

Utiliza como experiencia un dataset donde cada caso esta asociado a una etiqueta. La tarea de este clasificador es lograr predecir la etiqueta a partir de las caracteristicas de cada caso.

### No supervisado

En este tipo de algoritmos se utilizan datasets que continene diferentes caracteristicas y casos de estudio y la tarea es tratar de aprender propiedades utiles 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).

### Clasificación y Regresion (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 evaluacion de su desempeño va a diferir. En el primer camino nombrado, estamos en la presencia de un problema de regresion, donde se busca predecir una variable continua. En el segundo, el problema es de clasificacion, donde se busca predecir una variable discreta. A continuacion, paso a describir las caracteristicas, similutudes y diferencias entre estos dos tipos de modelos.

### Personales e Impersonales

### Descenso por el gradiente

### Métricas de evaluación

para evaluar las habilidades de un algoritmo de machine learning diseñarse una medida cuantitativa de su desempeño. por lo general está medida es específica a cada tipo de tarea diferente que el algoritmo de machine learning está llevando acabo. a continuación se detallan las métricas de evaluación para los diferentes tipos de tarea.

### Metricas de clasificacion

A la hora de evaluar el desempeño de un algoritmo de machine learning curso tarea resolver pertenece la tarea de clasificación podemos hallar varias medidas diferentes. cada una de estas medidas busca analizar de una forma diferente el desempeño del algoritmo.

* 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 clasificacion binaria, podemos observar que la matriz de confusion sera de y contara con cuatro celdas. Estas cuatro celdas representan los 4 tipos en los que pueden clasificarse las predicciones de un algoritmo de clasificacion. En el caso de tareas de clasificacion con mas de dos clases, el problema puede descomponerse en tareas de clasificacion binaria. Cabe aclarar que en un problema de clasificacion binaria las dos clases son comunmente denotadas de diferentes formas, como verdadero y falso, positivo y negativo, 1 y 0, etc. A continuacion se explican los valores con los que se completan las matrices de confusion. 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 termino verdadero se debe a que el clasificador a hecho una prediccion 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 termino falso se debe a que el clasificador a hecho una prediccion 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 confusion ha sido presentado, vamos a ver que cada una de las metricas que seran presentadas a continuacion son calculadas a partir de los 4 tipos de resultados de clasificacion.

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

### Accuracy:

Accuracy y Precission son dos metricas que llevan el mismo nombre si son traducidas al español, porque lo que mantendre sus nombre en ingles. Se denomina Accuracy al numero de predicciones correctas hechas por el modelos sobre la cantidad de predicciones hechas.

La metrica no debe ser utilizada cuando las clases estan desbalanceadas, es decir, que la cantidad de casos de una clase es mucho mayor a la cantidad de casos de la otra. Esto es asi porque la ecuacion de esta metrica es sensible al desbalanceo de clases. Supongamos, por ejemplo, que se aumenta en un factor el numero de casos de la clase negativa. Reescribiendo la ecuacion, obtenemos . Por ejemplo, en la tarea de predecir si una persona tiene cancer o no, supongamos que en el dataset con el que se testea al algoritomo el 99% de las persona no tiene cancer. Si el algoritmo de clasificacion simpremente precide siempre que la persona no tiene cancer, obtendria un Accuracy de 0.99. En estos casos, es preferible optar por otros tipos de metrica.

### 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.

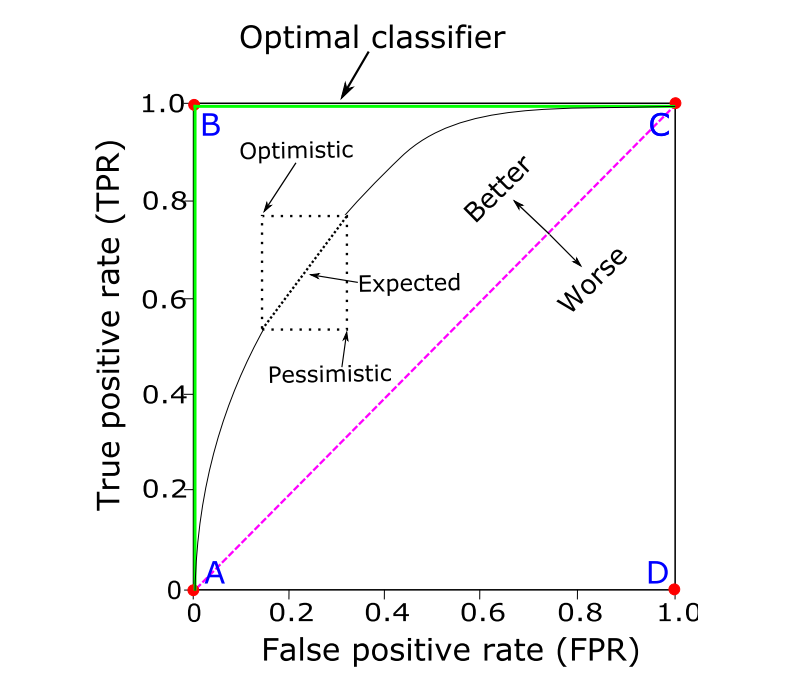
### Metrica F1 o F1 Score:

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 seran negativos. A continuacion, a partir de cada threshold se calculan TPR y FPR para obtener un punto en el grafico ROC.

### 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

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

Hasta el momento, solo he hablado de un trade off entre overfitting y under fitting a partir de aumentar o disminuir la capacidad de un algoritmo de aprendizaje. En realidad, esto es mas complejo, ya que es posible configurar a un algoritmo para preferir algunas funciones pertenecientes a su espacio de hipotesis por sobre otras de acuerdo a cierto criterio o regularizador. Es decir, las funciones menos preferidas van a ser elegidas solo si se desempeñan mucho mejor que las preferidas en el conjunto de entrenamiento. Puede interpretarse a la regularizacion como una forma general de variar la capacidad de un algoritmo de aprendizaje, ya que sacar a una funcion del espacio de hipotesis puede interpretarse como una preferencia infinita de las funciones que aun pertenecen al espacio de hipotesis por sobre la funcion excluida. Por lo tanto, las tecnicas de regularizacion tienen como objetivo disminuir la capacidad de un algoritmo de aprendizaje sin excluir explícitamente ninguna funcion del espacio de hipotesis, sino que dejando que el algoritmo defina cuales son las funciones mas apropiadas a aprender en el proceso de entrenamiento. La regularizacion tiene como objectivo disminuir el error de testeo sin afectar el error de entrenamiento. Dicho de otra manera, se busca que el modelo aprenda la funcion mas 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, supongase que se generan conjuntos de entrenamiento y testeo a partir de una funcion cuadratica. A continuacion, se entrena un modelo cuya capacidad son todas las funciones polinomicas. Si no se utilizase ninguna tecnica de regularizacion, el algoritmo apredenria una funcion utilizando un polinomio de grado (es decir, no tendra ninguna preferencia por aprender polinomio de grado 2, por ejemplo). De esta forma, el error de testeo sera elevado, ya que el modelo predecira el conjunto de testeo a partir del polinomio aprendido y no especificamente a partir de la funcion cuadratica. En este caso, una tecnica de regularizacion podria penalizar al algoritmo por cuanto mas grande sea el grado del polinomio. Debido a esto, el algoritmo debera minimizar, no solo el error de entrenamiento, sino que tambien el grado del polinomio. Por lo tanto, el algoritmo terminara aprendiendo la funcion cuadratica. Notese que el algoritmo no deberia aprender un polinomio de grado 1 (funcion lineal) ya que de esta forma aumentaria el error de testeo (suponiendo que el conjunto de entranamiento es lo suficientemente grande como para no parecerse a una funcion lineal).

Muchas tecnicas de regularizacion han sido desarrolladas en el ultimo tiempo, la mayoria pudiendo ser expresadas especificamente en algoritmos de Deep Learning, aunque muchas tambien son aplicables en algoritmos de Machine Learning. La tecnica de regularizacion mas comun

## Reproductibilidad de resultados

## Funcion de hipotesis

## Funcion de perdida

## Función de optimización

## Tecnicas de Regularizacion

# Deep Learning

## Modelos de DL usados

### DNN

### RNN

### CNN

### 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.

Los modelos de clasificacion pueden ser divididos entre modelos discriminativos y modelos generativos (Bishop 2006). Dado un conjunto de puntos de datos {𝑥 (1), ..., 𝑥(𝑚)}

asociados a un conjunto de resultados {𝑦 (1), ..., 𝑦(𝑚)}, los modelos descriminativos estiman 𝑃(𝑌| 𝑋) mientras que los modelos generativos estiman 𝑃(𝑋| 𝑌) y 𝑃(𝑌) para calcular la probabilidad posterior 𝑃(𝑌| 𝑋). En esta sección, se explicarán solo los modelos discriminativos ya que los modelos generativos no han sido utilizados.

Particularmente, se usaron la Regresion Logistica (LR) y Redes Neuronales Profundas (DNN) para explorar el problema de PCSF.

paper, we analyze the performance of discriminative models. Particularly, we use Logistic

Regression and Deep Neural Networks for analyzing whether the user will exhibit sedentary

behaviour in the next hour, given a set of features extracted from mobile devices’ sensors.

We chose these two machine learning techniques in order to compare the performance

between a linear and a non-linear approach.

Both Logistic Regression and Deep Neural Networks classifiers share similar

characteristics. The main problem they approach is to fit a function that maps an input

vector to a category . Formally, this mapping can be seen as a function ,

where is the set of parameters of the model, is the vector of features extracted from the

mobile devices’ sensors and , where 0 corresponds to non-sedentary behaviour

and 1 corresponds to sedentary behaviour. The parameters are learned by minimizing a loss

function. The method that minimizes the lost function is called optimizer. In general, and

albeit having many variations, the optimizer uses gradient descent to search the optimal

value of the parameters that minimize the loss function.

La regresion Logistica es un clasificador lineal que usa la funcion sigmoide como su hipotesis. La funcion sigmoide

Logistic Regression (LR) is a linear classifier that uses the sigmoid function as its

hypothesis. The logistic function is defined as and has the property that its codomain goes from 0 to 1, having a horizontal asymptote in both values. Thus, the output of

the sigmoid function can be seen as the confidence that the classifier has on that the

example corresponds to a particular class. For example, the closer to 1 the output, the

greater the confidence of the classifier in the fact that the input corresponds to the 1 class.

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.

Deep Neural Network (DNN) is an algorithm that belongs to the Deep Learning field.

DNNs can learn multiple levels of representation in order to model complex

relationships among data. In them, higher-level features and concepts are defined in

terms of lower-level ones. DNN is a type of model composed of a sequence of layers

(Goodfellow, Bengio, and Courville 2016). At the same time, each layer is composed of a

group of neurons, also called units. The first layer in a DNN is the input layer and

corresponds to the feature vector. The input layer is followed by one or more hidden

layers. The last layer in a DNN is the output layer. Each hidden unit computes the dot

product between the output of the previous layers and its corresponding weights -i.e the

parameters of the DNN-. Then, an activation function is applied to each unit to introduce

non-linear complexities to the model. In comparison to linear models, DNNs in general and

especially those used in this research, have many parameters that have to be learned. To

learn those parameters, DNNs use backpropagation with different optimization techniques.

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.

After finishing the process of tuning the DNN used in this work, we obtained a

network with 4 hidden layers architecture composed by 256x128x64x32 units, respectively.

As the activation function, we used Rectified Linear Units (ReLu), which uses the activation

function . To prevent overfitting, each hidden layer used Batch

Normalization (Ioffe and Szegedy 2015) and regularization. , also known as weight

decay, drives the weights closer to the origin by adding the sum of the square of each

parameter of the model to the loss function. The output layer has only one neuron with a

sigmoid activation function. We used binary cross-entropy as the loss function and Adaptive

Moment Estimation (Adam) (Kingma and Ba 2015) as the optimization method. Each DNN

model was fitted with 30 epochs and a batch size of 512, in the case of impersonal models,

and a batch size of 32, in the case of personal models.

All DNN architectures tested were designed and implemented using the deep

learning library Keras3 for Python. The Logistic Regression implementation used in our

experiments was the one provided by Scikit-Learn4, a machine learning library implemented

in Python.

erceptrón Multicapa

El Perceptrón Multicapa (MLP) es la arquitectura más sencilla dentro del campo de

Aprendizaje Profundo. Básicamente es una red de unidades computacionales

(neuronas) unidas por conexiones directas. 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 cuan fuertemente están

conectadas dos neuronas. Típicamente, la computación que lleva a cabo está separada

2 http://www.pewresearch.org/fact-tank/2018/09/28/internet-social-media-use-and-deviceownershipin-u-s-have-plateaued-after-years-of-growth/

en dos etapas: primero realiza la suma ponderada de todas sus conexiones entrantes y

luego aplica una función de activación [15].

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 .

2.2 Redes Neuronales Convolucionales

La Redes Neuronales Convolucionales (CNN) 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.

2.3 Redes Temporales Convolucionales

En Bai et al. 2018 [36] los investigadores presentan un tipo de CNN especialmente

diseñada para las tareas de predicción y modelado de secuencias, llamada Redes

Temporales Convolucionales (TCN). En dicho trabajo, los investigadores realizan una

evaluación sistemática de arquitecturas convolucionales y recurrentes.

Específicamente, se ponen a prueba tareas que han sido utilizadas en repetidas

ocasiones para comparar la efectividad de diferentes RNNs. 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 con mayor precesión.

Las TCN ya habían sido utilizada en otros trabajos [19, 22], pero en Bai et al. 2018

[36] se implementa su versión más simple, y es, además, la versión utilizada en el

presente trabajo. En dicha versión, se busca que las TCN tengan todas las ventajas que

presentan las CNN más simples –alta paralelización, no sufrir vanishing gradient o

exploding gradient, baja utilización de memoria-, pero siendo adaptadas para el

modelado de secuencias y la predicción de series de tiempo. Por esta razón, existen

varias diferencias claves entre ambas. Las diferencias más importantes son que las

TCN:

• No utilizan las capas de pooling para que la dimensión de entrada sea igual a la de

salida,

• Siempre aplican la convolución de una dimensión (la del tiempo).

• Son causales, es decir, las convoluciones solo consideran información del pasado y,

• Utilizan dilataciones para aumentar el campo receptivo exponencialmente y así

poder cubrir largas dependencias en los datos secuenciales.

2.4 Redes Neuronales Recurrentes

Las Redes Neuronales Recurrentes son modelos dedicados a secuencias que mantienen

un estado interno que es propagado a través del tiempo. Dicho estado interno funciona

como la memoria 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 tienen

como entrada un vector que representa una secuencia (por ejemplo: una oración) y las

neuronas procesan cada elemento de dicho vector uno a la vez e iterativamente [23].

Las RNN han ganado mucha popularidad por su buen desempeño en tareas

realacionadas al modelado del lenguaje [24] y traducción de máquina [25].

Desgraciadamente, las arquitecturas mas simples de RNN son difíciles de entrenar [26],

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 [27] y las GRU [28]

que añaden mayor cantidad de información a cada neurona en comparación con las

RNN más simples, como las compuertas de olvido.

# Descripción del problema

El problema que nos proponemos resolver está enmarcado dentro de la predicción del comportamiento sedentario futuro (FSBP, por sus siglas en inglés) y es el de predecir el gasto de energía (METs) que tendrá una persona en el futuro en base a datos recolectados de sensores de Smartphone. El gasto energético es medido en MET (Metabolic Equivalent of Tasks), 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 física a toda aquella actividad cuyo MET asociado sea menor o igual a 1.5.

El problema de predicción a resolver es de regresión, ya que, como queremos predecir el gasto de energía, y éste es un valor continuo, debemos usar algoritmos de regresión. Más específicamente, de todos los tipos de redes neuronales vistas durante la cursada, utilizaremos Redes Neuronales Recurrentes (RNN) y Redes Neuronales Convolucionales, que son especialmente aptas para el problema que trataremos porque los datos que usamos están organizados en series de tiempo, por lo que tenemos un problema donde la secuencialita juega un papel importante. Para generar las features que serán utilizadas como entrada para diferentes Redes Neuronales, pre-procesaremos un dataset llamado StudentLife[[3]](#footnote-2).

Es importante notar que el trabajo desarrollado plantea un nuevo enfoque al problema de FSBP ya que laforma de estimar el nivel de sedentarismo utilizada hasta ahora en el estado del arte no está de acuerdo a la propuesto por la comunidad científica ya que no toma en cuenta el valor de MET de cada tipo de actividad física llevada a cabo, a diferencia de nuestro trabajo, en el que si fue llevado a cabo el cálculo de MET.

En este trabajo, trataremos predecir el valor de MET para un usuario únicamente en la hora siguiente. Es decir, dados los datos de las horas *t-n, ..., t-2, t-1, t* utilizaremos NNs para hallar el gasto energético en el tiempo *t+1.* Nuestro objetivo principal es evaluar si es factible utilizar redes neuronales que puedan tratar con problemas de predicción que deban aprender de una secuencia de datos para el problema de FSBP. Además, nos interesa comparar la performance de las RNN y las CNN (mas específicamente, las TCN), con el objetivo de comprobar los resultados empíricos conseguidos en “An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling”, por S. Bai et. al. Para evaluar nuestras hipótesis diseñaremos arquitecturas con RNN, CNN normales, TCN y simple NN.

# Descripción del dataset

Analizamos y pre-procesamos el dataset StudentLife[[4]](#footnote-3) [(Wang et al. 2014)](https://paperpile.com/c/FQGDmo/eCuKI) para probar la validez de nuestros modelos. Este dataset contiene datos de los sensores de los Smartphone de 48 estudiantes a lo largo de 10 semanas en la primavera de 2013.

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. Los datos disponibles incluyen: datos de actividad física, datos de conversación, datos de ubicación, entre otros.

Diferentes tipos de datos fueron registrados con diferentes frecuencias, dependiendo del tipo de sensor y la carga de trabajo global del Smartphone. Por ejemplo, la actividad física se registró cada 2–3 segundos en 1 de cada 4 minutos, el uso de aplicaciones fue se registró cada 20 minutos, y los registros de ubicación cada 10 minutos.

# Pre-procesamiento del dataset

Todos los datos del dataset StudentLife poseen un *timestamp* asociado. Decidimos discretizar las series de tiempo en *buckets* de una hora ya que esta es la granularidad utilizada en la mayor parte del trabajo relacionado para la predicción de sedentarismo. Por lo tanto, todas las *features* 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 *bucket* específico puede corresponder al usuario 10 y la hora 2013-04-24 19: 00–20: 00. Los *buckets* para los cuales no teníamos información sobre la actividad física del usuario fueron eliminados. Además, se eliminó la información de un usuario en particular que presentaba inconsistencias en sus registros de actividad física. En total, obtuvimos 60,819 *buckets* para los 48 usuarios.

Los registros de actividad de cada usuario se están clasificados en el dataset como *estacionario*, *caminando* o *corriendo.* A partir de estos registros se calculó el valor del gasto energético para cada *bucket.* Tal como fue explicado anteriormente, el gasto energético es comúnmente medido de términos de Metabolic Equivalent of Tasks (MET). A cada tipo de actividad se le asignó un valor estático de MET de acuerdo al Compendio de Actividades Físicas[[5]](#footnote-4). Finalmente se calculó el promedio de MET para cada *bucket.*

Dado que discretizamos todos los datos en *buckets*  de una hora, tuvimos que tomar en cuenta que información iba a ser útil para predecir el valor del MET en las horas siguientes. Las *features* usadas fueron elegidas basándonos en previas investigaciones [(He and Agu 2016)](https://paperpile.com/c/FQGDmo/GSfPw)[(Cook and Krishnan 2015)](https://paperpile.com/c/FQGDmo/DZqSa), así como también otras fueron agregadas.

A continuación definimos las 20 *features* generadas pre-procesando el dataset StudentLife:

**GPS features**

* distanceTraveled: la distancia *haversine* recorrida calculada a partir de los registros de GPS;
* locationVariance: utilizado para medir la variabilidad en la ubicación GPS de un participante en cada *bucket*, computado como se describe en [(Saeb et al. 2015)](https://paperpile.com/c/FQGDmo/TbLIy).

**Time features**

* hourSine: transformación de seno de la hora;
* hourCosine: transformación de coseno de la hora;
* dayOfWeek: día de la semana;
* pastMinutes: el número de minutos transcurridos desde el comienzo del día; remainingMinutes: el número de minutos que quedan para terminar el día;

**Physical activity features**

* Stationary level: el porcentaje de instancias de actividad física clasificadas como estacionario/a’ en cada *bucket;*
* Walking level: el porcentaje de instancias de actividad física clasificadas como ‘caminando’ en cada *bucket;*
* Running level: el porcentaje de instancias de actividad física clasificadas como de ‘corriendo’ en cada *bucket;*
* activityMajor: el tipo de actividad física con más instancias en cada *bucket*;

**Audio features**

* SilenceLevel: el porcentaje de instancias de audio clasificadas como ‘silencio’ en cada bucket;
* voiceLevel: el porcentaje de instancias de audio clasificadas como ‘voz’ en cada bucket;
* noiseLevel: el porcentaje de instancias de audio clasificadas como ‘ruido’ en cada bucket;
* numberOfConversations: el número de conversaciones que ese estudiante tuvo en cada bucket;

**Other features**

* isCharging: si el Smartphone se estaba cargando;
* isLocked: si el Smartphone estaba bloqueado;
* isInDark: si el Smartphone estaba en la oscuridad;
* hasCalendarEvent: si tenía un evento programado en el calendario;
* wifiChanges: el número de cambio de conexiones wifi en cada bucket;
* sLevel: el valor de MET para la hora actual.

Finalmente, dado que las features *dayOfWeek* y *activityMajor* son categóricas, se realizó dummy-encoding sobre ellas. Con esta transformación, se obtuvieron 7 dummy features representando las categorías para la *feature* *dayOfWeek* y 3 para la feature *activityMajor*. Como resultado de esta última transformación, se finalizó el proceso con 28 *features* smartphone que serán luego usadas por los modelos predictivos para predecir el comportamiento sedentario de los usuarios.

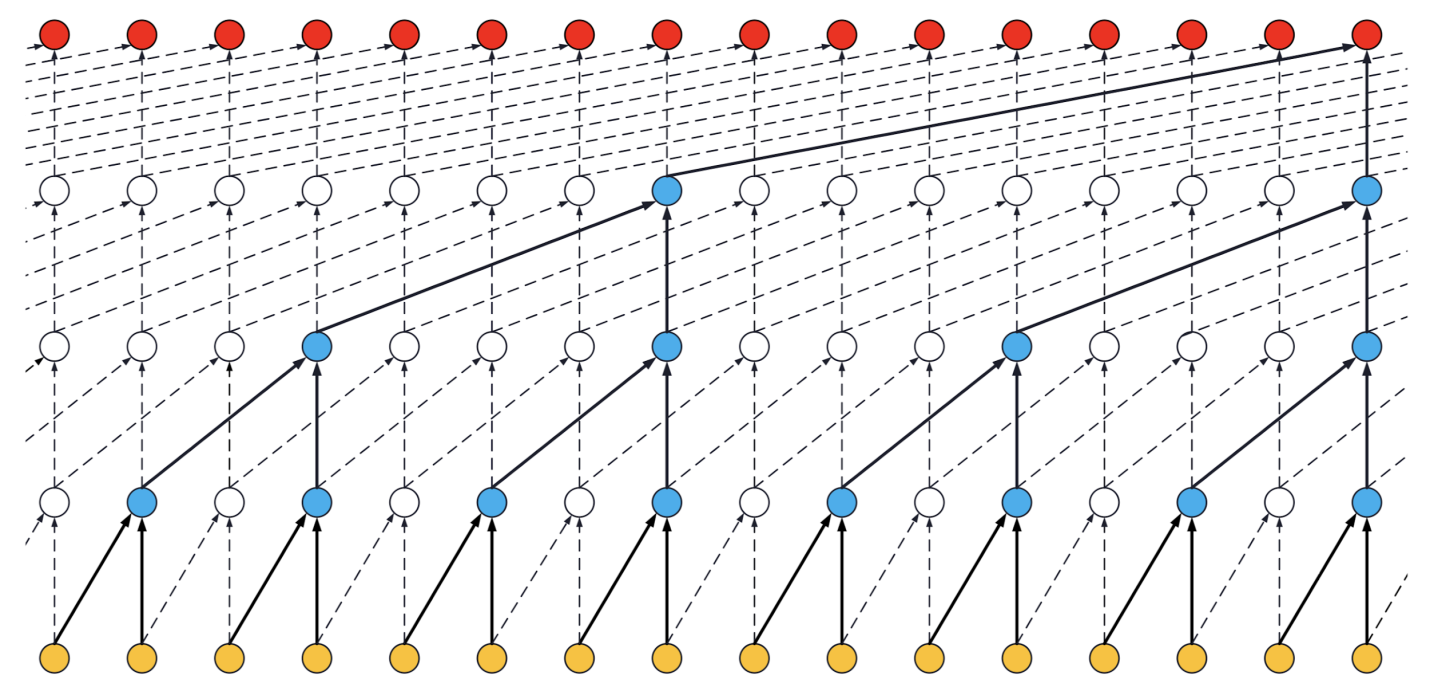
El siguiente paso del pre-procesamiento consistió en generar los time-lags. La cantidad de time-lags depende de cada arquitectura propuesta, siendo en algunos casos cero la cantidad de time-lags. Para ello, a una instancia del dataset que corresponde a un tiempo *t*  se le agregan las features de tiempos anteriores, dependiendo de la cantidad de time-lags que utilice la arquitectura. Este proceso aumenta la cantidad de features tantas veces como time-lags utilice la arquitectura, aumentando así el tamaño del dataset. Luego, se separa el dataset entre *X* e *y,* siendo *X* todas las *features* diferentes al *metLevel* del tiempo t (tiempo presente)*.* Por último, se dividen *X* e *y* entre casos de entrenamiento y de testeo. Para esta última división se usó un proporción de 2 a 1 en todos los casos.

El último paso del pre-procesamiento consiste en normalizar los datos. Para este paso se utilizaron normalizadores proveídos por la librería *Scikit-learn*. Varios normalizadores fueron tenidos en cuenta y, finalmente, se utilizaron diferentes según las características de cada usuario. Para evitar la fuga de datos, o también llamado *data lickeage,* para preparar los datos de entrenamiento de cada modelo primero realizamos una normalización de los datos de entrenamiento y, con la distribución aprendida por el normalizador, normalizamos los datos de entrenamiento. El *data lickeage* debe ser siempre tenido en cuenta ya que si se normalizan todos los datos al mismo tiempo se estaría perdiendo la posibilidad de evaluar la capacidad generalización de los predictores, ya que les estaríamos dando ya información de los datos que serán luego utilizados para testeo.

# Arquitecturas propuestas

Las arquitecturas propuestas fueron aquellas aptas para tratar con series de tiempo. Ellas son: Redes Neuronales Recurrentes (RNN) y Redes Neuronales Convolucionales (CNN). Clásicamente, las RNN fueron la elección para cualquier problema en el que la secuencia de los datos fuera importante (por ejemplo: traducción de texto). Recientemente, ciertos tipos de CNN han comenzado a utilizarse en los problemas para los que las RNN conseguían el mejor desempeño. Más específicamente, las CNN mencionadas utilizan operaciones convolucionales de una única dimensión. Entre las CNN propuestas, se encuentran aquellas implementadas con Keras por nosotros y las Temporal Convolutional Networks (TCN)[[6]](#footnote-5).

Las TCN son CNN que aplican la convolución de una dimensión, son causales, 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. 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-parametros -proceso también llamado “tuneo”- se realiza de una forma más rápida. La implementación utilizada para las TCN puede hallarse en el siguiente [link.](https://github.com/philipperemy/keras-tcn#why-temporal-convolutional-network) En Fig. 1 se puede observar un ejemplo de una sencilla TCN. En d icha figura se ve claramente el carácter causal de la red, ya que nunca se toma información del futuro. La red del ejemplo posee un único bloque residual y dilaciones de 1, 2, 4 y 8.

Fig. 1

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 t, si la cantidad de time-lags es 3, el input de la red neuronal estará formado por las features del tiempo anterior (t-1), así como también las features de los tiempos t-2 y t-3.

En general, los hiper-parámetros a tunear 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 kernel. Más aun, las TCN poseen hiper-parámetros que no poseen las CNN: como la cantidad de bloques residuales o la lista de dilataciones. Estos últimos hiperparámetros 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, las arquitecturas 1,2 y 3 son RNN, las arquitecturas 4 y 5 son CNN y la arquitectura número 6 es una NN simple. La arquitectura número 6 es una NN simple que utiliza únicamente información de la hora anterior para predecir el valor del MET de la hora siguiente y fue agregada a modo de comparación y para evaluar la utilidad de utilizar datos del pasado para FSBP. Esto quiere decir que si la arquitectura 6 supera a las RNN y CNN que utilizan información del pasado para predecir el valor de MET puede significar que, para este problema, la información del pasado no ayuda a mejorar la calidad de los predictores.

A continuación se muestra un cuadro que describe las arquitecturas propuestas. Las características más detalladas de cada arquitectura pueden ser analizadas en la función *get\_model* del script *tp\_dl\_train.py.*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Arq. 1 | Arq. 2 | Arq. 3 | Arq. 4 | Arq. 5 | Arq. 6 |
| Tipo de capas | LSTM - FC x 3 | LSTM x 2 - FC x 1 | LSTM x 3 - FC x 4 | Ver *model.summary()* | Conv1D x 1 - Flatten x 1 - FC x 1 | Dense x 3 |
| Neuronas por capa | 32-16-8-1 | 64-32-1 | 128-64-32-32-16-8-1 | Ver *model.summary()* | 32 - No aplica - 1 | 64 - 32 - 1 |
| Técnica de regularización | Dropout | Dropout | Dropout | Dropout | Dropout y Batch Normalization | Dropout |
| Número de time-lags | 8 | 12 | 12 | 8 | 4 | 1 |
| Algoritmo de optimización | Adam | Adam | Adam | Adam | Adam | Adam |
| Función de pérdida | MSE | MSE | MSE | MSE | MSE | MSE |
| Cantidad de épocas | 256 | 128 | 128 | 64 | 128 | 256 |
| Tamaño de batch | 64 | 64 | 64 | 64 | 64 | 128 |
| Relación Train/Test | ⅔-⅓ | ⅔-⅓ | ⅔-⅓ | ⅔-⅓` | ⅔-⅓ | ⅔-⅓ |
| Número de bloques residuales | No aplica | No aplica | No aplica | 1 | No aplica | No aplica |
| Lista de dilataciones | No aplica | No aplica | No aplica | 1-2-4 | No aplica | No aplica |
| Tamaño del kernel | No aplica | No aplica | No aplica | 2 | 2 | No aplica |
| Número de filtros | No aplica | No aplica | No aplica | 8 | 32 | No aplica |

FC = Fully Connected

# Métrica de evaluación seleccionada

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.

# Resultados

Es interesante notar que el dataset que generamos está formado por una gran cantidad de usuarios, donde cada usuario posee su propia rutina y características personales, que no tienen porque parecerse a la de los demás usuarios. Además, algunos usuarios poseen menos datos que otros. Debido a esto, es difícil hallar una arquitectura que funcione correctamente para todos los usuarios por las idiosincrasias de cada uno de ellos. Por este motivo, decidimos realizar un proceso de búsqueda de los hiper-parámetros que conseguían los mejores resultados para ciertos usuarios. Tres usuarios fueron seleccionados para medir la performance de las arquitecturas propuestas, ya que, realizar un análisis tan extenso (48 usuarios) no era el objetivo de la cátedra. Los usuarios seleccionados fueron elegidos específicamente debido a que tenían patrones de gasto energético diferentes. Los 3 usuarios son: 50 (bajo MET), 31 (MET medio), 4 (MET alto). Además los usuarios seleccionados poseen un diferente grado rutinario en su actividad física. 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. En Fig. 2, Fig. 3 y Fig. 4 se muestra el porcentaje del valor de MET a lo largo de los días de la semana y las horas del día de los 3 usuarios, respectivamente.

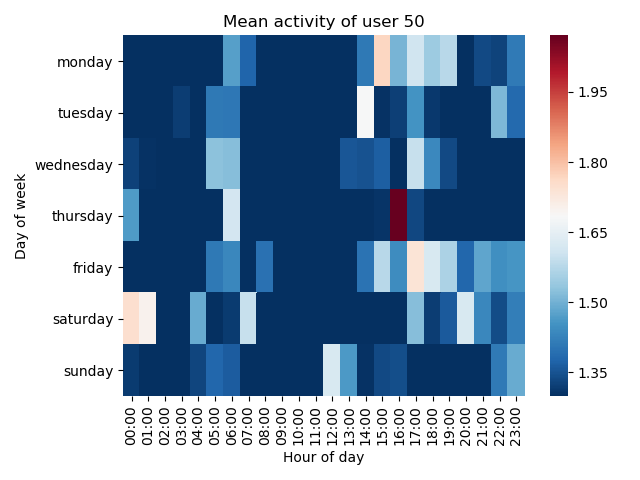


Fig. 2

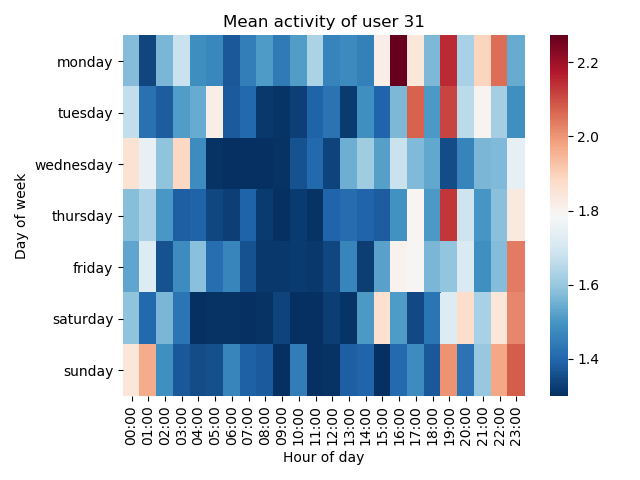


Fig. 3

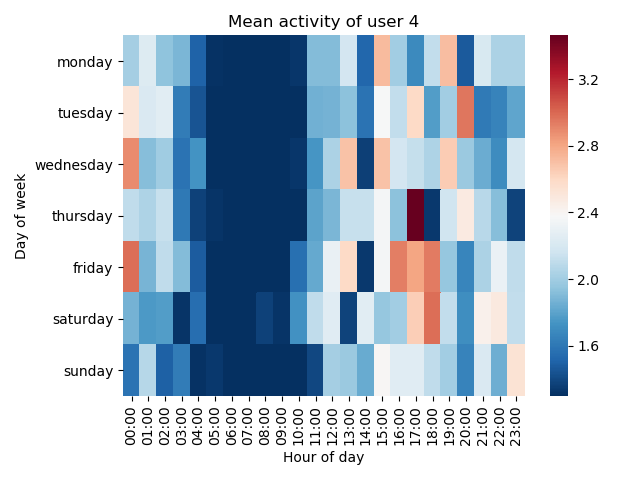


Fig. 4

A continuación, se presenta una tabla que muestra los valores de *mse* conseguidos para cada predictor. Se presenta el *mse* para entrenamiento y para el testeo. A pesar de que para evaluar la capacidad de generalización que posee un predictor debe proveerse su error prediciendo a partir de un conjunto de datos que no fue utilizado para entrenarlo, mostramos en la tabla el *mse* conseguido en el entrenamiento para poder analizar si los predictores produjeron *overfitting*.

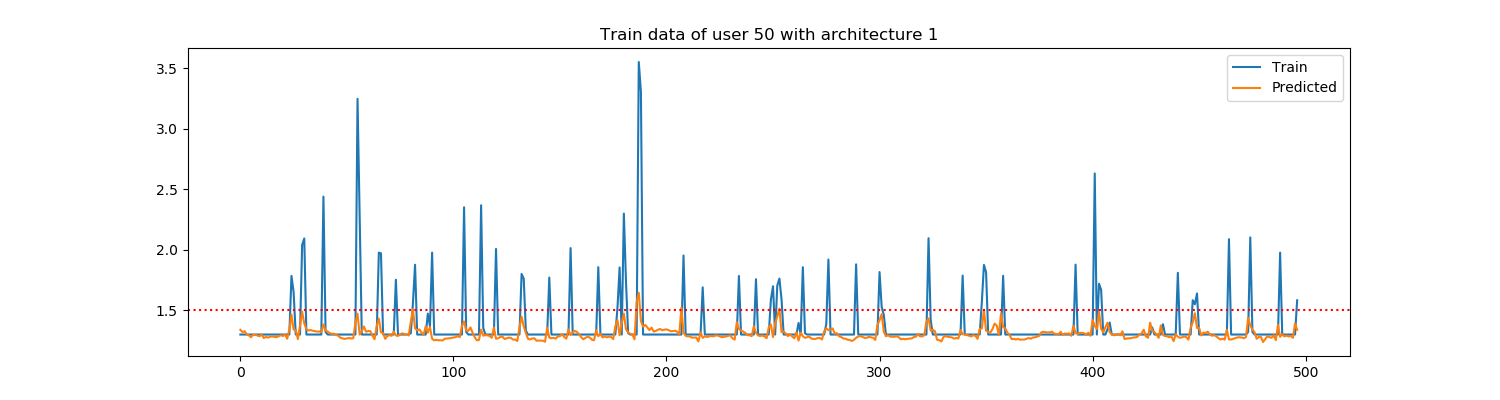
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| MSE Train/ MSE Test | Arquitectura 1 | Arquitectura 2 | Arquitectura 3 | Arquitectura 4 | Arquitectura 5 | Arquitectura 6 |
| Usuario 50 | 0.058/0.028 | 0.039/.0.04 | 0.043/0.035 | 0.05/0.028 | 0.042/0.029 | 0.034/0.047 |
| Usuario 31 | 0.143/0.309 | 0.047/0.0355 | 0.052/0.382 | 0.116/0.321 | 0.108/0.307 | 0.069/0.342 |
| Usuario 4 | 0.444/0.608 | 0.197/1.002 | 0.168/0.73 | 0.364/0.619 | 0.308/0.59 | 0.281/0.651 |

Si el entrenamiento se realiza nuevamente, los resultados de cada predictor pueden variar ya que no se consiguió que los resultados sean replicables fijando semillas mediante los generadores de números pseudo-aleatorios de TensorFlow y Numpy. Esto puede deberse a que las librarías que utiliza CudaNN para que los modelos sean entrenados aprovechando la GPU introduzcan cierto grado de aleatoriedad.

# Discusión

A continuación pasarán a discutirse los resultados obtenidos. Es interesante notar qué regresor tuvo un mejor desempeño para cada uno de los usuarios y analizar el porqué de dicho resultado.

El usuario más desafiante para los predictores resultó ser el número 50, ya que fue el menos rutinario y con menos casos de comportamiento no sedentario. Por lo tanto, resultó difícil que los NN aprendieran, a partir de las features del pasado, cuándo iba a tener un comportamiento no sedentario. Las arquitecturas 1 y 4 fueron las que mejor desempeño presentaron. Las Fig. 5 y 6 muestran las predicciones de entrenamiento y testeo para esta arquitectura.

Fig. 5

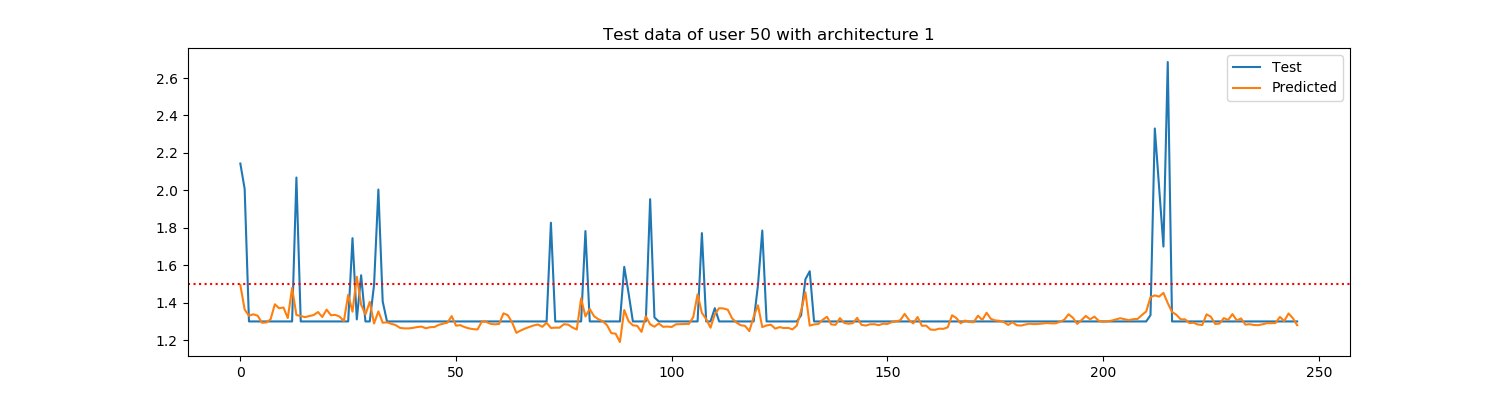


Fig. 6

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 *features* de tiempo, es entendible que los regresores se desempeñen mejor en estos casos. La arquitectura 5 (CNN) fue la que mejor performance tuvo en este caso, con un *mse* de testeo de 0.307. Las Fig. 7 y 8 muestran las predicciones de entrenamiento y testeo para esta arquitectura.

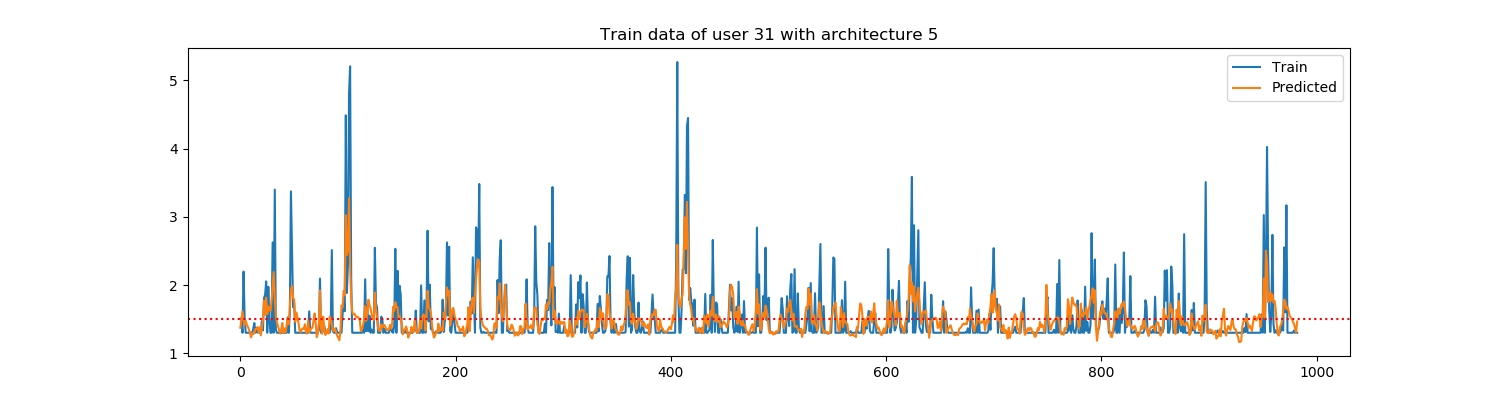


Fig. 7

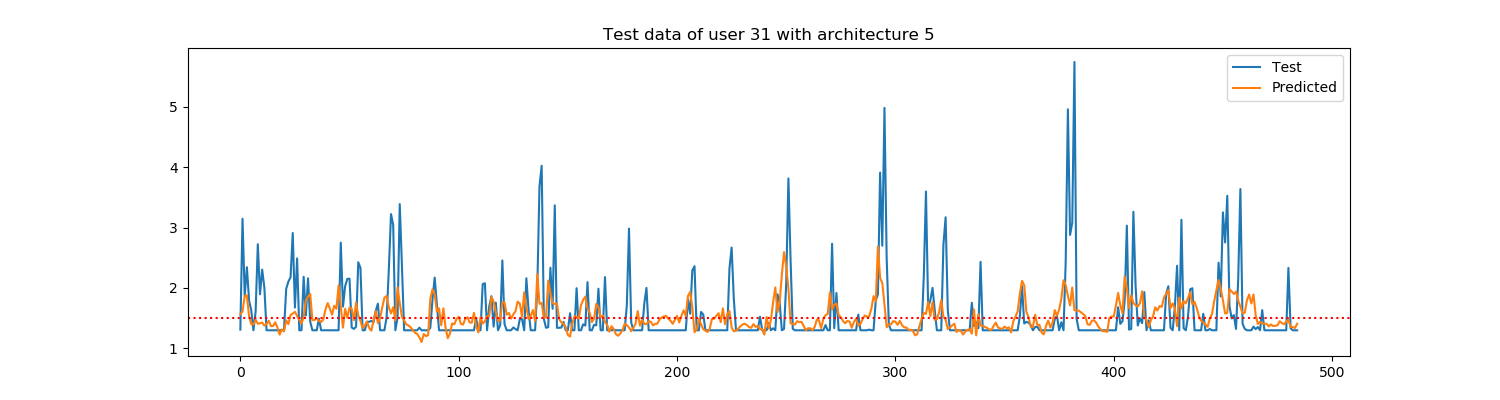


Fig. 8

Por último, los modelos sobre el usuario 4 fueron los que mejor desempeño tuvieron, a pesar de tener un mayor mse que los modelos sobre los otros usuarios. Esto se debe a que este usuario posee muchos mas 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 features 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 *mse.* La arquitectura que mejor se desempeñó en este caso fue la número 4 (TCN). Esto, y el hecho de que para todos los modelos las CNN tuvieron, en general, una mejor performance que las RNN, confirma nuestra hipótesis sobre que las CNN, y en particular las TCN, son más adecuadas para problemas de predicción que deban aprender de una secuencia de datos. Las Fig. 9 y 10 muestran las predicciones de entrenamiento y testeo para esta arquitectura.

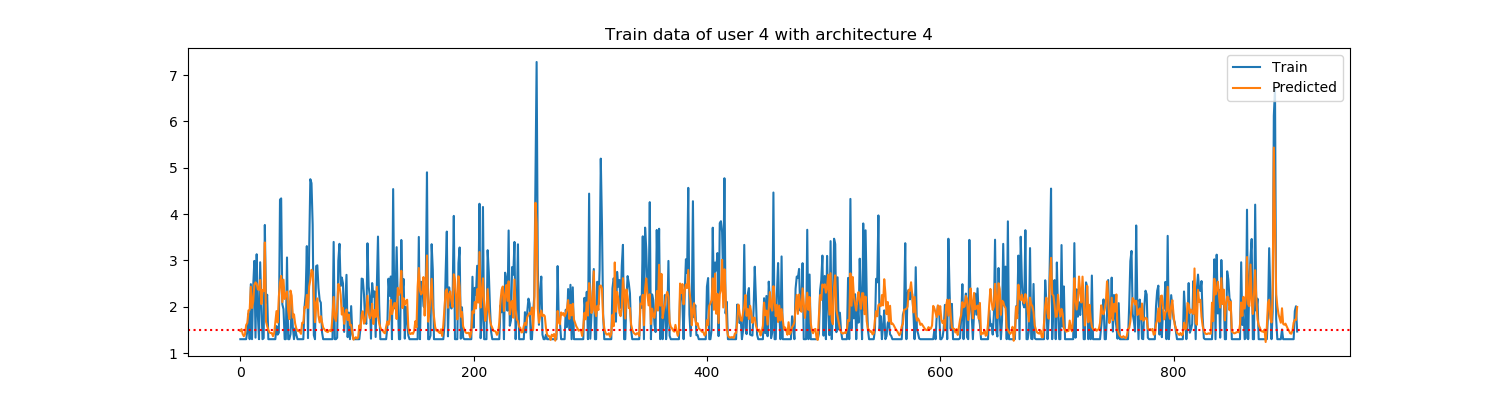


Fig. 9

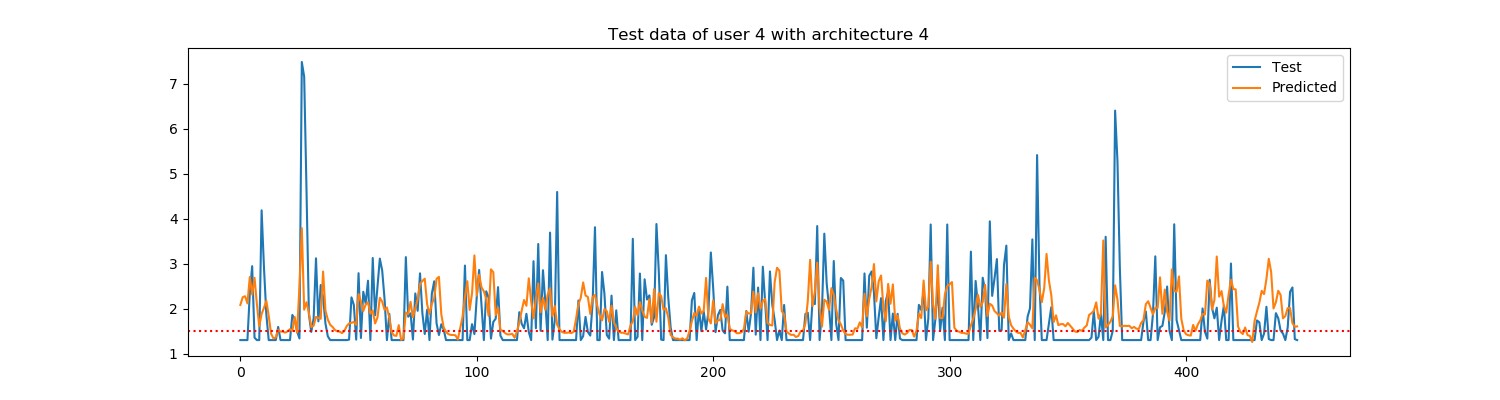


Fig. 10

# 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 propuestasy 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

[Cook, Diane J., and Narayanan C. Krishnan. 2015. *Activity Learning: Discovering, Recognizing, and Predicting Human Behavior from Sensor Data*. John Wiley & Sons.](http://paperpile.com/b/FQGDmo/DZqSa)

[Gong, Jiaqi, Yu Huang, Philip I. Chow, Karl Fua, Matthew S. Gerber, Bethany A. Teachman, and Laura E. Barnes. 2019. “Understanding Behavioral Dynamics of Social Anxiety among College Students through Smartphone Sensors.” *Information Fusion*. https://doi.org/](http://paperpile.com/b/FQGDmo/TxFr)[10.1016/j.inffus.2018.09.002](http://dx.doi.org/10.1016/j.inffus.2018.09.002)[.](http://paperpile.com/b/FQGDmo/TxFr)

[Harari, Gabriella M., Samuel D. Gosling, Rui Wang, Fanglin Chen, Zhenyu Chen, and Andrew T. Campbell. 2017. “Patterns of Behavior Change in Students over an Academic Term: A Preliminary Study of Activity and Sociability Behaviors Using Smartphone Sensing Methods.” *Computers in Human Behavior*. https://doi.org/](http://paperpile.com/b/FQGDmo/VGeQ)[10.1016/j.chb.2016.10.027](http://dx.doi.org/10.1016/j.chb.2016.10.027)[.](http://paperpile.com/b/FQGDmo/VGeQ)

[He, Qian, and Emmanuel O. Agu. 2016. “Smartphone Usage Contexts and Sensable Patterns as Predictors of Future Sedentary Behaviors.” In *2016 IEEE Healthcare Innovation Point-Of-Care Technologies Conference (HI-POCT)*. https://doi.org/](http://paperpile.com/b/FQGDmo/GSfPw)[10.1109/hic.2016.7797695](http://dx.doi.org/10.1109/hic.2016.7797695)[.](http://paperpile.com/b/FQGDmo/GSfPw)

[Kanjo, Eiman, Eman M. G. Younis, and Chee Siang Ang. 2019. “Deep Learning Analysis of Mobile Physiological, Environmental and Location Sensor Data for Emotion Detection.” *Information Fusion*. https://doi.org/](http://paperpile.com/b/FQGDmo/zeAi)[10.1016/j.inffus.2018.09.001](http://dx.doi.org/10.1016/j.inffus.2018.09.001)[.](http://paperpile.com/b/FQGDmo/zeAi)

[Pelletier, Charlotte, Geoffrey Webb, and François Petitjean. 2019. “Temporal Convolutional Neural Network for the Classification of Satellite Image Time Series.” *Remote Sensing*. https://doi.org/](http://paperpile.com/b/FQGDmo/qKiY)[10.3390/rs11050523](http://dx.doi.org/10.3390/rs11050523)[.](http://paperpile.com/b/FQGDmo/qKiY)

[Saeb, Sohrab, Mi Zhang, Christopher J. Karr, Stephen M. Schueller, Marya E. Corden, Konrad P. Kording, and David C. Mohr. 2015. “Mobile Phone Sensor Correlates of Depressive Symptom Severity in Daily-Life Behavior: An Exploratory Study.” *Journal of Medical Internet Research* 17 (7): e175.](http://paperpile.com/b/FQGDmo/TbLIy)

[Wang, Rui, Fanglin Chen, Zhenyu Chen, Tianxing Li, Gabriella Harari, Stefanie Tignor, Xia Zhou, Dror Ben-Zeev, and Andrew T. Campbell. 2014. “StudentLife.” In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing - UbiComp ’14 Adjunct*. https://doi.org/](http://paperpile.com/b/FQGDmo/eCuKI)[10.1145/2632048.2632054](http://dx.doi.org/10.1145/2632048.2632054)[.](http://paperpile.com/b/FQGDmo/eCuKI)

[Wu, Congyu, Mehdi Boukhechba, Lihua Cai, Laura E. Barnes, and Matthew S. Gerber. 2018. “Improving Momentary Stress Measurement and Prediction with Bluetooth Encounter Networks.” *Smart Health*. https://doi.org/](http://paperpile.com/b/FQGDmo/H0Qg)[10.1016/j.smhl.2018.07.017](http://dx.doi.org/10.1016/j.smhl.2018.07.017)[.](http://paperpile.com/b/FQGDmo/H0Qg)

[Zia, Jonathan, Arash Tadayon, Troy McDaniel, and Sethuraman Panchanathan. 2016. “Utilizing Neural Networks to Predict Freezing of Gait in Parkinson’s Patients.” *Proceedings of the 18th International ACM SIGACCESS Conference on Computers and Accessibility - ASSETS ’16*. https://doi.org/](http://paperpile.com/b/FQGDmo/SRc7)[10.1145/2982142.2982194](http://dx.doi.org/10.1145/2982142.2982194)[.](http://paperpile.com/b/FQGDmo/SRc7)

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. http://studentlife.cs.dartmouth.edu/ [↑](#footnote-ref-2)
4. Link al dataset: http://studentlife.cs.dartmouth.edu/ [↑](#footnote-ref-3)
5. https://sites.google.com/site/compendiumofphysicalactivities/ [↑](#footnote-ref-4)
6. https://arxiv.org/abs/1803.01271 [↑](#footnote-ref-5)