

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
SANTIAGO - CHILE



“FRAMEWORK PARA LA INTERFAZ CEREBRO-COMPUTADOR P300 SPELLER EN PYTHON.”

SEBASTIÁN ANDRÉS ÁVALOS HERNÁNDEZ

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

Profesor Guía: Dra. Carolina Saavedra
Profesor Correferente: Dr. Julio Sotelo

Diciembre - 2024

DEDICATORIA

A mi padre, a quien le hubiera encantado presenciar este momento.

AGRADECIMIENTOS

En primer lugar, me gustaría agradecer a mi familia dado que, a pesar de las dificultades su apoyo y sacrificio fue lo que me permite presentar este trabajo, también agradecer a mis compañeros y amigos que forme a lo largo de la carrera, que en muchas ocasiones no me dejaron rendirme y a veces haciendo más de lo que debían, y por último, a la profesora Carolina por su apoyo, ánimo y paciencia gracias a los cuales pude completar esta memoria.

RESUMEN

Resumen—

El P300 Speller corresponde a una interfaz cerebro-computador empleada para la comunicación por individuos que ven comprometida su capacidad del habla. El deletreador funciona a partir de la detección del estímulo P300 en base a la interacción del usuario con una matriz la cual sus filas o columnas son iluminadas en rápida sucesión, es por esto, que herramientas capaces de analizar grandes conjuntos de datos son necesarias para su funcionamiento, como objetivo de este trabajo se propone el desarrollo de un *framework* encargado de la capa analítica de la interfaz en el lenguaje de programación Python con el fin de presentar una alternativa de bajo costo. Para su validación, la propuesta fue comparada con una implementación similar existente en MATLAB obteniendo resultados positivos.

Palabras Clave— Interfaz Cerebro-Computador, P300, Maquinas de vectores de soporte, Python.

ABSTRACT

Abstract— The P300 Speller corresponds to a brain-computer interface used for communication by individuals whose speech ability is compromised. The speller works by detecting the P300 stimulus based on the user's interaction with a matrix whose rows or columns are illuminated in rapid succession, needing tools capable of analyzing large data sets for its operation. The objective of this work is to develop a framework in charge of the analytical layer of the interface in the Python programming language in order to present a low-cost alternative. For validation, the proposal was compared with a similar implementation existing in MATLAB, obtaining positive results.

Keywords— Brain Computer Interface, P300, Support Vector Machines, Python.

GLOSARIO

BCI: Brain-Computer Interface

EEG: Electroencefalografía

ERP: Event-related Potential

MEG: Magnetoencefalografía

ML: Machine Learning

MRI: Resonancia Magnética

RBF: Radial Basis Function

SVM: Support Vector Machine

ÍNDICE DE CONTENIDOS

RESUMEN	IV
ABSTRACT	IV
GLOSARIO	V
ÍNDICE DE FIGURAS	VIII
ÍNDICE DE TABLAS	IX
INTRODUCCIÓN	1
CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA	3
1.1 Objetivos	5
CAPÍTULO 2: MARCO CONCEPTUAL Y ESTADO DEL ARTE	6
2.1 Ondas cerebrales e Interfaces Cerebro-Computador	6
2.1.1 Electroencefalografía	6
2.1.2 Interfaces cerebro-computador	9
2.2 Filtrado de señal y modelos de clasificación	12
2.2.1 Filtros de señal	12
2.2.2 Clasificadores	12
2.2.3 Máquina de vectores de soporte	12
2.2.4 Métricas de desempeño	13
2.3 Lenguajes de programación y frameworks	16
2.3.1 BCI2000	16
2.3.2 OpenViBE	17
2.3.3 MATLAB	17
2.3.4 Python	17
CAPÍTULO 3: PROPUESTA DE SOLUCIÓN	20
3.1 Flujo de trabajo	21
3.2 Procesamiento	22
3.2.1 Estructura de los datos y codificación	22
3.2.2 Normalización	24
3.2.3 Filtrado de señales	24
3.2.4 Reordenamiento de los datos	24
3.3 Traducción	26
3.3.1 Clasificación y predicción	26
3.3.2 Interpretación y reporte de resultados	27
3.4 Interfaz	30
CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN	34

4.1	Metodología	34
4.2	Resultados	35
4.2.1	Pruebas de funcionamiento	35
4.2.2	Pruebas comparativas	45
CAPÍTULO 5: CONCLUSIONES Y TRABAJO FUTURO		50
ANEXOS		54
5.1	Resultados: Promedio de secuencias	54
REFERENCIAS BIBLIOGRÁFICAS		63

ÍNDICE DE FIGURAS

1	Diagrama del protocolo 10-20 original	7
2	Posicionamiento de los electrodos en el protocolo 10-10 o 10-20 extendido .	8
3	Diagrama del ciclo de un BCI	10
4	Ejemplo de matriz 6x6	11
5	Hiperplano y vectores de soporte	13
6	Curvas ROC	15
7	Elementos que componen el <i>framework</i> propuesto	20
8	Etapas del funcionamiento del pipeline	21
9	Codificación de los datos	22
10	Recreación del reorden de la señal.	25
11	Identificación de las filas/columnas en base al vector StimulusType obtenido de la SVM para una secuencia	28
12	Estructura del diccionario de resultados	29
13	Estructura de los diccionarios auxiliares	30
14	Pantalla principal de la interfaz de usuario	30
15	Visualizador de señal	31
16	Diccionarios creados a partir de la interfaz	32
17	Plantilla de la pantalla de resultados generales	33
18	Plantilla de la pantalla de resultados para sujetos individuales	33
19	Onda resultante al aplicar filtro de Butterworth de tipo paso alto.	36
20	Onda resultante al aplicar filtro de Butterworth de tipo paso alto.	37
21	Onda resultante al aplicar filtro de Butterworth de tipo pasa-banda.	38
22	Onda resultante al aplicar normalización de tipo suave.	39
23	Onda resultante al aplicar normalización de tipo fuerte.	39
24	Resultados generales para 1, 7 y 15 secuencias promediadas.	41
25	Resultados de LAC para todas las secuencias.	42
26	Resultados de LGP para todas las secuencias.	42
27	Resultados de JSC para todas las secuencias.	43
28	Resultados de ASR para todas las secuencias.	44
29	Resultados de GCE para todas las secuencias.	44
30	Resultados generales para los distintos promedios de secuencia.	46
31	Resultados de ACS para todas las secuencias.	54
32	Resultados de APM para todas las secuencias.	55
33	Resultados de ASG para todas las secuencias.	55
34	Resultados de DCM para todas las secuencias.	56
35	Resultados de DLP para todas las secuencias.	56
36	Resultados de DMA para todas las secuencias.	57
37	Resultados de ELC para todas las secuencias.	57
38	Resultados de FSZ para todas las secuencias.	58
39	Resultados de ICE para todas las secuencias.	58
40	Resultados de JLD para todas las secuencias.	59
41	Resultados de JLP para todas las secuencias.	59

42	Resultados de JMR para todas las secuencias.	60
43	Resultados de JST para todas las secuencias.	60
44	Resultados de LAG para todas las secuencias.	61
45	Resultados de PGA para todas las secuencias.	61
46	Resultados de WFG para todas las secuencias.	62
47	Resultados de XCL para todas las secuencias.	62

ÍNDICE DE TABLAS

1	Matriz de confusión para un problema binario.	14
2	Desempeño en porcentaje de la palabra obtenida para ambas implementaciones en base a la cantidad de secuencias promediadas.	47
3	Tiempos de ejecución en segundos a partir de la cantidad de secuencias promediadas.	48

INTRODUCCIÓN

Las interfaces cerebro-computador (o BCI por sus siglas en inglés) han sido estudiadas a lo largo de los años por los beneficios que éstas prestan para individuos que han visto sus capacidades motrices o cognitivas comprometidas, al ser un área relativamente nueva, los esfuerzos se han inclinado por perfeccionar las propuestas iniciales mientras que al mismo tiempo se busca producir soluciones mas accesibles y menos intrusivas sin comprometer la experiencia del paciente, si bien este tipo de tecnología en un inicio fue pensada con el fin de mejorar la calidad de vida de personas con algún grado de discapacidad, ésta también ha tenido aplicaciones en actividades recreativas destinadas al publico general como por ejemplo, los videojuegos.

La manera en que estos sistemas funcionan es a partir de la información que pueda ser extraída de los datos recopilados de la actividad cerebral, por lo tanto, los implementos necesarios para su extracción son también considerados como parte del BCI, la manera en que estos datos son procesados depende directamente de la funcionalidad que busca ser replicada y las características propias del tipo de onda cerebral empleadas para su análisis, en ese contexto, las interfaces cerebro-computador son una tecnología que conecta la disciplina de la Informática con la Medicina, mas específicamente la Neurociencia y los avances de ambas áreas contribuyen al refinamiento de estos sistemas.

En este contexto, el P300 Speller corresponde a un BCI empleado para la comunicación de individuos que presentan enfermedades donde la capacidad del habla se ve comprometida, como su nombre indica este sistema corresponde a un deletreador que opera haciendo uso del potencial relacionado con eventos P300, el cual es relativamente sencillo de evocar a partir de una estrategia denominada *oddball*. El método habitual de captura de datos corresponde al uso de Electroencefalografía, siendo este un método no invasivo para el paciente, no obstante, esto repercute en la cantidad de procesamiento necesario para la identificación de las ondas cerebrales deseadas, análisis que mayoritariamente se realiza empleando MATLAB, un software especializado para el manejo de estructuras de datos siguiendo notaciones matemáticas, el cual opera a partir de un modelo de licencias, lo cual limita el estudio y desarrollo de soluciones enfocadas a usuarios finales.

Como parte del desarrollo de esta memoria, se propone un *framework* para el P300 Speller encargado de realizar las labores del análisis de datos para señales provenientes de la actividad cerebral recopiladas mediante EEG haciendo uso de Python, un lenguaje de programación de código abierto y librerías asociadas para el análisis de datos y manipulación de estructuras de datos.

El análisis se realizará mediante el procesamiento y manipulación de las señales digitalizadas, para así entrenar un modelo de aprendizaje supervisado que pueda clasificar los datos según su asociación con el potencial P300. En el capítulo 1 se aborda el problema y su relevancia en mayor detalle, en el capítulo 2 se presentan las técnicas empleadas para el desarrollo del

P300 Speller, mientras que en el capítulo 3 se muestra la estructura del funcionamiento del *framework* propuesto para finalmente, en el capítulo 4 validar la solución propuesta a partir de la comparación de resultados con respecto a una implementación análoga en MATLAB.

CAPÍTULO 1

DEFINICIÓN DEL PROBLEMA

Las Interfaces Cerebro-Computador corresponden a modelos que permiten traducir información de la actividad cerebral de las personas en acciones concretas dentro de un sistema computacional, como consecuencia, su estudio se ha enfocado en poder simular funciones motrices en personas con algún grado de discapacidad (física o cognitiva, por ejemplo, amputaciones o problemas del habla respectivamente), analizar el grado de degradación de las funciones cognitivas en individuos que se encuentren con algún impedimento para realizar tareas cotidianas, pero también en la extensión de las capacidades humanas mediante elementos externos que puedan ser manipulados a través del uso consciente de la mente [Mak y Wolpaw, 2009], no obstante, su uso de forma recreativa también es una de las razones del estudio de estas interfaces, como por ejemplo, videojuegos manipulados mediante los estímulos producidos en el cerebro [Kaplan *et al.*, 2013].

En particular, existe una variante de *BCI* basando en electroencefalografía (EEG) que es de interés por lo beneficioso que resulta para personas con síndrome *lock-in* o en otras palabras con un alto grado de parálisis [Guy *et al.*, 2018], en particular, por que es menos invasivo para el paciente y a la vez es mas económico [Mak y Wolpaw, 2009], el P300 Speller, tal como su nombre indica, es una herramienta empleada para la comunicación que permite deletrear palabras. Este *BCI* hace uso del potencial P300, potencial que se produce aproximadamente 300ms posterior al inicio del estímulo que lo genera, la manera más común de evocar un P300 es mediante el paradigma *oddball*, el cual consiste en la identificación y enumeración de un estímulo anormal dentro de una serie de estímulos repetitivos.

Específicamente la implementación más común para este *speller*, es mediante una matriz de 6x6, en la cual se incluyen todas las letras del abecedario y otros caracteres que puedan ser de importancia (Como por ejemplo, números), durante secuencias cortas, las filas y columnas de la matriz son iluminadas de manera aleatoria, el potencial es registrado cuando la fila/columna que contiene el carácter objetivo se ilumina [Farwell y Donchin, 1988]. Al poder deletrear palabras de esta forma, los pacientes pueden entablar conversaciones con sus seres queridos, así como también poder dar indicaciones a sus cuidadores.

Uno de los problemas asociados a este *BCI* corresponde a la identificación del potencial P300. Debido a la naturaleza de la obtención de la señal, esta debe ser aislada de otras señales captadas durante la adquisición de señales de EEG, no solo eso, la señal debe ser filtrada de cualquier elemento no deseado como por ejemplo ruido o estática, más aun, el usuario debe estar concentrado durante la toma de datos debido a que las señales son fácilmente influenciadas por otros estímulos, volviendo más compleja la identificación del estímulo objetivo, si bien este último no es un problema que pueda ser solucionado a nivel de software, esto debido a las características propias de cada individuo, el estudio de esta interfaz en la actualidad se encuentra enfocada en aumentar la confiabilidad de los resultados ante la presencia de estas variables.

Sin embargo, existen otros problemas asociados al funcionamiento del BCI, la digitalización de las señales cerebrales implica en la práctica que se trabaja con un gran volumen de datos, los cuales deben ser procesados para poder realizar los análisis necesarios. Por mucho tiempo, debido a su fuerte enfoque en las matemáticas se ha utilizado MATLAB para realizar estos procesos, sin embargo, su modelo de monetización a partir de licencias es un factor a considerar al momento de trabajar con él, lo que repercute en tanto en los usuarios finales así como también en aquellos interesados en estudiar y/o implementar soluciones de este tipo; no solo eso, su carácter de software propietario influye directamente en cómo puede ser integrado con otros lenguajes de programación.

En ese sentido, poder realizar un *framework* para el procesamiento y traducción de señales cerebrales, en el contexto del BCI P300 Speller en un lenguaje con mayor adopción y de código abierto como Python, abre la oportunidad a seguir refinando el trabajo generado hasta ahora, en especial, considerando los avances que este ha tenido a lo largo de los años y el auge que está teniendo en el área del análisis de datos y *machine learning*, lo cual a la vez permite poder construir soluciones más accesibles y robustas que con los métodos actualmente disponibles.

La tasa de adopción y popularidad entre ambos lenguajes recopiladas por Stackoverflow, uno de los sitios relacionados a programación mas visitados, da una clara tendencia a marcar los beneficios que supone Python por sobre MATLAB, por ejemplo, de un total 87.585 encuestados, un 49.28 % es usuario de Python mientras que un 3.81 % lo es de MATLAB, siendo este ultimo una de la tecnologías peor evaluadas en términos de retención con un 18.33 % ¹, esto supone una gran diferencia en la cantidad de preguntas y ayuda que se pueden obtener del foro, lo que permite hacerse una idea de la disponibilidad de recursos como tutoriales, cursos o comunidades dispuestas a prestar soporte a sus miembros en otros sitios del internet, marcando una ventaja que puede mitigar el riesgo que implica un cambio de plataforma.

Tomando en consideración lo anterior, proyectos que utilizan Python para migrar trabajos anteriores, actualizar herramientas o proponer alternativas de código abierto existen y es esperable que con el paso del tiempo las implementaciones en Python se vuelvan un tema de discusión ya sea como alternativa o complemento a soluciones ya existentes. Por ejemplo: pyPPG [Goda *et al.*, 2024] corresponde a un *toolbox* para la medición de cambios en el volumen de la sangre en los tejidos que su principal objetivo corresponde a proponer una alternativa a partir de una metodología estandarizada y de código abierto, Scikit-ANFIS [Zhang y Chen, 2024] corresponde a una implementación del sistema ANFIS (mayoritariamente utilizada en MATLAB) compatible con la librería de Python scikit-learn mediante el uso de una interfaz de programación basada en las llamadas de dicha librería, MCR Toolkit 2.0 es una herramienta utilizada para la reconstrucción de tomografías computarizadas de rayos X, esta es una nueva versión desarrollada en Python que migra todas las funcionalidades de su primera versión desarrollada en MATLAB además de agregar nuevas características,

¹<https://survey.stackoverflow.co/2023/#most-popular-technologies-language>

finalmente, gMCSpy es una herramienta para el cálculo de conjuntos de corte mínimo en el contexto de modelos metabólicos a nivel del genoma, similar al proyecto anterior, esta es una nueva versión de un proyecto originario de MATLAB que añade nuevas características. Todos los proyectos anteriormente mencionados utilizan de referencia implementaciones de MATLAB como punto de comparación para medir la efectividad de sus propuestas, obteniendo ventajas en términos de rendimiento, optimización, resultados o compatibilidad con otras herramientas, demostrando los beneficios que supone Python y sus librerías para el desarrollo de proyectos en el ámbito científico.

Es por esta razón que en esta memoria se propone:

1.1. Objetivos

Objetivo General

- Desarrollar un *framework* basado en el P300 Speller para preprocesar, clasificar y presentar los resultados a partir de datos de la actividad cerebral utilizando el lenguaje de programación Python.

Objetivos Específicos

- Estudiar el funcionamiento de la interfaz cerebro-computador P300 Speller.
- Diseñar una propuesta de *framework* para una interfaz cerebro-computador P300 Speller utilizando el lenguaje de programación Python.
- Verificar y comparar el funcionamiento de la solución, haciendo uso de una implementación equivalente.

CAPÍTULO 2

MARCO CONCEPTUAL Y ESTADO DEL ARTE

2.1. Ondas cerebrales e Interfaces Cerebro-Computador

2.1.1. Electroencefalografía

La electroencefalografía (EEG) corresponde a un procedimiento en el cual mediante el uso de electrodos posicionados estratégicamente a lo largo de la parte superior de la cabeza se registran los impulsos eléctricos generados por las neuronas, estos registros son digitalizados e interpretados como ondas. Este es un procedimiento de carácter no invasivo, puesto que no es necesario realizar intervenciones quirúrgicas para la colocación de los electrodos.

Mediante este método es posible detectar los estados del ánimo así como también anomalías en la actividad cerebral, por lo que es usada principalmente como herramienta para el diagnóstico de la Epilepsia, Alzheimer, tumores cerebrales, entre otros [Blinowska y Durka, 2006]. Debido al relativo bajo costo de los implementos necesarios para su realización comparado a otras alternativas, es uno de los métodos más empleados para la identificación del potencial P300.

En base a la cantidad de los electrodos empleados y la ubicación de estos es como se detectan las ondas cerebrales, esto implica a la vez la cantidad de resultados que arroja el procedimiento, en particular, para la detección del P300 existen varias combinaciones que se han estudiado a lo largo del tiempo para obtener de manera consistente la detección del potencial, dado la presencia de elementos como el ruido, así como también otras señales de la actividad cerebral [Mak y Wolpaw, 2009]. Los canales comúnmente utilizados para la detección del P300 corresponden al F_z , C_z , P_z [Yordanova y Kolev, 1996, Krusienski *et al.*, 2008] que son los electrodos generalmente utilizados para distintos exámenes de EEG, sin embargo, considerar los electrodos PO_7 , PO_8 y O_z puede mejorar los resultados al realizar la clasificación [Krusiensi *et al.*, 2008].

Es necesario tener en consideración que independiente de la cantidad o tipo de electrodos empleados, la EEG no solo registra las respuestas provenientes del potencial P300, sino que es capaz de identificar distintos potenciales de la misma familia a la que éste pertenece, además de monitorizar distintas respuestas como los movimientos del cuerpo, ojos, entre otros; esto sumado a la presencia de ruido producto de interferencias en el electrodo, así como las generadas por la estructura ósea del cráneo, implica que la precisión de los datos obtenidos por este examen es variable, por lo que, es necesario realizar procesamiento de datos para poder ser utilizados.

Dado que la estructura ósea del cráneo varía de individuo en individuo y por ende, la forma de su cabeza es distinta, existe un protocolo encargado de indicar el posicionamiento de

los electrodos para reducir la variación de las señales medidas ante este fenómeno, este protocolo tiene por nombre 10-20, que como su nombre indica, especifica que la distancia de los electrodos debe seguir un ratio del 10 % del diámetro en los extremos de la cabeza, mientras que en sectores mas centrales, la distancia entre nodos debe ser del 20 % de su diámetro, tal como se puede ver en la figura 1.

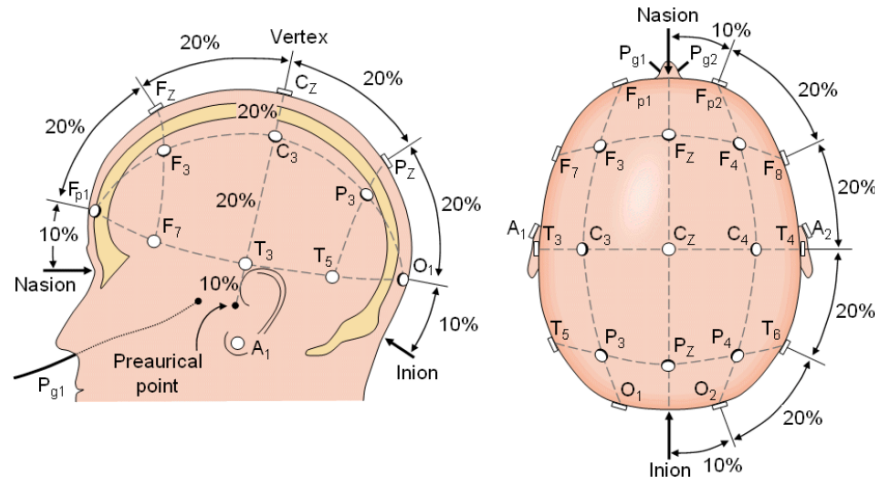


Figura 1: Diagrama del protocolo 10-20 original. Fuente: [Malmivuo y Plonsey, 1995]

Con el fin de aumentar la cantidad de electrodos a emplear en un sistema de EEG, se especificó una extensión del protocolo, que redujo la distancia desde cualquier punto a un 10 %, este protocolo también es conocido como 10-10 o 10-20 extendido, tal como se puede ver en la figura 2. Es importante mencionar que independiente del protocolo empleado, los electrodos son nombrados según el hemisferio y localizaciones del cerebro, por ejemplo, frontal (F), temporal (T), parietal (P), occipital (O) y central (C), mientras que aquellos electrodos localizados en el hemisferio izquierdo tienen como subíndice números impares, en el derecho números pares y aquellos ubicados en el centro la letra z.

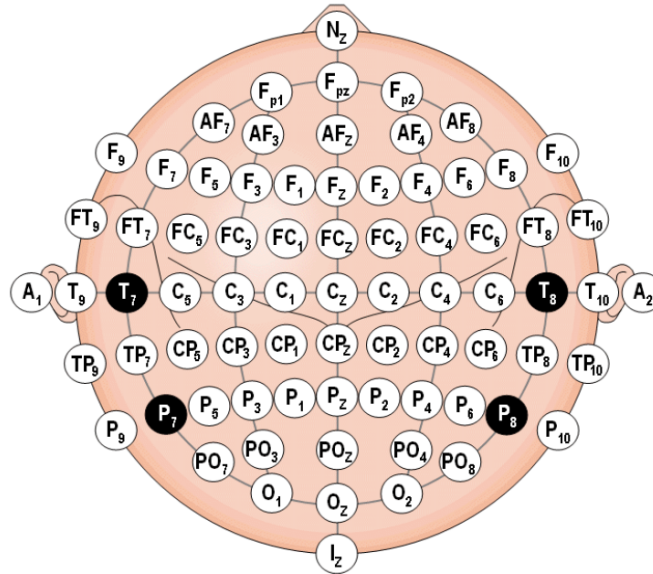


Figura 2: Posicionamiento de los electrodos en el protocolo 10-10 o 10-20 extendido. Fuente: [Malmivuo y Plonsey, 1995]

Potencial relacionado a evento P300

El P300 es un potencial relacionado con eventos (ERP, por su siglas en inglés), es decir, una variación en la actividad cerebral que se produce ante la presencia de un evento de carácter sensorial o cognitivo, las cuales quedan registrados mediante EEG [Blinowska y Durka, 2006], en particular, el P300 se caracteriza por manifestarse alrededor de 300ms posterior a la presencia del estímulo (Este se puede registrar entre los 250-500ms) [Yordanova y Kolev, 1996], la manera más frecuente para evocar este potencial es bajo el paradigma del bicho raro o paradigma *oddball*, el cual consiste en la detección de estímulos inusuales ante la presencia de elementos repetitivos [Krusienski et al., 2008, Farwell y Donchin, 1988]. Por ejemplo, el contar la cantidad de cartas de color amarillo que son trasapeladas rápidamente en un conjunto de cartas principalmente de color verde generaría una respuesta de tipo P300.

Debido al procedimiento empleado para su detección, la señal perteneciente al potencial P300 debe ser aislada de otras similares, para ello, contar con la presencia de más de un canal (en este caso, electrodo) que capture este potencial permite localizar los sectores donde este se produce, también es necesario tener en consideración que la respuesta del P300 difiere en magnitud con respecto a otras presentes en la EEG, por lo que el procedimiento deberá repetirse en múltiples ocasiones para su detección requiriendo promediar dichas señales, no obstante, la presencia de ruido es otra de las razones por la que es necesario procesar y promediar las señales para la identificación del P300. Un análisis donde las señales no son promediadas se denomina *Single trial*.

2.1.2. Interfaces cerebro-computador

Las interfaces cerebro-computador corresponden a sistemas computacionales que se encargan de traducir elementos de la actividad cerebral en acciones concretas a nivel de *hardware* o *software*. Para ello, deben procesar los datos obtenidos de la actividad cerebral, dependiendo de la precisión del método en que ésta fue registrada corresponde a la cantidad de pasos por los cuales los datos deben someterse antes de considerarse aptos para clasificarlos.

El sistema también debe poder traducir los datos de las ondas cerebrales en acciones a nivel *software* y/o *hardware* que posteriormente son comunicadas al usuario en forma de la acción que este busca realizar [Mak y Wolpaw, 2009]. Es importante mencionar que todo este proceso sucede en tiempo real, este modo de funcionamiento es conocido como análisis *online*, en contraste, cuando los datos son sometidos al mismo proceso en un periodo posterior a la toma de datos se denomina como análisis *offline*, este último es empleado para el estudio y mejoramiento de los sistemas ya desarrollados.

Dentro de las metodologías conocidas para la extracción de los datos están presentes los métodos invasivos, parcialmente invasivos y no invasivos, la elección de una técnica por sobre otra determina tanto el costo de operación como la precisión de los datos obtenidos, no solo eso, también repercute en la complejidad de los procedimientos y la capacitación necesaria, en ese sentido, procedimientos invasivos como la colocación de electrodos directamente en el cerebro son procedimientos complejos a la vez que costosos, pero que otorgan datos que requieren menor procesamiento, mientras que los procedimientos no invasivos como la electroencefalografía (EEG), magnetoencefalografía (MEG) y la resonancia magnética (MRI) resultan ser relativamente más económicos, requieren menor entrenamiento del personal a costa de la necesidad de un mayor procesamiento de los datos obtenidos [Leuthardt *et al.*, 2021]. La EEG es uno de los métodos para captura de datos más utilizados en el desarrollo de BCIs por lo accesible y práctico que resulta para producir soluciones que puedan ser empleadas fuera de hospitales o laboratorios [Jamil *et al.*, 2021].

La figura 3 muestra la arquitectura de un sistema BCI, partiendo por la recolección de los datos de la actividad cerebral, el procesamiento y traducción que es convertida en una acción que el usuario es capaz de interpretar.

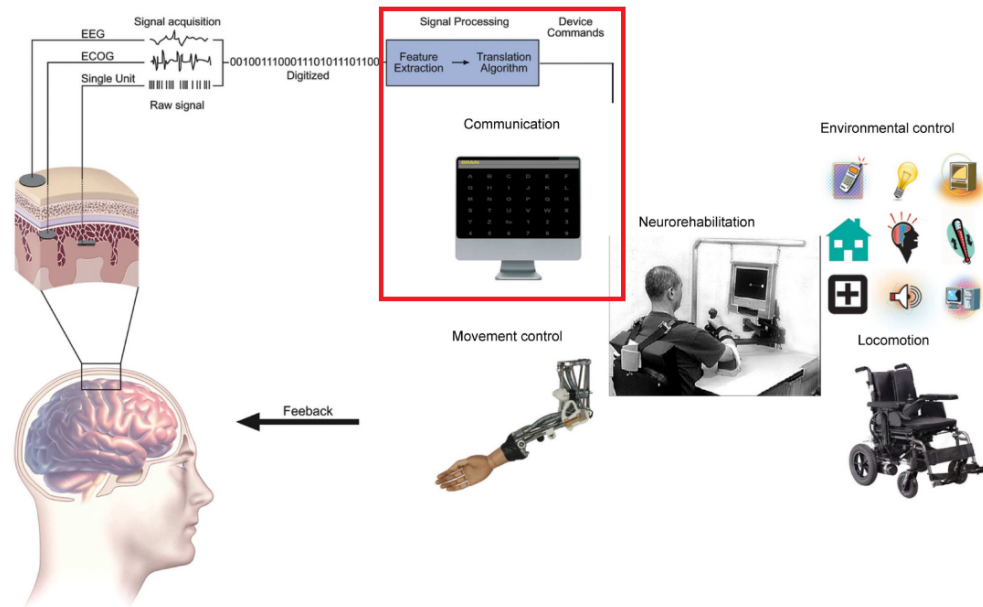


Figura 3: Diagrama del ciclo de un BCI: Al lado izquierdo se encuentra el proceso de adquisición de la señal a partir de la digitalización de ondas cerebrales, posteriormente, dichos datos son procesados para extraer las características relevantes que serán traducidas en *inputs* que serán empleados por aplicaciones que interactúen con el entorno. En lado derecho, se aprecia la retroalimentación al usuario con respecto a la acción que este busca realizar. En rojo, se identifica el alcance de la memoria. Fuente: [Mak y Wolpaw, 2009] (Modificado para adecuar al documento)

Dentro de los programas más conocidos para el desarrollo de estas interfaces se encuentra BCI2000 y OpenVibe, en particular, ofrecen una gran gama de herramientas para diseñar soluciones, cada una con su enfoque en particular. Estos serán introducidos en mayor detalle más adelante.

P300 Speller

El P300 Speller es un BCI que permite, tal como su nombre indica, deletrear palabras a partir del análisis del potencial P300. Específicamente, consiste en la proyección de una matriz de 6x6 en la cual se incluye el alfabeto y caracteres adicionales para completar todas las casillas de la matriz; según la finalidad del Speller, los elementos adicionales pueden corresponder a números o caracteres especiales. Existen 2 enfoques con respecto al uso del sistema; el primero de estos corresponde al guiado, donde se le indica al usuario la palabra a deletrear y en qué letra debe enfocarse, mientras que el segundo enfoque es libre, donde el usuario decide en qué letras enfocarse para deletrear una palabra [Guy *et al.*, 2018].

Cuando se encuentra en funcionamiento, una fila o columna de la matriz se iluminará aleatoriamente hasta que todas las filas/columnas hayan sido iluminadas una vez. El ciclo

completo de iluminaciones es denominado secuencia; este proceso se repite por una cantidad determinada de veces por letra. Para poder deletrear una letra, el usuario debe focalizar su atención en un carácter de la matriz y contar la cantidad de veces que la fila o columna a la que éste pertenece es iluminada. Esta acción genera un ERP, específicamente un P300 que es utilizado para identificar qué fila o columna fue iluminada. A partir de ello, se determina el carácter elegido por el usuario a partir de la intersección de la fila y columna que se iluminaron la mayor cantidad de veces. [Farwell y Donchin, 1988, Krusienski *et al.*, 2008] Una vez terminada la palabra, esta puede ser mostrada en pantalla o pronunciada mediante un sintetizador.

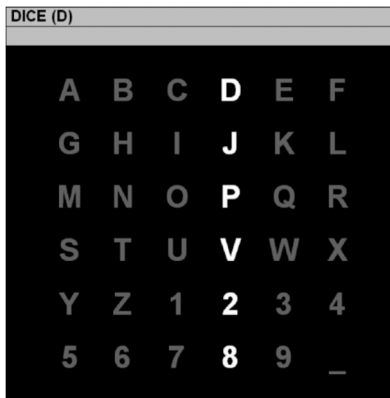


Figura 4: Ejemplo de matriz 6x6, en la sección superior se muestra la palabra a deletrear y la letra que el usuario debe enfocarse. En este caso, al iluminarse la columna objetivo, el P300 es registrado. Fuente: [Krusiensi *et al.*, 2008]

En términos del funcionamiento de un BCI, el sistema realiza la captura de datos a partir de las interacciones entre el usuario y la matriz, este proceso se puede realizar múltiples veces por letra con el fin de disminuir la incertidumbre al momento de la detección del estímulo al momento del análisis de los datos, la cantidad de veces que una misma letra fue registrada antes de pasar a la siguiente se denomina número de secuencias; una vez los datos fueron registrados, el sistema realiza varios procedimientos para el procesamiento de los datos, partiendo por filtros de señales para eliminar características indeseadas, recortar segmentos innecesarios de los datos (por ejemplo, cuando inicia/finaliza la captura de datos, al no contener datos útiles), promediar secuencias para generar una señal más clara y todo aquel paso que permita facilitar la identificación del estímulo deseado; finalmente, el sistema realiza una clasificación a partir de los datos para detectar los instantes en que se produjeron las respuestas P300, estas corresponden a filas/columnas de la matriz original, que son traducidas a los caracteres con los cuales se reconstruye la palabra que el usuario deletreó.

2.2. Filtrado de señal y modelos de clasificación

2.2.1. Filtros de señal

Los filtros de señal corresponden a técnicas de procesamiento de señales utilizados para eliminar elementos indeseados que se encuentren en ésta, la manera en que esto se logra es a partir de la remoción de frecuencias en la señal de origen. Esto se manifiesta mediante el rango de filtrado, para un filtro pasa bajo, todo elemento de la señal que esté por sobre un cierto umbral es anulado, por otra parte, para un filtro de pasa alto, el efecto es el contrario, para un cierto umbral cada elemento de la señal por debajo de este es eliminado. La aplicación de un filtro pasa bajo y pasa alto simultáneamente es denominado pasa banda.

Filtro de Butterworth

En el campo de los filtros de señales, el filtro de Butterworth se considera como uno de los filtros de señales más simples siendo su principal característica el aplanamiento de los *peaks* en la señal al ser aplicado un filtro de paso banda, lo cual suaviza la onda al momento de eliminar las señales que se encuentra fuera del rango de frecuencias.

2.2.2. Clasificadores

Los clasificadores o modelos de clasificación son técnicas de aprendizaje supervisado empleados en el contexto de *machine learning* para la predicción e identificación de clases en un conjunto de datos. El modelo debe ser entrenado a partir de un conjunto de datos en que las clases ya fueron identificadas (aprendizaje supervisado), de esta forma, mediante el conocimiento adquirido se puede discriminar y predecir las clases a las que pertenecen nuevos datos.

Los clasificadores se pueden dividir en base a la cantidad de clases presentes en los datos, ya sea binario o multi-clase, o tomando en consideración el tipo de algoritmo empleado para la clasificación, ya sea un algoritmo supervisado o no supervisado. En el contexto del P300 Speller, múltiples clasificadores han sido utilizados con resultados mixtos, dentro de los clasificadores más utilizados se encuentran: Linear Discriminant Analysis y sus variantes SWLDA y BLDA, y Support Vector Machine, sumándose en tiempos recientes métodos como Convolutional Neural Networks, Riemann Geometry, entre otros. [Lotte *et al.*, 2018]

2.2.3. Máquina de vectores de soporte

Máquinas de vectores de soporte (o SVM por sus siglas en inglés) corresponde a una técnica de aprendizaje supervisado para clasificación y regresión de problemas lineales, no obstante, mediante el uso de *kernels* el método puede ser adaptado para la resolución de problemas no lineales.

El principio detrás de este algoritmo es separar el *set* de datos en 2 clases y encontrar un hiperplano entre estas clases que permita clasificar nuevos datos a partir de su distancia

al plano, no obstante, este hiperplano podría estar en cualquier punto entre ambos set de datos, es por ello, que para minimizar la posibilidad de clasificar los datos de forma incorrecta, el hiperplano es determinado a partir de la máxima distancia entre elementos de las clases que se encuentran próximos al plano, aquellos puntos que resulten relevantes para generar dicho hiperplano son denominados los vectores de soporte y determinan las características de la solución [Kecman, 2005].

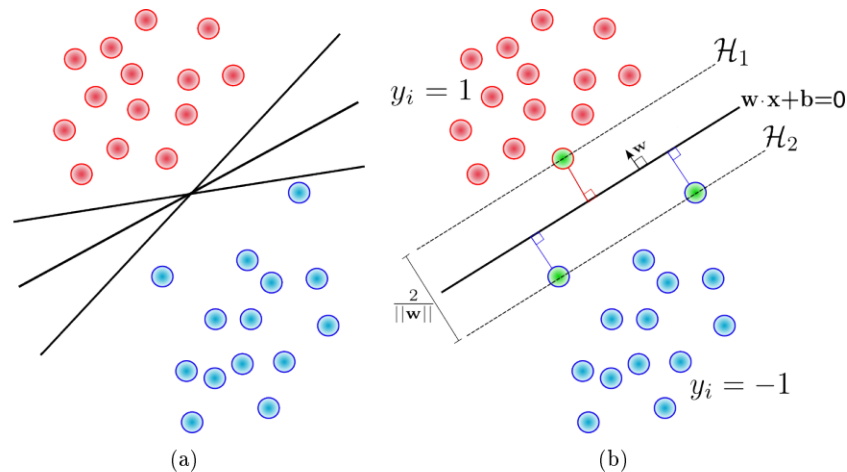


Figura 5: Hiperplano y vectores de soporte, en (a) se muestra algunos de los posibles planos a generar entre las clases, en (b) el plano que optimiza la distancia entre las clases a partir de los nodos más próximos a este. Dichos puntos son conocidos como vectores de soporte. Fuente: [Ruiz, 2013]

La figura 5 presenta los casos anteriormente descritos, en (a) se muestran algunos de los posibles planos que se pueden generar en torno a las 2 clases, mientras que (b) se enfoca en el plano que optimiza el problema de clasificación a partir de los nodos pertenecientes a las clases más próximos al plano.

No obstante, no todos los problemas son linealmente separables, es por ello, que para poder ser resueltos por la SVM se hace una modificación del funcionamiento de este a partir de los *kernels* y variables de holgura, donde estos problemas son llevados a un plano donde si puedan ser linealmente separables. Ejemplos de estos *kernels* son: Polinomial, radial (RBF), etc. [Kecman, 2005].

2.2.4. Métricas de desempeño

Los clasificadores de aprendizaje supervisado emplean ciertas métricas para determinar el rendimiento del algoritmo al momento de discernir entre las clases del conjunto de datos, estas métricas son calculadas a partir de la matriz de confusión del clasificador la cual indica cuantos elementos fueron asignados correctamente o no a su clase asociada.

		Valor Predicción	
		Positivo	Negativo
Valor Real	Positivo	Verdadero Positivo (TP)	Falso Negativo (FN)
	Negativo	Falso Positivo (FP)	Verdadero Negativo (TN)

Tabla 1: Matriz de confusión para un problema binario. En paréntesis, las siglas en ingles de los términos.

Fuente: Confección propia.

La Tabla 1 corresponde a una matriz de confusión de un problema binario general, en la cual el valor positivo corresponde a que el dato pertenece a la clase y negativo que no pertenece a la clase (por ejemplo, si el estímulo corresponde a un P300 o no). Las casillas de la matriz indican la cantidad de elementos catalogados en base a 4 casos, valores positivos que fueron clasificados positivos, valores positivos clasificados como negativos, valores negativos clasificados como positivos y valores negativos clasificados como negativos. En un caso ideal, la matriz solo contiene elementos en su diagonal y estos corresponden a la cantidad de datos positivos y negativos clasificados correctamente.

A partir de los 4 valores presentes en la matriz es posible calcular varias métricas para interpretar el rendimiento de un clasificador, las más utilizadas corresponden a la exactitud, precisión, sensibilidad, especificidad y F1-Score.

La **Exactitud** corresponde a la razón entre los valores correctamente clasificados con respecto al universo de valores en la matriz, en otras palabras, es la cantidad de predicciones positivas que fueron correctas. Su valor se determina a partir de la ecuación 1.

$$Exactitud = \frac{TP + TN}{(TP + TN + FN + FP)} \quad (1)$$

La **Precisión** corresponde a la razón entre los valores positivos clasificados correctamente entre todos los valores positivos asignados. Su valor se determina a partir de la ecuación 2.

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

La **Sensibilidad** o *recall* corresponde a la razón entre los valores positivos clasificados correctamente con respecto a todos los valores positivos que fueron clasificados correcta e incorrectamente. Su valor se determina a partir de la ecuación 3.

$$Sensibilidad = \frac{TP}{TP + FN} \quad (3)$$

La **Especificidad** es similar a la sensibilidad, no obstante, esta mide los valores negativos asignados correctamente con respecto a los valores negativos clasificados correcta e incorrectamente. Su valor se determina a partir de la ecuación 4.

$$Especificidad = \frac{TN}{TN + FP} \quad (4)$$

El **F1-score** corresponde a una métrica que encapsula la sensibilidad y la precisión en una sola variable, permitiendo conocer qué tan riguroso es el clasificador al momento de catalogar las clases; para valores más cercanos a 1, se entiende que la cantidad de falsos positivos y negativos decrece. Su valor se determina a partir de la ecuación 5.

$$F1Score = \frac{2 * Precision * Recall}{Precision + Recall} \quad (5)$$

Curva ROC

Las métricas anteriormente descritas no son las únicas que pueden ser utilizadas para determinar el rendimiento del clasificador, la curva ROC (Receiver Operating Characteristic) corresponde a una gráfica que representa la sensibilidad en base a la tasa de falsos positivos (o 1-especificidad) la cual permite determinar el punto de corte que el clasificador utilizada para separar las clases en base a las restricciones que se le busque imponer al modelo, por ejemplo, la tolerancia a los falsos positivos. No obstante, la curva ROC también puede ser empleada como métrica comparativa mediante el uso del área bajo la curva (AUC, por sus siglas en inglés) y la representación de la curva en sí.

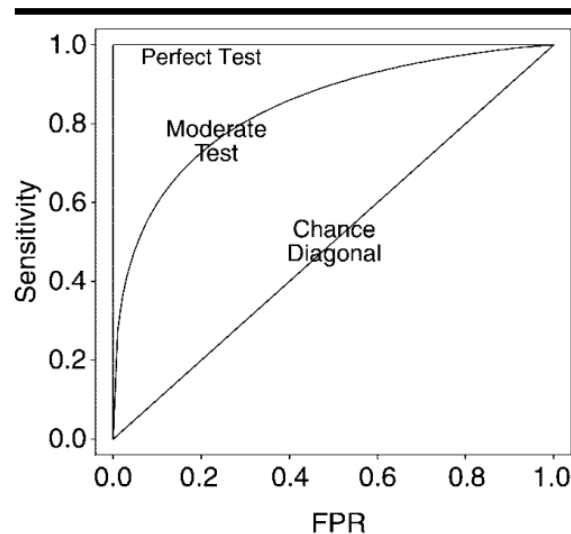


Figura 6: Curvas ROC. En la figura se presentan 3 curvas ROC correspondientes a tres clasificadores distintos: *Change diagonal* corresponde a un clasificador aleatorio que no tiene capacidad de discriminar positivos de negativos, mientras que una curva de tipo *Perfect test* corresponde a un clasificador que distingue entre las clases perfectamente, un clasificador intermedio entre ambos se representa por la curva de *Moderate test*. Fuente: [Obuchowski, 2003]

La figura 6 corresponde a un gráfico de 3 ejemplos de curvas ROC, un clasificador aleatorio es aquel que no puede discriminar entre las clases, este es usado de referencia para catego-

rizar el rendimiento del modelo, en el espacio ROC se caracteriza por ser una recta diagonal (*Chance diagonal*) con $AUC=0.5$, mientras que para un caso óptimo se tiene un clasificador ideal que identifica los datos correctamente, este es representado por una curva recta en el límite superior del gráfico (*Perfect test*) con $AUC=1.0$, mientras que un resultado intermedio entre ambas curvas se representa por una curva cuyo $AUC=0.75$ (*Moderate test*). Los resultados de un clasificador se determinan a partir de la cercanía a una de las 2 curvas descritas, y del valor bajo la curva corresponde al rendimiento de un clasificador, el cual es comparable entre los casos bases así como con respecto a otros clasificadores, no obstante, distintas curvas pueden tener valores similares de área bajo la curva, por lo que, por sí solo no es determinante al momento de comparar múltiples clasificadores.

2.3. Lenguajes de programación y frameworks

2.3.1. BCI2000

BCI2000² es un software de código abierto desarrollado en C++ para el sistema operativo Windows, se ha diseñado como una herramienta multipropósito para el desarrollo de implementaciones de BCIs, para ellos, presenta una arquitectura modular que permite realizar el registro de las ondas cerebrales, digitalización, procesamiento, y la traducción/interpretación de los datos, para ello cuenta con una suite de implementaciones de ciertos BCI conocidos, dentro de los que se incluye el P300 Speller. Además, contiene un módulo para el análisis *offline* y exportación de los datos [Schalk *et al.*, 2004]. Este último es particularmente útil, dado que es empleado para la captura de datos durante el EEG para realizar análisis *offline* en otros lenguajes de programación, en particular, tiene compatibilidad con MATLAB mediante el uso de plugins.

La compatibilidad con Python es posible haciendo uso del programa BCIPy2000, el cual se encarga de conectar los binarios de BCI2000 con una interfaz basada en Python para el desarrollo de implementaciones de BCIs, actualmente es compatible con Python 3.6.8.

P3 Speller

Corresponde a un *framework* perteneciente a BCI2000 que permite realizar la captura y filtrado de datos a partir de una implementación genérica del P300 Speller, permite configurar los elementos de la matriz a desplegar, definir las palabras objetivo entre otros, su uso está pensado para ser añadido a otros desarrollos propios de BCI2000 o externos mediante sus funciones de transferencia de datos. Los datos almacenados mediante este protocolo contienen la señal, parámetros empleados para la captura de datos así como también variables que permiten identificar y contextualizar los estímulos capturados.

²<https://www.bci2000.org/>

2.3.2. OpenViBE

OpenViBE³ también es un programa de código abierto desarrollado en C++ disponible en Windows y ciertas distribuciones de Linux. A diferencia de BCI2000 presenta un enfoque centrado en el estudio de la neurociencia, este permite registrar las señales del cerebro, digitalizarlas y visualiza de manera gráfica una interpretación de como los estímulos afectan el cerebro [Arrouët *et al.*, 2005], en el contexto de los BCI, OpenViBE es empleado principalmente para la extracción de datos, no obstante, contiene una interfaz de desarrollo que permite desarrollar sobre las características ya existentes puesto que también presenta demos de BCIs.

Existen plugins para su uso con MATLAB y Python en el ámbito del procesamiento de las señales tanto *online* como *offline*.

2.3.3. MATLAB

MATLAB⁴ o MATrix LABoratory, es un software especializado para el manejo de matrices que emplea el lenguaje de programación M, el cual funciona bajo un modelo de suscripción mensual/anual, con planes especiales para instituciones académicas. Contiene varios módulos para manejo de señales, análisis matemático e inteligencia artificial. Debido a su enfoque mas centrado en las matemáticas es empleado en la investigación, ingeniería y organizaciones debido que se diferencia de otros lenguajes por utilizar notaciones matemáticas para escribir código, resultando mas intuitivo para personas sin conocimientos de programación.

Debido a que es un software propietario, las herramientas a las que se pueden acceder están restringidas por los *toolbox* con los que MATLAB dispone, de la misma forma, integraciones con otros lenguajes de programación se ven limitadas por aquellas que tengan soporte oficial.

EEG Lab

EEG Lab⁵ corresponde a un *toolbox* de MATLAB para el procesamiento y visualización de datos de EEG, MEG, entre otros. Se caracteriza por tener una interfaz gráfica que permite realizar análisis de frecuencia vs tiempo, componentes independientes, estadísticas de ERP, así como también visualizar datos promediados o de tipo *single trial*.

2.3.4. Python

Python⁶ es un lenguaje de programación interpretado y multiparadigma, particularmente versátil por lo flexible que es con respecto a otros lenguajes del mismo tipo, el cual puede ser

³<https://openvibe.inria.fr/>

⁴<https://www.mathworks.com/products/matlab.html>

⁵<https://eeglab.org/>

⁶<https://www.python.org/>

extendido mediante módulos a través de su gestor de paquetes Pip volviéndolo de esta forma un lenguaje multi-propósito, estas mismas propiedades lo han vuelto uno de los lenguajes de programación más utilizados en el área del análisis de datos y *machine learning*.

Numpy

Numpy⁷ es una librería de análisis numérico y manejo matricial de Python que permite la creación y utilización de matrices multidimensionales denominadas numpy arrays las cuales resultan ser más eficientes que la implementación de listas nativas de Python, esto permite añadir funcionalidades presentes en MATLAB como aritmética matricial y álgebra lineal, por lo que esta librería resulta ser una de las más utilizadas para el análisis de datos, de manera que es empleada como dependencia de otras librerías enfocadas en la computación científica.

SciPy

Scipy⁸ es una librería de Python perteneciente a la computación científica que añade sobre las funcionalidades de Numpy, matrices dispersas, más elementos de álgebra lineal, interpolación, optimización, *eigenvalues*, procesamiento de señales entre otros. Debido a lo robusta que es, resulta fundamental para desarrollos de ingeniería, análisis numérico y manejo de datos. En particular, resulta de interés sus funciones para procesar señales, debido a la aplicación de filtros en señales digitales y la capacidad de leer binarios de MATLAB para cargar variables que hayan sido guardadas.

Matplotlib

Matplotlib⁹ es una librería de código abierto para Python enfocada en la generación de gráficos y visualización de datos, inicialmente desarrollado como una alternativa al *toolkit* para la creación de gráficos de MATLAB, se ha convertido en la librería por defecto para la visualización de datos y *Data Science* en Python, al contar con un repertorio de gráficos bastante robusto y que mediante su API pueden ser fácilmente desplegados en interfaces gráficas mediante librerías compatibles.

Scikit-Learn

Scikit-Learn¹⁰ es una librería enfocada en *machine learning* para Python diseñada inicialmente como una extensión de Scipy. Cuenta con algoritmos de clasificación, regresión, *clustering*, entre otros. Es de interés su implementación de SVM y los algoritmos de normalización.

⁷<https://numpy.org/>

⁸<https://scipy.org/>

⁹<https://matplotlib.org/>

¹⁰<https://scikit-learn.org/stable/>

PyQT

PyQT¹¹ es una librería enfocada en el desarrollo de interfaces gráficas, esta es un *binding* del *framework* QT desarrollado en C++, lo que permite adaptar las funcionalidades de este sin la necesidad de reescribir el código fuente. Una de sus ventajas es el uso de abstracciones para interactuar con elementos del sistema operativo, motores de base de datos, networking, etc.

PyQT también permite cargar elementos diseñados mediante QT Designer, una herramienta que permite diseñar rápidamente elementos de la interfaz a partir de un sistema *drag and click*.

¹¹<https://www.riverbankcomputing.com/software/pyqt/>

CAPÍTULO 3

PROPUESTA DE SOLUCIÓN

En la sección anterior se presentó a grandes rasgos el funcionamiento de un BCI, donde 3 actividades claves pueden identificarse: la recopilación de datos mediante la adquisición de señales de onda cerebral, el procesamiento de estos datos para poder extraer las características necesarias que permitan traducirse en acciones concretas y el uso de la información generada para manifestar dichas acciones de manera que el usuario final pueda recibir una retroalimentación con respecto a la acción que busca realizar. En este contexto la presente memoria busca abordar el paso asociado al procesamiento y traducción de los datos para el P300 Speller, a continuación, en la figura 7 se aprecia a grandes rasgos las herramientas empleadas en la solución propuesta.

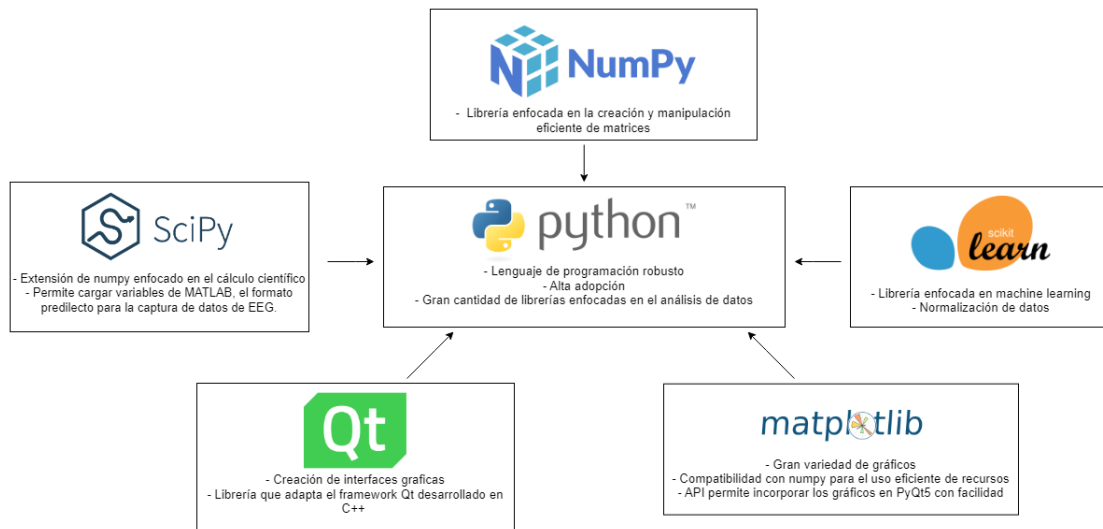


Figura 7: Elementos que componen el *framework* propuesto. Los elementos de los extremos corresponden a librerías de Python.

Fuente: Confección propia.

Con respecto a la figura 7, las versiones respectivas de Python y las librerías asociadas son las siguientes:

- Python 3.12
- Numpy 1.26.1
- Scipy 1.11.3
- Scikit-learn 1.3.2
- PyQt5 5.15.10

■ Matplotlib 3.8.2

Específicamente, el programa debe considerar aspectos como la capacidad de poder modificar parámetros referentes al análisis y mostrar los resultados obtenidos de manera gráfica como aspectos fundamentales, debido a que la relevancia de ciertas características sobre otras puede ser subjetiva, también debe ser posible exportar los resultados para que puedan ser analizados en etapas futuras por el usuario.

3.1. Flujo de trabajo

Un aspecto fundamental para entender la propuesta es conocer cómo es el flujo de trabajo esperado para el procesamiento de datos y traducción en el contexto del P300 Speller.

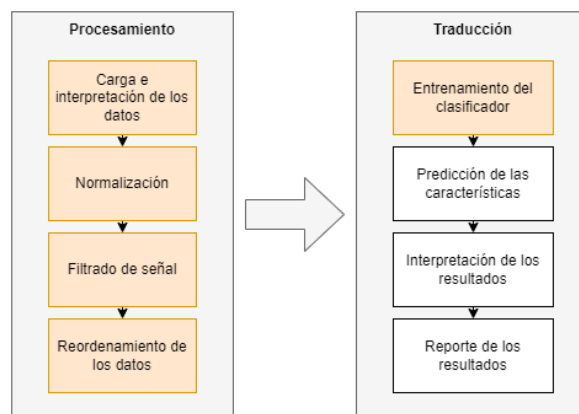


Figura 8: Etapas del funcionamiento del pipeline. En naranja se identifican los procesos en los cuales el usuario puede personalizar parámetros.

Fuente: Confección propia.

La figura 8 muestra las acciones que debe cumplir el *framework* para cada una de las dos etapas descritas con anterioridad, cada tarea marcada en color naranja son acciones sobre las cuales el usuario puede manipular parámetros para la ejecución del programa, por ejemplo, seleccionar el tipo de normalización que se aplicará a los datos o establecer el rango de filtrado de la señal. Es importante mencionar que la forma en cómo el usuario interactuará con dichas tareas es mediante la selección y configuración de parámetros en la interfaz de usuario, de la cual se explicará más adelante en la sección 3.4.

Dado que en la etapa de traducción es necesario entrenar un clasificador para poder detectar las respuestas de tipo P300, hay que tener en consideración que la etapa de procesamiento utiliza un conjunto de datos de entrenamiento como uno de prueba.

3.2. Procesamiento

Este subcapítulo abordará los procedimientos asociados a la carga, interpretación y manipulación de los datos para la obtención de una señal lo más limpia posible previo a la fase de traducción, en la cual se hará uso de los conjuntos de datos para identificar las respuestas deseadas.

3.2.1. Estructura de los datos y codificación

Algo importante a considerar es que el *framework* fue confeccionado para ser compatible con datos capturados mediante el protocolo P3 Speller del framework BCI2000 mencionado en la sección 2.3.1, una de las particularidades de este método es que los datos pueden ser registrados como variables de MATLAB método utilizado por varias bases de datos de EEG. Para su lectura se hará uso de la librería Scipy, dado que contiene el método `loadmat` que permite cargar y mapear estructuras de datos de MATLAB a equivalentes nativas de Python así como de Numpy para casos como matrices y vectores.

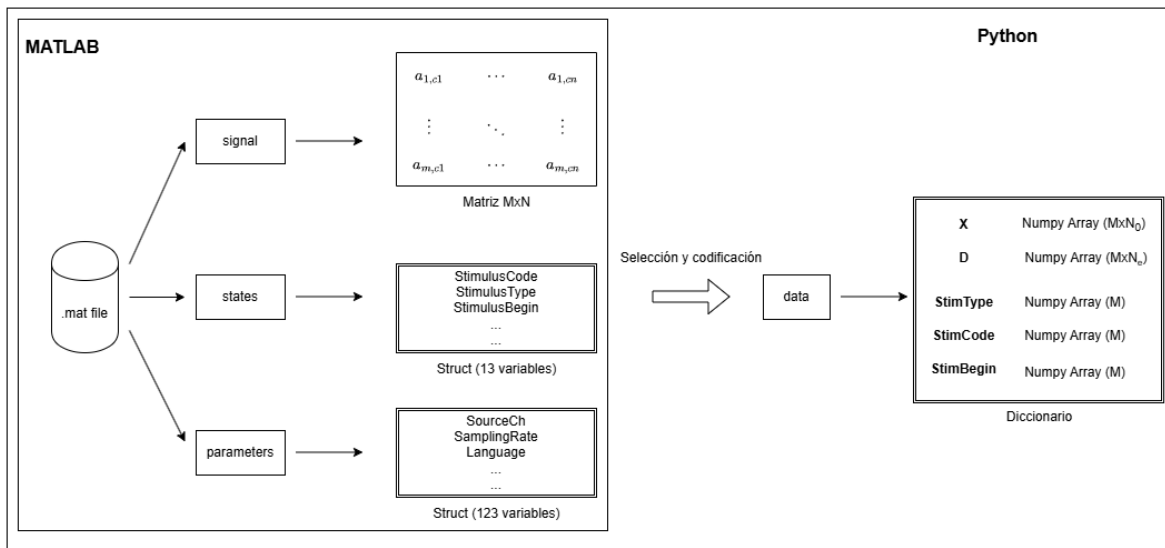


Figura 9: Codificación de los datos. Al lado izquierdo se muestra la estructura original de los datos cargados, los cuales son mapeados mediante SciPy a estructuras de Python, siendo seleccionados y codificados en un diccionario. M , N_0 y N_e corresponden al tamaño de la muestra, la cantidad de canales de EEG utilizados y la duración del ERP respectivamente.

Fuente: Confección propia.

En la figura 9 al lado izquierdo se muestra la estructura en que los datos son guardados bajo el protocolo P3, al cargar estas variables utilizando Scipy las estructuras son mapeadas a Numpy arrays, diccionarios, así como otros tipos de datos nativos de Python, en particular, una gran cantidad de metadatos son registrados en el struct *parameters* que no son útiles para el análisis sino que para dar contexto a cómo los datos fueron capturados, por lo que,

al codificar los datos se crea un diccionario que solo contiene los datos que se utilizarán a futuro. Es importante mencionar que la matriz *signal* es renombrada a *X*, como parte de la codificación.

La codificación en cuestión toma elementos como la señal de EEG y los vectores de prefijo "Stimulus" debido a su relevancia en las fases subsecuentes del *framework*, no obstante, también son utilizadas para computar la matriz *D* utilizada principalmente para debugging y para la creación del gráfico del subconjunto de la señal que contiene las respuestas P300.

Con respecto a ésta última, su importancia es relevante ya que mediante la matemática matricial es posible obtener subconjuntos de la señal que permiten validar el correcto manejo de los datos.

$$X = DA + N \quad (6)$$

En la ecuación 6 se aprecia una forma de representar las señales de electroencefalografía *X*. El primer término *DA* es la matriz de respuesta obtenida para el estímulo objetivo para cada canal. El segundo término *N* corresponde al ruido u otros estímulos. Donde *X* corresponde a la matriz de tamaño $M \times N_0$ siendo *M* el tamaño de la muestra y N_0 la cantidad de canales empleados en la EEG; *D* corresponde a una matriz Toeplitz de tamaño $M \times N_e$ cuya primera columna se define como $D(k, 1) = 1$, donde *k* es el inicio del estímulo *k*-ésimo objetivo con todos los demás elementos iguales a cero, siendo N_e el tiempo de duración del ERP; La matriz *A* de tamaño $N_e \times N_0$ corresponde al conjunto de ERPs para cada canal.

Es posible estimar la matriz *A* mediante el método de mínimos cuadrados, obteniendo la siguiente solución, expresada en la ecuación 7:

$$A = (D^T D)^{-1} D^T X \quad (7)$$

Sumado al diccionario *data*, otros tipos de datos suministrados por el usuario son empleados para complementar la información de la base de datos, como la palabra deletreada en el conjunto de entrenamiento de datos, la cantidad de secuencias y la frecuencia utilizada en la sesión de captura de datos, el tamaño del *trial*, etc.

3.2.2. Normalización

Como se mencionó al final de la sección 3.1, para poder entrenar el modelo de clasificación se contará con 2 conjuntos de datos, para lograr una correcta predicción de las características es esencial normalizar la escala de los datos de forma que minimizar los sesgos por orden de magnitud, en especial considerando que el rango de amplitud de la frecuencia puede oscilar en gran medida.

Mediante funciones provistas por Scikit-learn en su módulo de preprocesamiento, se definen 2 métodos de normalización para los datos:

- Suave: Hace uso de la clase `StandardScaler` para normalizar con media 0 y desviación estándar 1, restringiendo sus valores entre 0 y 1.
- Fuerte: Hace uso de la clase `MaxAbsScaler` para normalizar tomando en cuenta los máximos y mínimos del conjunto de datos, restringiendo sus valores al rango entre -1 y 1.

Ambos métodos normalizan el conjunto de prueba a partir de las métricas utilizadas para normalizar el conjunto de entrenamiento (ya sea $\bar{X}$ y σ o los máximos absolutos, según el método de normalización utilizado).

3.2.3. Filtrado de señales

Para el filtrado de la señal, se hace uso del módulo `signal` de Scipy para implementar un filtro de Butterworth, la forma en que los filtros son aplicados es mediante el cómputo de coeficientes los cuales son utilizados para reconstruir la señal filtrada a partir de la señal original.

Para ello se utilizará el método `butter` que calcula dichos coeficientes a partir del orden del filtro, el tipo, es decir, *lowpass*, *highpass* o *bandpass* y las frecuencias de corte, estos coeficientes pueden ser retornados en múltiples formatos, no obstante, el recomendado corresponde a secciones de segundo orden (SoS) para evitar problemas de precisión en el resultado, esto repercute en la forma en que la señal es reconstruida, dado que existen métodos para cada uno de los formatos, siendo `sosfilt` aquel que a partir de los coeficientes en formato SoS y la señal original devuelve la señal filtrada.

3.2.4. Reordenamiento de los datos

El último paso previo a la clasificación corresponde al reordenamiento de los datos, más específicamente, de la señal, de manera de facilitar el trabajo del modelo de ML. La forma en que esto se materializa es mediante la alteración de la disposición de la señal, y el promedio de señales si es que el usuario lo requiere.

Reestructuración de la señal

La disposición original de la señal discutida en la sección 3.2.1 corresponde al comportamiento de la actividad cerebral a lo largo del tiempo para cada uno de los canales utilizados en la EEG, dado que esta estructura es propia del protocolo de captura de datos para EEG, es necesario adaptar su disposición para su uso bajo el paradigma del P300 Speller.

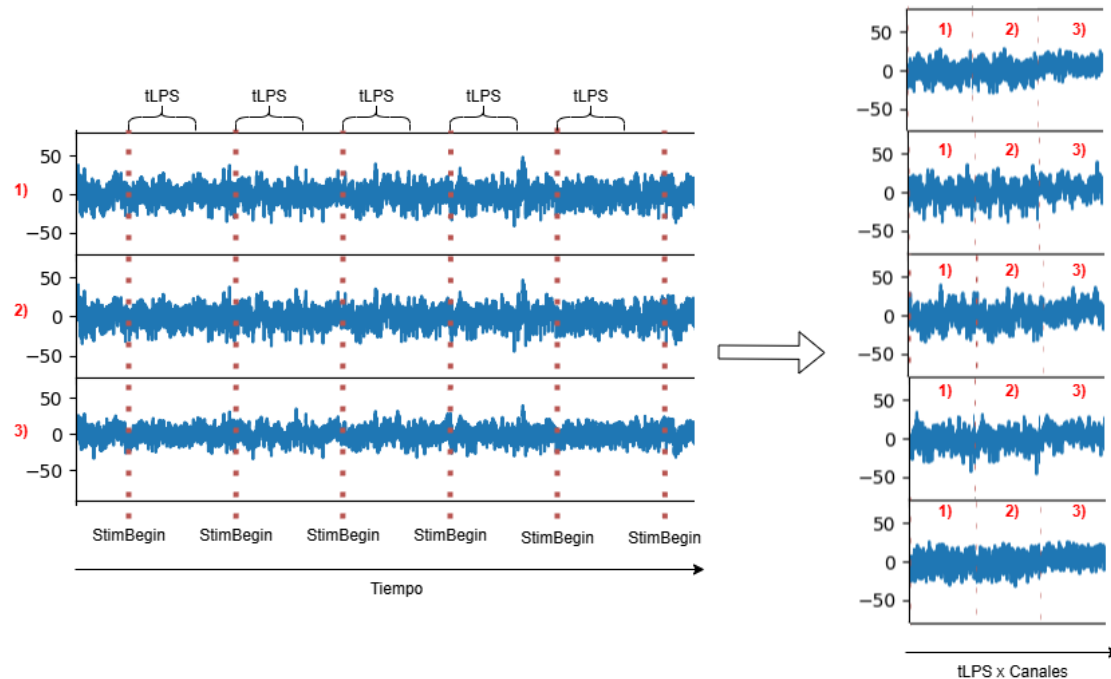


Figura 10: Recreación del reorden de la señal. Por cada canal existente, se extraen segmentos de una ventana de tiempo tLPS (Definida internamente) los cuales son agrupados, de esta forma, la nueva representación describe el estado de las distintas señales en un intervalo de tiempo específico.

Fuente: Confección propia.

En la figura 10 se aprecia un ejemplo de cómo la organización de los datos es llevada a cabo. Para ello, se calcula el largo de la ventana tLPS de tiempo a partir de información de la captura de datos como la frecuencia y el tamaño del *trial* y se agrupan bloques de dicho largo por cada canal presente, de forma que cada fila de la matriz describe el estado de la actividad cerebral para dicha ventana de tiempo. Este intervalo es lo suficientemente extenso como para registrar el estímulo P300 a pesar de su retraso de 300ms en aparecer, lo que permite asociar de mejor manera cuando el Speller registró o no una respuesta de tipo P300.

Promediar secuencias

Tal como se comentó en la sección 2.1.1 con respecto a EEG y el potencial P300, los datos extraídos mediante EEG deben ser procesados para ser utilizados como consecuencia de la

sensibilidad ante otros estímulos o interferencias propias de las características del individuo, lo que dificulta la detección de un estímulo P300 en análisis de tipo *single trial* (con solo una repetición o secuencia por letra), uno de los procedimientos comúnmente utilizados para la obtención de una señal con mayor definición corresponde al promedio de secuencias, debido a que permite reducir la presencia de ruido, interferencias y presencia de otros estímulos al tener varios registros donde solo la presencia de un P300 es una constante, lo que al promediar permite aislar de mejor manera dicho estímulo del resto. Para promediar secuencias, se debe identificar a lo largo de los datos en qué puntos un mismo flash se repite para ser promediada en base a la cantidad de secuencias que el usuario definió mediante la interfaz gráfica, de manera que estas nuevas señales amplificadas sustituyan a las anteriormente presentes. Este procedimiento se debe realizar teniendo en consideración la cantidad de secuencias por letra de forma de no promediar secuencias pertenecientes a una letra distinta.

Específicamente, la forma en que las secuencias son promediadas es considerando el total de repeticiones con respecto a la cantidad a promediar, determinando la cantidad de nuevas señales que contienen las secuencias promediadas. Por ejemplo: para 15 repeticiones y 5 secuencias promediadas, se obtienen 3 nuevas señales que sustituyen a las anteriores 15; por lo cual, para valores no enteros del cociente anterior, pueden existir secuencias no consideradas y descartadas del promedio.

3.3. Traducción

La fase de traducción considera los aspectos referente a la clasificación y predicción de características a partir de los datos procesados con anterioridad, los resultados de esta fase son empleados para identificar las filas y columnas objetivo en el *speller* con el fin de poder construir la palabra final. A partir de esta información, se calculan métricas de desempeño que son informadas al usuario para determinar la efectividad en la sesión.

3.3.1. Clasificación y predicción

Para la clasificación y predicción de características, se utilizará un algoritmo de clasificación bastante conocido y que obtiene buenos resultados para este problema en particular, *Support Vector Machine*, el cual fue descrito en la sección 2.2.3.

La implementación utilizada es la disponible en la librería Scikit-learn en su modulo SVM, la clase `SVC` dispone de los métodos necesarios para entrenar y predecir características a partir de los datos y las clases a la que estos pertenecen, en este caso, siendo la señal reorganizada y el vector `StimulusType` del conjunto de entrenamiento los utilizados para entrenar el modelo y la señal del conjunto de prueba utilizada como *input* para que el modelo pueda predecir las clases en la forma de un vector `StimulusType`.

La clase `SVC` al momento de ser instanciada permite definir parámetros que determinan el entrenamiento del modelo, como por ejemplo el tipo de kernel a utilizar y parámetros

asociados al kernel empleado, a continuación se indican los parámetros que son modificables por el usuario:

- Kernel: Las opciones disponibles son: 'linear', 'poly' y 'rbf'
- C: Corresponde al valor de regularización de la función a minimizar. Admite valores positivos distintos de 0, por defecto es 1.0.
- Degree: Corresponde al grado del polinomio utilizado por el kernel 'poly'. Solo admite valores enteros positivos, por defecto es 3.
- Gamma: Parámetro utilizado por los kernel 'poly' y 'rbf'. Las opciones son 'auto' 'scale' o un valor no negativo, por defecto es 'scale'.
- coef0: Término independiente utilizado en el kernel 'poly'. Admite valores positivo y negativos, por defecto es 0.0.

3.3.2. Interpretación y reporte de resultados

A partir del vector *StimulusType* generado por el modelo de clasificación, se debe interpretar su significado, esto mediante la obtención de la palabra que resulta del contraste con las filas/columnas del *speller*, sumado a ello, también deben calcular métricas de desempeño que permitan validar el correcto funcionamiento del clasificador.

La palabra es calculada mediante el uso de los vectores '*StimulusType*' y '*StimulusCode*' generados en la adquisición de los datos. A partir del contraste de ambos vectores es posible conocer el orden de iluminación de las filas y columnas que producen las letras que componen la palabra final; dado que el primer vector indica el instante en el tiempo que se reporta una respuesta P300, mientras que el segundo indica la fila o columna iluminada en el *speller* en un momento determinado. Teniendo en consideración que un ciclo o secuencia solo se compone de 12 iluminaciones en la matriz de 6x6 del BCI, es posible ir acotando los instantes en donde están las 2 instancias que producen una letra, de manera de no cruzar respuestas que pertenezcan a letras distintas.

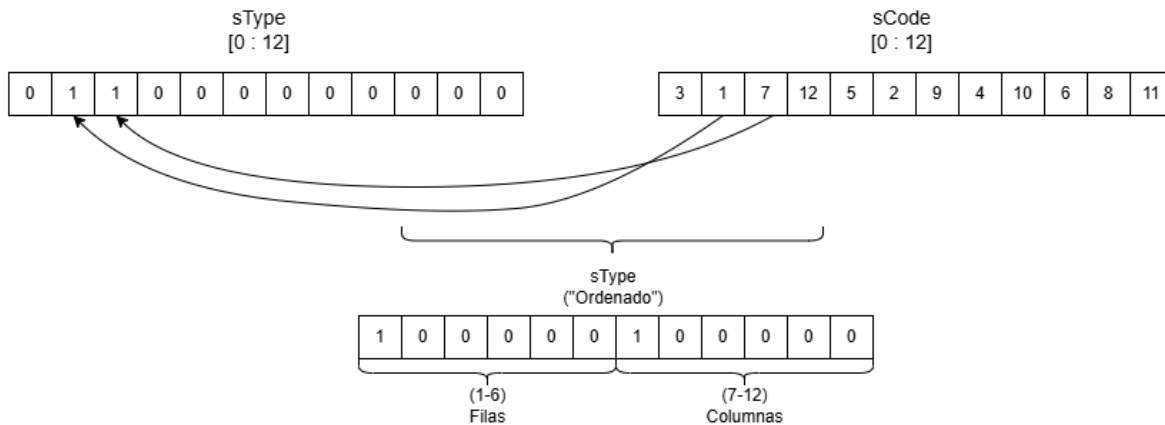


Figura 11: Identificación de las filas/columnas en base al vector `StimulusType` obtenido de la SVM para una secuencia. Al cruzar el número de fila/columna iluminado en un instante particular con el instante en que un P300 es registrado, es posible obtener que fila/columna se ilumina en dicho momento.

Fuente: Confección propia.

En la figura 11 se aprecia la forma en la que se identifica la fila y columna que compone una letra, dado que cada casilla es directamente asociable entre ambos vectores, es posible identificar la fila y columna del speller que resultaron producto de la detección de un P300. Al dividir ambos vectores en varios segmentos de largo 12 y realizando los cruces respectivos es posible obtener cada una de las letras que componen la palabra.

A partir de este punto, se calculan las métricas de desempeño comenzando con la exactitud de la palabra obtenida a partir del clasificador con respecto a la palabra objetivo, otras métricas relevantes son aquellas calculables a partir de la matriz de confusión, como *precision*, *recall* y F1 score, y finalmente la curva ROC con su respectiva área bajo la curva para determinar el rendimiento del clasificador.

La matriz de confusión es obtenida mediante el método `confusion_matrix` utilizando el vector `StimulusType` original y el obtenido del clasificador, mientras que la curva ROC es obtenida utilizando el método `RocCurveDisplay.from_estimator` a partir de la instancia del clasificador que contiene el modelo entrenado, la matriz que contiene la señal y el vector `StimulusType` original; y su área bajo la curva utilizado para ordenar las curvas en base al rendimiento es obtenida con el método `roc_auc_score` que recibe el vector `StimulusType` original y el entregado por el clasificador. Todos los métodos listados anteriormente pertenecen al módulo `metrics` de `Scikit-learn`.

Debido a que el *framework* puede operar bajo distintos valores de secuencias promediadas, los resultados deben ser almacenados de manera que su identificación no sea ambigua y el reporte final sea preciso. Para ello, se utiliza un diccionario anidado que almacena los resultados en base al sujeto y el número de secuencias promediadas.

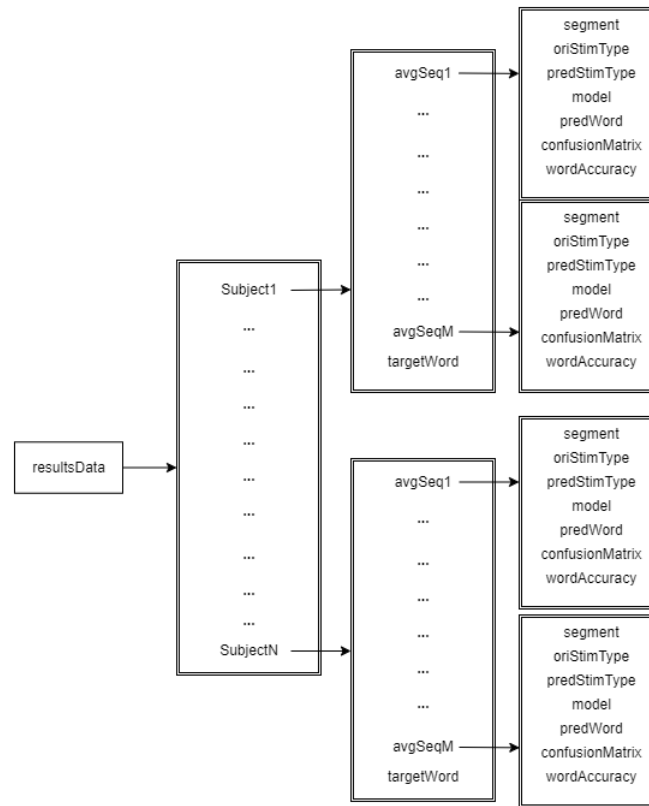


Figura 12: Estructura del diccionario de resultados. Este corresponde a una serie de diccionarios anidados en base al sujeto y la cantidad de secuencias promediadas.

Fuente: Confección propia.

La figura 12 describe la estructura utilizada por el diccionario para almacenar los resultados y datos necesarios para calcular las métricas anteriormente descritas. Ésta estructura almacena los datos de manera íntegra, por lo que para el reporte final, estructuras auxiliares más concretas son utilizadas.

El reporte de resultados considera resultados a nivel global e individual para cada sujeto, las métricas a mostrar son la variación de la precisión de la palabra obtenida del modelo y el F1 score, ambas en base a la cantidad de secuencias promediadas, así como también la peor y mejor curva ROC obtenida, esto para ambas secciones, la pantalla individual también muestra la palabra objetivo para tener un mejor contexto.

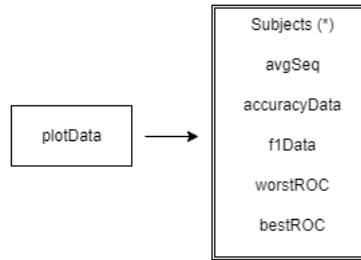


Figura 13: Estructura de los diccionarios auxiliares. (*) La llave Subjects no esta presente en todos los diccionarios.

Fuente: Confección propia.

La figura 13 muestra la estructura seguida por los diccionarios empleados para los gráficos de la interfaz, en si, sus estructuras son idénticas, no obstante el diccionario que utilizado para la pantalla global contiene una lista con los sujetos y las llaves accuracyData y f1Data son matrices a diferencia de los diccionarios individuales que son arreglos de una dimensión.

3.4. Interfaz

A lo largo de las secciones anteriores se ha discutido respecto a los procesos que componen al *framework* y como el usuario puede personalizar configuraciones para ciertos elementos de este, de forma de que se ejecute el tipo de análisis deseado. En ese sentido, la interfaz de usuario debe solicitar cada una de estas configuraciones, validar que sean correctas y agruparlas para ser utilizadas en cada proceso que lo requiera, no solo eso, también debe permitir al usuario poder seleccionar la ubicación de los archivos que contengan los conjuntos de datos a utilizar y, finalmente, presentar los resultados del análisis después de ejecutar el *pipeline* de procesos.

Figura 14: Pantalla principal de la interfaz de usuario.

Fuente: Confección propia.

En la figura 14 se aprecia la pantalla principal de la interfaz de usuario, en la parte superior se encuentran las 3 categorías principales que se corresponde con los pasos del *framework*, a la izquierda están los campos requeridos para la interpretación inicial de los datos y el promedio de secuencias para el reorden de estos; en la mitad están los campos que permiten regular el cómo la señal es filtrada y en la parte de derecha están los campos para normalizar y configurar el modelo de clasificación a utilizar. En la parte inferior se encuentra la sección dedicada a la selección de los datos que conformarán el análisis, ésta también permite previsualizar el estado de la señal inicial para determinar la viabilidad del conjunto para el análisis por parte del usuario como se puede ver en la figura 15.

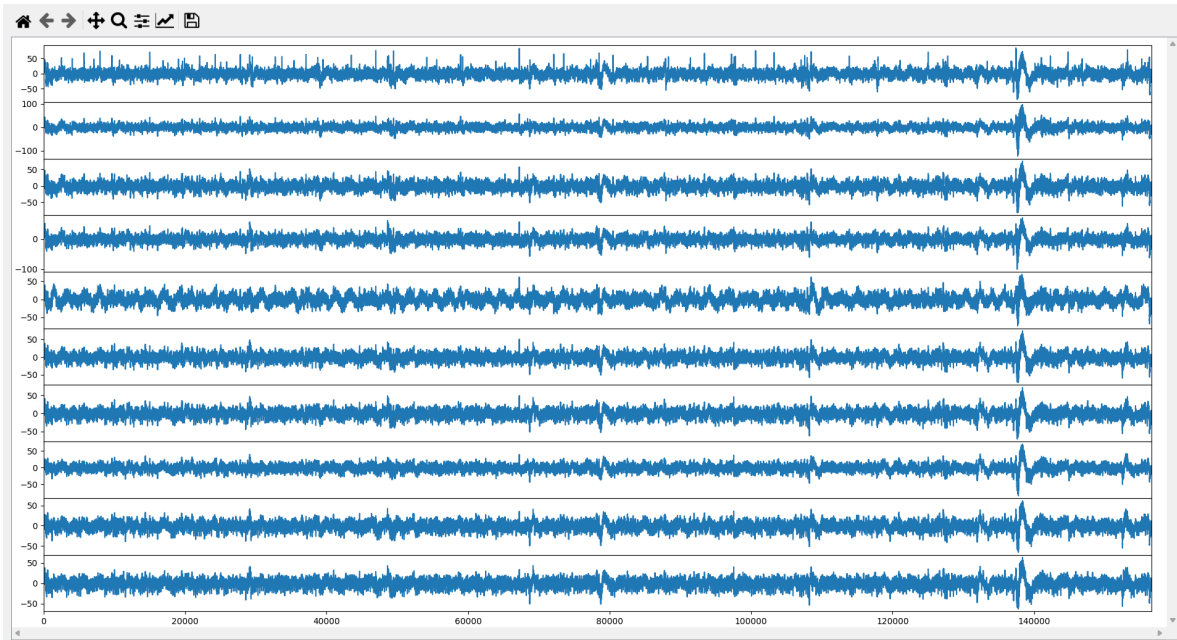


Figura 15: Visualizador de señal. Para un conjunto de datos seleccionado, el visualizador muestra la señal perteneciente a los datos de entrenamiento para cada uno de los canales presentes en la muestra.

Fuente: Confección propia.

La forma en que la interfaz se comunica con el resto del proyecto es mediante el uso de diccionarios que contienen la información que es posible recabar en la interfaz para ser utilizados al momento de la ejecución del *pipeline*.

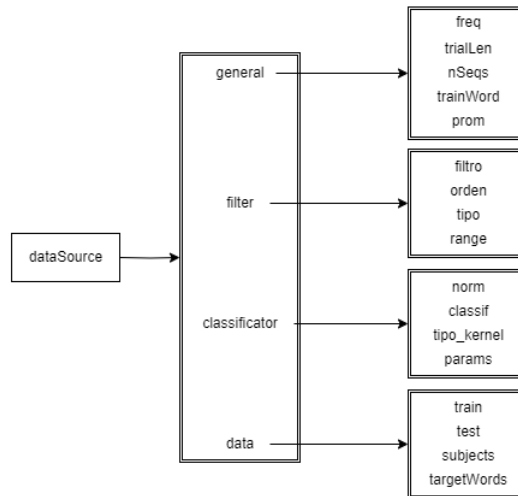


Figura 16: Diccionarios creados a partir de la interfaz.

Fuente: Confección propia.

La figura 16 muestra los diccionarios con los datos recopilados a partir de la interfaz, estos son agrupados en un diccionario para un rápido acceso.

La pantalla de resultados se crea a partir de la respuesta obtenida del pipeline y la creación de las estructuras auxiliares definidas al final de la sección 3.3.2, las cuales son creadas a partir de las plantillas definidas en las figuras 17 y 18. La opción de exportar resultados, permite generar un archivo csv con los resultados y un archivo pickle que contiene el diccionario definido en la figura 12.

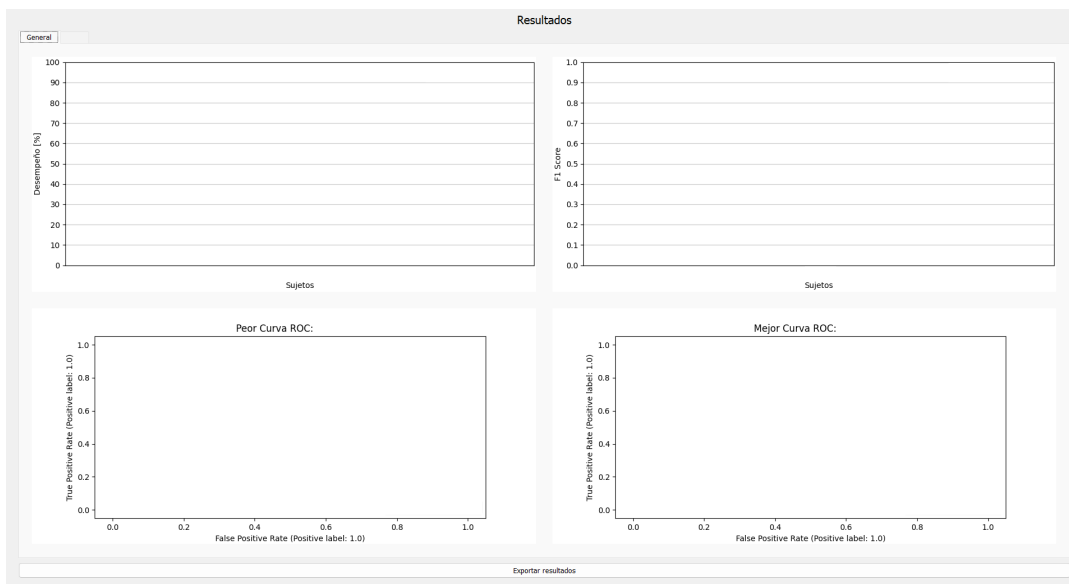


Figura 17: Plantilla de la pantalla de resultados generales. En los gráficos de la parte superior se muestran los resultados para el mejor sujeto, el peor y el sujeto promedio. La métricas utilizadas son el desempeño, F1 Score y las curvas ROC del mejor y peor modelo entre el universo de todos los sujetos.

Fuente: Confección propia.

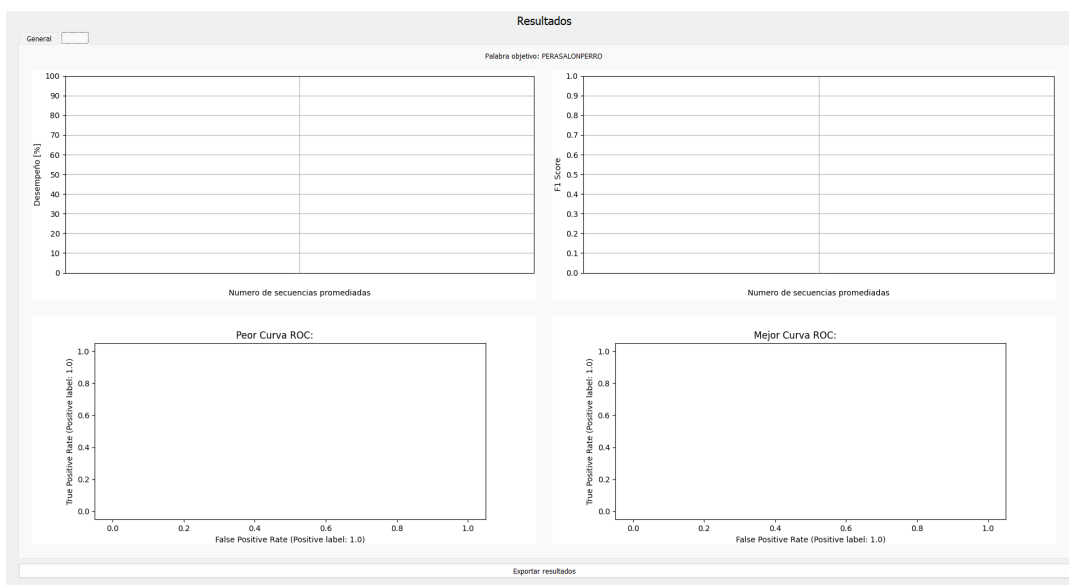


Figura 18: Plantilla de la pantalla de resultados para sujetos individuales. La pantalla muestra la palabra objetivo y los gráficos para el desempeño y F1 Score de un solo sujeto a elección. Además, se presentan la mejor y peor curva ROC del modelo entrenado con los datos del sujeto.

Fuente: Confección propia.

CAPÍTULO 4

VALIDACIÓN DE LA SOLUCIÓN

En el capítulo anterior se introdujo la propuesta de software a desarrollar, con cada uno de los componentes que lo conforman y la función que cumplen de cara al problema a solucionar. En este capítulo se presentará la metodología seguida para las pruebas efectuadas y sus respectivos resultados, sumado a ello se presentarán los antecedentes necesarios para contextualizar cómo dichas pruebas fueron llevadas a cabo.

4.1. Metodología

Para validar el funcionamiento del *framework* se realizarán 2 tipos de pruebas, la primera enfocada en verificar el correcto funcionamiento de la aplicación, entendiéndose, el correcto manejo y tratamiento de los datos, mientras que el segundo tipo de pruebas estará enfocado en evaluar el rendimiento a partir de una comparación con una implementación similar desarrollada en MATLAB.

Específicamente, las pruebas de funcionamiento evaluarán las siguientes características:

1. El filtro de señal es aplicado correctamente. Al comparar la señal previo y posterior al uso del filtro, estas son notoriamente distintas.
2. La normalización de los datos es aplicada correctamente. Las escalas de los datos previo y posterior de la normalización son distintas y corresponden al tipo de normalización utilizada.
3. Al promediar secuencias, los resultados deben mejorar para la mayoría de los sujetos.

Para las pruebas comparativas se evaluarán los siguientes aspectos:

1. El desempeño de los resultados. Para ambos experimentos, se considera como ganador aquel que presente mayor porcentaje de la palabra identificada correctamente.
2. El tiempo de ejecución. Para ambos experimentos, se considera como ganador aquel que obtenga el menor tiempo de ejecución.

Las configuraciones de la aplicación utilizadas para cada prueba serán mencionadas al momento de presentar los resultados.

El hardware utilizado para las pruebas fue el siguiente:

- CPU: Intel Core i5 11400H @ 2.7GHz
- RAM: 32GBs DDR4 @ 3200MHz
- OS: Windows 11 Home

Datos

Para efectuar las pruebas, se utilizará una base de datos pública con auditoría del Laboratorio de Imagenología de la Universidad Autónoma Metropolitana, México [Ledesma Ramírez *et al.*, 2010]. Está conformada por 30 individuos sin enfermedades de base identificados con siglas de 3 letras, haciendo uso del protocolo P3 para la captura de datos de EEG en BCI2000.

Esta base de datos esta conformada por 4 conjuntos de datos por sujeto, 2 sesiones de carácter guiado y 2 sesiones libres, cada sesión esta compuesta por 3 actividades de deletreado, a excepción de la última en la cual la cantidad de actividades es variable. Las palabras deletreadas en la sesiones guiadas son las mismas para cada sujeto, mientras que las palabras objetivo de las sesiones libres son distintas tanto por sesión como por individuos y son parte de la base de datos.

De la totalidad de la base de datos se utilizarán los datos de 22 sujetos, los cuales tenían todos los archivos necesarios para la ejecución del experimento: ACS, APM, ASG, ASR, DCM, DLP, DMA, ELC, FSZ, GCE, ICE, JLD, JLP, JMR, JSC, JST, LAC, LAG, LGP, PGA, WFG y XCL. Además, para el conjunto de entrenamiento y prueba se hará uso de los datos de la primera y tercera sesión, correspondientes a una sesión guiada y libre respectivamente.

Implementación de MATLAB

Para realizar la comparativa de desempeño, se hará uso de una implementación utilizada para el estudio de métodos que permitan mejorar los resultados de clasificación para análisis de tipo *single trial*, es decir, sin secuencias promediadas [Saavedra *et al.*, 2019]. Esta implementación hace uso de métodos de filtrado y otras técnicas más complejas que las desarrolladas, no obstante, es posible hacer uso de las mismas configuraciones que las utilizadas en el *framework*, lo cual permite que su desempeño pueda ser comparado. El código al no ser de público fue provisto por la profesora Carolina Saavedra.

La versión de MATLAB utilizada para correr el código corresponde a MATLAB R2023b Update 3 (23.2.0.2409890).

4.2. Resultados

4.2.1. Pruebas de funcionamiento

Filtrado de señal

Para esta prueba se utilizó la siguiente configuración:

- Filtro: Highpass 10Hz, Lowpass 30Hz, Passband entre 10 y 30 HZ
- Sujetos: APM

El objetivo de esta prueba es verificar que los filtros de señal son aplicados correctamente, esto mediante la comparación de la onda original con su contra-parte filtrada.

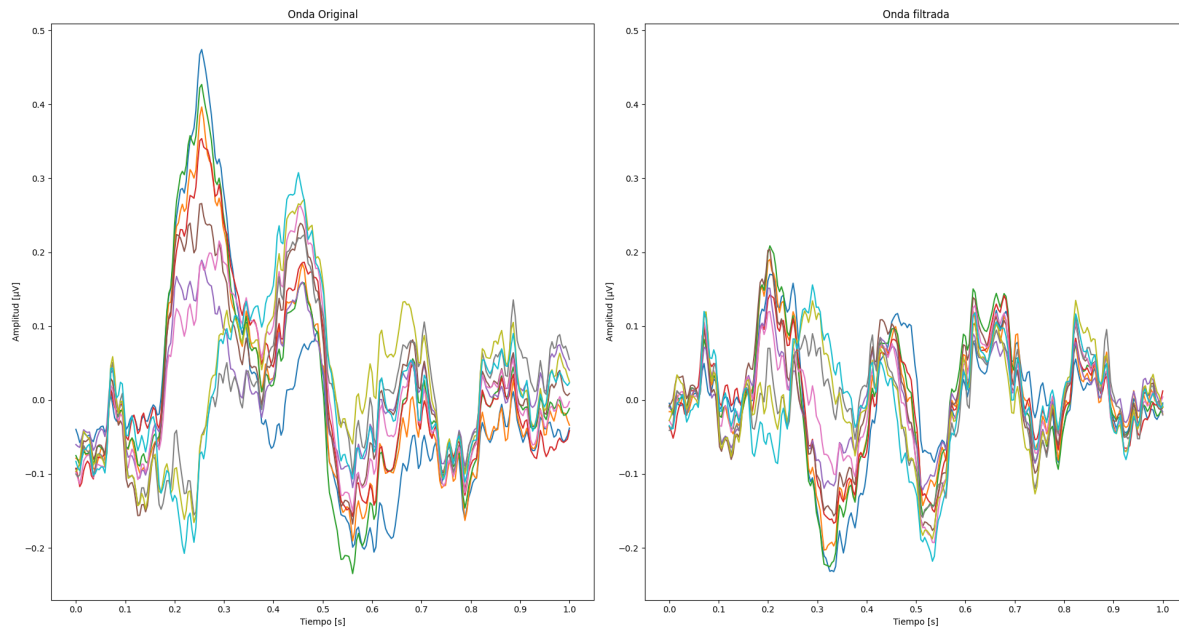


Figura 19: Onda resultante al aplicar filtro de Butterworth de tipo paso alto. El filtro fue configurado para excluir frecuencias bajo los 10Hz.

Fuente: Confección propia.

La figura 19 describe el funcionamiento del filtro en modalidad paso alto de 10Hz, eso quiere decir que frecuencias que se encuentran por debajo de ese valor son excluidas de la señal, esto produce que la onda resultante tenga periodos mas cortos produciendo que la onda resultante se compacte.

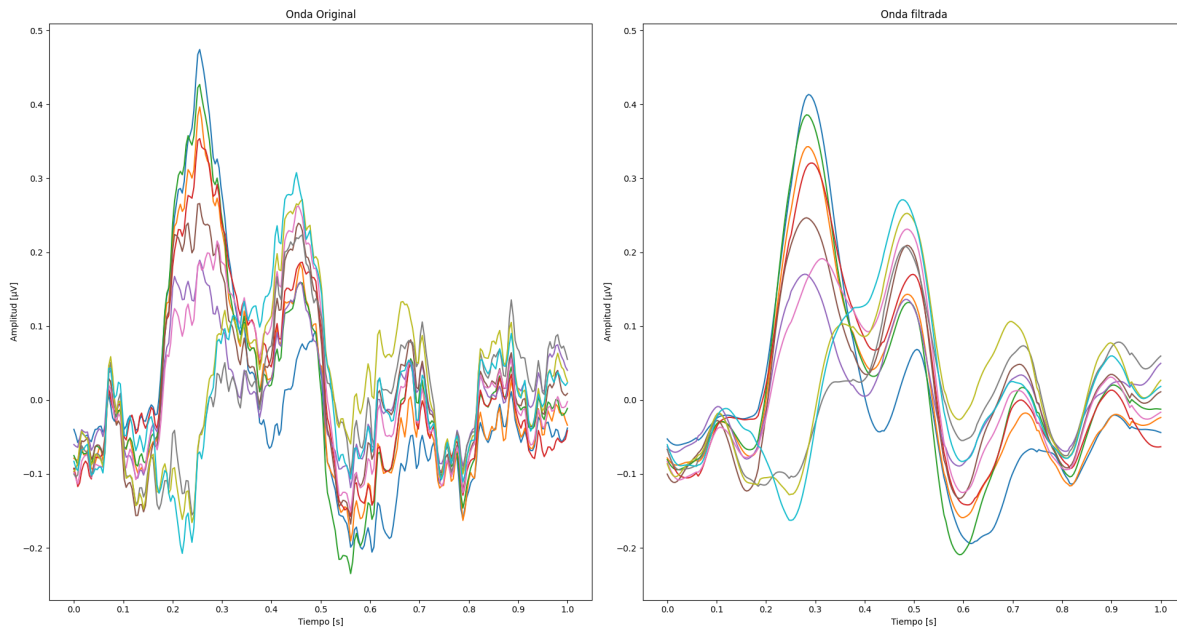


Figura 20: Onda resultante al aplicar filtro de Butterworth de tipo paso bajo. El filtro fue configurado para excluir frecuencias sobre los 30Hz.

Fuente: Confección propia.

Por otra parte, la figura 20 describe el funcionamiento del filtro paso bajo de 30Hz, lo cual excluye las frecuencias por sobre este valor, produciendo que la onda resultante sea más suave que la original, aplanando las crestas.

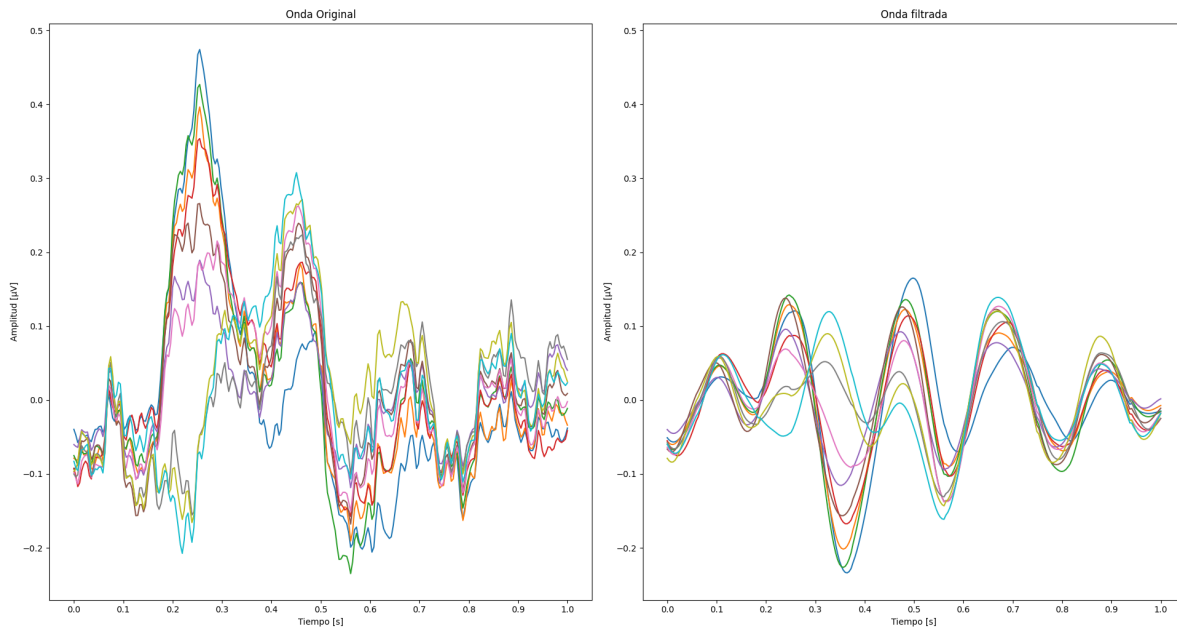


Figura 21: Onda resultante al aplicar filtro de Butterworth de tipo pasa-banda. El filtro fue configurado para excluir frecuencias fuera del rango entre los 10Hz y 30 Hz.

Fuente: Confección propia.

Finalmente, la figura 21 describe el funcionamiento de filtro de paso banda entre los 10Hz y 30 Hz, dado que este filtro se puede interpretar como la aplicación de los 2 filtros anteriores al mismo tiempo, la onda resultante presenta características de cada uno de estos.

Para todos los casos descritos el comportamiento es el esperado, lo cual indica que los filtros son aplicados correctamente.

Normalización

Para esta prueba se utilizó la siguiente configuración:

- Normalización: Suave y Fuerte
- Sujetos: APM

El objetivo de esta prueba es validar que al aplicar las respectivas normalizaciones, los datos son escalados correctamente.

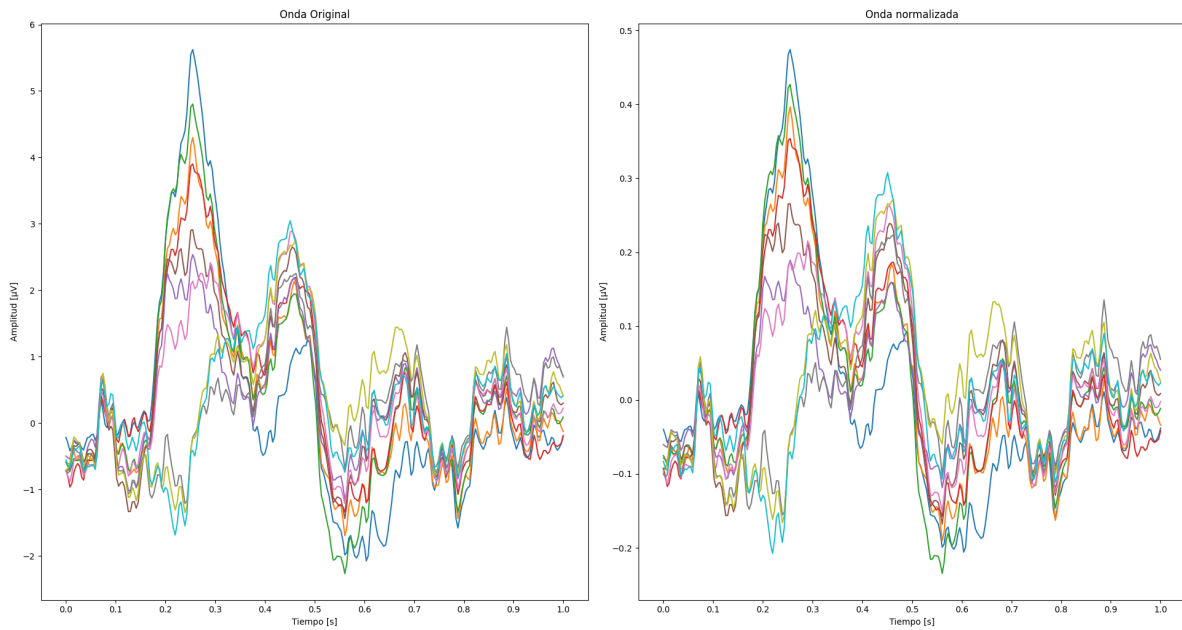


Figura 22: Onda resultante al aplicar normalización de tipo suave.
Fuente: Confección propia.

La figura 22 muestra la onda original y resultante después de haber utilizado la configuración Suave de normalización. A grandes rasgos la distribución de los datos permanece casi intacta, resaltandose en el eje vertical la diferencia en escala que toman los nuevos valores, lo que se condice con lo esperado

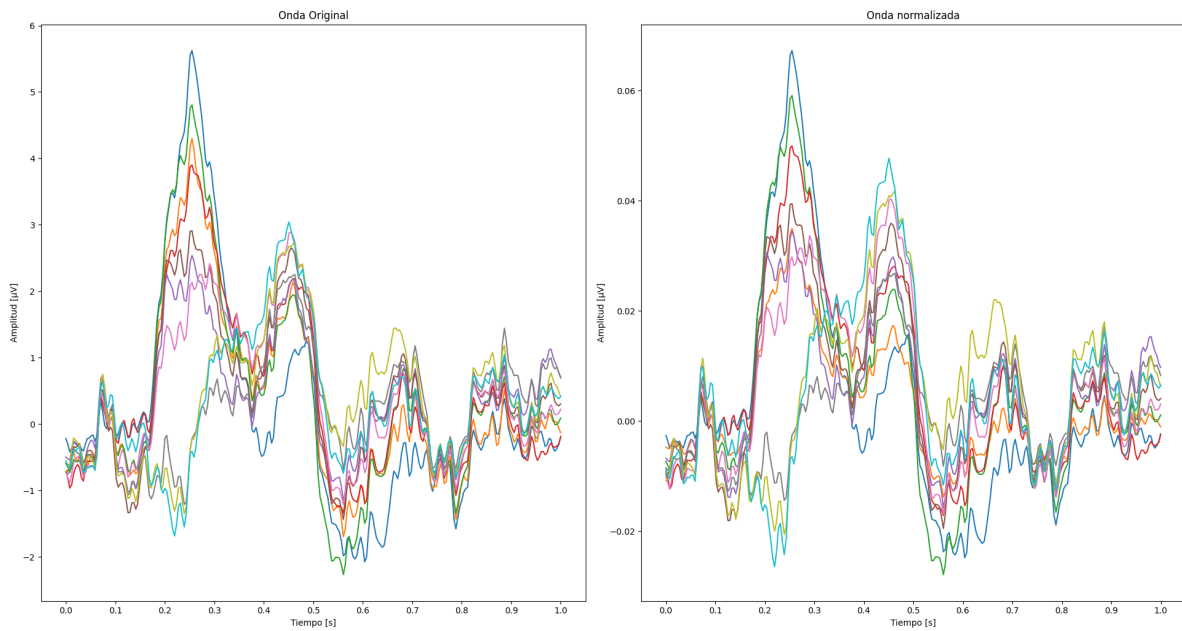


Figura 23: Onda resultante al aplicar normalización de tipo fuerte.
Fuente: Confección propia.

La figura 23 corresponde a la comparativa de la señal original y la resultante posterior a la normalización utilizando la configuración fuerte. Similar al caso anterior, la distribución o forma general de la señal es comparable, no obstante, varias diferencias son perceptibles, esto presumiblemente por el cambio en la escala de los datos, respecto a esto ultimo, los valores se encuentran dentro del rango esperado, sin embargo, dado que se utilizan los máximos absolutos para normalizar y estos no se encuentran representados, se puede concluir que son valores que sesgan los datos.

Promedio de secuencias

El objetivo de esta prueba es validar el supuesto de que en general, al promediar mayor número de secuencias se obtiene una señal con menos ruido, lo que repercute en una mejora del desempeño de los resultados. De esta forma, el método empleado para promediar secuencias esta bien aplicado.

Para esta prueba se utilizó la siguiente configuración:

- Filtro: Passband entre 0.1 y 20Hz
- Normalización: Suave
- Clasificador: Support Vector Machine
- Kernel: linear (Parámetros por defecto)
- Secuencias promediadas: 1-15
- Sujetos: Todos

Objetivo:

- Metrica de interes: Desempeño [%] y F1 Score

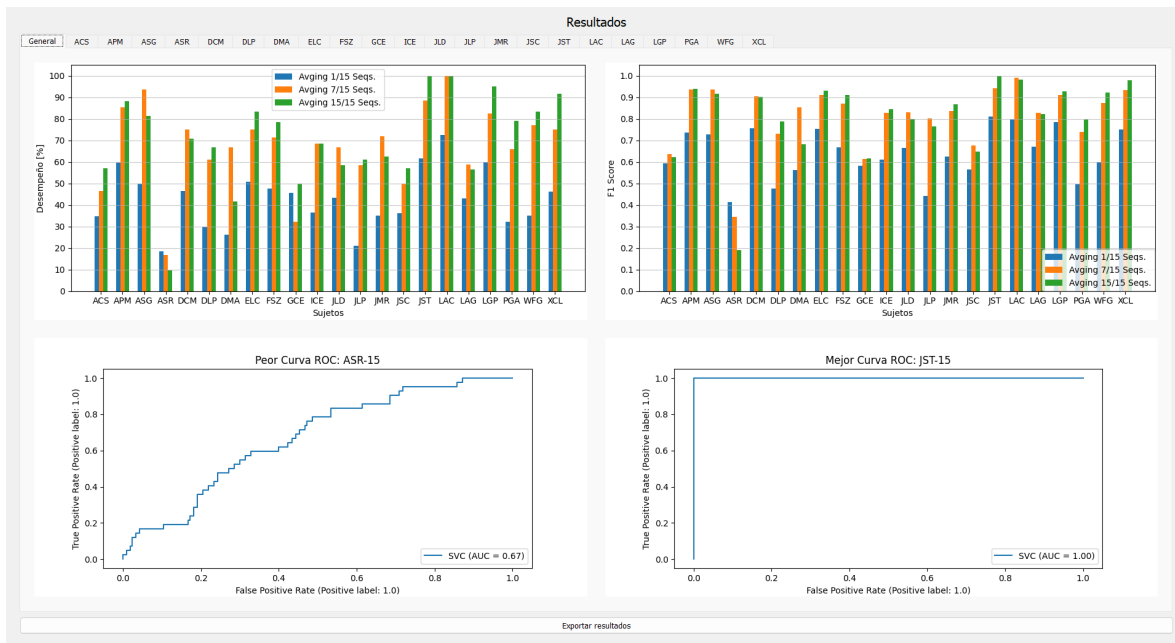


Figura 24: Resultados generales para 1, 7 y 15 secuencias promediadas. Los resultados se muestran para todos los sujetos según el promedio de secuencias más bajo, más alto y uno intermedio dado el gran volumen de promedios calculados.

Fuente: Confección propia.

En la figura 24 se muestran los resultados generales del experimento sólo mostrando 3 de los posibles promedios de secuencias. A continuación se presentan los 5 resultados más relevantes de la prueba, aquellos que muestran los casos más notables obtenidos, para más información con respecto al resto de los resultados, estos pueden ser revisados en el Anexo.

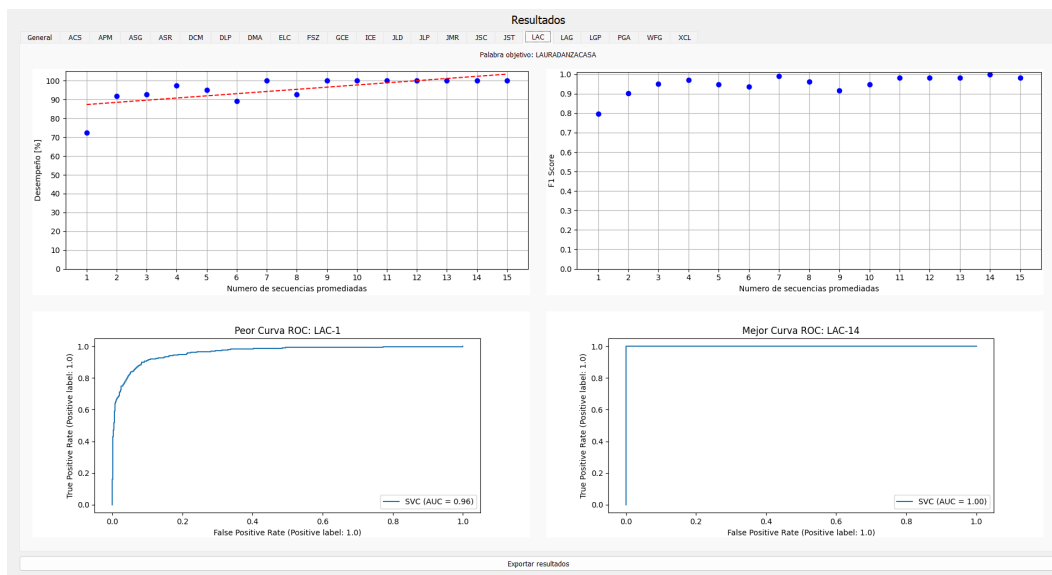


Figura 25: Resultados de LAC para todas las secuencias. Al lado izquierdo se presenta el gráfico de desempeño por letra, al lado derecho el F1 Score, ambos con respecto a la cantidad de secuencias promediadas, en la parte de inferior se muestra la peor y mejor curva ROC del conjunto.

Fuente: Confección propia.

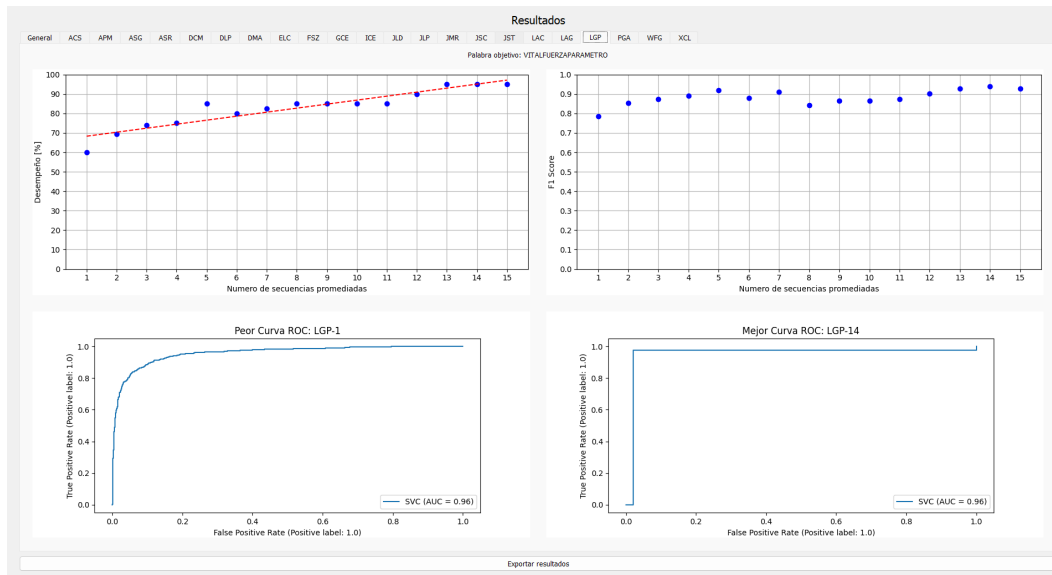


Figura 26: Resultados de LGP para todas las secuencias. Al lado izquierdo se presenta el gráfico de desempeño por letra, al lado derecho el F1 Score, ambos con respecto a la cantidad de secuencias promediadas, en la parte de inferior se muestra la peor y mejor curva ROC del conjunto.

Fuente: Confección propia.

Las figuras 25 y 26 corresponden a los mejores casos obtenidos en la prueba, en general, la gran mayoría de los resultados presenta líneas de tendencia similares a estos 2 sujetos, no obstante, difieren en el desempeño y el F1 Score, materias para las cuales los resultados descritos se condicen con lo esperado, es decir, que en la medida que se promedian más secuencias, el desempeño del modelo aumenta.

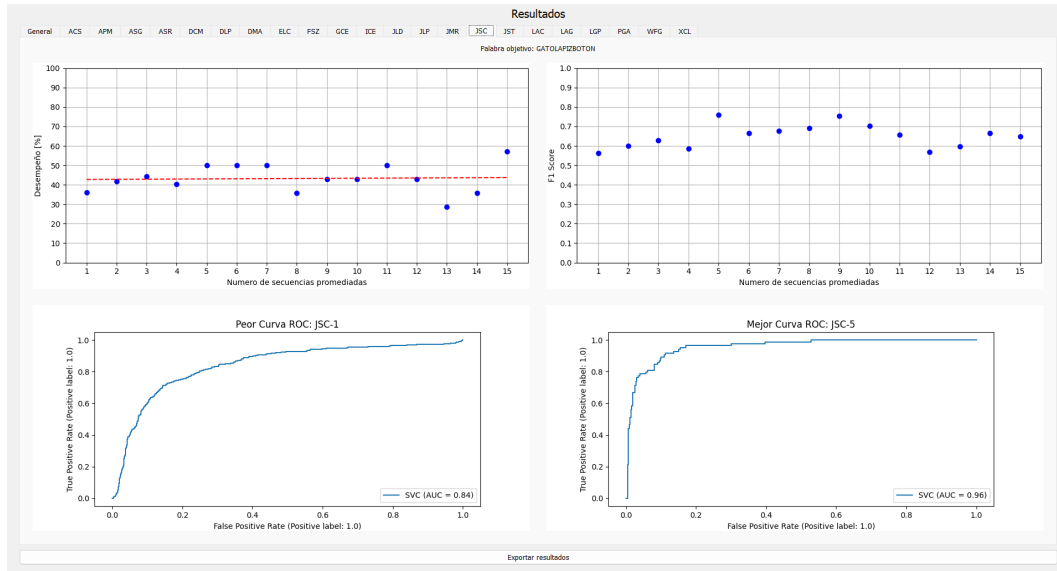


Figura 27: Resultados de JSC para todas las secuencias. Al lado izquierdo se presenta el gráfico de desempeño por letra, al lado derecho el F1 Score, ambos con respecto a la cantidad de secuencias promediadas, en la parte de inferior se muestra la peor y mejor curva ROC del conjunto.

Fuente: Confección propia.

La figura 27 presenta uno de los primeros casos que no se condicen con lo esperado dada la fluctuación que existe en la medida que se promedian más secuencias, no obstante, son pocos los promedios donde el desempeño es peor que el caso inicial y la gráfica del F1 Score resulta ser un poco más estable, lo que advierte que esta diferencia en resultados simplemente pueda ser un problema que se presenta al cruzar las respuestas del Speller para obtener las letras de la palabra.

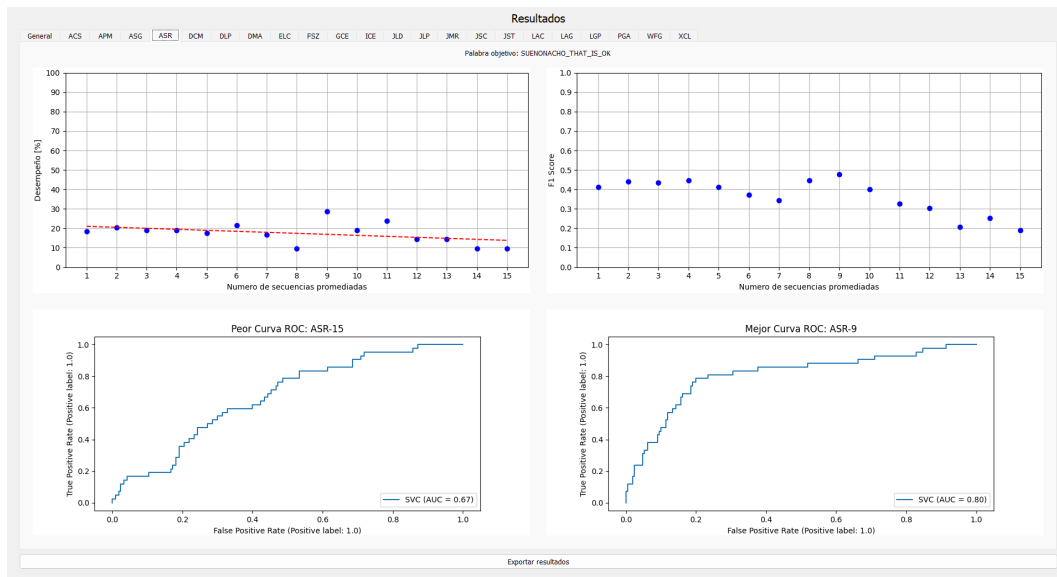


Figura 28: Resultados de ASR para todas las secuencias. Al lado izquierdo se presenta el gráfico de desempeño por letra, al lado derecho el F1 Score, ambos con respecto a la cantidad de secuencias promediadas, en la parte de inferior se muestra la peor y mejor curva ROC del conjunto.

Fuente: Confección propia.

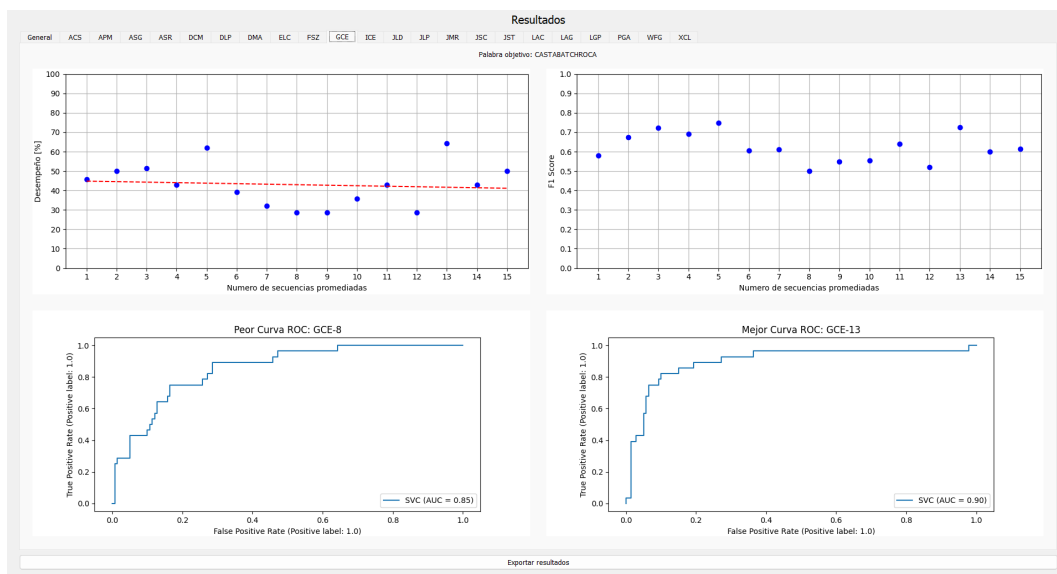


Figura 29: Resultados de GCE para todas las secuencias. Al lado izquierdo se presenta el gráfico de desempeño por letra, al lado derecho el F1 Score, ambos con respecto a la cantidad de secuencias promediadas, en la parte de inferior se muestra la peor y mejor curva ROC del conjunto.

Fuente: Confección propia.

Finalmente, las figuras 28 y 29 corresponden a los peores casos encontrados en la prueba, se caracterizan por tener tendencias a empeorar en la medida que se promedian más secuencias, siendo ASR el caso más pesimista, donde incluso el F1 Score tiende a la baja. Esto se puede deber a problemas existentes en los datos, como por ejemplo, poca rigurosidad al realizar los ensayos o características propias del individuo que dificulten la búsqueda del estímulo P300, también se puede deber a que el promedio esté eliminando información debido a que el estímulo se encuentre en instantes distintos durante la ventana de estudio entre una secuencia y otra, por ejemplo, que para una secuencia se encuentre al principio y en otra próximo al fin de la ventana, al promediar dichas secuencias, el resultado sería una onda que difícilmente represente la información sobre la ubicación de los P300.

4.2.2. Pruebas comparativas

Para ambas pruebas se utilizó la siguiente configuración:

- Filtro: Passband entre 0.1 y 20Hz
- Normalización: Suave
- Clasificador: Support Vector Machine
- Kernel: linear (Parámetros por defecto)
- Secuencias promediadas: 1, 5 y 15
- Sujetos: Todos

Desempeño por letra

- Métrica de interés: Desempeño [%]

El objetivo de esta prueba es comparar el rendimiento que se tiene a nivel de porcentaje de letras correctamente identificadas entre ambas implementaciones, por lo que ambos códigos son ejecutados con las mismas configuraciones.

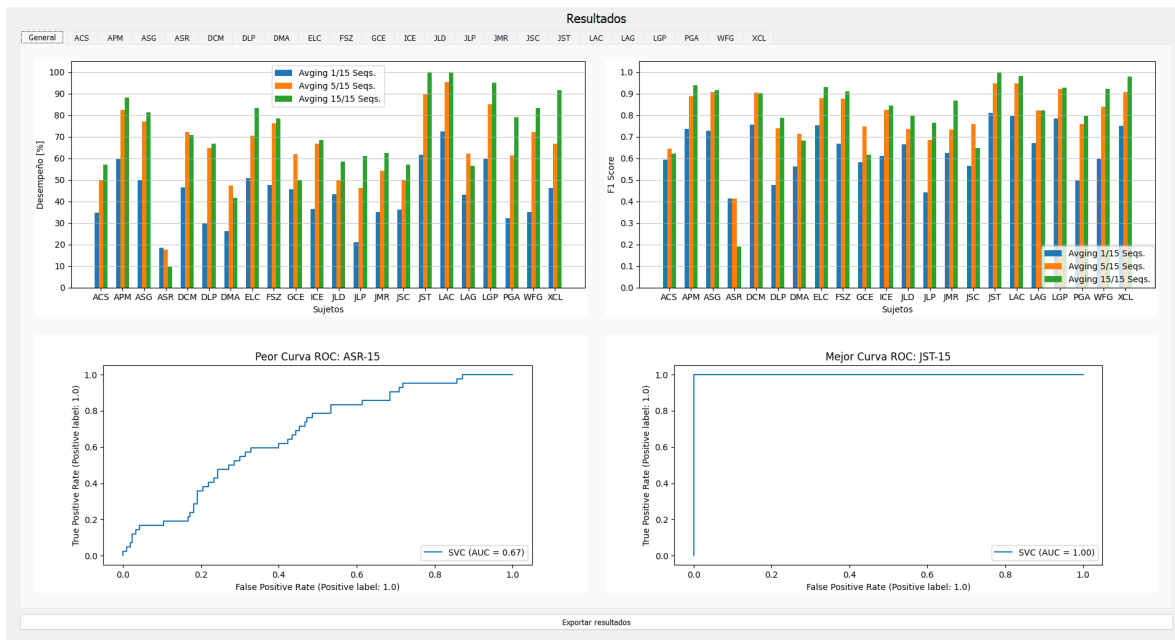


Figura 30: Resultados generales para los distintos promedios de secuencia.
Fuente: Confección propia.

La figura 30 muestra los resultados de la prueba de rendimiento, es decir, el porcentaje de palabras correctamente identificadas para la solución descrita en este trabajo, con el fin de comparar ambas soluciones, estos resultados y los provenientes de la implementación de MATLAB han sido recopilados y ordenados en una tabla.

Sujeto	Secuencias promediadas					
	Python			MATLAB		
	1	5	15	1	5	15
ACS	34.76	50.00	57.14	33.33	50.00	57.14
APM	59.61	82.35	88.24	59.61	82.35	88.24
ASG	50.00	77.08	81.25	48.75	77.08	81.25
ASR	18.41	17.46	9.52	18.10	25.40	9.52
DCM	46.39	72.22	70.83	44.44	69.44	62.50
DLP	30.00	64.81	66.67	30.00	62.96	72.22
DMA	26.11	47.22	41.67	28.33	36.11	41.67
ELC	50.74	70.37	83.33	50.74	70.37	88.89
FSZ	47.62	76.19	78.57	47.62	76.19	78.57
GCE	45.71	61.90	50.00	43.81	66.67	50.00
ICE	36.49	66.67	68.42	36.14	63.16	68.42
JLD	43.33	50.00	58.33	43.33	41.67	58.33
JLP	21.11	46.30	61.11	22.59	48.15	66.67
JMR	35.00	54.17	62.50	35.00	50.00	68.75
JSC	36.19	50.00	57.14	32.86	47.62	57.14
JST	61.54	89.74	100.00	61.54	76.92	100.00
LAC	72.38	95.24	100.00	70.95	95.24	100.00
LAG	42.90	62.32	56.52	41.16	59.42	52.17
LGP	60.00	85.00	95.00	58.00	78.33	95.00
PGA	32.28	61.40	78.95	30.53	61.40	78.95
WFG	35.00	72.22	83.33	35.56	66.67	83.33
XCL	46.11	66.67	91.67	46.11	63.89	91.67
Promedio	42.35	64.52	70.01	41.75	62.23	70.47
Desv. Est.	13.24	17.03	20.80	12.90	16.14	21.06

Tabla 2: Desempeño en porcentaje de la palabra obtenida para ambas implementaciones en base a la cantidad de secuencias promediadas. Los resultados marcados con verde, amarillo y rojo corresponden a los casos donde la solución propuesta tiene mejor, igual o peor rendimiento respectivamente.

Fuente: Confección propia.

La tabla 2 recopila los resultados para las pruebas de desempeño ejecutadas en Python y MATLAB por cantidad de secuencias promediadas, a modo de comparación, los resultados de la solución propuesta fueron marcados considerando si estos eran mejores, iguales o peores que en MATLAB.

A grandes rasgos, la solución propuesta tiene mejor o igual desempeño por letra que el código en MATLAB, hay casos excepcionales como JLP donde para todas las secuencias el desempeño no es favorable. En general, aproximadamente un 38 % de los casos resultó ser favorable para la propuesta mientras que un 15 % de los casos el rendimiento no fue

favorable, lo cual es un balance bastante positivo si se toma en cuenta que el resto de los casos obtuvieron un rendimiento similar en ambos casos, mientras que en promedio para 1 y 5 secuencias promediadas, el código de Python tiene mejor desempeño que su equivalente en MATLAB.

Considerando las diferencias existentes en cómo Python y MATLAB manejan sus tipo de datos y las operaciones aplicables a estos, en este caso, la manipulación matricial, sumado a la diferencia que puede existir entre las implementaciones de las funciones pertenecientes a las distintas librerías utilizadas, los resultados son favorables para el trabajo desarrollado.

Tiempo de ejecución

Para realizar esta prueba, ambos códigos fueron ejecutados posterior al reinicio del sistema, para reducir la carga que este pueda tener y reducir la cantidad de procesos al mínimo, es importante mencionar que el tiempo de ejecución solo considera el tiempo correspondiente a la ejecución del pipeline descrito en el capítulo 3 sin considerar las operaciones adicionales para mostrar los resultados en pantalla, es decir, solo considera la carga, filtrado y procesamiento de datos, la clasificación de las respuestas P300 y el cálculo de métricas de desempeño, de manera que la comparativa fuera justa entre ambas implementaciones.

Ambos códigos fueron ejecutados 10 veces para obtener una medición que disminuya el error aleatorio.

Nº de iteracion	Secuencias promediadas					
	Python			MATLAB		
	1	5	15	1	5	15
1	420.67	27.81	24.90	405.14	53.93	48.82
2	422.26	27.72	25.21	402.83	51.98	48.19
3	418.23	27.68	24.78	402.52	53.56	48.40
4	419.23	27.72	24.57	402.02	53.60	47.79
5	420.18	27.68	24.84	403.34	53.60	48.16
6	421.97	27.98	24.73	401.40	54.06	47.98
7	419.80	27.87	24.57	402.10	51.70	47.89
8	419.28	27.78	24.67	401.38	51.47	48.29
9	420.60	27.77	24.88	397.66	53.02	47.95
10	417.84	27.92	24.47	402.33	53.74	48.09
Promedio	420.01	27.79	24.76	402.07	53.07	48.16
Desv. Est.	1.37	0.10	0.20	1.80	0.93	0.28

Tabla 3: Tiempos de ejecución en segundos a partir de la cantidad de secuencias promediadas.

Fuente: Confección propia.

La tabla 3 muestra los tiempos de ejecución por secuencia para cada implementación, tomando en consideración los promedios, para 1 secuencia la propuesta resulta ser un

4.36 % más lento, mientras que para 5 y 15 secuencias resulta ser 62.53 % y 64.18 % más rápido respectivamente.

Dado que la diferencia de tiempo en la medida que se promedian más secuencias disminuye considerablemente, es posible asumir que gran parte del tiempo termina siendo de pre-procesamiento al disminuir el tiempo que toma clasificar los datos, en vista que el patrón se repite para ambas implementaciones. Si bien para el caso base resulta ser ligeramente más lento, es posible que la diferencia favorable del tiempo de ejecución en Python se deba a la optimización existente en librerías como Numpy, Scipy y Scikit-learn para manipular los datos.

En general, los resultados muestran una tendencia a favorecer tanto en desempeño como en rendimiento la solución propuesta por sobre la implementación de MATLAB, si bien existen casos puntuales donde este último obtiene mejores resultados, en general, los resultados tienden a ser iguales o incluso mejores en cuanto a desempeño y considerablemente más rápidos en términos de tiempo de ejecución.

CAPÍTULO 5

CONCLUSIONES Y TRABAJO FUTURO

A lo largo de este trabajo se exploró la necesidad de diversificar la oferta en torno a soluciones de software pensadas para la interfaz cerebro-computador P300 Speller, partiendo por las limitaciones de las alternativas actualmente existentes ya sea a nivel económico por el costo que implica la programación en MATLAB (producto de la necesidad de contar con licencias para su uso), así como los cambios en las tendencias en lo que respecta al uso de nuevas tecnologías como es el caso del aumento de la cuota de mercado de lenguajes de código abierto como Python producto de la especialización de librerías enfocadas en el análisis de datos e inteligencia artificial, convirtiéndose de esta forma en una alternativa altamente robusta para diseñar soluciones y de bajo costo para este tipo de problemas.

Debido a las inquietudes que supone la migración de un lenguaje de programación a otro en términos técnicos, se propuso el diseño de un *framework* que aborde los aspectos que componen la lógica analítica del P300 Speller, en ese sentido, la propuesta hace uso de librerías como Numpy, Scipy, Scikit-learn, Matplotlib y PyQt5 para realizar tareas de lectura y manipulación de datos, filtrado de señal, y el entrenamiento de un clasificador para la detección de los estímulos de tipo P300 que permiten determinar las palabras deletreadas por el usuario. En base a los resultados, la solución ideada es capaz de procesar la información recopilada para así mostrar métricas de desempeño del experimento de manera gráfica, todo esto mediante una interfaz de usuario.

Para validar que la solución es acorde al problema a solucionar, se realizaron distintos tipos de pruebas enfocadas a verificar que las tareas que componen el *framework* son ejecutadas de manera correcta y se ajustan a lo esperado con respecto al procedimiento que están realizando, además, pruebas que buscan comparar la solución desarrollada con una implementación similar en MATLAB. Para ello se hizo uso de una base de datos pública de EEG para ejecutar las pruebas.

Para el primer tipo de pruebas, en general los resultados coinciden con lo esperado, siendo el caso de las pruebas con respecto al promedio de secuencias aquellas que mostraron una gran variación en los resultados, se presume que las causas puedan deberse a limitaciones en sí de la idea de promediar secuencias junto con características propias de los datos al momento de su captura. Con respecto a las pruebas comparativas se obtuvo que para el 37.88 % de los casos, la solución planteada obtiene mejores resultados que su contra-parte en MATLAB, no obstante, para el 15.15 % de los casos los resultados obtuvieron un peor rendimiento, mientras que para el resto de los casos se obtuvieron exactamente los mismos resultados. Finalmente, se determinó que a grandes rasgos el código desarrollado en Python resulta ser mas rápido, teniendo que para 5 y 15 secuencias promediadas este resulta ser un 62.53 % y 64.18 % más rápido y que para 1 secuencia es un 4.36 % más lento que el código ejecutado en MATLAB.

Cumplimiento de objetivos

A continuación se validará el cumplimiento de los objetivos planteados a manera de dar cierre al trabajo realizado, comenzando por los objetivos específicos para dar lugar al objetivo general:

- **Estudiar el funcionamiento de la interfaz cerebro-computador P300 Speller.**

El capítulo 2 referente al marco teórico, se abordó los principios bajos los cuales la interfaz cerebro-computador P300 Speller se fundamenta, explicando en qué consiste y como opera una interfaz cerebro-computador, las particularidades propias del P300 Speller como por ejemplo, el potencial evocado P300, el protocolo de captura de datos, la clasificación de las respuestas, la interpretación de los resultados, etc.

- **Diseñar una propuesta de *framework* para una interfaz cerebro-computador P300 Speller utilizando el lenguaje de programación Python.**

El capítulo 3 define la propuesta de solución planteada en base a lo aprendido en el capítulo anterior, estableciendo el stack tecnológico a utilizar y como cada librería presente se acomoda a los pasos de procesamiento y manipulación de datos, filtrado de señal, clasificación de la respuesta P300 y análisis de los resultados obtenidos.

- **Verificar y comparar el funcionamiento de la solución, haciendo uso de una implementación equivalente.**

El capítulo 4, la validación de la solución contiene los resultados de las pruebas ejecutadas para verificar que el funcionamiento del *framework* es el correcto en lo que respecta a filtrado de señal, normalización de los datos y promedios de secuencias, así como también, los experimentos realizados para comparar el rendimiento del trabajo realizado con otro similar desarrollado en MATLAB a partir del desempeño por letra obtenido y los tiempos de ejecución para ambos casos.

En lo que respecta al objetivo general: **Desarrollar un *framework* basado en el P300 Speller para preprocesar, clasificar y presentar los resultados a partir de datos de la actividad cerebral utilizando el lenguaje de programación Python**, al estar los objetivos específicos contenidos en este, se da por realizado el propósito de este trabajo.

Trabajo Futuro

El propósito de este trabajo fue demostrar la factibilidad de desarrollar una herramienta capaz de realizar la carga analítica de la interfaz cerebro-computador P300 Speller sin sacrificar lo logrado en MATLAB, en ese aspecto, se demostró que se podía obtener incluso mejores resultados, lo cual da pie a considerar posibles mejoras que permitan robustecer aún más la propuesta realizada, en esa línea, hay una serie de mejoras a realizar así como también la inclusión de nuevas características con el fin de mejorar la herramienta en caso de ser del interés del lector.

- **Implementar más métodos de filtrado.**

La investigación con respecto al P300 Speller y otras interfaces cerebro-computador dependientes de datos de EEG se ha centrado en poder aplicar nuevos métodos de filtrado de datos para su clasificación y mejorar los resultados en *single trial* a partir de la información obtenible de estos mismos, para sortear las limitantes que implica el método de promedio de secuencias para reducir el ruido y mejorar la consistencia, uno de los desafíos es determinar que herramientas sería pertinente implementar en el *framework* para labores de filtrado como sustituto o complemento a las existentes.

- **Implementar nuevos clasificadores.**

Una de las razones de utilización de SVM como técnica de clasificación corresponde a lo práctico que resulta para resolver este tipo de problemas, no obstante, como se vio en la sección del marco teórico, no es el único clasificador utilizado en la actualidad dado que nuevos métodos están siendo estudiados para mejorar los resultados que hay hasta la fecha. Al igual que en el caso de los filtros, el desafío está en determinar cuales pueden resultar buenas adiciones teniendo en cuenta que se podría requerir otro tipo de características no implementadas en la solución actual.

- **Creación y carga de plantillas de configuraciones.**

Uno de los aspectos que se volvió evidente durante las pruebas fue la necesidad de poder guardar las configuraciones utilizadas en los experimentos para volver a ser utilizadas en experimentos posteriores de manera de evitar estar rellendo campos ya que la información adicional del conjunto de datos es constante en el tiempo. Esto repercute en la necesidad de poder guardar dicha información e implementar métodos para su lectura, inicialización de variables y reflejar los cambios en la interfaz.

- **Robustecer el visualizador de señal.**

La implementación actual carece de varias características como poder visualizar la señal por ventanas o segmentos en vez de una vista general; una aplicación interesante sería separar la señal correspondiente a una sesión en las actividades que la componen para tener una mejor idea de anomalías presentes al momento de capturar los datos, así como también poder visualizar mas de una sesión a la vez, o sobreponer cada uno de los señales o el promedio de todas la secuencias, los criterios sobre que características implementar deben estar fundamentadas en el aporte que producen en la experiencia de usuario.

- **Flexibilizar las restricciones de funcionamiento.**

La implementación final tuvo una gran influencia de la estructura de la base de datos utilizada al momento de cargar los datos, lo cual en ciertos aspectos puede resultar un poco restrictiva si se intenta utilizar la aplicación con datos de otra procedencia, teniendo que formatearlos a la estructura correcta, debido a supuestos que tuvieron que establecerse, por ejemplo, considerar que una sesión tiene 3 actividades de deletreo, que todos los sujetos seleccionados emplean la misma palabra para entrenar el modelo o la imposibilidad de seleccionar o indicar los *dataset* de entrenamiento y

prueba, esto también supone un desafío a nivel visual sobre cómo ordenar la interfaz de usuario para evitar confusiones.

ANEXOS

5.1. Resultados: Promedio de secuencias

A continuación se muestra el resto de resultados para la prueba de validación con respecto al promedio de secuencias

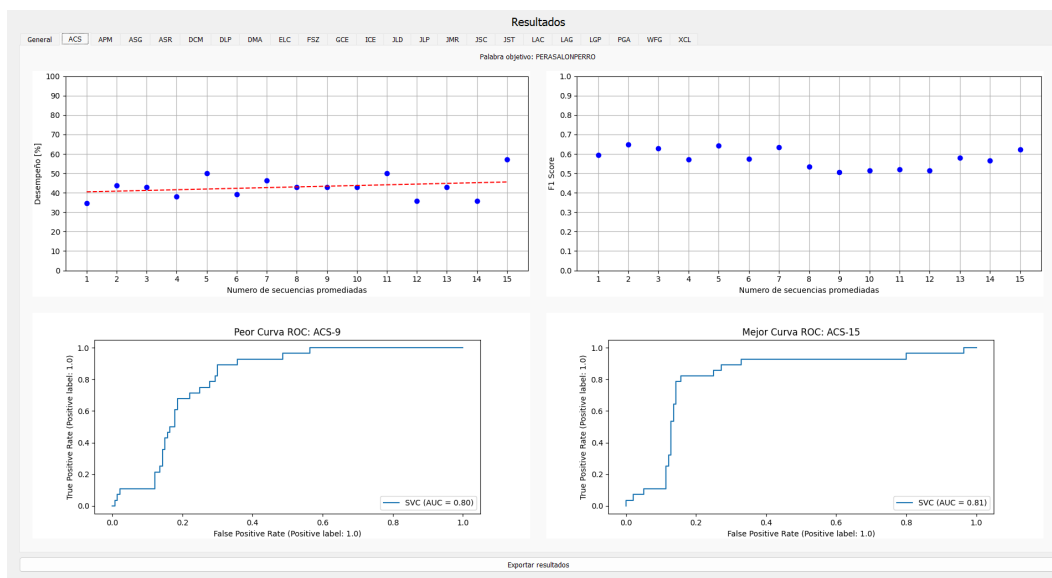


Figura 31: Resultados de ACS para todas las secuencias. Al lado izquierdo se presenta el gráfico de desempeño por letra, al lado derecho el F1 Score, ambos con respecto a la cantidad de secuencias promediadas, en la parte de inferior se muestra la peor y mejor curva ROC del conjunto.

Fuente: Confección propia.

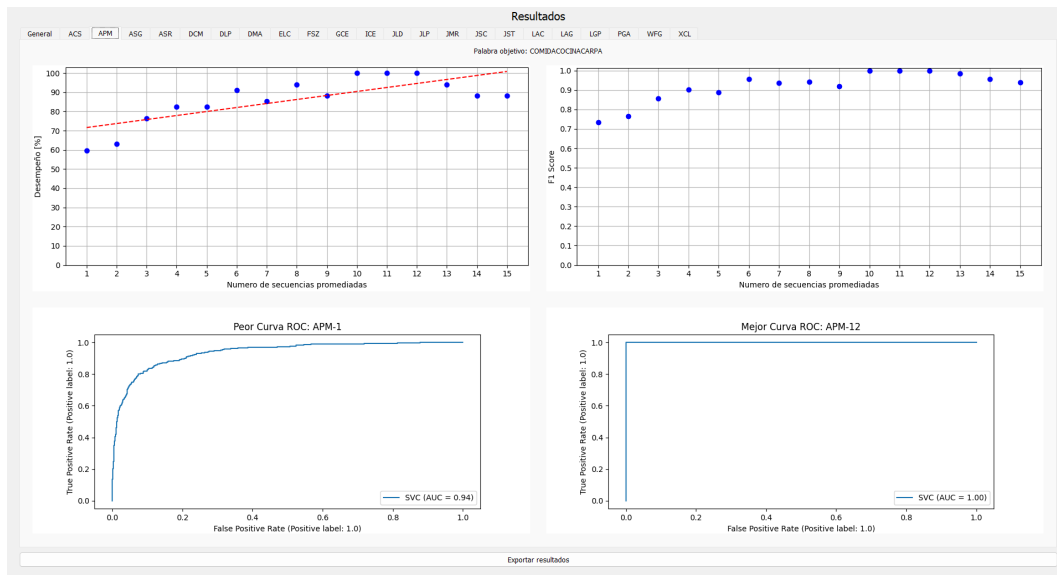


Figura 32: Resultados de APM para todas las secuencias. Al lado izquierdo se presenta el gráfico de desempeño por letra, al lado derecho el F1 Score, ambos con respecto a la cantidad de secuencias promediadas, en la parte de inferior se muestra la peor y mejor curva ROC del conjunto.

Fuente: Confección propia.

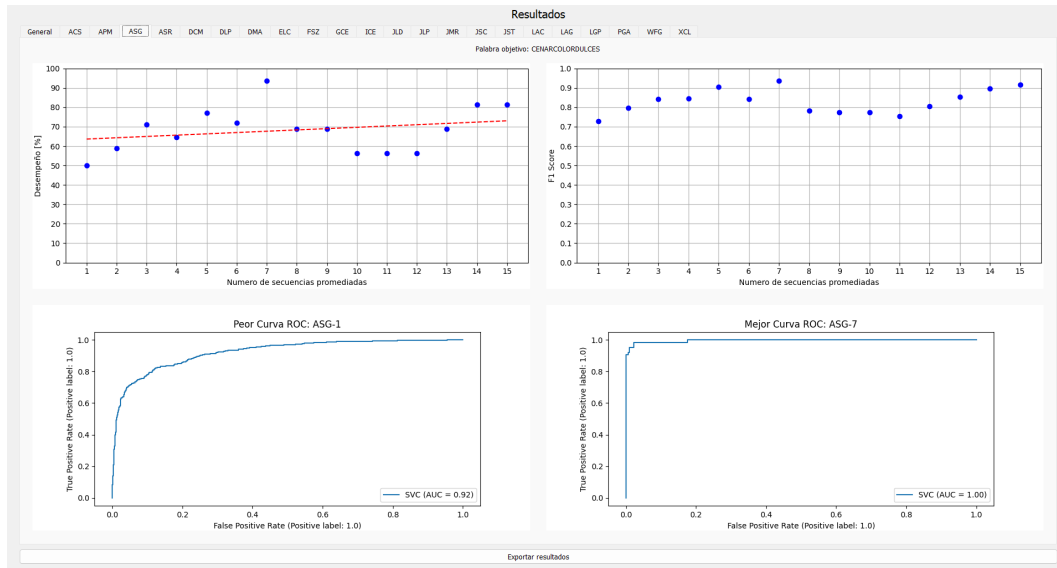


Figura 33: Resultados de ASG para todas las secuencias. Al lado izquierdo se presenta el gráfico de desempeño por letra, al lado derecho el F1 Score, ambos con respecto a la cantidad de secuencias promediadas, en la parte de inferior se muestra la peor y mejor curva ROC del conjunto.

Fuente: Confección propia.

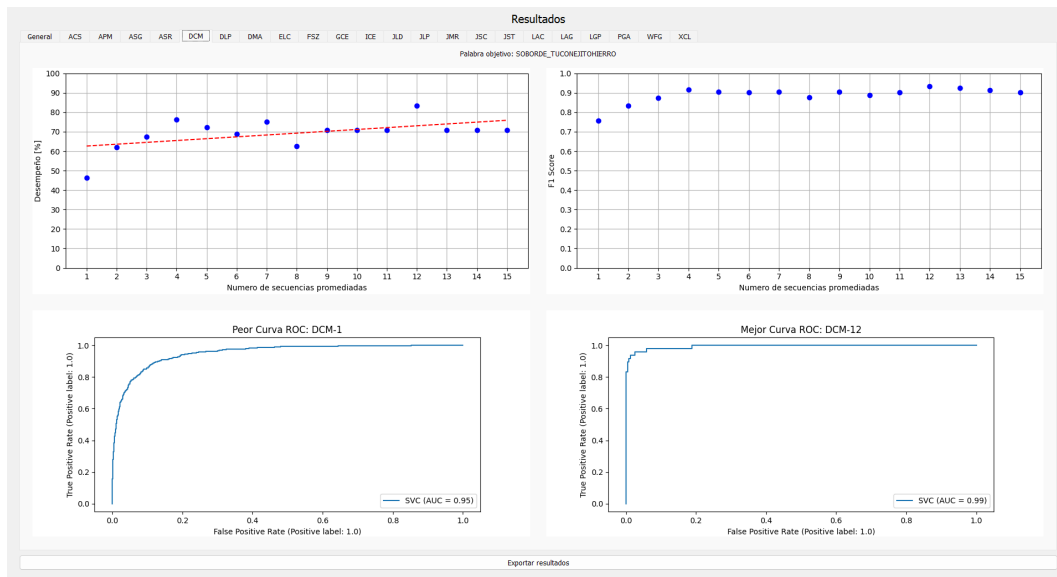


Figura 34: Resultados de DCM para todas las secuencias. Al lado izquierdo se presenta el gráfico de desempeño por letra, al lado derecho el F1 Score, ambos con respecto a la cantidad de secuencias promediadas, en la parte de inferior se muestra la peor y mejor curva ROC del conjunto.

Fuente: Confección propia.

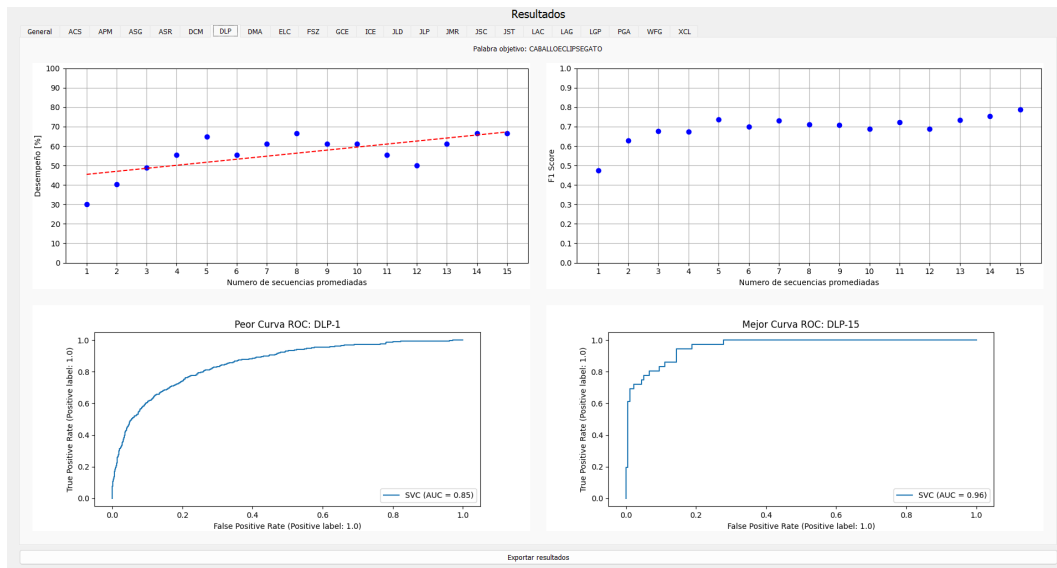


Figura 35: Resultados de DLP para todas las secuencias. Al lado izquierdo se presenta el gráfico de desempeño por letra, al lado derecho el F1 Score, ambos con respecto a la cantidad de secuencias promediadas, en la parte de inferior se muestra la peor y mejor curva ROC del conjunto.

Fuente: Confección propia.

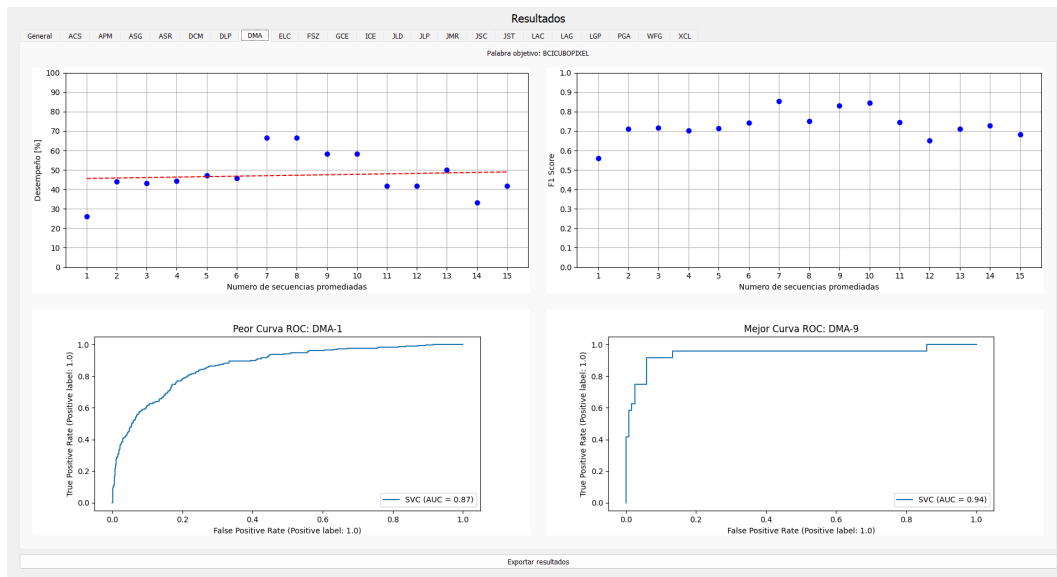


Figura 36: Resultados de DMA para todas las secuencias. Al lado izquierdo se presenta el gráfico de desempeño por letra, al lado derecho el F1 Score, ambos con respecto a la cantidad de secuencias promediadas, en la parte de inferior se muestra la peor y mejor curva ROC del conjunto.

Fuente: Confección propia.

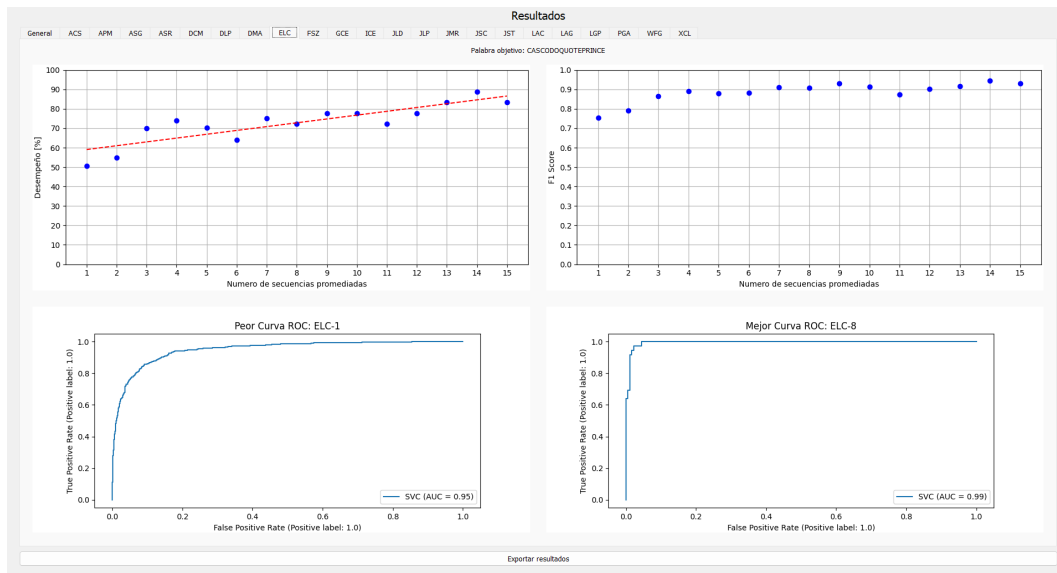


Figura 37: Resultados de ELC para todas las secuencias. Al lado izquierdo se presenta el gráfico de desempeño por letra, al lado derecho el F1 Score, ambos con respecto a la cantidad de secuencias promediadas, en la parte de inferior se muestra la peor y mejor curva ROC del conjunto.

Fuente: Confección propia.

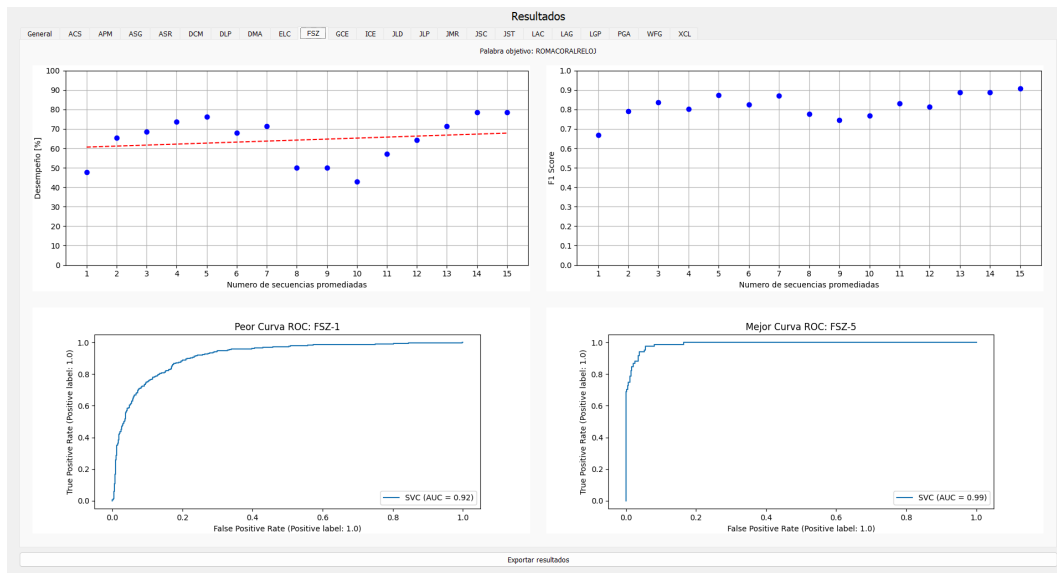


Figura 38: Resultados de FSZ para todas las secuencias. Al lado izquierdo se presenta el gráfico de desempeño por letra, al lado derecho el F1 Score, ambos con respecto a la cantidad de secuencias promediadas, en la parte de inferior se muestra la peor y mejor curva ROC del conjunto.

Fuente: Confección propia.

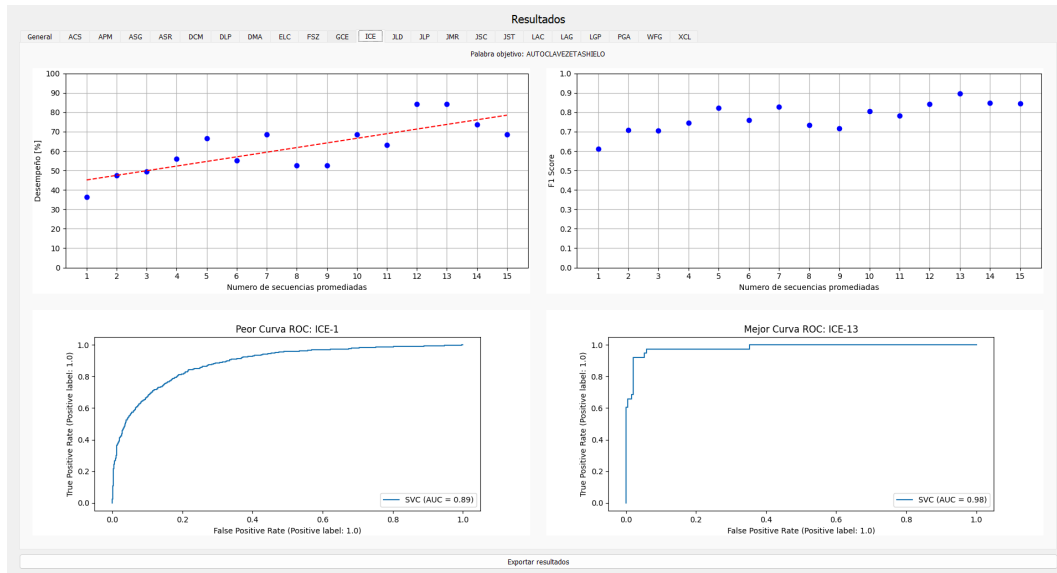


Figura 39: Resultados de ICE para todas las secuencias. Al lado izquierdo se presenta el gráfico de desempeño por letra, al lado derecho el F1 Score, ambos con respecto a la cantidad de secuencias promediadas, en la parte de inferior se muestra la peor y mejor curva ROC del conjunto.

Fuente: Confección propia.

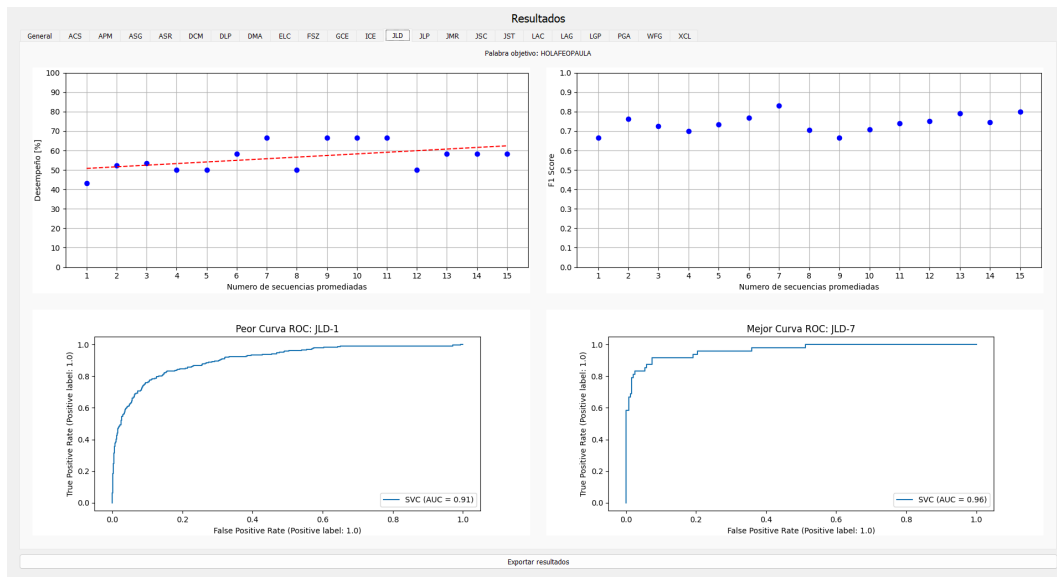


Figura 40: Resultados de JLD para todas las secuencias. Al lado izquierdo se presenta el gráfico de desempeño por letra, al lado derecho el F1 Score, ambos con respecto a la cantidad de secuencias promediadas, en la parte de inferior se muestra la peor y mejor curva ROC del conjunto.

Fuente: Confección propia.

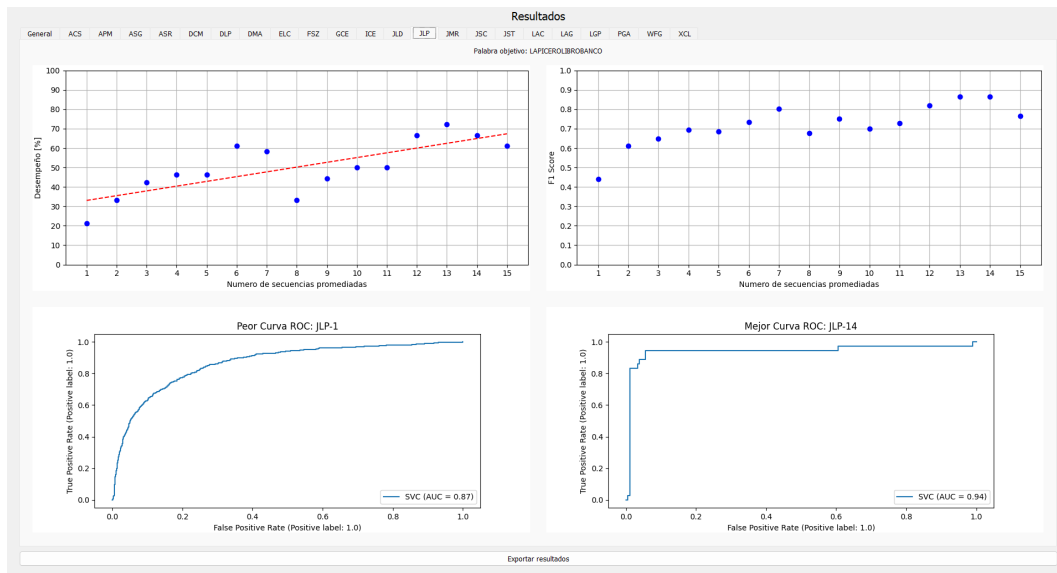


Figura 41: Resultados de JLP para todas las secuencias. Al lado izquierdo se presenta el gráfico de desempeño por letra, al lado derecho el F1 Score, ambos con respecto a la cantidad de secuencias promediadas, en la parte de inferior se muestra la peor y mejor curva ROC del conjunto.

Fuente: Confección propia.

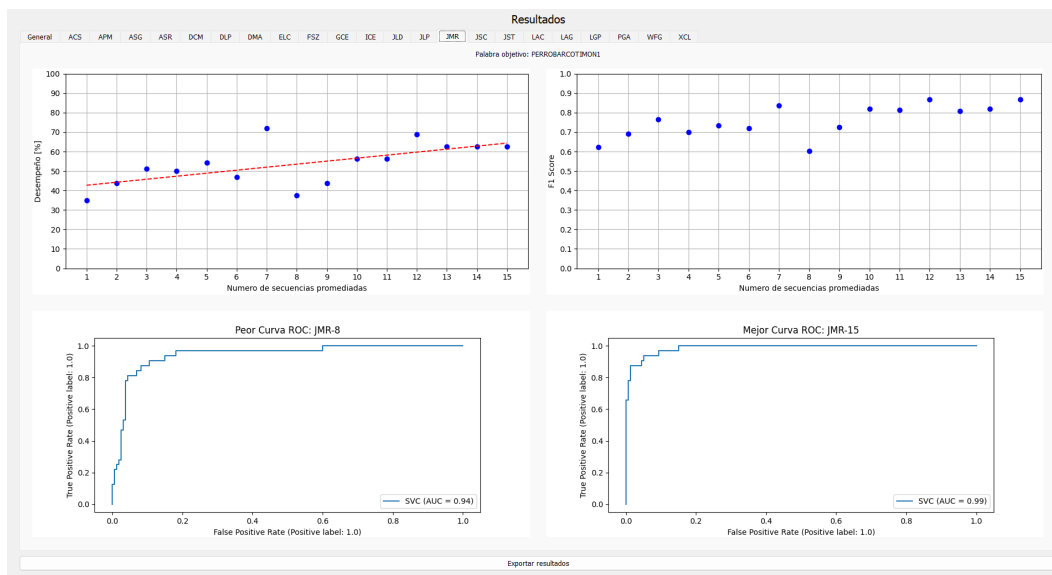


Figura 42: Resultados de JMR para todas las secuencias. Al lado izquierdo se presenta el gráfico de desempeño por letra, al lado derecho el F1 Score, ambos con respecto a la cantidad de secuencias promediadas, en la parte de inferior se muestra la peor y mejor curva ROC del conjunto.

Fuente: Confección propia.

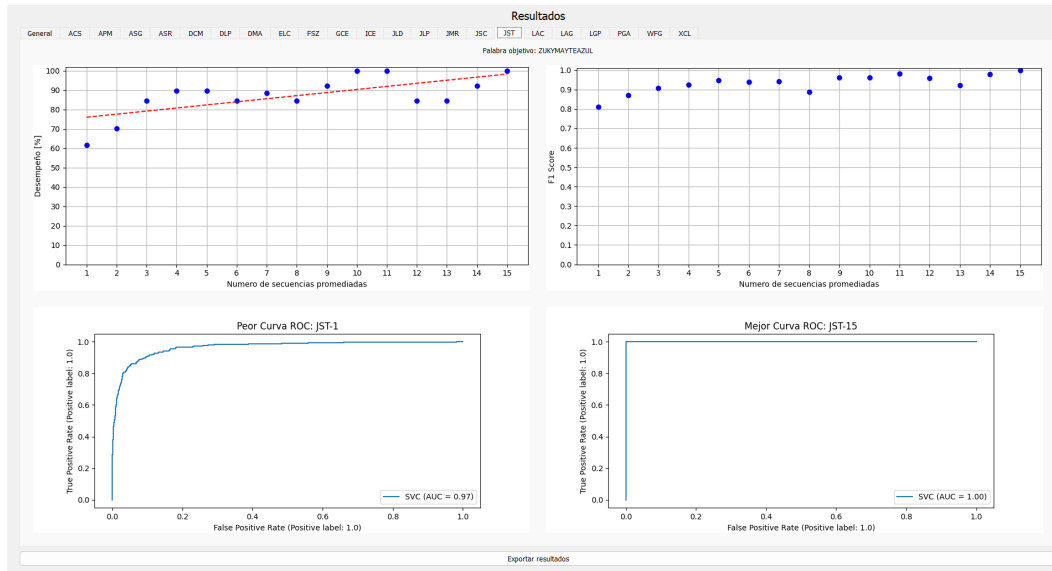


Figura 43: Resultados de JST para todas las secuencias. Al lado izquierdo se presenta el gráfico de desempeño por letra, al lado derecho el F1 Score, ambos con respecto a la cantidad de secuencias promediadas, en la parte de inferior se muestra la peor y mejor curva ROC del conjunto.

Fuente: Confección propia.

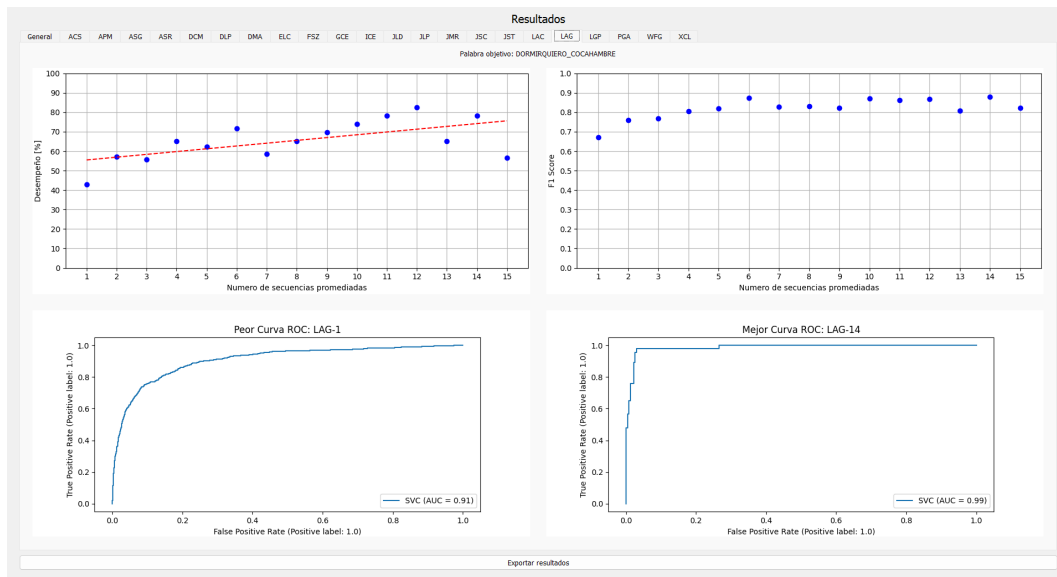


Figura 44: Resultados de LAG para todas las secuencias. Al lado izquierdo se presenta el gráfico de desempeño por letra, al lado derecho el F1 Score, ambos con respecto a la cantidad de secuencias promediadas, en la parte de inferior se muestra la peor y mejor curva ROC del conjunto.

Fuente: Confección propia.

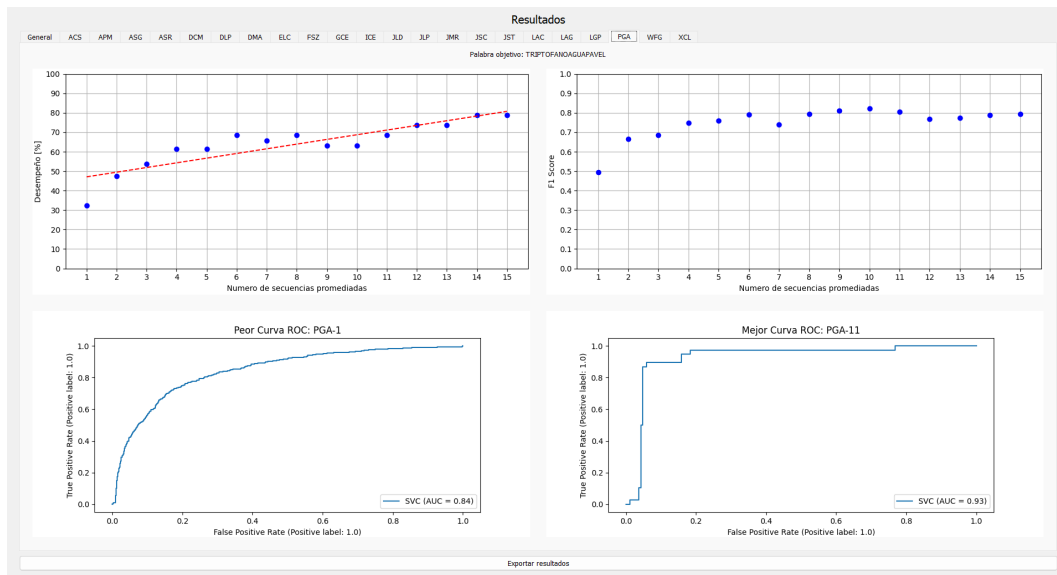


Figura 45: Resultados de PGA para todas las secuencias. Al lado izquierdo se presenta el gráfico de desempeño por letra, al lado derecho el F1 Score, ambos con respecto a la cantidad de secuencias promediadas, en la parte de inferior se muestra la peor y mejor curva ROC del conjunto.

Fuente: Confección propia.

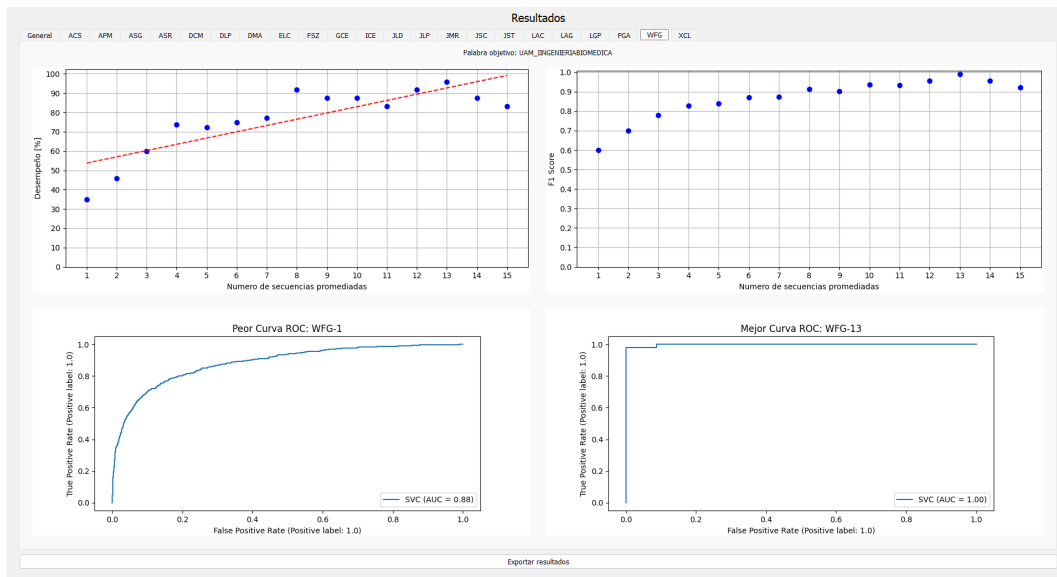


Figura 46: Resultados de WFG para todas las secuencias. Al lado izquierdo se presenta el gráfico de desempeño por letra, al lado derecho el F1 Score, ambos con respecto a la cantidad de secuencias promediadas, en la parte de inferior se muestra la peor y mejor curva ROC del conjunto.

Fuente: Confección propia.

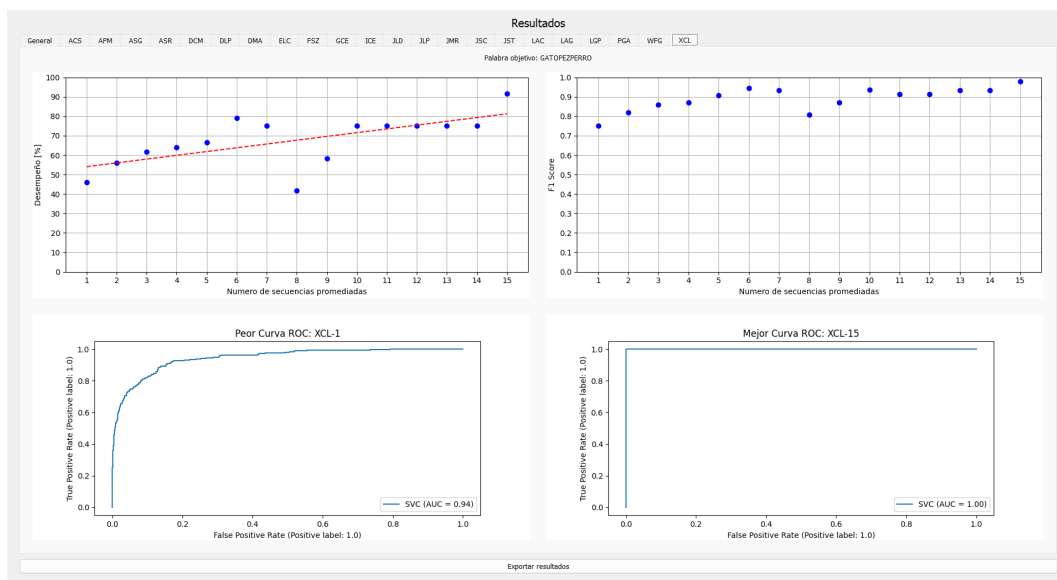


Figura 47: Resultados de XCL para todas las secuencias. Al lado izquierdo se presenta el gráfico de desempeño por letra, al lado derecho el F1 Score, ambos con respecto a la cantidad de secuencias promediadas, en la parte de inferior se muestra la peor y mejor curva ROC del conjunto.

Fuente: Confección propia.

REFERENCIAS BIBLIOGRÁFICAS

- [Arrouët *et al.*, 2005] Arrouët, C., Congedo, M., Marvie, J.-E., Lamarche, F., Lécuyer, A., y Arnaldi, B. (2005). Open-ViBE: a three dimensional platform for real-time neuroscience. *Journal of Neurotherapy*, 9(1):3–25.
- [Blinowska y Durka, 2006] Blinowska, K. y Durka, P. (2006). Electroencephalography EEG. *Wiley encyclopedia of biomedical engineering*.
- [Farwell y Donchin, 1988] Farwell, L. A. y Donchin, E. (1988). Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials. *Electroencephalography and clinical Neurophysiology*, 70(6):510–523.
- [Goda *et al.*, 2024] Goda, M. Á., Charlton, P. H., y Behar, J. A. (2024). pyppg: A python toolbox for comprehensive photoplethysmography signal analysis. *Physiological Measurement*, 45(4):045001.
- [Guy *et al.*, 2018] Guy, V., Soriani, M.-H., Bruno, M., Papadopoulo, T., Desnuelle, C., y Clerc, M. (2018). Brain computer interface with the P300 speller: Usability for disabled people with amyotrophic lateral sclerosis. *Annals of Physical and Rehabilitation Medicine*, 61(1):5–11.
- [Jamil *et al.*, 2021] Jamil, N., Belkacem, A. N., Ouhbi, S., y Lakas, A. (2021). Noninvasive electroencephalography equipment for assistive, adaptive, and rehabilitative brain-computer interfaces: a systematic literature review. *Sensors*, 21(14):4754.
- [Kaplan *et al.*, 2013] Kaplan, A. Y., Shishkin, S. L., Ganin, I. P., Basyul, I. A., y Zhigalov, A. Y. (2013). Adapting the P300-based brain-computer interface for gaming: A review. *IEEE Transactions on Computational Intelligence and AI in Games*, 5(2):141–149.
- [Kecman, 2005] Kecman, V. (2005). Support vector machines—an introduction. En *Support vector machines: theory and applications*, pp. 1–47. Springer.
- [Krusienski *et al.*, 2008] Krusienski, D., Sellers, E., McFarland, D., Vaughan, T., y Wolpaw, J. (2008). Toward enhanced P300 speller performance. *Journal of Neuroscience Methods*, 167(1):15–21. Brain-Computer Interfaces (BCIs).
- [Ledesma Ramírez *et al.*, 2010] Ledesma Ramírez, C., Bojorges, E., Yanez-Suarez, O., Saavedra, C., Bougrain, L., y Gentiletti, G. (2010). An open-access P300 speller database.
- [Leuthardt *et al.*, 2021] Leuthardt, E. C., Moran, D. W., y Mullen, T. R. (2021). Defining surgical terminology and risk for brain computer interface technologies. *Frontiers in Neuroscience*, 15:599549.
- [Lotte *et al.*, 2018] Lotte, F., Bougrain, L., Cichocki, A., Clerc, M., Congedo, M., Rakotomamonjy, A., y Yger, F. (2018). A review of classification algorithms for eeg-based brain-computer interfaces: a 10 year update. *Journal of neural engineering*, 15(3):031005.

- [Mak y Wolpaw, 2009] Mak, J. N. y Wolpaw, J. R. (2009). Clinical applications of brain-computer interfaces: current state and future prospects. *IEEE reviews in biomedical engineering*, 2:187–199.
- [Malmivuo y Plonsey, 1995] Malmivuo, J. y Plonsey, R. (1995). *Bioelectromagnetism: Principles and Applications of Bioelectric and Biomagnetic Fields*. Oxford University Press.
- [Obuchowski, 2003] Obuchowski, N. A. (2003). Receiver operating characteristic curves and their use in radiology. *Radiology*, 229(1):3–8.
- [Ruiz, 2013] Ruiz, C. V. S. (2013). *Méthodes d'analyse et de débruitage multicanaux à partir d'ondelettes pour améliorer la détection de potentiels évoqués sans moyennage: application aux interfaces cerveau-ordinateur.(Wavelet-based semblance methods to enhance single-trial ERP detection)*. Tesis doctoral, Citeseer.
- [Saavedra et al., 2019] Saavedra, C., Salas, R., y Bougrain, L. (2019). Wavelet-based semblance methods to enhance the single-trial detection of event-related potentials for a bci spelling system. *Computational Intelligence and Neuroscience*, 2019(1):8432953.
- [Schalk et al., 2004] Schalk, G., McFarland, D. J., Hinterberger, T., Birbaumer, N., y Wolpaw, J. R. (2004). BCI2000: a general-purpose brain-computer interface BCI system. *IEEE Transactions on biomedical engineering*, 51(6):1034–1043.
- [Yordanova y Kolev, 1996] Yordanova, J. y Kolev, V. (1996). The relationship between P300 and event-related theta EEG activity. *Psychology*, 7.
- [Zhang y Chen, 2024] Zhang, D. y Chen, T. (2024). Scikit-anfis: A scikit-learn compatible python implementation for adaptive neuro-fuzzy inference system. *International Journal of Fuzzy Systems*, pp. 1–19.