Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Clasificación de señales cerebrales utilizando técnicas de aprendizaje automático

Tesis de Licenciatura en Ciencias de la Computación

Martín Julián Caravario

Directores: Pablo Brusco y Laura Kaczer

Buenos Aires, 2018

RESUMEN

En esta tesis de licenciatura nos proponemos realizar un trabajo interdisciplinario, que consiste en aplicar técnicas de aprendizaje automático al estudio de la actividad cerebral en humanos en el procesamiento de palabras nuevas, frente a la existencia de lo que se conoce como priming morfológico. El objetivo fue investigar si los modelos predictivos entrenados a partir de datos recolectados en experimentos biológicos pueden constituir una nueva herramienta de análisis, que complementen (o incluso mejoren) las técnicas estadísticas empleadas en la actualidad. Para eso nos basamos en resultados de trabajos previos y utilizamos los datos allí obtenidos como corpus de datos para realizar diversos experimentos de aprendizaje automático. Sobre estos datos, extrajimos características de las señales cerebrales para luego experimentar construyendo modelos de clasificación para predecir condiciones de priming. Los resultados obtenidos fueron prometedores para algunos sujetos en donde obtuvimos valores cercanos a 0.7 utilizando la métrica $AUC\ ROC$ sobre datos de evaluación. En segundo lugar, abordamos el problema de construir clasificadores que utilicen instancias de distintos sujetos para luego predecir sobre un nuevo sujeto no visto durante la etapa de entrenamiento y de esta manera encontrar modelos que generalicen propiedades entre distintas personas. Los resultados muestran las dificultades encontradas ante esta nueva tarea y posibles caminos para continuar el análisis. Finalmente, dejamos a disposición un framework de análisis de datos construido sobre el lenguaje Python para futuras tareas de aprendizaje automático sobre señales cerebrales.

Palabras claves: Aprendizaje automático, Electroencefalograma, Potenciales evocados, Priming, Importancia de variables, Selección de modelos, Procesamiento de series temporales.

ABSTRACT

We propose an interdisciplinary study that consisted in applying machine learning techniques to the study of human brain activity under a morphological priming protocol. Our work relied on building data-driven predictive models using collected data from biological experiments. The main goal was to develop machine learning tools for analysis that complement the classical statistical techniques. We performed several experiments by extracting features from brain signals data and then, by creating classifiers that predict one possible priming condition. Promising results were obtained for some subjects where we reach scores around 0.7 of $AUC\ ROC$ on evaluation data. A second part of the study aimed at building inter-subject classifiers with the objective of finding patterns and properties that generalize among different subjects. Results showed several difficulties associated to this task. We presented possible solutions to these difficulties where further analysis is required. Finally, we contribute by releasing a python-based framework built to carry out new machine learning experiments with EEG data.

Keywords: Machine learning, Electroencephalogram, Evoked potentials, Priming, Feature importance, Model selection, Times series processing

AGRADECIMIENTOS

- A mis directores Pablo y Laura por la dedicación que pusieron durante todo el trabajo, por sus enseñanzas y por su buena onda y optimismo que me sirvieron para no bajar los brazos.
- A mi familia por haberme bancado siempre para que pueda terminar la carrera y por buscar siempre lo mejor para mi.
- A la UBA y a mi país por haberme dado la oportunidad de formarme en una de las mejores universidades publicas con excelentes profesores.
- A los amigos que hice en la facultad (Chino, Fede, Guido, Lucas, Bruno, Flor, Lucho, Nox, Chris, Mica, Peter, Marto, Toba, Sherman) que estuvieron en los momentos difíciles e hicieron que todas las cursadas sean más llevaderas.
- A Euge por creer siempre en mi, por apoyarme siempre y darme todo el amor y contención que necesite en los momentos más difíciles de la carrera.
- A mis amigos de toda la vida, por todas las juntadas, comidas y por acompañarme en todo momento
- Al equipo de Retargetly, que supo entender mis tiempos y me permitió trabajar y estudiar al mismo tiempo.

Índice general

1. INTRODUCCIÓN

En esta tesis de licenciatura nos proponemos realizar un trabajo interdisciplinario que consiste en aplicar técnicas de aprendizaje automático al estudio de la actividad cerebral en humanos en el procesamiento de palabras nuevas. El objetivo es investigar si los modelos predictivos entrenados a partir de datos recolectados en experimentos neurobiológicos pueden constituir una nueva herramienta de análisis, que complementen (o incluso mejoren) las técnicas estadísticas empleadas en la actualidad.

Nuestro vocabulario es, en teoría, infinito. Podemos crear nuevas palabras combinando otras existentes, en un mecanismo recursivo que permite ampliar nuestro repertorio lingüístico. Uno de los interrogantes dentro del área de las neurociencias cognitivas es entender el procesamiento de estas nuevas palabras de modo tal de determinar cómo se integran exitosamente al léxico mental. Una palabra que se integra es capaz de interactuar con otras palabras asociadas, a nivel semántico o bien morfológico. En el caso de estudio que analizaremos en esta tesis [Kaczer et al., 2015], se utilizó un protocolo de *priming* asociado a un registro electroencefalográfico (EEG). La lógica de los protocolos de *priming* reside en que el acceso a una determinada palabra (por ej., "médico") puede verse facilitado por la presentación previa de otra palabra relacionada, como "enfermera" [Meyer and Schvaneveldt, 1971]. En una primera sesión, los sujetos (hablantes nativos del holandés) aprendieron una serie de palabras compuestas novedosas (por ej., appelgezicht, "cara de manzana"). Luego, realizaron un protocolo de *priming*: cada palabra compuesta es leída en voz alta y constituye el *prime* de un dibujo *target* que debe nombrarse (por ej., el dibujo de una manzana, appel). Se realizó una segunda sesión dos días después de la primera. En ambas sesiones se determinó la latencia de producción de las palabras target y los potenciales evocados (*Event-Related Potentials*, ERPs) ante estos, para determinar en qué medida las palabras nuevas pueden constituirse efectivamente como facilitadoras o *primes*. Los resultados comportamentales indican que las palabras nuevas generan una disminución en la latencia para nombrar los dibujos, demostrando un efecto de *priming* significativamente mayor que las palabras conocidas. Por otro lado, se analizó el potencial N400 (así denominado por tratarse de una onda negativa que aparece a los 400 milisegundos luego de la aparición del estímulo), dado que ha demostrado ser un indicador de integración semántica y morfológica [Kutas and Federmeier, 2011]. Se observó un decremento en la amplitud del componente N400 para las palabras relacionadas, que es más pronunciado para las palabras nuevas (nueva > familiar > no relacionada).

La técnica de electroencefalografía asociada a los potenciales evocados (ERPs) constituye una herramienta muy poderosa para el estudio de procesos cognitivos, debido a que permite medir con gran precisión temporal la actividad sincrónica de la corteza cerebral ante la presentación de estímulos de interés. Los componentes de los ERPs pueden constituir indicadores de los aspectos temporales y los mecanismos neurales implicados en la memoria de palabras. El análisis de los potenciales evocados (ERPs) suele realizarse sobre la base del promedio de un gran número de ensayos libres de artefactos, que permiten mejorar la relación señal-ruido. Sin embargo, este abordaje tiene una serie de limitaciones. En primer lugar, se asume una consistencia de la respuesta frente al estímulo, lo cual no necesariamente ocurre. Asimismo, el promedio no toma en cuenta la variabilidad entre los ensayos debido a los procesos cognitivos bajo estudio (por ejemplo, por la habituación

del sujeto). Finalmente, en estudios de producción de lenguaje, una gran proporción de ensayos suelen ser excluidos debido a que poseen artefactos asociados al habla. Por lo tanto, es deseable contar con métodos de análisis de la señal que permitan analizar los datos ensayo por ensayo, de modo tal de mejorar el poder estadístico y evitar estimaciones sesgadas en el análisis de los ERPs. En este sentido, el abordaje de aprendizaje automático permite obtener modelos predictivos a partir del análisis de los ensayos individuales [James et al., 2013, Michalski et al., 2013].

Esta tesis tiene como objetivo principal estudiar el poder descriptivo de la aplicación de técnicas de aprendizaje automático supervisado en tarea de clasificación de señales provenientes de EEG. A diferencia del enfoque basado en promedios antes mencionado, cada ensayo será utilizado de manera individual como ejemplo etiquetado para la construcción de un clasificador automático. Además, los métodos que se proponen constan de menos etapas de preprocesamiento y preselección de datos que permitirían disminuir tanto la subjetividad como el trabajo manual requerido en problemas similares.

Como objetivo secundario, priorizaremos la construcción de una pieza de software reutilizable que permita generalizar las técnicas aplicadas a nuevos problemas de la misma índole.

En este framework, el enfoque que utilizaremos será el de estudiar y aplicar técnicas de extracción de características principales (o features) sobre señales provenientes del EEG para luego intentar predecir la condición a la cual pertenecen dichas señales. Una vez obtenidos los modelos, investigaremos qué características hicieron posibles una buena predicción y cuáles no son informativas. Además, exploraremos la cantidad necesaria de datos para realizar esta tarea.

El gran problema que surge al querer predecir la condición a la que pertenece una señal de manera aislada es la ya nombrada relación señal-ruido en datos provenientes de EEG [Luck, 2014]. Además, la extracción de features de señales temporales y múltiples canales correlacionados impone desafíos computacionales y estadísticos relacionados a qué variables contienen más información y cuáles menos en los modelos obtenidos [Meng et al., 2012]. Este problema metodológico, que aún no tiene clara solución, provoca una difícil interpretación de los resultados desde el punto de vista científico que intenta explicar los fenómenos subyacentes. En este trabajo, estudiaremos maneras de contemplar estos problemas para el caso particular de señales de EEG.

En relación con la experimentación, este trabajo está planteado en dos etapas distintivas. En primer lugar, una primera etapa que consistirá en comprender qué caracteriza a las señales de cada uno de los sujetos por separado (experimentos intrasujeto). Es decir, estudiaremos el poder de predicción de nuestras herramientas al ser entrenadas y validadas sobre datos de una misma persona. Por otra parte, una segunda etapa que consistirá en comprender las señales de los sujetos de una forma más general que en la primera. Para ello, entrenaremos modelos que incluyan información proveniente de varios sujetos e intentaremos predecir sobre señales provenientes de sujetos no vistos en la etapa de entrenamiento (experimentos intersujeto).

Diversos trabajos han mostrado formas de aplicar técnicas de aprendizaje automático sobre datos de EEG. Por ejemplo, trabajos como [Chan et al., 2011, Amini et al., 2013, Geuze et al., 2013, Meng et al., 2012, Stahl et al., 2012], muestran distintas maneras de obtener características principales útiles para diversas tareas de clasificación. En cuanto a modelos de aprendizaje automático, haremos experimentos tanto con modelos clásicos de clasificación, tales como árboles de decisión y *Random Forest* [Breiman, 2001], como mode-

los competitivos del estado del arte que últimamente demuestran su poder en competencias de aprendizaje automático, como por ejemplo *XGBoost* [Chen and Guestrin, 2016].

Sobre estos modelos y la distinta variedad de formas de extraer características, buscaremos combinaciones de parámetros óptimas para cada tarea. Además, dada la gran dimensionalidad que surge en datos provenientes de señales temporales, realizaremos experimentos con métodos de reducción de dimensionalidad tales como *PCA* y *K-best*. Finalmente, una vez obtenidos los mejores clasificadores, analizaremos la importancia que tuvo cada uno de los features a la hora de clasificar para, de esta manera, lograr replicar las conclusiones obtenidas con el análisis previo de ERPs y expandir en detalles propios de estas herramientas.

2. MATERIALES Y MÉTODOS

2.1. Corpus

El corpus utilizado para la realización de la tesis fue obtenido a partir de datos provenientes de la experimentación realizada en el trabajo [Kaczer et al., 2015].

El dataset se compone de datos de electroencefalografía (EEG) obtenidos a partir de un experimento de *priming* morfológico realizado sobre 22 sujetos de origen holandés de entre 19 y 25 años, de los cuales el 70 % fueron mujeres. Para este experimento se realizaron dos sesiones separadas por dos días de diferencia. En la primera sesión, los sujetos realizaron las etapas de aprendizaje y *priming* descriptas a continuación, mientras que en la segunda sesión solamente realizaron la tarea de *priming*. Es importante mencionar que se aleatorizaron los estímulos presentados entre una sesión y otra y que, a pesar de haber utilizado el mismo casco de EEG, la posición de cada electrodo y la conectividad del mismo podrían haber variado entre sesiones.

El experimento constó de dos etapas diferenciadas. En una primera parte (aprendizaje), se les mostró a los participantes una serie de palabras compuestas junto con su definición correspondiente. Estas palabras podían ser inventadas, es decir una nueva combinación de dos palabras existentes en su vocabulario, como por ejemplo *appelgezicht* (cara de manzana), o bien existentes, como puede ser *appelmoes* (salsa de manzana). Estas palabras que fueron leídas por los sujetos en la primera etapa son denominadas *primes*, debido a la función que tendrán en la segunda etapa del experimento.

La segunda etapa consistió en realizar la tarea de *priming*. Los sujetos debían nombrar una serie de dibujos ('targets') en voz alta mientras se mide la latencia de producción de estas palabras en conjunto con el registro de su actividad cerebral (EEG). Por cada uno de los dibujos target, los sujetos observaron previamente una palabra *prime* seguida de una serie de estímulos de relleno ('fillers') de modo tal que el sujeto no percibiera la lógica del diseño experimental.

Los *primes* podían pertenecer a alguna de las siguientes categorías: (a) estar relacionadas o no con los dibujos, o (b) podían ser nuevas o familiares. De este modo, se constituyen tres condiciones experimentales de acuerdo a la relación entre la palabra *prime* y el dibujo target: Relacionada-familiar (**condición 1**), Relacionada-nueva (**condición 2**), y No relacionada (**condición 3**). La Figura 2.1 muestra la dinámica explicada del experimento con ejemplos para cada posible condición del experimento.

En la totalidad de la tarea, cada sujeto vio tres veces cada dibujo pero siempre precedido por un *prime* diferente. Todos los *primes* utilizados fueron palabras compuestas, mientras que los *targets* fueron palabras simples. Un subconjunto de las palabras utilizadas en el experimento pueden encontrarse en la Tabla 2.1

Para nuestro trabajo, utilizamos cada uno de los ensayos del corpus. Un ensayo se corresponde con una medición de EEG de un sujeto en el momento de observar el dibujo target, acompañada con información que indica de qué condición (1, 2 o 3) se trata. Cada ensayo contiene las señales de todos los canales del casco de EEG desde los 200 milisegundos previos al estímulo (presentación del dibujo), hasta 550 milisegundos posteriores al evento. El equipo de EEG utilizado para medir a los sujetos fue el *BioSemi Active Two*, compuesto por 32 canales y las señales fueron tomadas a 256 Hz. Se utilizó un filtro de banda sobre

las señales obtenidas para quedarse únicamente con las frecuencias entre 0.1 y 30 Hz.

Dibujo	Prime Familiar	Prime Nuevo	Prime no relacionado
appel (apple)	appelmoes (apple sauce)	appelgezicht (Apple-looking face)	aardbeiveld (strawberry field)
beer (bear)	ijsbeer (polar bear)	kajakbeer (Kayak for bears)	aktetas (briefcase)
broek (trouser)	zwembroek (swimming trunks)	schrijfbroek (Pants to write)	blokfluit (recorder)
brug (bridge)	loopbrug (gangway)	brughark (Rake for bridges)	borstvoeding (breastfeeding)
kasteel (castle)	kasteelheer (lord of the castle)	glijbaankasteel (Castle with slides)	brandstof (fuel)
diamant (diamond)	diamantmijn (diamond mine)	diamantoog (Diamond eye)	briefpost (letter)
goud (gold)	bladgoud (gold leaf)	sigaargoud (Dark gold)	bushalte (bus stop)
hond (dog)	waakhond (watchdog)	hondtoon (Dog accent)	dansvloer (dancefloor)
jas (jacket)	jaszak (jacket pocket)	jasgom (Eraser for jackets)	deurklink (heck)

Tab. 2.1: Subconjunto de palabras utilizadas en el experimento. Los dibujos (target) deben nombrarse precedidos por alguno de los diversos *primes*.

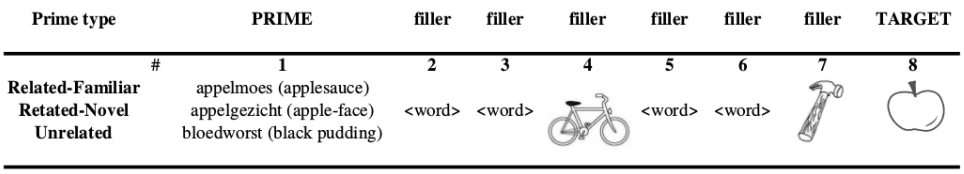


Fig. 2.1: Dinámica del experimento de *priming* morfológico.

2.2. ERPs

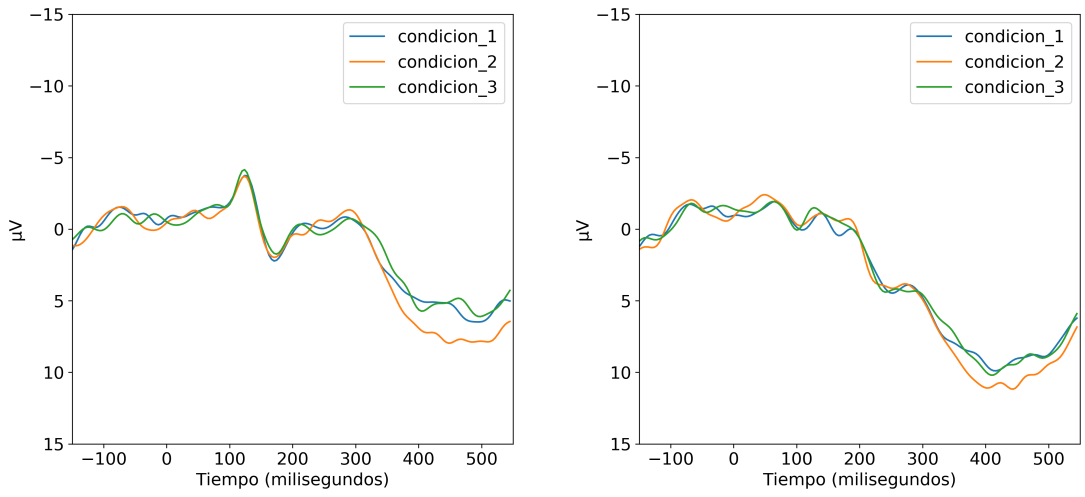
El origen de la actividad que se registra en el EEG proviene fundamentalmente de las neuronas de la corteza cerebral. El registro, realizado a nivel del cuero cabelludo, es el resultado de la suma de los potenciales post-sinápticos de un grupo de neuronas. Sin embargo, la magnitud de la actividad eléctrica percibida en el EEG (en el orden de μV) es mucho menor que la generada por una sola neurona (en el orden de mV), debido a que es filtrada y atenuada al pasar por las diferentes capas de tejido que separan la superficie cortical del electrodo de registro. Además de estos factores, la señal de EEG medida se ve afectada por diversas fuentes de actividad neuronal, lo cual dificulta la tarea de aislar procesos cognitivos individuales como los que queremos analizar. Estos factores

intermediarios hacen que la señal obtenida de la medición contenga mucho ruido y se atenúe en comparación con la señal original que buscamos analizar. A esta relación se la conoce como *relación señal-ruido* que en el caso de las mediciones de EEG suele ser baja.

La técnica de los potenciales evocados *(Event-Related Potentials, ERPs)* es utilizada para extraer las respuestas cerebrales asociadas a eventos sensoriales y cognitivos de las señales de EEG. Esta técnica consiste en realizar un promedio por canal de todos los ensayos obtenidos del EEG por cada una de las condiciones presentes según el evento presentado. El objetivo de la técnica es eliminar el ruido presente en las señales medidas por el EEG mediante el cálculo del promedio entre todas ellas. A estos promedios se los llama potenciales evocados ya que representan un potencial eléctrico de la señal relacionado a un evento en particular.

En el caso del trabajo sobre el que obtuvimos los datos [Kaczer et al., 2015], se analizó el potencial N400 en el cual se observa un pico de negatividad alrededor de los 400 milisegundos post estímulo. En la Figura 2.2 se encuentra graficado el promedio de todas las señales por condición para los canales Cz y P3. En ambas figuras, la negatividad está graficada hacia arriba en el eje y y podemos observar, a partir de los 400 milisegundos, un pico de negatividad entre los ERPs de las condiciones 1 y 3 (familiar y no relacionada) con respecto a la condición 2 (nueva). Generalmente, la separación entre condiciones no puede verse en todos los canales por lo que decidimos graficar algunos de los canales en donde esta separación está más marcada.

A continuación, presentamos una técnica alternativa a la de ERPs como herramienta para describir el efecto de las distintas condiciones sobre las señales eléctricas registradas.



(a) Canal Cz

(b) Canal P3

Fig. 2.2: ERPs

2.3. Aprendizaje automático como herramienta descriptiva

2.3.1. Introducción

En la presente tesis ponemos a prueba la utilización de **aprendizaje automático supervisado**, en particular técnicas de **clasificación binaria**, como herramienta de descripción de datos en contraste con las técnicas de promedios clásica (*ERPs*). La manera en que utilizamos estas técnicas fue mediante la construcción de modelos automáticos que aprendan a realizar la tarea específica de predecir a qué condición pertenece una señal. Una vez obtenido un modelo, lo utilizamos como intermediario para **inferir** qué patrones explican los efectos encontrados [James et al., 2013, p. 19].

A diferencia de la técnica de los ERPs, en la que se necesitan todos los ensayos del experimento para poder realizar un promedio, en nuestro caso, la tarea consistió en construir un modelo que aprenda a predecir entre dos condiciones del experimento a partir de atributos extraídos de un ensayo de EEG (como se explica en la Sección 2.3.2).

El tipo de técnicas utilizadas pertenece al área de aprendizaje supervisado debido a que el corpus de datos que utilizamos contiene la etiqueta (condición) correspondiente a cada uno de los ensayos. La tarea realizada fue de clasificación ya que, en este caso, contamos con tres clases diferentes a predecir (una por cada condición del experimento).

Una decisión importante que tomamos fue la de utilizar clasificación binaria. Es decir, construimos modelos que discriminan entre dos condiciones a la vez en lugar de construir clasificadores multiclase. Esta decisión se debe a que por una parte, simplifica el análisis del problema ya que la mayor parte de las técnicas y métricas para evaluar resultados se enfocan en problemas binarios y por otra parte, se debe a que nos interesa ver cómo se distinguen de a pares las condiciones y no simplemente construir un discriminador de condiciones.

Cabe aclarar que, pese a que las técnicas de aprendizaje automático tienen como principal objetivo predecir nuevos ejemplos no vistos por el modelo para entrenar, nuestro interés consistió en utilizar la herramienta para explicar un corpus de datos fijo. En otras palabras, intentamos entender propiedades y características de los mejores modelos obtenidos para poder así explicar y entender el problema biológico desde el punto de vista de un modelo computacional.

La experimentación en este trabajo consistió en una serie de pasos: en primer lugar, separamos el corpus en dos conjuntos: desarrollo y control. Tanto la explicación de cómo realizamos la separación como la finalidad de esta se encuentra detallada en la Sección 2.3.4. Una vez separados los datos, utilizamos el conjunto de desarrollo para entrenar modelos usando distintas configuraciones de parámetros mediante la combinación de la técnica de *grid search*, explicada en la Sección 2.3.8, con la de *cross validation* explicada en la Sección 2.3.4 en donde evaluamos nuestros modelos con la métrica *AUC ROC*, detallada en la Sección 2.3.5. Una vez construidos los modelos, analizamos cuáles fueron las variables predictoras más importantes para los mejores modelos obtenidos utilizando la métrica de importancia de features que se encuentra desarrollada en la Sección 2.3.3.

2.3.2. Extracción de features

Los modelos de clasificación que utilizamos durante la experimentación deben ser entrenados con instancias representadas por vectores numéricos. Es por esto que debemos

transformar nuestras instancias, es decir cada ensayo, en vectores de atributos representativos de estas. Es decir, dada la instancia i, extraeremos un vector:

$$\mathbf{x}^{(i)} = x_1^{(i)} \dots x_m^{(i)} \qquad (x_n^{(i)} \in \mathbb{R})$$

Estos vectores característicos o features intentan capturar la información presente en las señales de la instancia que permite diferenciarlas unas de otras en cada una de las condiciones.

Una forma posible de construir el vector numérico sería condensando todas las señales de todos los canales del ensayo en un único vector de dimensión 6144 (32 canales x 192 puntos de señal en cada canal). El problema que tiene este enfoque es que generalmente los modelos utilizados para clasificar no son lo suficientemente poderosos para aprender a diferenciar las diferentes clases con las que contamos, sobre todo por la baja cantidad de datos que tenemos. Es por esto que debemos extraer características de las señales que logren condensar la información presente en el ensayo y que pueda ser representada como un vector numérico.

A la tarea de comprimir la información de una señal se agrega la complejidad de tener que hacerlo por cada uno de los canales de EEG. En nuestro caso contamos con 32 canales, es decir que tenemos 32 señales distintas por cada instancia. Tomamos la decisión de extraer features de todos los canales del EEG. Otros enfoques posibles descartan o seleccionan un subconjunto de canales sobre los cuales extraer características para así reducir la dimensión y problemas relacionados a la correlación entre estas señales.

En la Sección 2.4 abordamos las distintas técnicas de extracción de features elegidas para este trabajo.

2.3.3. Clasificadores

Para poder aprender distintas características de cada ensayo que produzca efectos en la condición del mismo decidimos utilizar la técnica de CLASIFICACIÓN, es decir, construir clasificadores que aprendan a predecir las distintas condiciones de nuestros ensayos.

Un clasificador es un modelo estadístico construido a partir de ejemplos de entrenamiento (ensayos acompañados de sus respectivas condiciones) que una vez entrenado puede utilizarse para predecir la clase de nuevos ensayos no vistos durante la etapa de entrenamiento. Es por lo tanto esencial de los clasificadores no solo que aprendan patrones particulares de las instancias de entrenamiento, sino también que aprendan a generalizar patrones que permitan predecir etiquetas de instancias no utilizadas para entrenar.

Existen diversos algoritmos que construyen clasificadores. En este trabajo decidimos utilizar clasificadores basados en ÁRBOLES DE DECISIÓN debido al gran poder predictivo y a la transparencia que poseen para analizar los modelos a posteriori a través del cálculo de la importancia de variables.

Algoritmos de clasificación: árboles de decisión y Random Forest

Un árbol de decisión es un clasificador que utiliza reglas de decisión para separar las instancias en clases. Este modelo puede verse como un árbol binario que será utilizado como camino para tomar la decisión sobre a qué clase pertenece una instancia. En particular, cada nodo interno contiene una pregunta en función del valor de alguno de los features de la instancia (por ejemplo, ¿es el valor del feature $x_3 > 20$?) que en caso de cumplirse la condición, implica seguir recorriendo el árbol por la rama "positiva" en donde

encontraremos nodos de decisión y finalmente, nodos hojas en donde obtendremos la clase predicha para dicha instancia.

Una forma de entender el entrenamiento de este clasificador, es la siguiente: Dado un conjunto de entrenamiento, para crear un nodo es necesario elegir el feature (por ejemplo x_3) y un corte (por ejemplo > 20) que "mejor separe" las instancias entre clases según algún criterio particular. Existen diversos criterios para realizar esta separación, en nuestro caso particular decidimos utilizar ganancia de información [Michalski et al., 2013, p.57]. Una vez elegido el feature y el corte, se crean dos nuevos nodos hijos, uno para la rama positiva, otro para la negativa, se reparten las instancias del conjunto de entrenamiento según cumplan o no con la condición del nodo padre y luego se repite el proceso sobre cada uno de los nodos internos. Este proceso continúa hasta que se cumpla algun criterio de parada definido previamente, como puede ser que todas las instancias asignadas a un nodo pertenezcan a una única clase, o que el arbol haya alcanzado una profundidad determinada. Una vez cumplido el criterio de parada, al nodo hoja se le asigna la clase correspondiente a la que sea mayoría dentro del conjunto de instancias que hayan llegado hasta ese nodo.

Una vez finalizado el entrenamiento, tendremos un árbol en donde cada nodo representa una toma de decisión de acuerdo a un feature y a un corte dado, cada rama refleja la decisión elegida para ese feature, y cada hoja representa una posible clase. Al momento de recibir una instancia nueva para clasificar, se aplican las reglas definidas en el árbol sobre los features de esta, partiendo desde la raíz hasta llegar a una hoja, la cual tendrá la clase predicha para esa instancia.

Un ejemplo de un árbol de decisión entrenado en el que los features son las características de un jugador de fútbol (edad, cantidad de goles y cantidad de lesiones) y debe clasificar si este tiene que ser convocado a la Selección, se muestra en la Figura 2.3.

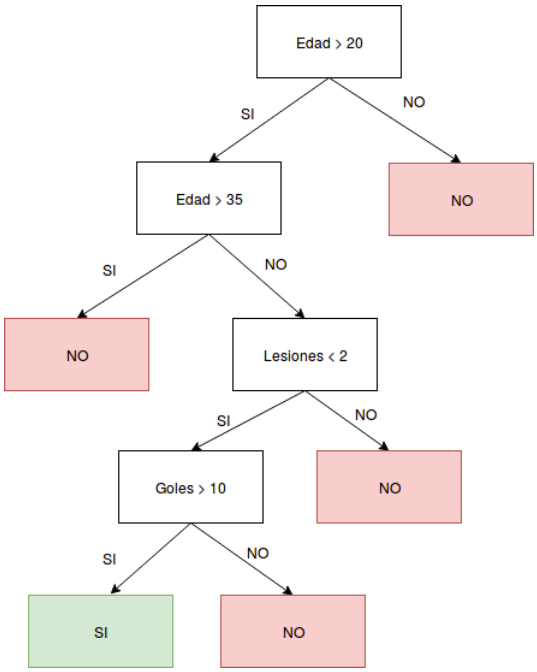


Fig. 2.3: Árbol de decisión para convocar a un delantero.

El principal problema que tienen los árboles de decisión es que son modelos con mucha

varianza. Es decir, que pequeños cambios en los datos producen modelos muy distintos. El alto grado de variabilidad lleva a que sea más probable que el modelo se ajuste a los datos sobre los que entrena para construir el árbol y no logre generalizar de forma adecuada. Para evitar este problema, suele utilizarse la técnica de *bagging (Bootstrap Agreggating)*, en la cual se parte el conjunto de datos de entrenamiento en subconjuntos construidos mediante muestreo con reemplazo (*bootstrap*). Sobre estos nuevos conjuntos de datos se entrenan modelos diferentes entre sí con la particularidad de que cada uno se adapta a los datos de manera diferente y luego se combinan para tener un único modelo. Al momento de predecir una nueva instancia se la clasifica con todos los sub-modelos generados y luego se elige la clase ganadora en base a una votación realizada sobre las predicciones generados por cada uno.

El objetivo de bagging es el de buscar reducir la varianza mediante el promedio (o combinación) de varios estimadores con alta varianza. De todas maneras, los árboles construidos no contienen la independencia necesaria para lograr este efecto. Esto se debe a que si alguno de los features es un predictor fuerte, es decir que es el que mejor separa las clases de las instancias en la mayoría de los casos, entonces siempre se ubicará cerca de la raíz, lo que generaría que todos los árboles se parezcan entre sí y por lo tanto, al promediarlos no se lograría disminuir la varianza.

Random Forest [Breiman, 2001] es un modelo que utiliza la técnica de *bagging* para construir árboles de decisión distintos entre sí y con baja varianza. La diferencia con respecto a *bagging* es que al construir cada árbol, particularmente al momento de seleccionar el "feature que mejore separe los datos", se considera solamente un subconjunto de features elegidos al azar lo que genera una mayor variabilidad entre los árboles construidos.

Decidimos utilizar el modelo de *Random Forest* como uno de los clasificadores para nuestros experimentos ya que ha probado ser competitivo en cuanto a poder predictivo y, además, tiene la posibilidad de conocer la importancia que tuvo cada uno de los features al momento de clasificar una instancia. Esto nos permite hacer un análisis para responder preguntas relacionadas sobre los canales y momentos en donde existe más información.

La importancia de features es utilizada para saber cuáles son los features que permiten separar de mejor forma las clases a predecir. En árboles de decisión, para calcular la importancia es necesario definir una métrica de impureza, es decir, una forma de medir cuánto cambia la impureza de las muestras al clasificar con un determinado feature. En nuestro caso particular, utilizamos como métrica el coeficiente de *Gini*, provisto por la implementación utilizada de los clasificadores.

La importancia de *Gini* para un feature [Breiman, 2017, Louppe et al., 2013], se calcula como el decremento de impureza de *Gini* en promedio sobre cada uno de los nodos donde el feature es utilizado.

La Ecuación 2.1 muestra cómo se calcula la importancia para un determinado feature x_i.

$$Importancia(x_i) = \frac{\sum_T \sum_{n \in T: v(s_n)=x_i} p(n) \Delta impureza(s_n)}{N_T} \tag{2.1}$$

En donde $p(n)$ es la proporción de instancias que llegan al nodo n, $v(s_n)$ es el feature utilizado en el split del nodo n, N_T es la cantidad de árboles totales y $\Delta impureza(s_n)$ es la variación de impureza en el split del nodo n, que en nuestro caso será el coeficiente de *Gini*.

2.3.4. Separación de datos para desarrollo y control de modelos

En cualquier tarea de aprendizaje automático es importante poder evaluar qué tan bien funcionará nuestro modelo sobre datos no vistos durante la etapa de entrenamiento. Es por ello que se establece una clara separación entre datos para desarrollo y datos para control. El conjunto de desarrollo es utilizado para buscar combinaciones de parámetros del modelo, mientras que el de control tiene el objetivo de validar la performance del modelo final escogido.

Inicialmente contamos con 19 sujetos de los cuales uno de ellos fue descartado ya que la longitud de las señales en sus ensayos difería de la del resto de los sujetos. Cada ensayo está compuesto por 32 canales, cada uno con 192 mediciones de la señal en el tiempo. En primer lugar, se separaron de forma aleatoria los sujetos 8, 10, 12 y 21 con el objetivo de utilizarlos para testear el clasificador intersujeto. Luego, por cada uno de los sujetos restantes se separó para testear el 10 % de sus ensayos de manera balanceada entre las tres condiciones posibles [1]. La Tabla 2.2 muestra la cantidad de instancias con las que se realizó la experimentación por cada condición junto con la cantidad utilizada en el conjunto de control. Por ejemplo, para el sujeto 7, se tiene un total de 93 instancias, de las cuales 32 pertenecen a la condición 1, 28 a la condición 2 y 33 a la condición 3. Sobre cada uno de estos conjuntos se separaron 3 instancias para utilizarlas como control. Los sujetos que fueron separados para el clasificador intersujeto (Experimento 5) no fueron incluidos en la experimentación, por lo que no se separaron sus datos.

Sujeto	**Condición 1**		**Condición 2**		**Condición 3**		**Total**
	Total	Control	Total	Control	Total	Control	
sujeto 2	21	2	26	3	26	2	73
sujeto 3	24	3	30	3	30	2	84
sujeto 5	20	3	26	2	24	2	70
sujeto 6	17	2	17	1	19	2	53
sujeto 7	32	3	28	3	33	3	93
sujeto 8	26		25		27		78
sujeto 9	31	3	31	3	29	3	91
sujeto 10	33		30		35		98
sujeto 11	32	4	36	3	35	3	103
sujeto 12	13		14		7		34
sujeto 14	30	3	35	3	33	4	98
sujeto 15	28	2	26	3	29	3	83
sujeto 18	33	3	34	3	35	4	102
sujeto 19	27	3	29	3	31	3	87
sujeto 20	34	3	33	4	31	3	98
sujeto 21	6		2		3		11
sujeto 24	28	3	22	2	25	3	75
sujeto 25	20	2	20	2	17	2	57

Tab. 2.2: Cantidad de instancias de la sesión 1.

[1] Generalmente se utiliza el 20 % de las instancias para testear, pero en este caso particular se decidió usar el 10 % de los ensayos, ya que al tener tan pocas instancias por sujeto (menos de 100), se priorizó tener un mayor número de instancias para el entrenamiento del modelo

Cross Validation

Pese a tener datos separados que fueron utilizados para estimar la performance real del mejor modelo elegido, es importante conocer valores realistas de qué tan bien están prediciendo nuestros modelos durante la etapa de desarrollo y búsqueda de parámetros. El problema surge al momento de construir un modelo sobre los datos de desarrollo ya que si entrenamos sobre todos ellos, no tendremos datos disponibles sobre los que evaluar el modelo para medir qué tan bien generaliza.

La validación cruzada o *cross validation* es una técnica que se utiliza para evaluar modelos de una manera realista en la etapa de su construcción que no necesita datos extras más allá de los destinados al entrenamiento.

Para evitar el problema de *sobre-estimación de resultados* en los datos de desarrollo y así poder construir modelos robustos y que generalicen bien, utilizamos la técnica conocida como *k fold cross validation* que consiste en separar el conjunto de desarrollo en k subconjuntos de igual tamaño y sin solapamiento llamados *folds*.

Al momento de entrenar, se realizaron k iteraciones en las que en cada una se utilizaron todos los *folds* para entrenar, menos uno sobre el cual se testeó el modelo y se obtuvo un resultado (métrica) propio de la iteración. Una vez que se hayan completado las iteraciones, es decir que se hayan utilizado todos los *folds* para testear, se combinan todos los resultados obtenidos en cada iteración, tal como se observa en la **Figura** 2.4.

Al utilizar *k fold cross validation* nos aseguramos de que en cada iteración estamos entrenando con particiones de los datos y evaluando sobre un *fold* distinto a todas estas. Además, al finalizar las iteraciones habremos utilizado cada uno de los *folds* para evaluar, lo que nos lleva a reducir la posibilidad de sub o sobre estimación de los resultados.

2.3.5. Métrica de evaluación

La métrica que se utilizó para evaluar nuestros modelos fue el $AUC\ ROC$ (área bajo la curva ROC) calculado de forma **global** entre nuestros *folds*. La curva ROC mide la relación entre el TPR (tasa de verdaderos positivos) y el FPR (tasa de falsos positivos) al clasificar las instancias del conjunto utilizando un determinado umbral de decisión. En las Ecuaciones 2.2 y 2.3 se muestra cómo se calculan el FPR y TPR respectivamente.

$$FPR = \frac{fp}{fp\ +\ tn} \tag{2.2}$$

$$TPR = \frac{tp}{tp\ +\ fn} \tag{2.3}$$

En donde *tp* representa la cantidad de positivos verdaderos; *fp*, la cantidad de falsos positivos; *fn*, la cantidad de falsos negativos y *tn*, la cantidad de negativos verdaderos.

Para construir esta curva, se debe definir una de las clases a comparar como clase positiva y la otra como clase negativa. Luego, se utiliza el clasificador para obtener puntajes relacionados a la probabilidad de pertenecer a la clase positiva de cada instancia y se las ordena de manera ascendente según este puntaje. En nuestro caso particular, el cálculo se realizó de forma global, es decir, se calculó el puntaje de cada instancia por cada *fold* de validación dentro de cada iteración de *cross validation*. Una vez que se obtuvieron y se

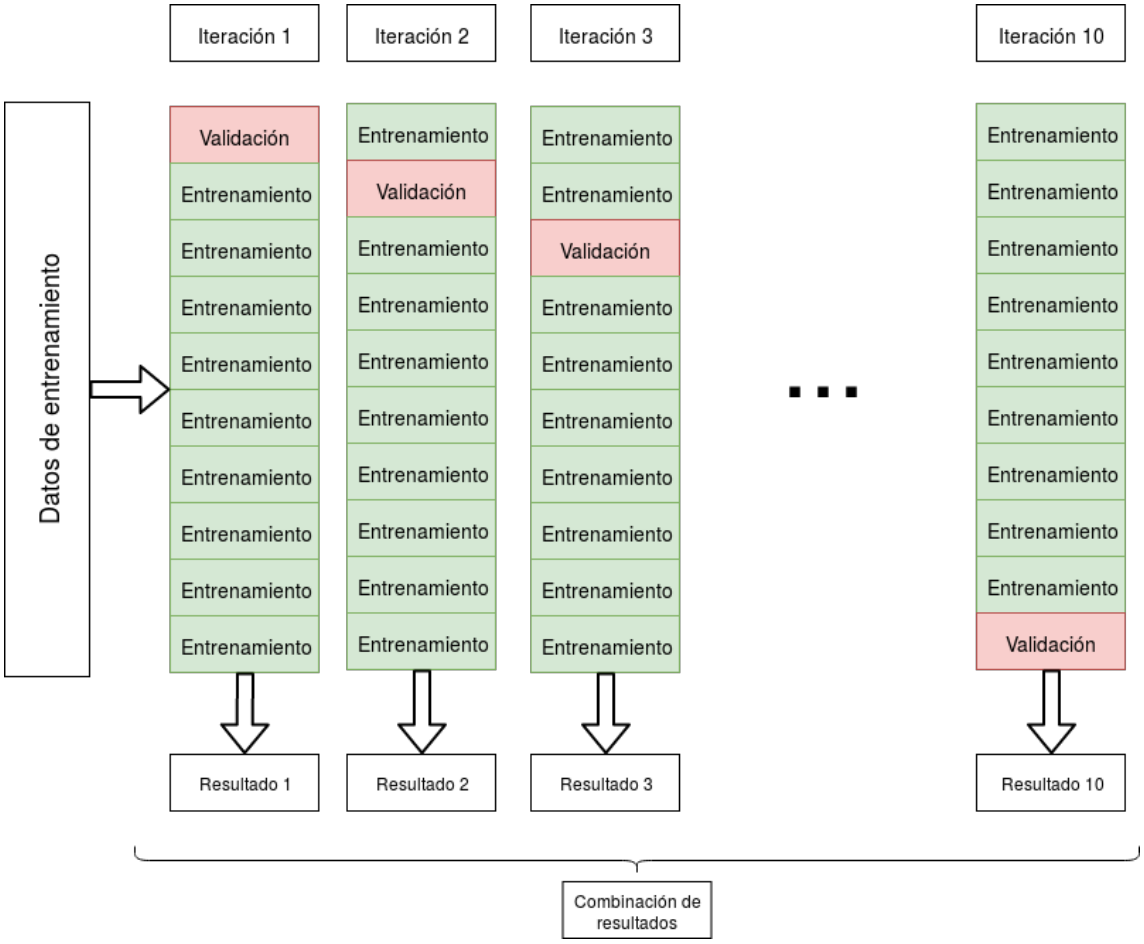


Fig. 2.4: Ejemplo de *k fold cross validation* con 10 *folds*.

ordenaron los puntajes de todos los *folds*, se varió el umbral de decisión[2] entre el mínimo y máximo puntaje y se calcularon los valores de *FPR* y *TPR* para cada umbral dado.

La curva ROC consiste en graficar el *FPR* vs. el *TPR* por cada uno de los umbrales elegidos. En este caso, utilizamos el área que se encuentra por debajo de ella como métrica de performance de nuestros modelos. El proceso de construcción de la curva ROC global puede verse representado en el Algoritmo 1.

Otra forma de calcular la performance de un modelo es calcular el *AUC ROC* por cada uno de los *folds* y hacer un promedio de estos valores. En nuestro caso, decidimos calcularlo de forma global ya que si hubiésemos utilizado la forma no global, cada uno de los *folds* hubiese quedado con muy pocas instancias de test (aproximadamente 6 instancias en promedio). Esto significaría que cada uno de los valores obtenidos por *fold* no sean lo suficientemente confiables y representativos.

2.3.6. Curvas de aprendizaje

Una manera de medir cuánto aprende un algoritmo es ver cómo varía su performance al variar la cantidad de datos sobre los cuales se entrena.

[2] El umbral que utiliza el clasificador para decidir si una instancia pertenece a la clase positiva o negativa.

Algorithm 1 Cálculo de la curva ROC Global.

procedure ROC GLOBAL(X, Y) ▷ X: Instancias, y: Etiquetas
  for $fold \leftarrow 0, 1, 2, .., 10$ **do** ▷ por cada fold del cross validation
    $indices\_train \leftarrow todos - fold$
    $indices\_val \leftarrow fold$
    $modelo \leftarrow entrenar(X_{indices\_train}, Y_{indices\_train})$
    $scores_{indices\_val} \leftarrow modelo.predecir\_scores(X_{indices\_val})$
  end for
  for $n \leftarrow 1..|scores|$ **do** ▷ Habrá tantos umbrales como scores
    $umbral \leftarrow ordenar(scores)[n]$
    $Y\_preds \leftarrow asignar\_clases(umbral, scores)$ ▷ para cada instancia i, si $score[i] >$ umbral se le asignará clase positiva si no, clase negativa
    $fpr, tpr \leftarrow medir(Y\_preds, Y)$ ▷ Predicciones vs. etiquetas
    dibujar(fpr,tpr)
  end for
end procedure

Las *curvas de aprendizaje* muestran cómo cambia la performance del modelo a medida que se agregan instancias de entrenamiento a un modelo dado. Para construirlas, se entrena un modelo con un conjunto inicial de instancias y se grafica la performance obtenida. Luego, se itera agregando sucesivamente nuevas instancias al conjunto inicial y así se obtienen nuevos resultados hasta que el conjunto con el que se entrena contiene una cantidad suficiente de instancias (en nuestro caso, todas las instancias de un sujeto). Con este tipo de gráfico se pretende observar cómo varía la performance a medida que contamos con más instancias de entrenamiento y se busca poder obtener un resultado realista de la performance del modelo.

En la experimentación realizada utilizamos la métrica *AUC ROC* global para medir la performance en cada iteración. La Figura 2.5 muestra un ejemplo de curva de aprendizaje en la que la performance mejora a medida que se agregan instancias de entrenamiento.

2.3.7. Reducción dimensional

En los casos donde se cuenta con mucha información para entrenar el modelo puede suceder que estemos complicando o entorpeciendo al clasificador para aprender a realizar la tarea específica. En nuestro caso, contamos con vectores de dimensión 672 (ver Sección 2.4) para el entrenamiento, lo que puede perjudicarnos al entrenar el clasificador con features de más. Para combatir este problema, se utilizó la técnica de reducción de dimensionalidad, que consiste en reducir la dimensión de los vectores originales a otros de menor dimensión sin perder la información contenida en los vectores originales. Una vez reducidos, se utilizan los vectores para entrenar el modelo. A continuación explicamos algunas de estas técnicas y su funcionamiento.

KBest

Esta técnica consiste en realizar un test estadístico *(F-test)* por cada uno de los features del conjunto. Este test toma un feature, agrupa sus valores según condición (1,2 o 3) y luego compara las medias de los conjuntos resultantes para analizar si hay diferencias

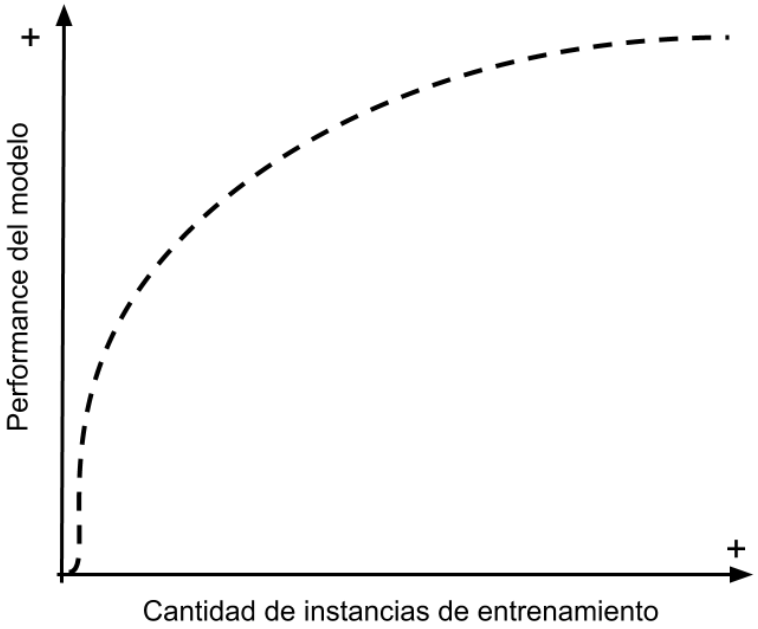


Fig. 2.5: Ejemplo de curva de aprendizaje.

significativas. Al valor obtenido de esta comparación se lo llama *F-valor* y representa cuan diferentes son las medias de un feature al agrupar sus valores por condición. Una vez finalizados los cálculos, se obtiene un *F-valor* por cada uno de los features del conjunto, de los cuales seleccionaremos los k que tengan el valor más grande. El Algoritmo 2 muestra un pseudocódigo de cómo funciona la técnica de selección de features *Kbest*.

Esta técnica es utilizada para reducir la dimensión ya que estamos reduciendo nuestro vector original a uno de tamaño k que contiene a los mejores features del vector original.

PCA

PCA (*principal component analysis*) [Bishop, 2006, p. 561] es una técnica estadística que consiste en encontrar las componentes (ejes provenientes de una rotación de los datos) que capturan mayor varianza sobre un conjunto de datos. A estas componentes se las denomina componentes principales y tienen la particularidad de ser ortogonales entre sí, de manera que la proyección de datos sobre estas componentes contiene menor correlación entre las variables.

Para encontrar estas componentes, se utiliza la técnica de descomposición en valores singulares (SVD) con el objetivo de obtener los autovectores de la matriz de covarianza de los features que, en este caso, se corresponden con las componentes principales. Cabe destacar que todas las componentes son ortogonales y que la primera componente será la que explique mayor varianza, seguida de la segunda componente y así sucesivamente.

Algorithm 2 Técnica de selección de features kbest.

procedure KBEST(X, y, k) ▷ X: matriz de features, y: vector de targets, k: cantidad de features a seleccionar
    $Fvalores = \{\}$
    for $feature \leftarrow 0, 1, 2, 3, ..., |X_{columnas}|$ **do**
        $conjunto_1, conjunto_2 \leftarrow obtener\_valores\_por\_condicion(X, feature, y)$
        $Fvalores[feature] = calcular\_f\_valor(conjunto_1, conjunto_2)$
    end for
    $kbest = obtener\_k\_mejores(k, Fvalores)$
return *kbest*
end procedure

Esta técnica puede ser utilizada para reducir la dimensión de los datos. Es decir, si encontramos las componentes principales sobre un conjunto de datos, podemos elegir las que mayor varianza expliquen y luego utilizar la proyección de los datos sobre esas componentes e ignorar el resto de las componentes.

2.3.8. Grid Search

Al momento de elegir un clasificador es necesario determinar una serie de hiperparámetros para optimizar su funcionamiento. Es por eso que debemos explorar una serie de combinaciones de estos que nos ayuden a obtener la mejor performance posible. La búsqueda de parámetros óptimos en grilla, o *grid search*, consiste en buscar la mejor combinación de parámetros para un modelo dados posibles valores para cada parámetro. Para esto se construye una grilla en la cual se define un conjunto de valores por cada uno de los parámetros del modelo, sobre los que luego se realizan todas las permutaciones posibles. Luego, por cada modelo resultante se clasifican las instancias del conjunto de desarrollo y se evalúa la métrica obtenida. Finalmente, se seleccionan de la grilla aquellos parámetros con los cuales el modelo obtuvo su mejor funcionamiento.

2.4. Técnicas de extracción de features sobre series temporales

Las técnicas de extracción de features que se explican a continuación fueron ejecutadas en cada uno de los canales del ensayo de manera independiente a partir del momento en el que el sujeto recibe el estímulo. Una vez calculados los valores extraídos para cada una de estas señales, optamos por concatenarlos todos y así obtener un único vector representativo (de dimensión 672) de la instancia que, a su vez, intenta condensar toda la información disponible en los distintos canales del ensayo.

A continuación, explicamos las técnicas de extracción de features que fueron utilizadas.

2.4.1. Extractor 1: "Promedio global"

Este método consiste en realizar un promedio de toda la señal post estímulo. De esta forma, obtuvimos un vector de 32 puntos (uno por canal), en donde cada uno de los puntos de ese vector habrá sido calculado tal como se observa en la Figura 2.6.

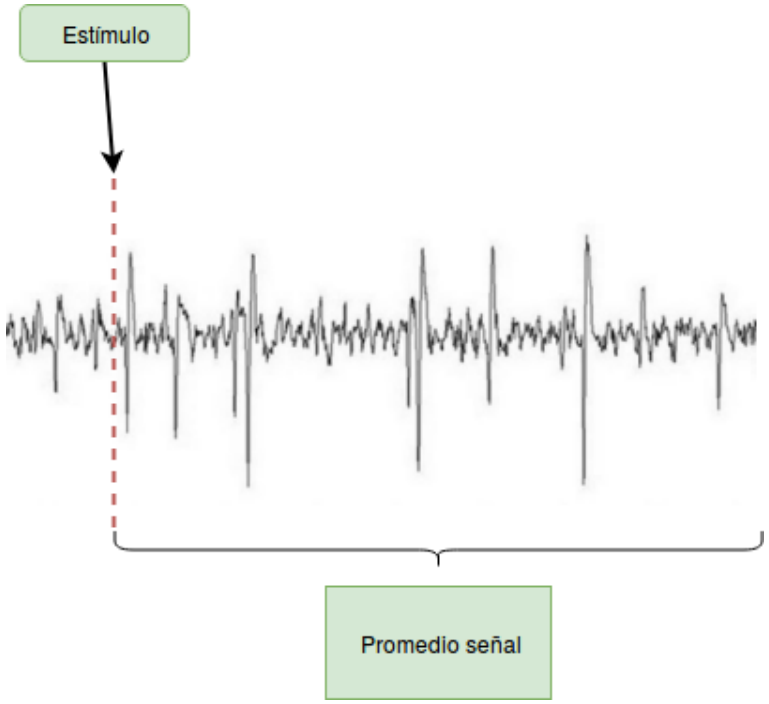


Fig. 2.6: Extractor 1: Promedio global sobre un canal.

2.4.2. Extractor 2: "Promedios sobre ventanas"

Este método consiste en tomar ventanas móviles de a 100 milisegundos sin solapamiento sobre la señal post estímulo y calcular el promedio sobre esas ventanas, tal como se muestra en la Figura 2.7. Esto nos dará 5 puntos por canal, por lo que el vector final posee una dimensión de 160. Una de las particularidades de este método es que intenta capturar cambios de amplitud en el tiempo, por lo que debería ser útil para capturar los picos de negatividad que se observan en el potencial N400. Esta última observación no es posible hacerla en el método del promedio global ya que se pierde la noción de tiempo al realizar el promedio sobre todo el canal.

2.4.3. Extractor 3: "Frecuencias"

Este método calcula la energía para cada una de las frecuencias de la señal según el método de *Welch* sobre cada canal. Una vez calculado, se realiza un promedio de las energías por banda de frecuencia. Las bandas elegidas para analizar fueron las siguientes:*theta* (4Hz - 7Hz), *alpha* (8Hz - 12Hz) y *beta* (13Hz - 30Hz). Estas bandas fueron elegidas ya que son las más analizadas en los diversos trabajos consultados, como por ejemplo en [Kepinska et al., 2017]. Cabe destacar que otra de las bandas que se suele analizar es la banda *gamma* que va de 30Hz a 100Hz. Sin embargo, en nuestro caso no resulta relevante abordarla ya que las señales sobre las que trabajamos fueron filtradas con frecuencias menores a 30Hz.

Para calcular la energía para cada frecuencia, el método de *Welch* crea una ventana móvil de tamaño fijo y calcula la energía de cada frecuencia de la señal en esa ventana. Este cálculo de energía lo realiza mediante la transformada de Fourier. Luego se va moviendo esa ventana y se repite el proceso hasta obtener N ventanas y por cada una

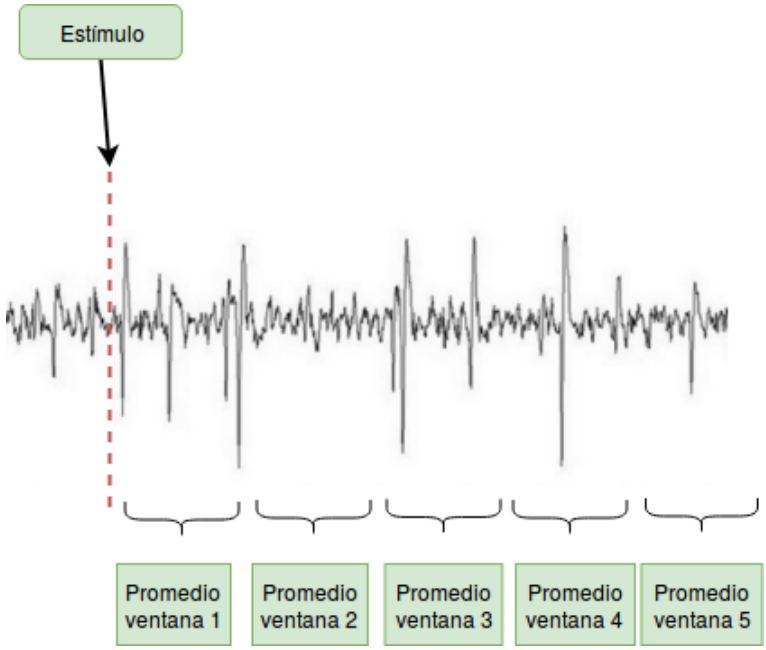


Fig. 2.7: Extractor 2: Promedio sobre ventanas en una sola señal.

de ellas 128 frecuencias junto con su respectiva energía. Finalmente, se calcula el promedio para cada una de las frecuencias obtenidas en cada ventana. El Algoritmo 3 muestra el pseudocódigo de este proceso para calcular las frecuencias sobre ventanas. La función $calcular\_energia\_alpha(S, ventana)$ se encarga de calcular la energía presente en todas las frecuencias dentro de la banda alpha, para una determinada *ventana* de la señal S. Esta forma de calcular las energías de las frecuencias presentes mediante ventanas resulta más robusta que calcular la transformada sobre toda la señal.

Este método, a diferencia de los previamente explicados, intenta capturar información sobre las frecuencias presentes en la señal dejando de lado el factor del tiempo.

Algorithm 3 Cálculo de energía por frecuencia.

procedure WELCH(S, N) ▷ S: señal, N: cantidad de ventanas a utilizar
  $energia\_alpha \leftarrow []$ ▷ banda de frecuencias de 8 a 12 Hz
  $energia\_beta \leftarrow []$ ▷ banda de frecuencias de 13 a 30 Hz
  $energia\_theta \leftarrow []$ ▷ banda de frecuencias de 4 a 7 Hz
  for $ventana \leftarrow 0, 1, 2, 3, ..., N$ **do**
    $energia\_alpha \leftarrow energia\_alpha + calcular\_energia\_alpha(S, ventana)$
    $energia\_beta \leftarrow energia\_beta + calcular\_energia\_beta(S, ventana)$
    $energia\_theta \leftarrow energia\_theta + calcular\_energia\_theta(S, ventana)$
  end for
  $promedio\_alpha \leftarrow calcular\_promedio(energia\_alpha)$
  $promedio\_beta \leftarrow calcular\_promedio(energia\_beta)$
  $promedio\_theta \leftarrow calcular\_promedio(energia\_theta)$
end procedure

2.4.4. Extractor 4: "Estadísticas de la señal"

Esta serie de métodos consiste en tomar "valores estadísticos clásicos" sobre cada uno de los canales de la señal, tal como se explica en el trabajo [Amini et al., 2013]. Las estadísticas que consideramos fueron:

- Tiempo (t_{max}) en el que aparece el máximo valor de la señal
- Máximo valor de la señal (S_{max})
- Suma de los valores positivos de la señal
- Suma de los valores negativos de la señal
- Diferencia entre el máximo y mínimo valor de la señal

2.4.5. Extractor 5: "Muestras de la señal"

Este método consiste en tomar 7 muestras de la señal separadas por un intervalo de 80 milisegundos, tal como se explica en [Chan et al., 2011]. El proceso se aplica a cada una de las señales de la instancia y se concatenan todos los puntos obtenidos en un único vector de dimensión 224. Este método, al igual que el de promedios sobre ventanas, tiene la particularidad de capturar cambios en el tiempo de una señal. La Figura 2.8 muestra el funcionamiento del extractor sobre un solo canal del ensayo.

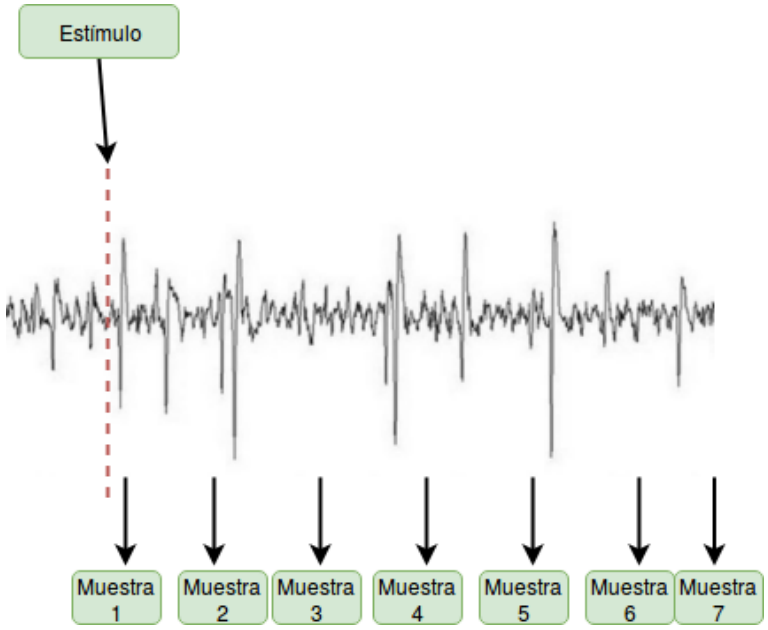


Fig. 2.8: Extractor 5: Muestras de la señal sobre una sola señal.

3. EXPERIMENTOS

A continuación, presentamos una serie de experimentos de aprendizaje automático destinados a contestar las preguntas mencionadas en la Introducción: ¿será posible que los datos con los que contamos por sujeto contengan información que permitan distinguir entre todas las condiciones tal como se observa en los ERPs? ¿Será posible que nuestros modelos aprendan con la cantidad de datos disponibles? ¿Cuáles son los canales y los momentos más importantes a la hora de discriminar cada ensayo?

Los experimentos 1, 2 y 4 buscan resolver los problemas encontrados al construir un clasificador, mientras que el experimento 3 intenta profundizar sobre el modelo hallado y, además, busca analizar los features mas relevantes al momento de clasificar una instancia. Finalmente, el experimento 5 comenta pruebas preliminares en la tarea de clasificación de instancias intersujeto, intentando armar un modelo capaz de entrenar con instancias de todos los sujetos que sea capaz de generalizar a nuevos sujetos.

Todos los experimentos que explicamos a continuación fueron realizados comparando todos los posibles pares de condiciones de *priming* explicados en la sección 2.1 (1 vs. 2, 2 vs. 3 y 1 vs. 3). Para facilitar el análisis, nos centramos en la comparación de las condiciones 2 y 3 que se corresponden con *priming nuevo* y *priming no relacionado* respectivamente.

3.1. Experimento 1: Búsqueda de un modelo de clasificación

En este primer experimento, analizamos la información contenida en los ensayos del corpus de datos. Particularmente nos enfocamos en responder la siguiente pregunta: ¿será posible que los datos con los que contamos por sujeto contengan información que permita distinguir entre todas las condiciones, tal como se observa en los ERPs?.

A diferencia de los ERPs, en los que se realiza un promedio de todos los ensayos por condición, nuestra experimentación se centra en el análisis ensayo por ensayo de cada uno de los sujetos. Utilizamos las técnicas de *aprendizaje automático* previamente explicadas para responder esta pregunta y, además, tratamos a cada ensayo como una instancia etiquetada con su respectiva condición.

El objetivo de este experimento es **construir un clasificador para cada uno de los sujetos** que aprenda a clasificar los ensayos según su condición. Para ello, debemos extraer features de algunos de los ensayos del sujeto para, finalmente, predecir condiciones sobre ensayos separados para evaluación del mismo.

En primer lugar, se decidió separar los datos en un conjunto de desarrollo y otro de control, tal como fue explicado en la Sección 2.3.4. Luego, se procedió a ejecutar el pipeline esquematizado en la Figura 3.1 en donde se muestran los pasos necesarios para entrenar un clasificador sobre los datos de desarrollo. A continuación, se seleccionaron las instancias para entrenar que en este caso fueron todas las correspondientes al conjunto de desarrollo. Luego, se extrajeron features de los ensayos utilizando todas las técnicas detalladas en la Sección 2.4. Una vez extraídos los features, se emplearon las técnicas de reducción de dimensionalidad, desarrolladas en la Sección 2.3.7, sobre estos y finalmente se entrenó un clasificador sobre los features reducidos.

Una vez que el clasificador fue entrenado, procedemos a clasificar las instancias de evaluación, tal como se presenta en la Figura 3.2.

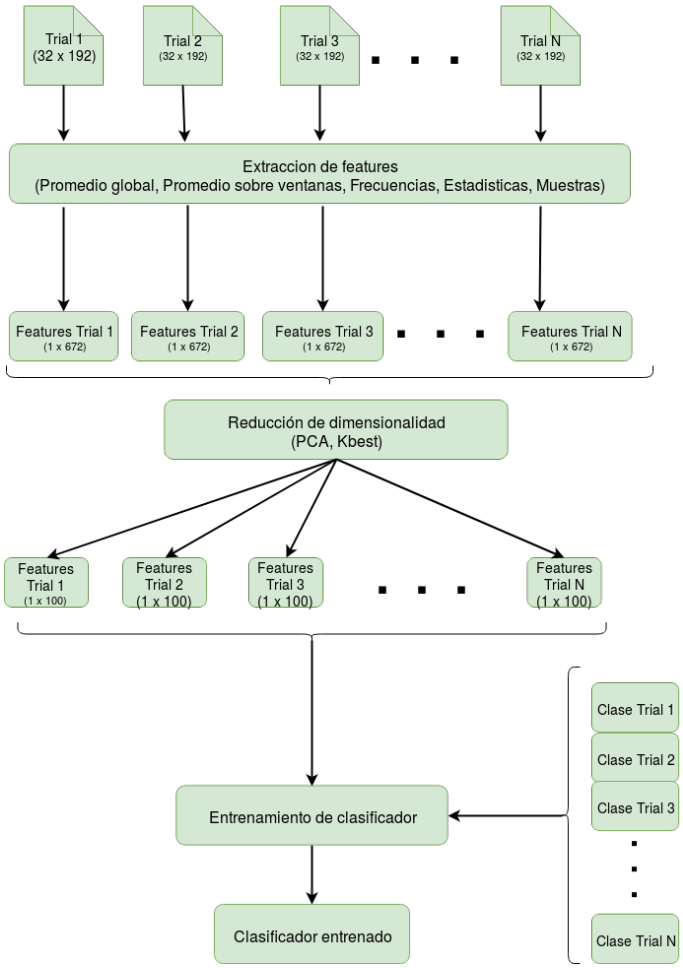


Fig. 3.1: Pipeline de aprendizaje automático para entrenar un clasificador.

Una vez extraídos los features, decidimos visualizar los puntos obtenidos a través de técnicas de reducción dimensional para analizar la posibilidad de hallar estructuras subyacentes en los datos que indiquen si una separación entre condiciones es posible. Si una proyección muestra puntos muy separados por clase, sugiere que los datos en dimensión alta contienen estructura que permite separar entre condiciones. Por otra parte, una proyección que no muestre separación no necesariamente indica que no haya estructura, sino que el método de reducción parece no estar captando la posible información subyacente.

En consecuencia, decidimos reducir la dimensión del espacio de features obtenidos a un espacio de dimensión 2 con el objetivo de poder visualizar estos datos de alta dimensión en un solo gráfico. Para ello, utilizamos la técnica de *PCA* explicada en la Sección 2.3.7 y luego analizamos las proyecciones obtenidas.

La Figura 3.3 muestra cómo puntos generados al aplicar la técnica de *promedio de ventanas* sobre cada ensayo de los sujetos 20 y 11 respectivamente son proyectados a 2 dimensiones. Estos gráficos muestran puntos coloreados según la clase del vector de features, es decir, según la condición de la instancia procesada que luego es proyectada bajo los ejes obtenidos a partir de la aplicación de la técnica.

En este caso, pudimos observar una leve separación en subconjuntos de puntos agru-

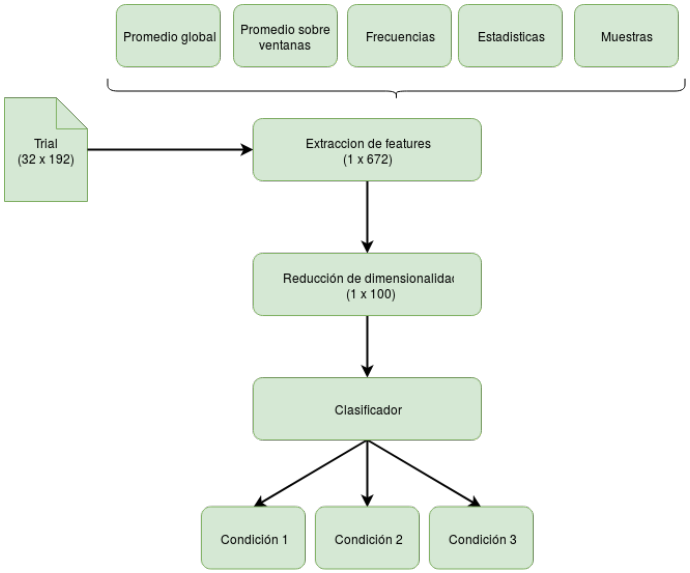
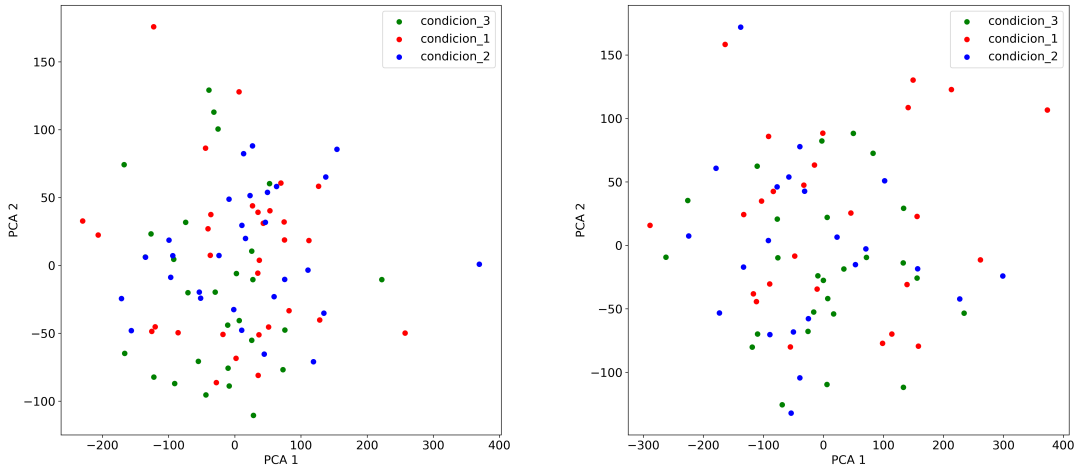


Fig. 3.2: Pipeline de aprendizaje automático para predecir una instancia.



(a) Sujeto 20 *(b)* Sujeto 15

Fig. 3.3: Visualización de features con utilizando la técnica de promedios sobre ventanas

pados por color para cada una de las condiciones. Esto nos permitió pensar que los datos posiblemente contengan una estructura en una dimensión más alta y captada a través de la técnica de promedios sobre ventanas.

Fue importante para el desarrollo de los métodos repetir este procedimiento variando las técnicas de extracción de features para tener una idea de qué métodos eran más factibles y qué métodos no presentaban características positivas en cuanto a poder capturar información en los datos.

Una vez visualizados los features, decidimos continuar con la construcción del modelo

de clasificación. Para esto debimos tomar una serie de decisiones relacionadas con las técnicas a utilizar y los parámetros del modelo a elegir. Se decidió utilizar la ya mencionada técnica de *grid search* (Sección 2.3.8), que nos permitirá iterar sobre posibles combinaciones de parámetros para nuestro modelo, con el objetivo de elegir la que mejor puntaje tenga.

En este punto, decidimos experimentar con el modelo de clasificación *Random Forest* que fue explicado en la Sección 2.3.3. Asimismo, decidimos experimentar con el modelo *XGBoost* [Chen and Guestrin, 2016] que, al igual que Random Forest, construye un conjunto de árboles de decisión (*ensemble*) de una forma diferente. Para cada modelo, experimentamos variar la cantidad de árboles a utilizar en cada clasificador, los valores fueron 100 y 200 árboles. Con respecto a los modelos de reducción de dimensionalidad, probamos *PCA* y *KBest* (Sección 2.3.7) buscando reducir los features a las dimensiones 30, 50 y 100. Además, decidimos probar los clasificadores sin realizar reducción de dimensionalidad, es decir entrenando con los 672 features originales para analizar la performance. La grilla que combina todas estas configuraciones y que fue utilizada en este experimento se puede observar en la Tabla 3.1.

Además, al momento de construir nuestros modelos decidimos utilizar la técnica de *k-fold cross validation*, explicada en la Sección 2.3.4. Al igual que la separación de datos en conjuntos de desarrollo y control, esta técnica nos permite evitar problemas de sobreestimación de la performance del modelo sobre conjuntos de validación. En nuestro caso particular, elegimos utilizar 10 *folds*, ya que es el valor comúnmente utilizado en la literatura [Meng et al., 2012, Geuze et al., 2013] para las tareas de aprendizaje automático sobre señales de EEG. La métrica utilizada para evaluar los modelos tanto en la etapa de desarrollo como en la de test fue el $AUC\ ROC$ global, tal como fue definido en la Sección 2.3.5.

Las Figuras 3.4, 3.5, 3.6 y 3.7 corresponden a la comparación de diversos parámetros entre las condiciones 2 y 3. Cada figura propone explorar cada una de las decisiones de parámetros tomadas anteriormente. En el eje x de las imágenes se encuentran cada uno de los sujetos, mientras que en el eje y se muestra el $AUC\ ROC$ en promedio y el desvío estándar que se obtuvo a partir de todas las permutaciones de parámetros para cada elección de parámetros. Con una línea punteada se marca el $AUC\ ROC$ correspondiente a 0.5 (azar).

La Figura 3.4 muestra la comparación entre los modelos *Random Forest* y *XGBoost*, explicados en la Sección 2.3.3. Como se puede observar, no pareciera haber diferencias marcadas entre ambos clasificadores, razón por la cual decidimos descartar el modelo *XGBoost* debido a que necesita una gran optimización de sus hiperparámetros para que su funcionamiento sea bueno. Por otro lado, el modelo *Random Forest* no solo suele tener un buen rendimiento sin muchas modificaciones de hiperparámetros, sino que además resulta ser un modelo más simple.

La Figura 3.5 muestra la performance de cada sujeto al comparar la cantidad de árboles utilizados al clasificar con los modelos. En principio esperábamos que cuantos más árboles utilizáramos mejor sería el puntaje obtenido. Sin embargo, según lo observado en la Figura 3.5, la cantidad de árboles a utilizar no presentó diferencias importantes entre sus dos posibles valores. Al igual que en la comparación de modelos, decidimos utilizar el modelo más simple que en este caso resultó ser el modelo de 100 árboles[1].

La Figura 3.6 muestra la comparación al utilizar distintos métodos de reducción de

[1] Es importante aclarar que por cuestiones de poder de cómputo no es posible probar grandes cantidades de configuraciones y, por lo tanto, utilizamos valores que suelen funcionar en tareas similares.

Clasificador	**# Árboles**	**Método de reducción**	**Dimensión resultante**
Random Forest	100	PCA	30
XGBoost	100	PCA	30
Random Forest	100	PCA	50
XGBoost	100	PCA	50
Random Forest	100	PCA	100
XGBoost	100	PCA	100
Random Forest	100	KBest	30
XGBoost	100	KBest	30
Random Forest	100	KBest	50
XGBoost	100	KBest	50
Random Forest	100	KBest	100
XGBoost	100	KBest	100
Random Forest	200	PCA	30
XGBoost	200	PCA	30
Random Forest	200	PCA	50
XGBoost	200	PCA	50
Random Forest	200	PCA	100
XGBoost	200	PCA	100
Random Forest	200	KBest	30
XGBoost	200	KBest	30
Random Forest	200	KBest	50
XGBoost	200	KBest	50
Random Forest	200	KBest	100
XGBoost	200	KBest	100
Random Forest	100	Sin reducción	672
XGBoost	100	Sin reducción	672
Random Forest	200	Sin reducción	672
XGBoost	200	Sin reducción	672

Tab. 3.1: Grilla utilizada para la búsqueda de hiperparámetros en el Experimento 1.

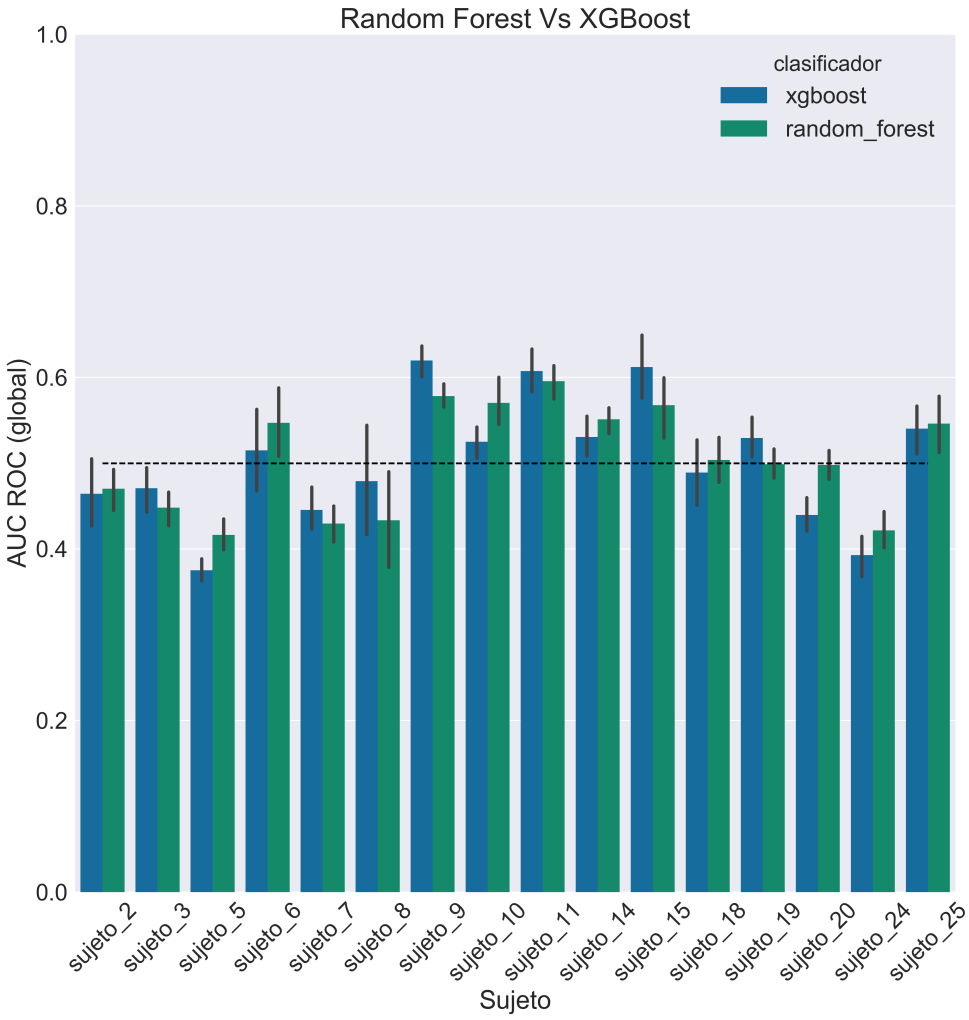


Fig. 3.4: Comparación de clasificadores.

dimensionalidad, todos ellos explicados en la Sección 2.3.7. En esta comparación esperaríamos que al aplicar alguna técnica de reducción de dimensionalidad mejore la performance del clasificador, ya que estaríamos seleccionando solamente los features más relevantes (o componentes) y descartando otros que pueden introducir ruido. Esto se corresponde con lo observado en la Figura 3.6 en donde para la mayoría de los casos se muestra que la mejor performance se alcanza aplicando *KBest*. A pesar de esto, las diferencias tampoco son lo suficientemente determinantes como para optar por alguna de las técnicas. Por otro lado, el sujeto que mejor puntaje tuvo sobre el total fue el sujeto 15 como se observa en la Figura 3.6, el cual se alcanza sin realizar reducción de dimensionalidad. Decidimos utilizar el modelo del mejor caso pues consideramos que para los sujetos en los que hay una diferencia marcada en sus señales, el modelo aprende a diferenciarlas con una performance superior a la del resto de los sujetos, como sucede en este caso con el sujeto 15.

Finalmente, la Figura 3.7 compara la cantidad de dimensiones a reducir. En este caso,

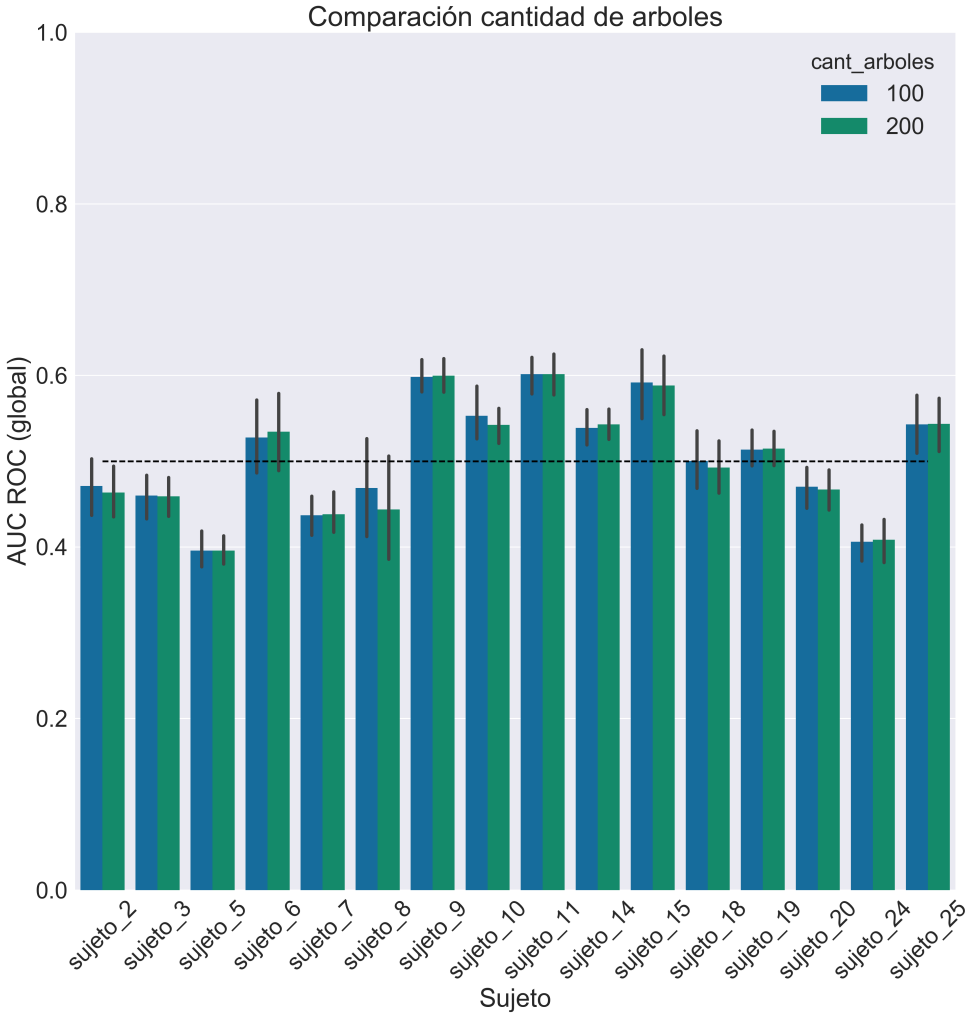


Fig. 3.5: Comparación cantidad de árboles.

observamos nuevamente que los resultados no se diferencian con claridad como para elegir cierta cantidad de dimensiones por sobre otra. Notamos que la mejor performance la obtiene el sujeto 15 al utilizar todos los features, es decir sin hacer reducción alguna. Nuevamente, considerando las figuras 3.6 y 3.7 con respecto a los mejores sujetos, tomamos la determinación de utilizar el modelo más simple en cuanto a decisiones o pre-procesamiento de datos que este requiere, que en este caso es sin reducción de dimensionalidad.

Con respecto a las condiciones restantes (1 vs. 2 y 1 vs. 3), los resultados obtenidos fueron similares a los comentados y tampoco encontramos evidencias marcadas para decidir por un modelo particular. A diferencia de lo observado en la Figura 3.6, al comparar las condiciones 1 vs. 2 (*priming familiar* vs *priming nuevo*), se obtuvo que para la mayoría de los sujetos el mejor método fue *PCA*. Utilizando la técnica de *PCA* podemos saber cuáles son las componentes principales pero, al ser cada componente una combinación lineal de los features en el espacio original, resulta complicado saber qué porcentaje de cada feature

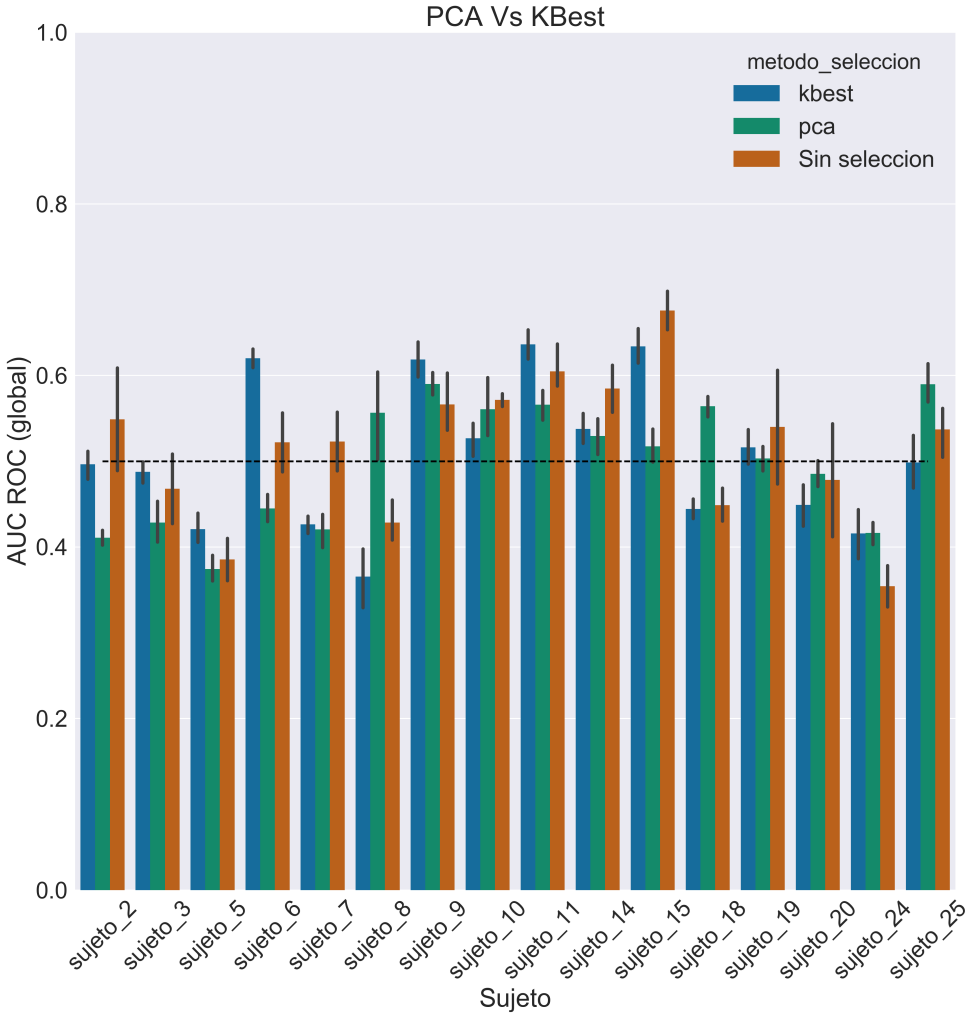


Fig. 3.6: Comparación métodos de reducción de dimensionalidad.

impacta en la importancia de features. De esta manera, decidimos descartar la idea de *PCA* por el hecho de no haber sido ampliamente superior y utilizar el mismo modelo para comparar todas las condiciones, que terminó siendo **Random Forest con 100 árboles y sin reducción de dimensionalidad**.

Resultados sobre datos de control

Luego de evaluar las distintas combinaciones, decidimos elegir un modelo que consideramos suficientemente bueno al menos en una cantidad razonable de sujetos. Este resultó ser *Random Forest* con 100 árboles y sin reducción de dimensionalidad.

Bajo esta combinación de parámetros, entrenamos un modelo sobre todo el conjunto de desarrollo que luego utilizamos para clasificar sobre el conjunto de control previamente separado. La tabla 3.2 muestra los resultados de dicha ejecución comparando todas las

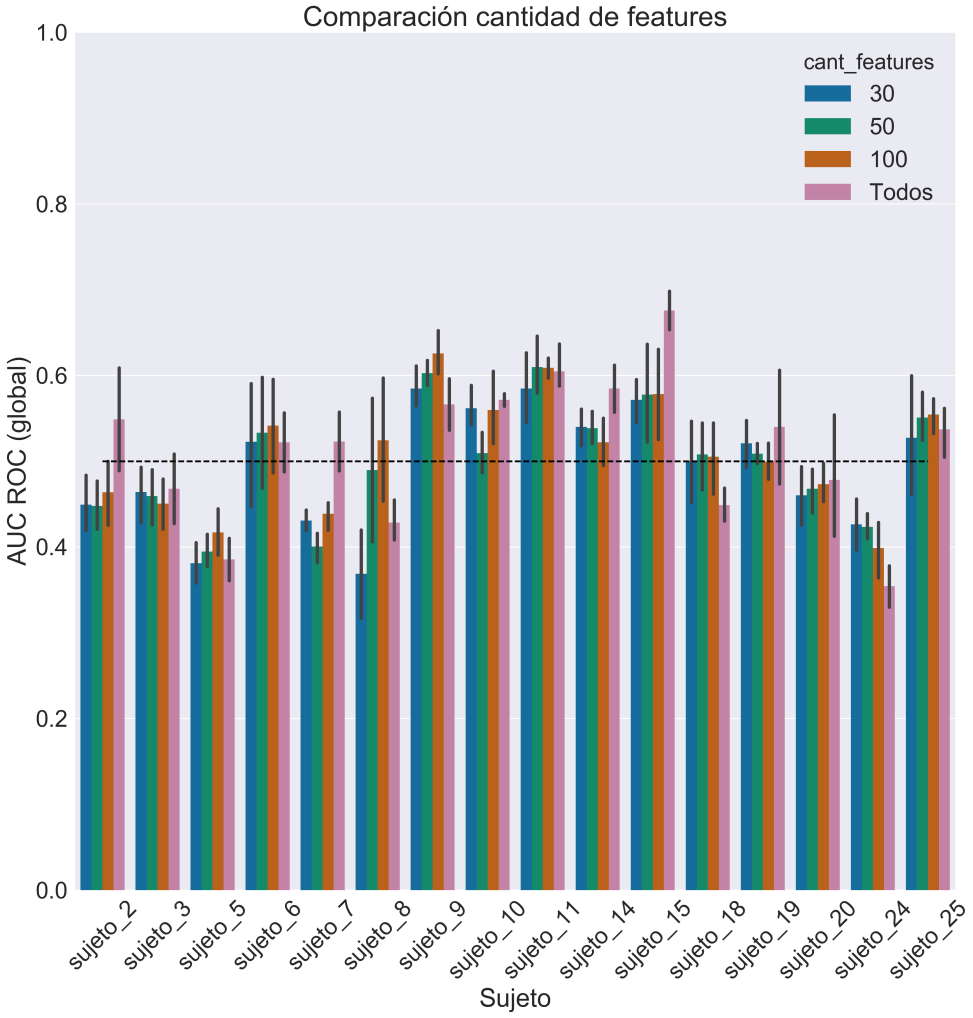


Fig. 3.7: Comparación cantidad de features.

condiciones. Como puede observarse allí, hay sujetos que poseen muy pocas instancias de test. Por ejemplo, en el caso del sujeto 6, al comparar las condiciones 1 vs. 2 y 1 vs 3, el *AUC ROC* es 1, mientras que al comparar 2 vs. 3 la métrica es 0. Esto se debe a que al haber tan pocas instancias para testear (3, 4 y 6 en el caso del sujeto mencionado) los resultados obtenidos no fueron para nada concluyentes ni confiables ya que errar la clase de alguna instancia dentro del conjunto de test tiene un gran impacto a nivel de cualquier performance que midamos sobre estos datos. De esta manera, no podemos basarnos en los resultados obtenidos para poder medir la performance de nuestro modelo debido a que no contamos con suficientes instancias de test.

	1 vs 2			2 vs 3			1 vs 3		
Sujeto	**AUC**	**Acc**	**#Test**	**AUC**	**Acc**	**#Test**	**AUC**	**Acc**	**#Test**
2	0.5	0.6	5	0.33	0.4	6	0.5	0.25	4
3	0.22	0.33	6	1.0	1.0	6	0.83	0.6	5
5	1.0	1.0	5	1.0	1.0	6	0.33	0.4	5
6	1.0	1.0	3	0.0	0.33	6	1.0	1.0	4
7	0.66	0.83	6	0.77	0.66	6	0.33	0.5	6
9	0.33	0.66	6	0.66	0.83	6	0.11	0.33	6
11	0.83	0.71	7	0.77	0.83	6	0.16	0.57	7
14	0.55	0.33	6	0.33	0.57	6	0.5	0.57	7
15	0.0	0.2	5	0.44	0.33	6	0.33	0.4	5
18	0.55	0.66	6	0.58	0.42	6	0.58	0.42	7
19	0.55	0.33	6	0.66	0.5	6	0.11	0.33	6
20	0.66	0.42	7	0.33	0.42	6	0.77	0.66	6
24	0.66	0.4	5	0.5	0.6	6	0.33	0.33	6
25	0.5	0.25	4	0.0	0.5	6	0.5	0.25	4

Tab. 3.2: Resultados del Experimento 1 sobre datos de control.

Conclusión del experimento 1

Gracias a este experimento pudimos darnos una idea de qué tan separables son las clases a través de los gráficos de features. Pudimos ver que las clases parecen estar separadas, lo cual indica que un clasificador robusto sería capaz de clasificarlas correctamente. Además, pudimos concluir que no siempre es conveniente reducir la dimensionalidad de los datos ya que en este caso particular, la diferencia entre hacerla o no, no termina siendo determinante.

Pudimos ver, también, que la decisión de separar los datos en desarrollo y control no es conveniente en este caso ya que terminan quedando muy pocas instancias para testear y se perdería robustez en las métricas calculadas. Por esta razón, en el experimento 4 abordamos una forma de medir la performance al testear sobre un conjunto de datos de test más grande que provienen de la segunda sesión del experimento original.

Con respecto a la pregunta planteada al inicio del experimento podemos afirmar que, a pesar de no contar con la cantidad de datos suficientes para construir un modelo de clasificación robusto que permita clasificar las instancias de todos los sujetos, pudimos observar resultados prometedores para algunos de ellos. Es por esto que nos planteamos una nueva pregunta sobre la que trabajamos en el experimento 2: ¿será que realmente el clasificador está aprendiendo a discriminar condiciones en ciertos sujetos o los resultados son producto del azar?

3.2. Experimento 2: Medición de aprendizaje

Ante resultados prometedores para algunos de los sujetos, cabe preguntarse si se debieron al azar o si nuestros algoritmos realmente están aprendiendo a partir de los datos.

En el siguiente experimento, analizamos qué tan robusto es un modelo construido con los parámetros elegidos en el experimento anterior y cómo aprende a medida que variamos la cantidad de instancias sobre las que se entrena. La pregunta concreta a contestar, en

este caso, será: ¿está nuestro modelo realmente aprendiendo a clasificar, o los resultados obtenidos fueron producto del azar?

Para responder esta pregunta construimos *curvas de aprendizaje*, desarrolladas en la Sección 2.3.6, utilizando todos los datos de la sesión 1, es decir sin realizar la separación en datos de desarrollo y control.

Realizamos estas curvas con una modificación particular dada la cantidad de datos con los que contamos. Decidimos realizar 10 curvas de aprendizaje para cada sujeto en donde para cada una partimos de un subconjunto inicial aleatorio compuesto por 20 instancias balanceadas entre las dos clases a comparar (10 de cada clase). Luego, agregamos de a dos instancias balanceadas al conjunto inicial hasta obtener el conjunto con todas las instancias de entrenamiento. El Algoritmo 5 muestra el pseudocódigo del método propuesto.

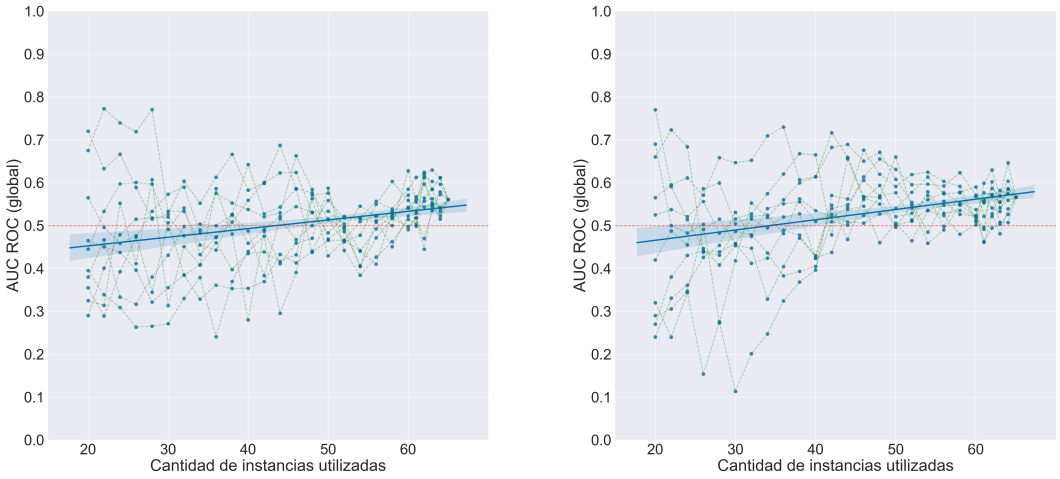
La razón por la cual se decidió realizar esta modificación fue la gran variabilidad entre nuestros datos. Cuando no es sencillo encontrar patrones en las instancias debido a la dificultad de la tarea, las curvas de aprendizaje suelen necesitar cientos de instancias para marcar una tendencia clara y converger. Ya que no contamos con dicha cantidad, decidimos generar varias versiones de las curvas de aprendizaje en donde en conjunto esperamos encontrar tendencias que permitan determinar si existe o no aprendizaje.

Algorithm 5 Modificación del algoritmo de curvas de aprendizaje

```
1: procedure MULTIPLES LEARNING CURVES(X, y)        ▷ X: Instancias, y: Etiquetas
2:     n_0 ← 20                                      ▷ tamaño inicial de cada conjunto
3:     repeat
4:         curva ← curva vacía
5:         X_0, y_0 ← elegir al azar n_0 elementos de X, y
6:         for j ← 0, 2, 4, 6, .. do                 ▷ mientras alcancen las instancias
7:             X_dev ← X_0 + tomar j elementos al azar de X \ X_0
8:             y_dev ← etiquetas de X_dev
9:             curva_{j+n_0} ← calcular_cross_val_auc(X_dev, y_dev)
10:        end for
11:        dibujar(curva)
12:    until se alcance cantidad de curvas
13:    dibujar regresión lineal entre curvas
14: end procedure
```

En la Figura 3.8a se observan caídas repentinas de las curvas al agregar nuevas instancias y, de esta manera empeora la performance haciendo que el *AUC ROC* decrezca, a pesar de estar entrenado con una mayor cantidad de instancias. Esto puede deberse a que la señal que diferencia una condición de otra tenga más de una forma si comparamos las utilizadas para entrenar y para validar. En otras palabras, podría pasar que al momento de construir la curva de aprendizaje estemos agregando instancias con patrones distintos a las que se usan para validar, llevando a que el armado del modelo use una forma de señal particular y que se equivoque al predecir sobre instancias de validación que tenían una forma diferente. Este problema se presenta cuando no tenemos instancias suficientes como para que el modelo pueda generalizar de forma correcta.

También observamos que, a pesar de tener saltos repentinos de vales de *AUC ROC* a medida que aumentamos la cantidad de instancias, las curvas del sujeto parecen mejorar aproximándose siempre a valores por encima de la línea del azar. Esto nos permite suponer



(a) Utilizando todos los features

(b) Utilizando el extractor de promedios de ventana

Fig. 3.8: Comparación entre las curvas de aprendizaje para el sujeto 10 utilizando distintas técnicas de extracción de features. Las curvas fueron generadas utilizando el Algoritmo 5 En el eje x se muestran la cantidad de instancias que se utilizaron para entrenar el modelo (X_{dev}), mientras que en el eje y se muestra el $AUC\ ROC$ global. La línea roja punteada que se encuentra en el eje y indica el valor 0.5 de $ROC\ AUC$, mientras que la línea azul muestra una regresión lineal aplicada a los puntos obtenidos de las curvas de aprendizaje junto con su desvío estándar.

que existen puntos en los que el modelo está realmente captando información que permite discriminar las instancias entre sus condiciones y que cuantas más instancias tiene, más estable parece ser su performance sobre el conjunto de validación. Además en la Figura 3.8a observamos que la pendiente de la regresión (recta azul) es ascendente y que además supera la línea del azar, lo que nos permitiría suponer que el clasificador está aprendiendo.

Estas observaciones no se cumplieron para todos los sujetos de nuestro conjunto de datos ya que en varios casos las pendientes resultaron ser decrecientes y estuvieron muy por debajo del umbral correspondiente al azar. Es por esta razón que decidimos mostrar solamente algunos de los sujetos para los que obtuvimos mejores resultados, debido a que para el resto no encontramos evidencias suficientes para suponer que hubo algún tipo de aprendizaje del modelo. En la Tabla 3.3 se muestran cuáles fueron los sujetos para los que se observó algún tipo de aprendizaje. Es decir, que todas las curvas de aprendizaje convergieron a un valor por encima de la línea del azar y, además, la pendiente de la regresión fue creciente, y cuales no tuvieron aprendizaje alguno. Como se observa en la Tabla 3.3, podemos suponer que las condiciones más separables son las 2 y 3, ya que en 6 de los 15 sujetos logramos ver aprendizaje en sus curvas.

Al obtener resultados prometedores para algunos sujetos, nos preguntamos si al simplificar la información de las instancias con las que entrenamos el modelo, es decir utilizar menos features para entrenar, podríamos continuar observando indicios de aprendizaje en los mismos sujetos. Contamos con un vector de 672 features por instancia, lo cual consi-

sujeto condición	1 vs 2	2 vs 3	1 vs 3
sujeto_18	✓		
sujeto_6			
sujeto_15	✓	✓	✓
sujeto_7			
sujeto_3			
sujeto_9		✓	
sujeto_14		✓	
sujeto_5			
sujeto_19			
sujeto_11	✓	✓	
sujeto_20		✓	
sujeto_2			
sujeto_25			
sujeto_24			✓
sujeto_10		✓	✓

Tab. 3.3: Aprendizaje para cada sujeto por condición.

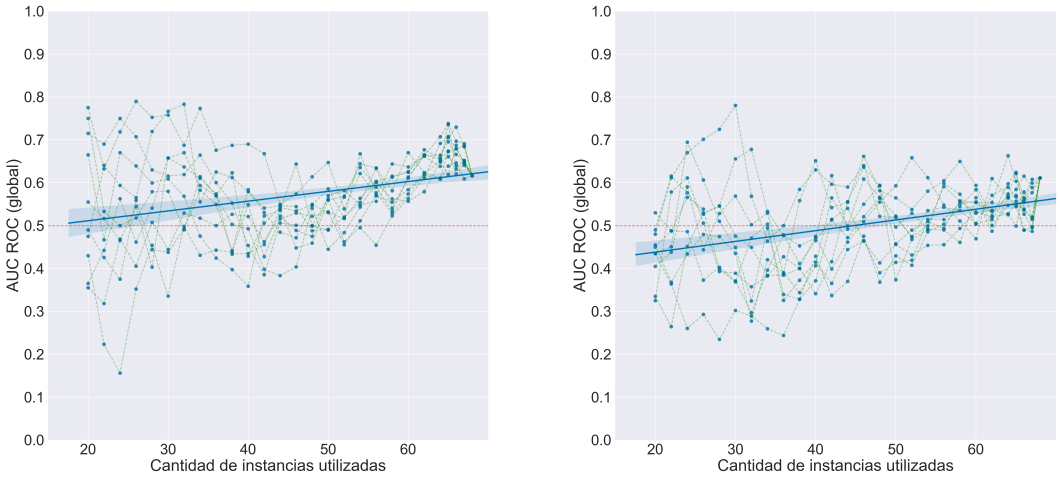
deramos que puede ser un factor que está entorpeciendo de alguna manera al clasificador. Decidimos, entonces, utilizar solamente el extractor de promedios de ventana, explicado previamente en la Sección 2.4. Tomamos esta decisión ya que el vector de features resultante tiene una dimensión de 160 por cada instancia, lo que resulta ser un valor más razonable que el anterior.

En la Figura 3.8b se muestran las nuevas curvas generadas con el extractor de promedios de ventana. Podemos observar una mejora con respecto a las curvas realizadas con todos los features ya que la interpolación de los puntos (línea azul) se encuentra por encima del umbral del 0.5 a partir de la utilización de 30 instancias, mientras que en la Figura 3.8a esto sucede a partir de las 40 instancias. Notamos que en todas las curvas la pendiente sigue siendo creciente y, a partir de cierto punto, todas superan el umbral del 0.5. Esto es un indicio de que el modelo aprende a medida que entrena con más instancias. Estos resultados permiten suponer que al entrenar solamente con el extractor de promedios sobre ventanas, el modelo continúa capturando la información necesaria para poder diferenciar las clases.

Con respecto al resto de las condiciones, también se produjo una mejora en la performance al utilizar menos features, excepto al comparar las condiciones 1 vs. 2 (*priming familiar* vs. *priming nuevo*) en donde la performance empeoró. Esto puede observarse en la comparación de la Figura 3.9 en donde notamos que la línea azul se acerca al valor del azar y que los puntajes obtenidos no son tan uniformes como al usar todos los features, sino que hay una mayor variabilidad.

Conclusión experimento 2

Concluimos que no es necesario entrenar con todos los features, sino que solamente al usar el extractor de promedios de ventana estamos capturando la información necesaria para que el modelo pueda aprender. También notamos que, en este caso, no siempre tener una mayor cantidad de instancias implica tener una mejor performance del algoritmo.



(a) Utilizando todos los features

(b) Utilizando el extractor de promedios de ventana

Fig. 3.9: Comparación entre las curvas de aprendizaje para el sujeto 11 utilizando distintas técnicas de extracción de features.

Como discutimos anteriormente, esto puede deberse a instancias que contienen patrones distintos a los que se encuentran en el conjunto de validación y, al agregarlas, producen que la performance decaiga o simplemente que la tarea sea demasiado difícil para la cantidad de datos con los que contamos. Asimismo, hay sujetos que son más fáciles de clasificar que otros en términos de performance del algoritmo de clasificación, lo que puede verse reflejado en las curvas de aprendizaje que superan el umbral del 0.5 y a las tendencias que muestran sus curvas de aprendizaje. A priori, no hay una única explicación para este hecho. Sin embargo, podemos suponer que los features que estamos extrayendo para los sujetos en los que vemos aprendizaje están logrando capturar las diferencias que separan cada una de las condiciones. Otra posible explicación sería que las condiciones de los sujetos para los que el modelo logra aprender están más diferenciadas en las señales medidas y contienen una menor cantidad de ruido en comparación con el resto de los sujetos.

Con respecto a la pregunta formulada al inicio del experimento, podemos concluir que obtuvimos indicios suficientes como para suponer que nuestros modelos están aprendiendo en muchos casos al basarnos en las curvas de aprendizaje obtenidas. Si bien el aprendizaje no fue observado con claridad en todos los sujetos, pudimos notar que algunos de ellos tienden a mejorar su performance con una mayor cantidad de instancias. Por lo tanto, concluimos que necesitaríamos más instancias sobre el resto de los sujetos para determinar si la falta de precisión de nuestro algoritmo se debe al modelo elegido, a las características del sujeto, o a la cantidad de datos.

3.3. Experimento 3: Importancia de features

Los resultados del experimento anterior son prometedores ya que hay indicios de que el clasificador está realmente aprendiendo a medida que entrena con más instancias en

algunos sujetos. Por lo tanto, a continuación quisimos averiguar qué es lo que está teniendo en cuenta a la hora de clasificar las instancias para distinguir entre las distintas condiciones. Particularmente intentamos responder la pregunta: ¿cuáles son los canales y los momentos más importantes a la hora de clasificar una instancia?

Para eso utilizamos la importancia de features calculada por cada modelo y analizamos cuáles fueron los features que más relevancia tuvieron al momento de distinguir las condiciones en cada momento de tiempo. La importancia de features es una métrica provista por el modelo *Random Forest* que indica la relevancia de cada uno de los features al momento de clasificar una instancia. La manera en la que se calcula esta importancia ha sido explicada en la Sección 2.3.3. En nuestro caso, consideramos modelos que solo utilizan los features correspondientes al extractor de promedios de ventanas para poder analizar momentos y canales más relevantes del clasificador. El cálculo es realizado sobre los sujetos cuyas curvas de aprendizaje realizadas con el extractor de promedios de ventana fueron prometedoras. Decimos que un resultado es prometedor cuando la linea azul de la regresión es creciente y termina por encima del umbral del azar (linea punteada roja). Esto nos da una idea de cuáles son las ventanas temporales más importantes para el modelo y, de esta manera, podemos analizar si se corresponden con los resultados de los ERPs o no.

En la Figura 3.10 se muestra la importancia de features a través del tiempo para un clasificador construido con los datos del sujeto 10 para distinguir entre las condiciones 2 y 3 del experimento. En esta figura, las ventanas con mayor importancia se encuentran en el intervalo de los 456 a 550 milisegundos. Este resultado se corresponde con el potencial N400 observado en los ERPs, en donde se observaban picos de negatividad en la señal a partir de los 400 milisegundos. Estos parecen ser capturados por la ventana que promedia los valores entre los 450 y 550 milisegundos, ya que resulta ser el feature más importante a la hora de separar las condiciones.

La Figura 3.11 indica cuáles fueron los canales mas importantes en cada ventana temporal. Mayor intensidad de rojo significa que el canal ubicado en esa posición de la cabeza tuvo una mayor importancia frente al resto. De esta manera, los canales más importantes para el modelo fueron Fz,P3,PO3 y Pz. Todos ellos pertenecen a la zona central izquierda del casco, tal como se observa en la Figura 3.11. Estos canales se corresponden con aquellos donde mayor diferencia se obtuvo en los ERPs. Estos resultados son congruentes con la bibliografía sobre neurobiología del lenguaje [Pujol et al., 1999], donde se muestra que en el cerebro las funciones del habla y procesamiento del lenguaje podrían estar ubicadas en las regiones laterales izquierdas.

Este análisis acerca de los momentos más importantes también fue realizado para el resto de los sujetos que habían sido prometedores en el experimento anterior. Para los sujetos restantes, notamos que los momentos más importantes se encuentran en las ventanas correspondientes al intervalo entre 0 y 224 milisegundos (primeras dos ventanas temporales). Estos resultados no se corresponden con el potencial N400 ya que las diferencias las vemos en las primeras ventanas y no en aquellas correspondientes a los 400 milisegundos. Pese a no haber sido explorado en [Kaczer et al., 2015], resulta de interés explorar en el futuro estas diferencias tempranas entre condiciones.

Además, es importante destacar que al contar con features que están muy correlacionados entre sí, como es nuestro caso debido a la relación temporal y espacial que existe entre ellos, los resultados obtenidos no resultan completamente confiables. Esto se debe a que la importancia de variables similares podría diluirse incluso por debajo del nivel de importancia de features que solo contienen ruido como se muestra en [Meng et al., 2012].

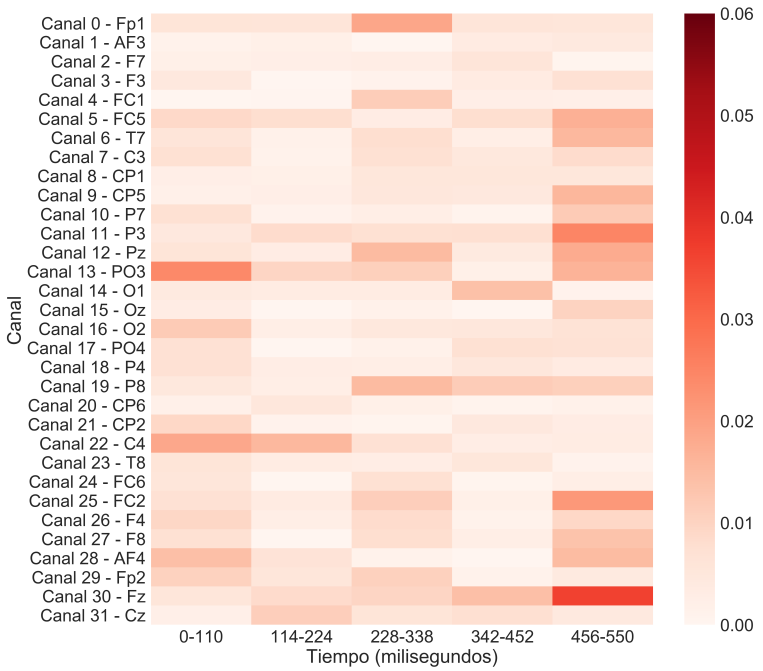


Fig. 3.10: Importancia de features a través del tiempo para el sujeto 10 comparando las condiciones 2 y 3. En el eje x se encuentran los valores en milisegundos de las ventanas tomadas, mientras que en el eje y se muestran todos los canales del casco utilizado. Cada celda de la matriz contiene el valor de la importancia que se obtuvo para dicha ventana de tiempo en ese canal. Esta importancia tiene un valor entre 0 y 1 y, cuanto más cercano a 1 es el valor, el color de rojo en el gráfico se vuelve más intenso.

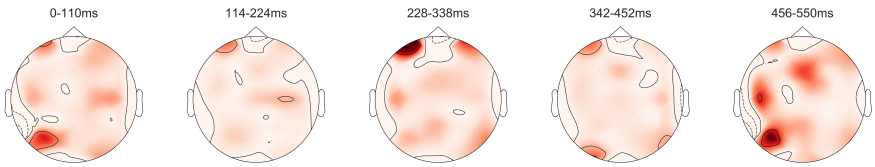


Fig. 3.11: Importancia de features por canal a través del tiempo para el sujeto 10, comparando las condiciones 2 y 3. El valor de importancia tiene un valor entre 0 y 1 y, cuanto más cercano a 1, el color de rojo en el gráfico se vuelve más intenso.

Conclusión experimento 3

En este experimento observamos que los features que capturan de mejor manera la información necesaria para separar las condiciones para un modelo que parece estar apren-

diendo se corresponden con la última ventana tomada en cada canal, es decir entre los 456 y 550 milisegundos. Este resultado se corresponde con el potencial N400 en donde se observa un pico de negatividad a partir de los 400 milisegundos post estímulo. También notamos que los canales más importantes se encontraban en la zona central-izquierda (Fz, P3, PO3), lo cual se corresponde con los canales en los que mejor se diferencia la separación entre condiciones (Sección 2.2). Si bien estos resultados tienen su correspondencia con los ERPs obtenidos en el trabajo [Kaczer et al., 2015], los indicios provistos por la importancia de features en variables que pueden contener un grado alto de correlación no constituyen información robusta y confiable, tal como mencionamos anteriormente. Es por eso que dejamos como trabajo a futuro reproducir la medición de importancia de features con alguna técnica que sea robusta y estable ante features correlacionados con el objetivo de aportar evidencia a las conclusiones antes mencionadas.

3.4. Experimento 4: Validación del modelo sobre datos de la sesión 2

Tal como fue desarrollado al final del Experimento 1, los datos que habíamos separado para testear el modelo no fueron suficientes para poder obtener métricas robustas y analizar los resultados obtenidos. Es por este motivo que para este experimento decidimos utilizar un nuevo conjunto de control compuesto por más datos.

De esta manera, en nuestro conjunto de desarrollo ahora incluimos todas las instancias de la sesión 1, mientras que el conjunto de evaluación estuvo compuesto por los datos de la sesión 2, que fueron detallados en la Sección 2.1. Al usar este nuevo conjunto de datos, nos aseguramos tener una mayor cantidad de instancias sobre las que validamos el modelo, permitiéndonos obtener métricas mas robustas y confiables para analizar. Cabe destacar que a pesar de tratarse de un experimento intrasujeto, las condiciones en las que se posicionó el casco, el estado en el que se encuentra el sujeto, la experiencia de haber completado la sesión 1 una semana antes y otros factores pueden haber variado drásticamente los patrones que distinguen a cada condición.

La cantidad de instancias sobre las que testeamos a cada sujeto se encuentran indicadas en la Tabla 3.4

Nuevamente intentamos responder la pregunta planteada en el Experimento 1 y realizamos las curvas de aprendizaje utilizando los features de promedios sobre ventanas. Los parámetros seleccionados para la construcción del modelo a utilizar fueron los obtenidos en el Experimento 1, es decir, *Random Forest* con 100 árboles y sin reducción de dimensionalidad.

Construimos las curvas de aprendizaje de la misma manera que en el Experimento 2, tal como se explicó en el Algoritmo 5, con la particularidad de que al momento de calcular el *AUC ROC* en cada iteración no lo hicimos utilizando la técnica de *k fold cross validation*. En su lugar, entrenamos el modelo con todas las instancias propias de la iteración (en el algoritmo se ve reflejado en la variable X_dev) y luego lo evaluamos sobre todas las instancias del sujeto en la sesión 2. Finalmente, ordenamos las instancias de esta segunda sesión según el puntaje obtenido al clasificarlas con el modelo y variamos el umbral de detección para calcular el *AUC ROC*.

Las curvas fueron calculadas sobre los sujetos con los que se obtuvieron mejores resultados en el Experimento 2 utilizando solamente el extractor de promedios sobre ventanas. En la Figura 3.12 pudimos notar una mejora de performance al compararla con las obtenidas en el Experimento 2 (Figura 3.8b), en donde la pendiente de la regresión (línea azul)

Sujeto	Condición 1	Condición 2	Condición 3	Total
2	25	24	30	79
3	33	36	35	104
5	25	27	30	82
6	23	24	26	73
7	34	36	36	106
8	31	32	30	93
9	31	29	36	96
10	32	30	36	98
11	30	31	25	86
12	17	19	21	57
14	12	11	14	37
15	28	29	32	89
18	16	18	24	58
19	29	30	30	89
20	25	29	25	79
21	31	28	31	90
24	25	25	25	75
25	19	19	20	58

Tab. 3.4: Cantidad de instancias de la sesión 2.

es siempre creciente y los valores de $AUC\ ROC$ obtenidos son cercanos al 0.7. Además de esto, la linea azul (regresión) supera la línea del azar con un margen considerablemente mayor al obtenido en la Figura 3.8b. Finalmente, pudimos observar que sigue habiendo instancias que al agregarlas para entrenar empeoran la performance al igual que sucedía en las curvas del Experimento 2.

Conclusión experimento 4

Con este experimento concluimos que el modelo utilizado está logrando aprender a clasificar las instancias de la sesión 2, habiendo entrenado previamente sobre la sesión 1. Lo importante a considerar de este resultado es que los conjuntos de entrenamiento y test, a diferencia del Experimento 1, fueron obtenidos bajo condiciones diferentes (días distintos, distintas posiciones de electrodos en el cuero cabelludo, entre otros factores) y aun así pudimos observar un aprendizaje a medida que se agregaban instancias.

También notamos que la cantidad de instancias con las que se disponen son un factor determinante al momento de entrenar un modelo. Esto se debe a que, como ya vimos en las curvas de aprendizaje, la performance del modelo mostró una mejora considerable con respecto a las obtenidas en el Experimento 2, donde contábamos con menos instancias de test. Asimismo, comprobamos que hay instancias que al agregarlas empeoran al modelo sin importar la cantidad de instancias con las que contemos. En principio no conocemos por qué empeoran la performance, pero intuimos que la causa de la variabilidad puede estar relacionada a ensayos en los que el sujeto se encontraba desconcentrado o realizando algún movimiento que haya introducido ruido. Por lo contrario, también es posible que esto se deba a la variabilidad intrínseca de los modelos aleatorios que estamos utilizando.

Finalmente, notamos cómo disminuye la variabilidad de las curvas en comparación

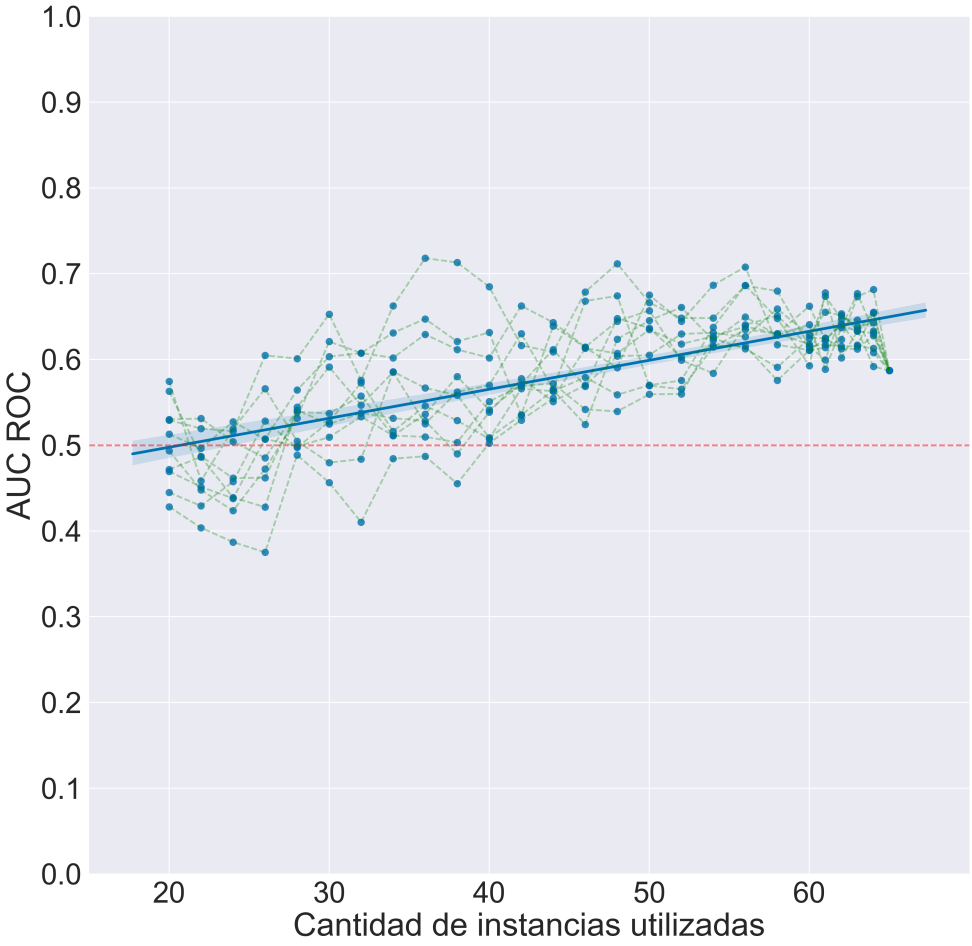


Fig. 3.12: Múltiples curvas de aprendizaje testeando sobre la sesión 2 para el sujeto 10, utilizando el extractor de promedios sobre ventanas y comparando las condiciones 2 y 3.

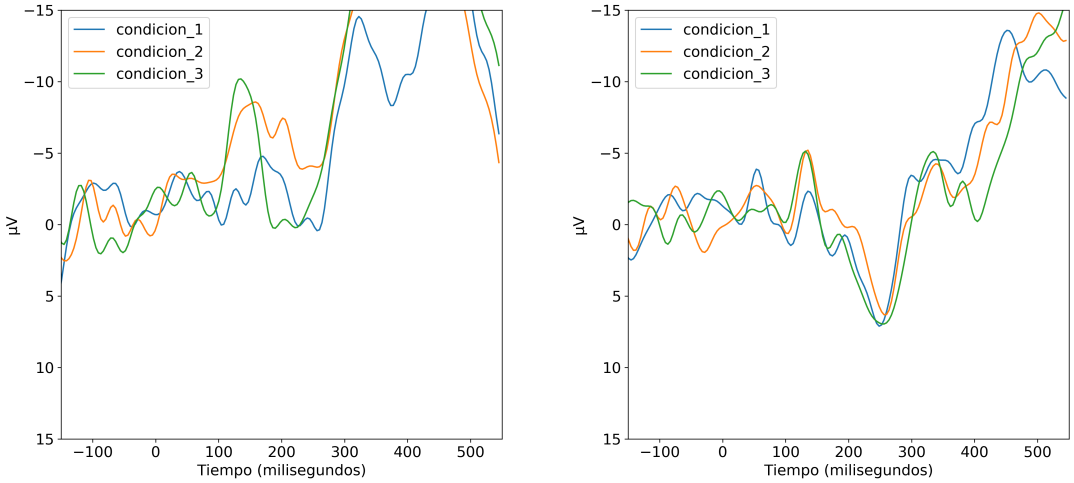
con las obtenidas en el Experimento 2, es decir, que las curvas son más uniformes y la distancia entre puntos de cada una de las iteraciones es menor. De todas formas, es importante mencionar que esta comparación no es del todo justa, ya que en el Experimento 2 realizamos la evaluación sobre una cantidad de instancias que dependió directamente de la cantidad que usamos para entrenar, mientras que en este experimento la cantidad de instancias de evaluación fue siempre la misma para cada uno de los puntos de la curva (en donde sólo varió la cantidad de datos utilizados para entrenar cada modelo).

3.5. Experimento 5: Clasificación intersujeto

Hasta este momento nos habíamos propuesto construir un clasificador por cada uno de los sujetos del conjunto de datos. Sin embargo, en este experimento intentamos construir un

único clasificador entrenado con todos los sujetos. El objetivo es poder predecir las clases para un nuevo sujeto cuyas instancias no fueron previamente utilizadas por el clasificador para entrenar. Específicamente, intentamos responder la pregunta: ¿será posible entrenar un modelo con instancias de varios sujetos para luego poder predecir sobre instancias de un sujeto nuevo?

Este nuevo enfoque presenta algunas diferencias con respecto a los clasificadores de los experimentos anteriores. En primer lugar, al estar entrenando con diferentes sujetos, debemos normalizar las instancias de alguna forma, ya que las señales de cada uno se mueven en un rango diferente de valores. En la Figura 3.13 se muestran los ERPs para dos sujetos distintos (6 y 7) y, así, podemos observar cómo las señales promediadas de cada uno en un mismo canal tiene rangos de valores muy diferentes. La normalización realizada consistió en restarle el promedio y dividir por el desvío estándar a cada ensayo, por cada uno de los sujetos en el conjunto de entrenamiento.



(a) ERP sujeto 6 *(b)* ERP sujeto 7

Fig. 3.13: Comparación ERPs sobre el canal AF3.

Por otra parte, debemos cambiar la forma de entrenar los modelos, debido a que la técnica de *cross validation* explicada previamente utilizaba instancias de un mismo sujeto para evaluar, mientras que en este momento necesitamos hacerlo sobre instancias de un sujeto no visto. Para ello, utilizamos otro tipo de cross validation llamado *leave one group out cross validation* en la que en cada iteración se dejan todas las instancias de un sujeto para evaluar y se entrena sobre los sujetos restantes.

Al tener instancias muy heterogéneas debido en parte a las diferencias individuales entre las personas, decidimos agregar una técnica de extracción de features más robusta que las mencionadas anteriormente. Este nuevo extractor divide al conjunto de canales en 4 clusters agrupados por región y luego calcula el promedio por ventanas sobre estos clusters. La agrupación de canales fue elegida de la misma manera que en [Kaczer et al., 2015], tal como se observa en la Figura 3.14.

La Figura 3.15 muestra la performance obtenida al clasificar combinando distintos ex-

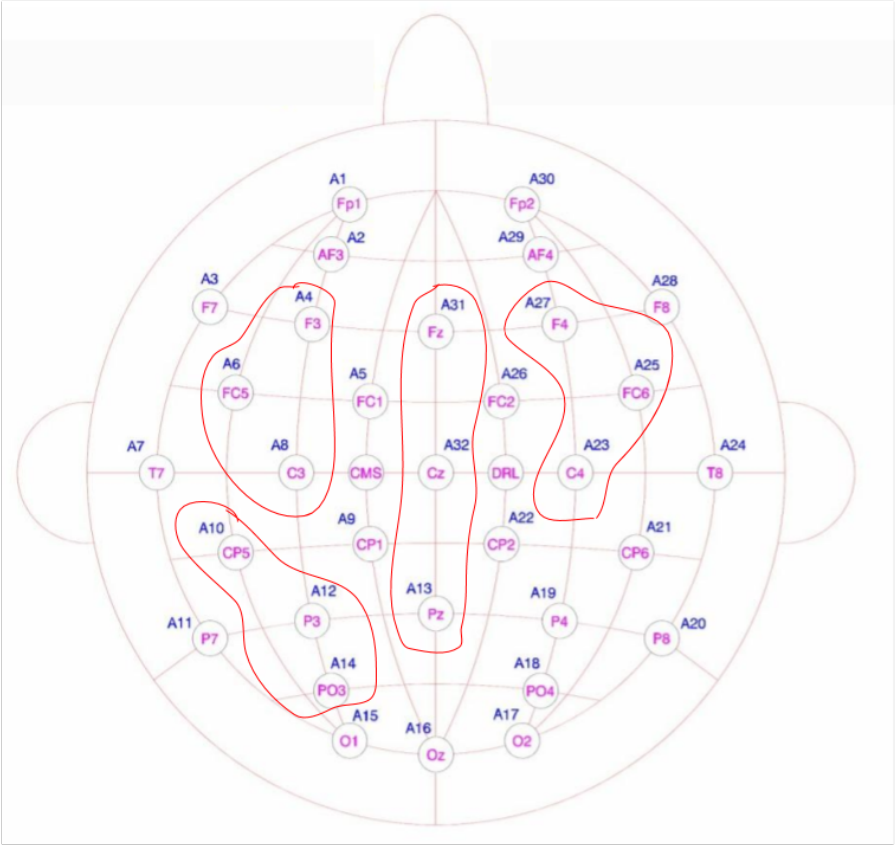


Fig. 3.14: Agrupación de canales por clusters. Cluster 1: F3-FC5-C3; Cluster 2: F4-FC6-C4; Cluster 3: CP5-P3-PO3; Cluster 4: FZ-CZ-PZ.

tractores de features utilizando la técnica de *cross validation* mencionada, la métrica *AUC ROC global* y el modelo *Random Forest* con 100 árboles y sin reducción de dimensionalidad. Los resultados obtenidos no resultan alentadores ya que para ninguna combinación de técnicas de extracción pudimos observar una performance que supere con margen la línea punteada del azar. En otras palabras, no pudimos observar indicios de aprendizaje para el modelo con ninguna de las técnicas de extracción.

Esto nos lleva a pensar que el problema de clasificación intersujeto es un problema difícil para atacar con métodos de extracción de features que no son del todo robustos, ya que no logramos capturar la información presente en las señales como para generalizar sobre las señales de un nuevo sujeto.

Conclusión experimento 5

Luego de analizar los resultados, inspeccionamos cuidadosamente las señales de los sujetos y notamos que, luego de realizar la normalización, la forma de las señales seguía siendo muy diferente en cuanto a las tendencias y los patrones que siguen. Al continuar encontrando estas diferencias luego de normalizar, consideramos que la tarea a la que nos enfrentamos es compleja ya que ante dos instancias de una misma condición, las magnitudes son muy diferentes, lo cual complicará la tarea de generalizar y aprender al clasificador.

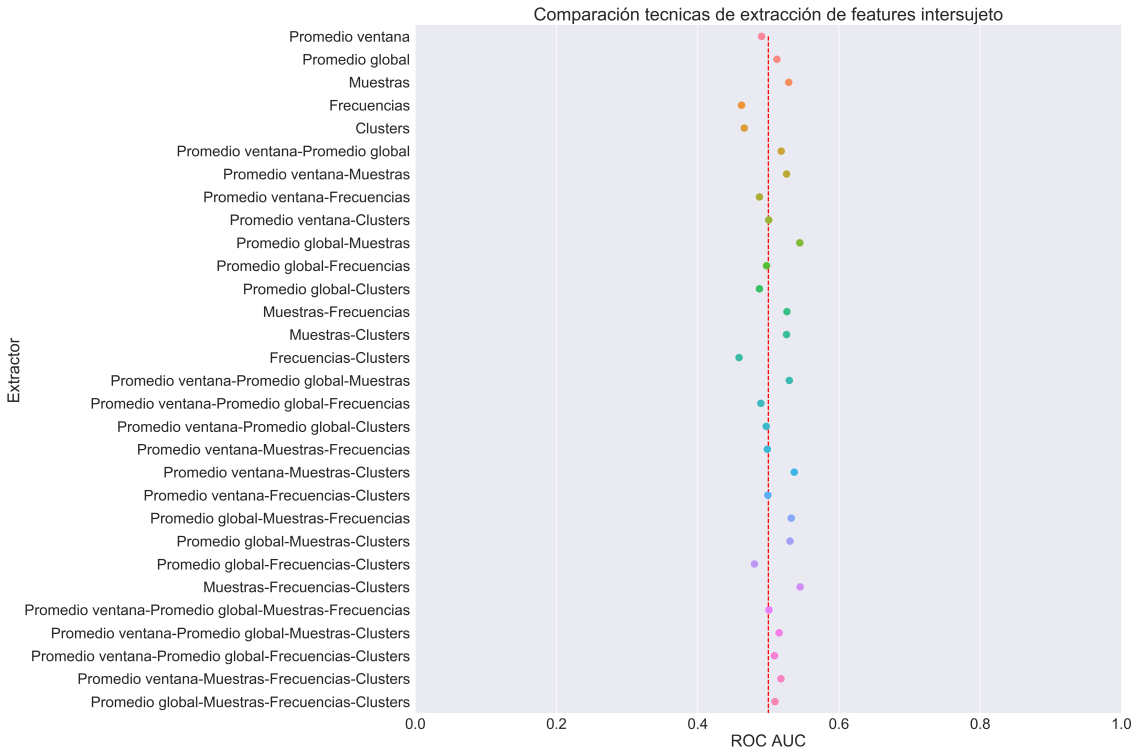


Fig. 3.15: Comparación de técnicas de extracción de features intersujeto.

Concluimos, entonces, que el problema planteado en este experimento es lo bastante complejo como para atacarlo con las herramientas con las que contamos actualmente. De esta manera, dejamos como trabajo a futuro la búsqueda de métodos más robustos de extracción de features y un análisis más profundo sobre las causas por las cuales los métodos utilizados no parecen estar aprendiendo.

4. CONCLUSIONES

El trabajo realizado tuvo como objetivo aplicar técnicas de aprendizaje automático para poder analizar la actividad cerebral en sujetos al aprender palabras nuevas. Para ello, realizamos diversos experimentos con los que intentamos reproducir resultados encontrados en el trabajo de [Kaczer et al., 2015].

En los primeros experimentos nos preguntamos si era posible construir un clasificador que nos permitiera diferenciar instancias según su condición (relacionada existente, relacionada nueva y no relacionada) y, luego, realizamos una búsqueda de evidencia de aprendizaje. Para lograrlo, realizamos visualizaciones de los vectores de features en dos dimensiones en los que encontramos evidencia que nos permitió suponer que las técnicas de extracción estaban capturando información. Posteriormente, utilizamos la técnica *grid search* para explorar posibles configuraciones de parámetros para los modelos a utilizar y tomamos decisiones con respecto a qué configuración utilizar en base a los resultados más prometedores. Al analizar los resultados sobre los datos de evaluación, concluimos que en el caso de contar con tan pocas instancias no es conveniente realizar esta separación, ya que las métricas obtenidas no son confiables ni representativas de la performance del modelo.

A continuación, nos propusimos analizar la confiabilidad de los resultados obtenidos y nos preguntamos si nuestro modelo realmente estaba aprendiendo o si los resultados obtenidos fueron producto del azar. Para eso visualizamos diversas curvas de aprendizaje. El resultado mostró evidencia de aprendizaje en más de un sujeto y en más de una comparación entre condiciones. Es importante mencionar que la cantidad de instancias y de sujetos con los que suele contarse para este tipo de experimentos biológicos resulta una limitante fundamental para las tareas de aprendizaje automático.

A partir de los sujetos más prometedores, buscamos los canales y momentos más determinantes para el clasificador en la tarea de discriminación entre condiciones. En este experimento pudimos observar que los momentos más importantes se correspondieron a las ventanas temporales entre los 456 y 550 milisegundos, al igual que lo observado en [Kaczer et al., 2015] sobre el potencial N400.

Además, comprobamos la robustez de nuestros modelos al utilizar instancias correspondientes a distintos días para entrenar y para testear. Los modelos mostraron evidencia de aprendizaje, lo que nos lleva a considerar que se logró generalizar de manera correcta aun ante instancias medidas bajo condiciones distintas.

Por último, diseñamos un experimento con el que intentamos construir un único clasificador que utilice las instancias de diversos sujetos para entrenar y que además aprenda a generalizar sobre sujetos no vistos. Al enfrentarnos con esta tarea, nos encontramos con diversos problemas al utilizar instancias de sujetos distintos para entrenar. Los resultados obtenidos no fueron buenos. Por esta razón, dejamos como trabajo a futuro la utilización de técnicas más robustas de extracción de features, como puede ser *Wavelets*, y la experimentación con distintas formas de normalizar instancias provenientes de distintos sujetos.

Producto de este trabajo, dejamos a disposición una herramienta de software creada para poder reproducir los experimentos expuestos en esta tesis. Esta servirá para trabajar sobre nuevos corpus de datos de EEG, entrenar modelos sobre estos y luego analizar la

importancia de variables, tal como se realizó en este trabajo.

Trabajo Futuro

Al haber observado en las curvas de aprendizaje que al agregar nuevas instancias de entrenamiento al modelo no siempre la performance aumentaba, nos preguntamos qué es lo que llevó a que esto sucediera. Por lo tanto, dejamos como trabajo a futuro experimentar realizando un análisis y selección de instancias al momento de entrenar para utilizar únicamente las mejores y no introducir instancias que decrementen la performance. En conjunto con este análisis, consideramos que podría ser útil realizar un análisis sobre la importancia de cada uno de los canales y descartar aquellos que aporten menos información o que puedan introducir ruido.

Por otro lado, es importante mencionar que al utilizar features con un alto grado de correlación, como es nuestro caso, los resultados obtenidos al realizar la importancia de features no son del todo confiables para analizar. Queda pendiente utilizar otras técnicas más robustas y confiables ante features correlacionados para medir su importancia, como por ejemplo las abordadas en el trabajo [Meng et al., 2012].

Además de esto, dejamos pendiente la realización de un análisis sobre los sujetos cuya importancia de features no se correspondió con lo analizado en el potencial N400 con el objetivo de encontrar nueva evidencia e información con respecto a la tarea o simplemente corroborar si se trataba de ruido.

Asimismo, sugerimos utilizar la técnica de extracción de features por clusters utilizada en el experimento intersujeto, sobre los experimentos 1, 2, 3 y 4, para analizar la correspondencia con los resultados obtenidos en este trabajo o conseguir nuevos resultados.

Otro enfoque distinto al presentado en este trabajo, podría ser la utilización de otro tipo de clasificadores más poderosos en cuanto a poder predictivo, como por ejemplo las redes neuronales.

Finalmente, dejamos como trabajo a futuro la realización de un nuevo experimento en el que se evalúe si es útil utilizar información de otros sujetos para mejorar el poder de predicción de la tarea intrasujeto. Es decir, aprender patrones generales y luego adaptar el modelo para un sujeto dado.

Bibliografía

[Amini et al., 2013] Amini, Z., Abootalebi, V., and Sadeghi, M. T. (2013). Comparison of performance of different feature extraction methods in detection of p300. *Biocybernetics and Biomedical Engineering*, 33(1):3–20.

[Bishop, 2006] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg.

[Breiman, 2001] Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

[Breiman, 2017] Breiman, L. (2017). *Classification and regression trees*. Routledge.

[Chan et al., 2011] Chan, A. M., Halgren, E., Marinkovic, K., and Cash, S. S. (2011). Decoding word and category-specific spatiotemporal representations from meg and eeg. *Neuroimage*, 54(4):3028–3039.

[Chen and Guestrin, 2016] Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM.

[Geuze et al., 2013] Geuze, J., van Gerven, M. A., Farquhar, J., and Desain, P. (2013). Detecting semantic priming at the single-trial level. *PloS one*, 8(4):e60377.

[James et al., 2013] James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An introduction to statistical learning*, volume 112. Springer.

[Kaczer et al., 2015] Kaczer, L., Timmer, K., Bavassi, L., and Schiller, N. O. (2015). Distinct morphological processing of recently learned compound words: An erp study. *brain research*, 1629:309–317.

[Kepinska et al., 2017] Kepinska, O., Pereda, E., Caspers, J., and Schiller, N. O. (2017). Neural oscillatory mechanisms during novel grammar learning underlying language analytical abilities. *Brain and language*, 175:99–110.

[Kutas and Federmeier, 2011] Kutas, M. and Federmeier, K. D. (2011). Thirty years and counting: finding meaning in the n400 component of the event-related brain potential (erp). *Annual review of psychology*, 62:621–647.

[Louppe et al., 2013] Louppe, G., Wehenkel, L., Sutera, A., and Geurts, P. (2013). Understanding variable importances in forests of randomized trees. In *Advances in neural information processing systems*, pages 431–439.

[Luck, 2014] Luck, S. J. (2014). *An introduction to the event-related potential technique*. MIT press.

[Meng et al., 2012] Meng, J., Meriño, L. M., Shamlo, N. B., Makeig, S., Robbins, K., and Huang, Y. (2012). Characterization and robust classification of eeg signal from image rsvp events with independent time-frequency features. *PloS one*, 7(9):e44464.

[Meyer and Schvaneveldt, 1971] Meyer, D. E. and Schvaneveldt, R. W. (1971). Facilitation in recognizing pairs of words: evidence of a dependence between retrieval operations. *Journal of experimental psychology*, 90(2):227.

[Michalski et al., 2013] Michalski, R. S., Carbonell, J. G., and Mitchell, T. M. (2013). *Machine learning: An artificial intelligence approach*. Springer Science & Business Media.

[Pujol et al., 1999] Pujol, J., Deus, J., Losilla, J. M., and Capdevila, A. (1999). Cerebral lateralization of language in normal left-handed people studied by functional mri. *Neurology*, 52(5):1038–1038.

[Stahl et al., 2012] Stahl, D., Pickles, A., Elsabbagh, M., Johnson, M. H., Team, B., et al. (2012). Novel machine learning methods for erp analysis: a validation from research on infants at risk for autism. *Developmental neuropsychology*, 37(3):274–298.