CIMAT

Centro de Investigación en Matemáticas, A.C.

CLASIFICADOR DE LA DIFICULTAD DEL USO DE LA MEMORIA DE TRABAJO POR LA MEDICIÓN DE CARGA COGNITIVA A TRAVÉS DE SEGUIMIENTO OCULAR

T E S I S

Que para obtener el grado de

Maestro en Ingeniería de Software

con Orientación en

Interacción Humano-Computadora

Presenta

Jorge Sánchez Rodríguez

Director de Tesis:

Dr. Hugo Arnoldo Mitre Hernández

Codirector de Tesis:

M. en E. Sergio Martín Nava Muñoz

Autorización de la versión final

Zacatecas, Zac., 19 de abril de 2021

Agradecimientos

A mi familia, en especial a mi mama Georgina Rodríguez, mis abuelos Anselmo Rodríguez y Oralia Castro, por ser un ejemplo para mí, por todo su apoyo, por su cariño y por las enseñanzas que me han dado.

A mi asesor el Dr. Hugo Arnoldo Mitre por su apoyo y dedicación para tener una mejor experiencia durante mi estancia en CIMAT, así como todos sus aportes dentro de este trabajo de investigación.

A mi co-asesor el M. en E. Sergio Martín Nava Muñoz por todas sus aportaciones y apoyo para culminar este trabajo.

a mis compañeros por su amistad y hacer de esta estancia algo más ameno.

A todas aquellas personas que conocí durante este tiempo de maestría y se volvieron parte importante en esta experiencia.

A la comunidad de CIMAT, en especial CIMAT-Zacatecas por la oportunidad de ingresar a esta maestría y el apoyo brindado durante mis estudios.

Al Consejo Nacional de Ciencia y Tecnología (CONACyT) por la beca otorgada durante mis dos años de maestría.

Resumen

La memorización es un proceso cognitivo clave en el contenido de aprendizaje. Un curso con tareas de memoria difícil, puede provocar una sobrecarga cognitiva y frustración en un grupo de alumnos y, por su parte, con tareas de memoria fácil, se puede evocar baja carga cognitiva que llevaría a los alumnos al aburrimiento; ambos casos con malos resultados para el aprendizaje. Para controlar la carga cognitiva en los cursos de aprendizaje, primero hay que medirla. Sin embargo, actualmente no hay clasificadores de carga cognitiva ni de dificultad de tareas de memoria. En este trabajo se propone un clasificador de dificultad de tareas de memoria de trabajo, tomando como base características de carga cognitiva con datos oculares. Se realizaron dos experimentos, uno en donde la carga cognitiva es inducida por tareas de memoria auditivas y el otro con tareas de memoria visual. En el primero se identifican dos niveles de dificultad: fácil y difícil. Estos niveles se consiguieron solicitándole a los participantes que hablaran con la verdad en una entrevista o respondieran con una mentira. Posteriormente, fueron clasificados con una máquina de soporte vectorial (*SVM, del ingles: Support Vector Machine*) y el clasificador de análisis discriminante lineal (*QDA, del ingles: Quadratic Discriminant Analysis*). En el segundo caso, se realiza un análisis de los datos de la pupila a partir de tres tareas de memorización con dificultad fácil, media y alta. Se seleccionaron las características con diferencias significativas y se entrenaron los clasificadores más precisos utilizados en la investigación de *eye tracking* como SVM, árbol de decisión (*DT, del ingles: Decision Tree*), análisis discriminante lineal (*LDA, del ingles: Linear Discriminant Analysis*) y bosque aleatorio (*RF, del ingles: Random Forests*).

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

El término memoria de trabajo se refiere a un sistema cerebral que proporciona almacenamiento temporal y manipulación de la información necesaria para tareas cognitivas tan complejas como la comprensión, el aprendizaje y el razonamiento del lenguaje (Baddeley, 1992), por lo tanto, la memorización es un proceso cognitivo clave del cerebro (Wang, 2009). En el aprendizaje, una nueva tarea debe crear una nueva rutina cognitiva que detalla la secuencia de procesos cognitivos necesarios para realizar la tarea, en la cual se involucra el rendimiento de la memoria de trabajo.

En un futuro sería interesante controlar la dificultad de una tarea, para ello es importante medir la carga cognitiva mientras los usuarios la estén realizando. Desde hace tiempo, se han investigado diferentes enfoques mediante señales fisiológicas para la medición de carga cognitiva, en su mayoría, estos presentan un problema: son invasivos para el usuario (Lin y Kao, 2018; Nourbakhsh y col., 2017). Entre los distintos estudios que se han llevado a cabo, hay uno en especial que destaca por no ser invasiva con el usuario; esta técnica obtiene información mediante rastreadores oculares remotos.

Los rastreadores oculares ofrecen información valiosa como: la cantidad de saltos sacádicos (Vrij y col., 2015), dilatación de la pupila (Siegle, Ichikawa y Steinhauer, 2008), número de parpadeos (Fukuda, 2001), entre otros. Esta información ha sido estudiada

por diferentes investigadores en los últimos años para relacionarla con la carga cognitiva. Se ha logrado comprobar que algunas variables oculares muestran cambios en base a la demanda de carga mental que las actividades requieran (Siegle, Ichikawa y Steinhauer, 2008).

En resumen, medir la carga cognitiva es relevante para controlar la dificultad en un futuro próximo, por lo que tener un clasificador que sea capaz de dividir dicha carga en diferentes niveles de dificultad, es importante para el control de la misma.

1.1. Sentencia del problema

La teoría de la carga cognitiva (Sweller, 1988) desempeña un papel importante en la investigación de la interacción humano-computadora. Existe una necesidad imperiosa de una medida no invasiva de la carga cognitiva de los individuos, ya que puede guiar a los diseñadores de sistemas interactivos para evitar sobrecargar a los usuarios.

En los últimos años, la carga cognitiva se midió empleando encuestas como el Índice de carga de tareas de la NASA (Hart, 2006) o medidas fisiológicas con electroencefalograma (*EEG, del íngles: Electroencephalogram*) (Lin y Kao, 2018), respuesta galvánica de la piel (*GSR, del íngles: Galvanic Skin Responses*) (Nourbakhsh y col., 2017) y rastreadores oculares (Marandi y col., 2018). Sin embargo, las encuestas y varios dispositivos de contacto fisiológico con la piel (EEG y GSR) pueden ser invasivos para el usuario, con excepción del rastreador ocular remoto que se coloca debajo de la pantalla a una distancia de 50-70 centímetros del usuario. Por esta razón, este trabajo de investigación se basa en el seguimiento ocular.

En los estudios de este último, el tamaño de la pupila (Kun y col., 2013a; Palinko y Kun, 2012) es una respuesta fisiológica utilizada para medir la carga cognitiva durante tareas de memoria, decisión y de percepción. También existen modelos de clasificación que manejan características oculares con diversas aplicaciones como tipos de lectura

(Biedert y col., 2012; Kun y col., 2013a), estados mentales (Eivazi y Bednarik, 2011), nivel de fatiga mental (Li y col., 2020) y comportamiento de conducción (Deng y col., 2019). Sin embargo, no se han encontrado clasificadores de carga cognitiva para la detección de dificultad en tareas de memoria de trabajo.

1.2. Motivación

Tenemos una cantidad limitada de capacidad de procesamiento disponible en los canales verbales y visuales (Mayer y Moreno, 2003). Esta capacidad puede sobrecargarse por la carga cognitiva generada durante las tareas de memoria de trabajo, también llamada sobrecarga cognitiva. La medición de esta carga y clasificación de dificultad, podría permitir que un sistema responda de manera adecuada, ya sea disminuyendo o aumentando el nivel de dificultad de la tarea.

Por estas razones, el contenido de aprendizaje debe diseñarse de manera que se minimice cualquier carga cognitiva innecesaria (Mayer y Moreno, 2003). Existe una complejidad implícita en el contenido de aprendizaje que necesita un equilibrio para evitar la sobrecarga cognitiva, especialmente en aquellos estudiantes con bajo rendimiento (Yang y col., 2020). El contenido de aprendizaje con alta complejidad puede llevar a la frustración y la baja complejidad al aburrimiento; un minuto o más de frustración en el campo de la educación está asociado con malos resultados de aprendizaje (DeFalco y col., 2017); Además, el aburrimiento es la emoción más persistente en los contenidos de aprendizaje y está asociado con un aprendizaje deficiente (Baker y col., 2010). Podemos decir que la carga cognitiva que genera el contenido de aprendizaje –con sus tareas de memoria de trabajo implícitas– en los estudiantes, debe ser medida y controlada para reducir la sobrecarga cognitiva y el aburrimiento.

1.3. Objetivos

El objetivo general de esta tesis es crear y validar un clasificador capaz de diferenciar entre diferentes niveles de dificultad (fácil, medio y difícil) evocada por tareas de memoria de trabajo, esto utilizando características obtenidas a través de seguimiento ocular.

Se tomaron en cuenta los siguientes objetivos específicos:

1. Realizar una revisión literaria de clasificadores multiclase y la medición de carga cognitiva en tareas de memoria por seguimiento ocular.

2. Diseñar y ejecutar un experimento para la medición de carga cognitiva por el uso de memoria con estímulos auditivos.

3. Con los resultados del experimento realizar una análisis estadístico para encontrar diferencias significativas de las variables obtenidas con datos oculares.

4. Realizar un análisis estadístico para encontrar diferencias significativas a partir de un conjunto de datos utilizado para la medición de la carga cognitiva por el uso de la memoria de trabajo.

5. Diseñar, desarrollar y validar un clasificador con las características oculares más significativas a partir de un conjunto de datos utilizado para la medición de la carga cognitiva por el uso de la memoria de trabajo.

6. Discutir los resultados obtenidos de la experimentación y concluir con las contribuciones relevantes.

1.4. Estructura de tesis

Los siguientes capítulos están estructurados de la siguiente forma: Capítulo 2 Marco teórico, donde se encuentran los conceptos necesarios para el entendimiento del tema principal. Capítulo 3 Trabajos relacionados, aquí se expone una revisión literaria sobre medición de carga cognitiva y clasificadores que utilizan datos oculares. Capítulo 4 Clasificación de dificultad de tareas de memoria auditivas con datos oculares, en el cual se explica el procedimiento y los resultados de un primer experimento donde se utilizaron tareas de memoria auditivas. Capítulo 5 Clasificación de dificultad de tareas de memoria visuales con datos oculares, donde se expone un segundo experimento que utilizó tareas de memoria visuales. Capítulo 6 Conclusiones y trabajo futuro, con los resultados generales y posibles trabajos que se puedan seguir desarrollando a partir de esta investigación.

Capítulo 2

Marco teórico

2.1. Carga cognitiva por memoria de trabajo

La teoría de la carga cognitiva (Sweller, 1988) describe estructuras de aprendizaje en términos de un sistema de procesamiento de información que involucra memoria a largo plazo, el cual es capaz de almacenar eficazmente todos nuestros conocimientos y habilidades durante un tiempo que se cree puede llegar a ser permanente y, también, la memoria de trabajo que realiza las tareas intelectuales asociadas con la conciencia. La información solo se puede almacenar en la memoria a largo plazo después de ser atendida y procesado por la memoria de trabajo. La memoria de trabajo, sin embargo, es extremadamente limitada tanto en capacidad como en duración.

En términos generales, la carga cognitiva se refiere a la actividad mental impuesta en la memoria de trabajo durante un periodo de tiempo, es decir, si la carga mental aumenta, la carga cognitiva será mayor.

El hecho de que la memoria a corto plazo sea en capacidad extremadamente menor que la memoria a largo plazo no quiere decir que esta no sea utilizada comúnmente, al contrario, sin ella no seríamos capaces de recordar las cosas que realizamos cotidianamente, ni mucho menos llegar a tener un recuerdo de algo que sucedió hace mucho

tiempo. Esto sucede ya que la información guardada dentro de la memoria a largo plazo no puede ser utilizada directamente, sino que la memoria de trabajo es la encargada de recabar información dentro de la memoria a largo plazo para que esta sea utilizada. Esto quiere decir, que siempre mantendremos un nivel de carga cognitiva y mientras la información que va de la memoria a largo plazo a la memoria de trabajo sea mayor, la carga cognitiva se verá en aumento.

La memoria de trabajo no se encarga solo de recabar lo que hay en la memoria a largo plazo, también es dentro de ella que se crea y procesa nueva información, ambos procesos dan a la carga cognitiva más trabajo, por consiguiente, cuando demandamos recabar, procesar y crear nuevos recuerdos es cuando mayores niveles de carga cognitiva presentamos.

2.2. Datos oculares para medir la carga cognitiva por uso de memoria de trabajo

Un rastreador ocular es un dispositivo que permite a los investigadores observar la posición del ojo para entender dónde una persona está mirando. La mayoría de los rastreadores oculares modernos se basan en un método llamado reflejo corneal para detectar y realizar un seguimiento de la ubicación del ojo mientras este se mueve. La reflexión corneal utiliza una fuente de luz para iluminar el ojo, lo que provoca una reflexión que es detectada por una cámara de alta resolución. La imagen capturada por la cámara se utiliza para identificar el reflejo de la fuente de luz sobre la córnea, la distancia entre el centro de la pupila y la reflexión sobre la córnea es calculada para saber la ubicación de la pupila (ver figura 2.1). En la figura 2.1 se muestra un ejemplo donde en rojo se observa el centro de la pupila, verde el centro del reflejo corneal provocado

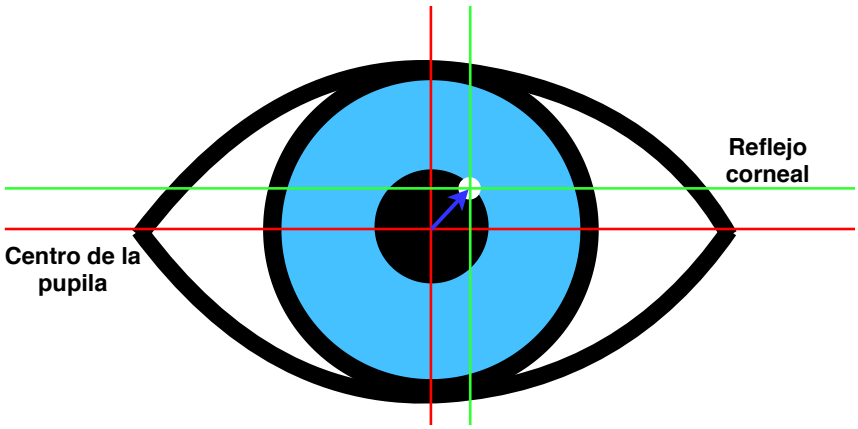

FIGURA 2.1: Ejemplo del funcionamiento de un rastreador ocular con reflexión corneal.

por la fuente de luz y la flecha azul representa la distancia entre los dos puntos que es calculada para obtener parámetros de los movimientos de los ojos.

El seguimiento ocular ha sido aplicado en numerosos campos de estudio, incluyendo factores humanos, psicología cognitiva, marketing, experiencia de usuario y el amplio campo de la computación centrada en el humano. El proceso de observación de una persona se divide en una gran cantidad de variables; para el desarrollo de este trabajo, se tomó en cuenta la dilatación de la pupila, movimientos oculares y parpadeos.

El movimiento ocular es una habilidad que no necesariamente se desarrolla de la misma manera en cada ojo. Esto quiere decir que en una misma persona sus ojos pueden reaccionar diferente a un mismo estímulo. Se dividen en tres movimientos distintos: fijación, seguimiento y saltos sacádicos. Las fijaciones son las pausas del movimiento de los ojos en un área específica del campo visual. Durante las fijaciones, los ojos no permanecen totalmente inmóviles, por el contrario, realizan constantes movimientos involuntarios de pequeña amplitud (Gila, Villanueva y Cabeza, 2009).

La dispersión del tiempo en las fijaciones de los ojos en la pantalla expresada como entropía durante respuestas fáciles y difíciles en tareas de memoria auditiva, esta relacionada con la carga cognitiva (Mitre-Hernandez y col., 2019). Además, la longitud (Mitre-Hernandez y col., 2019), la velocidad máxima (Di Stasi y col., 2010) y los

movimientos (Marandi y col., 2018) de los saltos sacádicos pueden describir la carga cognitiva. Aunque para el cálculo correcto de fijaciones y movimientos sacádicos se necesita un rastreador ocular remoto a 60 Hz o más.

La dilatación de la pupila ocurre cuando se incrementa el tamaño de la pupila, generalmente se relaciona con los cambios de luz, siendo ésta un mecanismo para que estos cambios no afecten la visión. Hoy en día, está comprobado que hay más factores que ocasionan una dilatación de la pupila, uno de estos son los cambios en la carga cognitiva (Hess y Polt, 1964).

El uso del tamaño de pupila como indicador de carga cognitiva se remonta a Hess y Polt, 1964, quienes demostraron que existe una relación entre la dilatación de la pupila y la dificultad de la tarea (el diámetro de la pupila aumenta, a su vez, con la dificultad del problema). Actualmente, los datos sobre cambios en el diámetro de la pupila, medidos con tecnología de seguimiento ocular, se utilizan ampliamente para estudiar tareas de carga cognitiva como conducir un vehículo mientras se escucha un diálogo (Kun y col., 2013b), interactuar con interfaces para la toma de decisiones (Lallé y col., 2015), realizar ejercicios matemáticos , memorización de números y percepción de estímulos visuales (Beatty, 1982) y realización de operaciones mentales-aritméticas (Papesh, Goldinger y Hout, 2012a), entre otras.

El parpadeo es una señal de comportamiento (Irwin y Thomas, 2010) también denominada como una respuesta psicofisiológica (Tanaka y Yamaoka, 1993) que es controlada por el sistema nervioso central. Los parpadeos están influenciados por los procesos cognitivos, como ejemplo la frecuencia de parpadeo durante tres tareas visuales diferentes, categorizadas por nivel de dificultad, se analizaron en sujetos jóvenes y revelaron diferencias estadísticamente significativas en la frecuencia de parpadeo (Marandi y col., 2018). En otros estudios, la frecuencia de parpadeo se vio afectada por los cambios entre las tareas auditivas –preguntas para respuestas planificadas (fáciles) o espontáneas (difíciles) (Mitre-Hernandez y col., 2019), entrevistas (Frosina y col., 2018a)

y preguntas sí/no (George y col., 2017). Otros enfoques para caracterizar el efecto de la carga cognitiva sobre el parpadeo incluyen los realizados durante tareas como conducir, es decir, la frecuencia de parpadeo disminuye cuando aumenta la complejidad del entorno de conducción (Faure, Lobjois y Benguigui, 2016). Se puede argumentar que la frecuencia de parpadeo es apropiada para la medición de la carga cognitiva; sin embargo, para no perder el registro de parpadeos rápidos, la frecuencia del dispositivo debe ser de 100 Hz o más.

2.3. Clasificadores supervisados

En el aprendizaje supervisado, al elemento de aprendizaje se le da el valor correcto (o, aproximado) de la función para entradas particulares y cambia su representación para tratar de coincidir con la información proporcionada por la retroalimentación.

En palabras más simples, el algoritmo aprende de supervisar acciones históricas (puntos de datos) para producir una salida. Dentro del aprendizaje automático supervisado, hay docenas de enfoques diferentes, pero este estudio se centra en máquina de soporte vectorial, análisis discriminante lineal, árbol de decisiones y bosque aleatorio.

2.3.1. Máquinas de soporte vectorial

Las máquinas de soporte vectorial (*Support Vector Machine*, SVM) son un método de aprendizaje supervisado, cuya propuestas fue por Cortes y Vapnik, 1995. Inicialmente, se aplicaron para problemas de clasificación de dos grupos. Su funcionamiento se basa en localizar la entrada en un espacio multidimensional, donde se ubica el hiperplano óptimo que separa las dos clases. Más tarde, se propusieron diferentes técnicas para utilizar SVM en problemas de clasificación con más de dos clases (Chauhan, Dahiya y Sharma, 2019), tales como uno contra uno, uno contra el resto, etc.

Una SVM ubica los datos de entrada a un espacio de características de dimensión mayor, esto con la finalidad de que se vuelvan más separables en comparación con el espacio de entrada original, para después encontrar el hiperplano que los separe y maximice el margen m que existe entre el hiperplano y las clases. Maximizar el margen m es un problema de programación cuadrática y puede ser resuelto por su problema dual introduciendo multiplicadores de Lagrange. Para la selección del hiperplano, se consideran los datos de entrenamiento que caen en las fronteras de los márgenes; a estos se les conoce como vectores de soporte. Algunas de las razones por las que este método ha tenido éxito, es que no padece de mínimos locales y el modelo solo depende de los datos con más información, los cuales son vectores de soporte.

Dicho de otra manera, una SVM es un modelo lineal para problemas de clasificación. Puede resolver problemas lineales y no lineales y, a su vez, funcionar bien para muchos problemas prácticos. La idea de SVM es sencilla: el algoritmo crea una línea o un hiperplano que separa los datos en clases.

Para entender mejor a qué se refieren los conceptos como margen, hiperplano, etc., se presenta un ejemplo en donde se aprecia su funcionamiento dentro de un problema de clasificación lineal y no lineal. Trabajar con una SVM implica tener un conjunto de datos etiquetados para ser utilizados en el entrenamiento. Pensemos en un conjunto de datos S de dos clases que sean separables y se encuentren debidamente etiquetados como se observa en la figura 2.1.

$$S = \{(x_1, y_1), ..., (x_n, y_n)\} \tag{2.1}$$

Suponiendo que este conjunto de datos representa a dos clases (figrua 2.2). Para resolver este problema se necesita clasificar los puntos rojos de los puntos azules. En una primera aproximación, lo que hacen las SVM es encontrar un hiperplano entre los datos de las dos clases. SVM es un algoritmo que toma los datos como entrada y genera

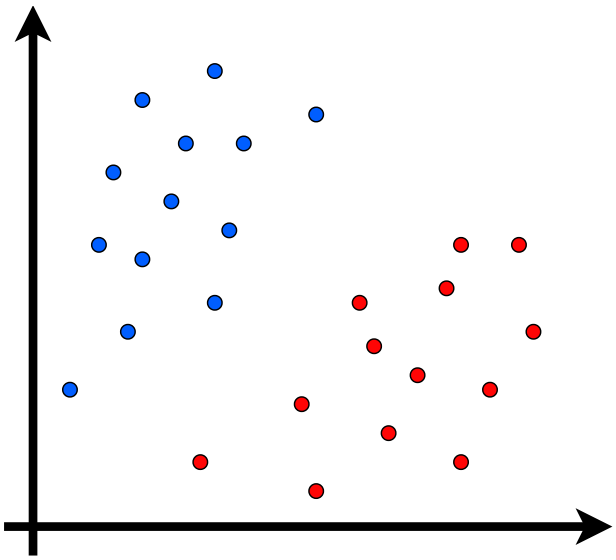

FIGURA 2.2: Distribución de datos de entrada de dos dimensiones, con dos grupos debidamente etiquetados que son linealmente separables.

una línea que separa esas clases si es posible. Entonces, su tarea es encontrar la línea ideal que separe este conjunto de datos en dos clases (puntos rojos y azules).

No es una tarea difícil, pero como se puede notar, no hay una línea única que resuelva el problema. De hecho, tenemos infinitas líneas que pueden separar estas dos clases, esta situación se observa en la figura 2.3. Las lineas amarillas se muestra bastante cerca de las clases roja y azul. Aunque clasifican los conjuntos de datos actuales, no es una línea generalizada y en el aprendizaje automático nuestro objetivo es obtener un separador óptimo. En cambio, la línea verde es la que estamos buscando. Es visualmente intuitivo en este caso que la línea verde clasifica mejor.

De acuerdo con el algoritmo SVM, primero se encuentran los puntos más cercanos a la línea de ambas clases; estos puntos se denominan vectores de soporte. Después, se calcula la distancia entre la línea y los vectores de soporte, esta distancia se llama margen y su objetivo es maximizar ese margen. El hiperplano para el cual, el margen es máximo, es el hiperplano óptimo. Estos conceptos se pueden ver gráficamente en la figura 2.4.

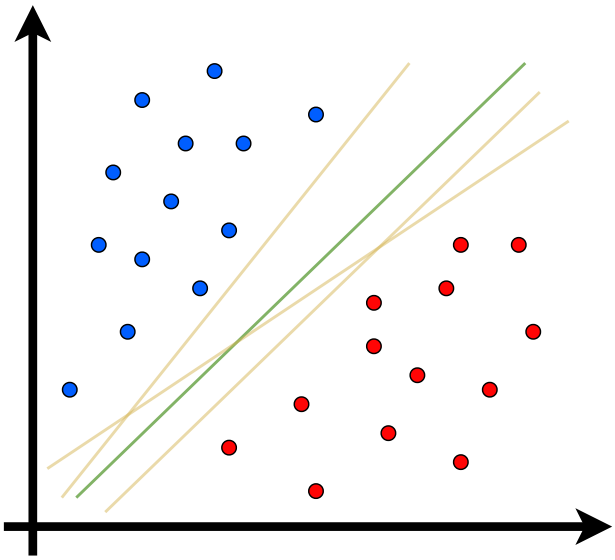

FIGURA 2.3: Posibles lineas que puedan separar los dos grupos.

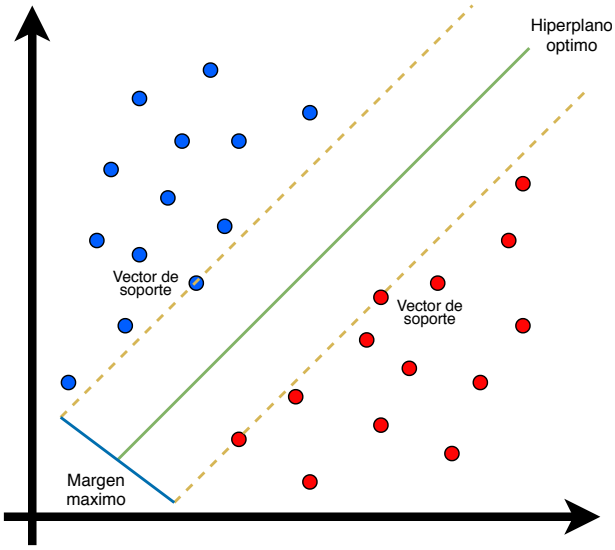

FIGURA 2.4: Hiperplano óptimo que separa las clases.

Por lo tanto, SVM intenta establecer un límite de decisión de tal manera que la separación entre las dos clases sea lo más amplia posible. Este es un caso bastante simple, pero cuando los datos no se pueden separar linealmente, tal como se muestra en la figura 2.5, habrá que tomar otras consideraciones. No se puede trazar una línea recta que pueda clasificar estos datos. Pero estos datos se pueden convertir en datos linealmente separables en una dimensión superior. Si se agrega una dimensión más, la cual

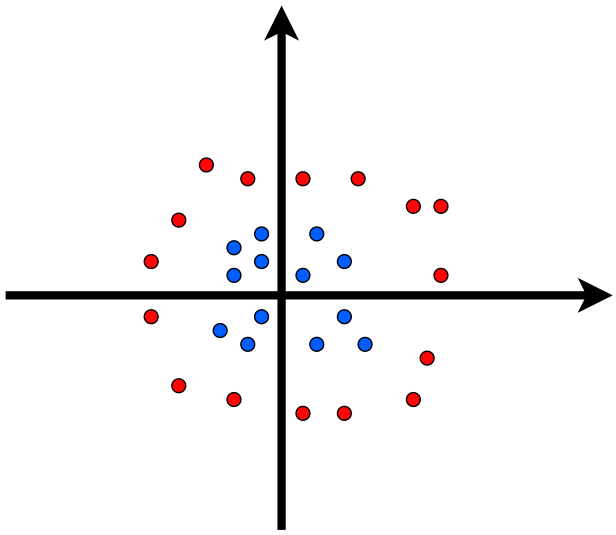

FIGURA 2.5: Distribución de datos de entrada de dos dimensiones con dos grupos debidamente etiquetados que no son linealmente separables.

llamaremos eje z y sus coordenadas están dadas por la ecuación 2.2, tenemos que:

$$z = x^2 + y^2 \tag{2.2}$$

Entonces, básicamente la coordenada z es el cuadrado de la distancia entre el punto y el origen. Si se gráfica tomando en cuenta el nuevo eje z, ahora los datos son claramente separables linealmente (ver figura 2.6).

Por lo tanto, es posible clasificar los datos agregándoles una dimensión adicional para que se vuelvan separables linealmente y luego proyectar el límite de decisión de nuevo a las dimensiones originales mediante una transformación matemática. En el área de las SVM a esta transformación es conocida como kernel.

El kernel es una función que simula la proyección de los datos iniciales en un espacio de características con mayor dimensión (Yekkehkhany y col., 2014), para que en esta nuevo espacio los datos sean linealmente separables. Los tres kernels mas usados son:

- Kernel polinomial $K(x, x_i) = (\langle x, x_i \rangle + 1)^p$
- kernel sigmoide $K(x, x_i) = tanh(\langle x, x_i \rangle + 1)$

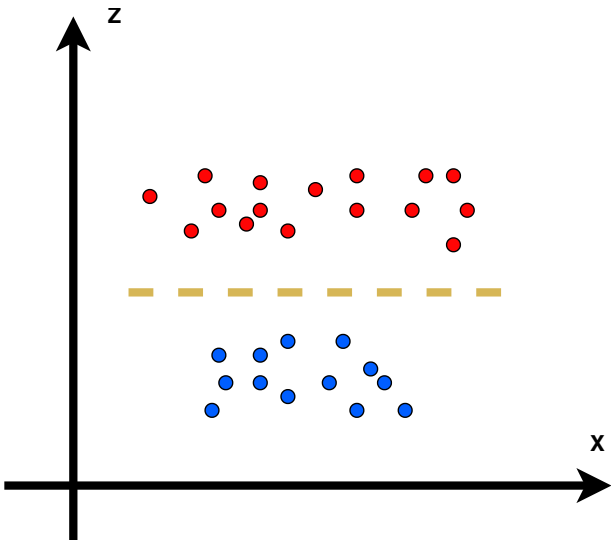

FIGURA 2.6: Separación de dos grupos que no son linealmente separables al convertirlos a una dimensión mayor.

- Kernel Función de base radial (RBF, del ingles: Radial basis function) $K(x, x_i) = exp^{\frac{|x-x_i|^2}{2\sigma^2}}$

2.3.2. Análisis discriminante lineal

Los clasificadores basados en análisis discriminante lineal (*Linear Discriminant Analysis*, LDA), originalmente propuestos por R. A. Fisher (Fisher, 1936), se encuentran entre los algoritmos más simples. Basado en el uso del principio de máxima verosimilitud, LDA resulta ser el clasificador óptimo que logra la tasa de clasificación errónea más baja con el supuesto de que los datos son gaussianos con estadísticas perfectamente conocidas y una matriz de covarianza común entre las clases (Hastie et al., 2001). Sin embargo, en la práctica la matriz de covarianza y las medias asociadas con cada clase no pueden conocerse perfectamente.

A LDA también se le conoce como una técnica de reducción de dimensionalidad. Como su nombre lo indica, este método disminuye el número de dimensiones (características) en un conjunto de datos mientras retienen la mayor cantidad de información

posible.

El objetivo de esta técnica es discriminar diferentes clases en un espacio de baja dimensión reteniendo los componentes que contienen valores de características que tienen la mejor separación entre clases.

Para la clasificación, LDA está diseñado para separar dos o más clases de observaciones basadas en una combinación lineal de características, siendo la designación lineal, el resultado de que las funciones discriminantes sean lineales. Además, se basa en el teorema de Bayes, el cual formula que podemos relacionar la probabilidad de que X pertenezca a cada clase con la probabilidad de que la entrada tome un valor dentro de cada clase.

$$P(Y|X) = \frac{P(Y|X)P(Y)}{P(X)} \tag{2.3}$$

Entonces, LDA asume que $P(X|Y)$ como una función de densidad, la cual es la función de una distribución normal. Al sustituir con la fórmula de una distribución normal por $P(X|Y)$, nos queda la siguiente función:

$$\delta(X) = \frac{\mu_i X}{\sigma^2} - \frac{1}{2}\frac{\mu_i^2}{\sigma^2} + \log(P(Y = i)) \tag{2.4}$$

Esta es una suposición clave de LDA. La técnica funcionará mejor si las variables de entrada provienen de una distribución aproximadamente normal y, cuanto menos se mantenga esta aproximación, peor será el rendimiento. Con la ecuación 2.4 podemos clasificar X que pertenece a la clase que produce el mayor valor de función discriminante. Tenga en cuenta que la fórmula es lineal en X, que es donde se origina su nombre.

2.3.3. Árbol de decisión

Los árboles de decisión son un algoritmo de aprendizaje automático que sirve para realizar una clasificación o regresión de datos a través de una estructura jerárquica.

Dicho de otra forma, su funcionamiento consiste en identificar los factores clave que diferencian entra las distintas clases de nuestros datos. Al hacerlo, los árboles de decisión pueden tomar datos de entrada y predecir una clase al ser ejecutados a través de un conjunto de preguntas que difieren entre sí, estas mismas, se forman mediante el aprendizaje automático.

Las preguntas generadas a través del aprendizaje automático son preguntas con respuestas del tipo si/no. Es decir, después de cada pregunta existen dos rutas que el árbol de decisiones puede tomar, la ruta si y la ruta no. Cada uno de estos caminos conducirá a otra pregunta o al resultado final. Este resultado se obtiene cuando ya no existen más preguntas por contestar.

Existen tres conceptos clave para entender el funcionamiento de un árbol de decisiones, los cuales definiré a continuación:

- Clases: Cuando se realiza una clasificación, los diferentes grupos a los que pueden pertenecer los conjuntos de datos.
- Nodos: Las distintas preguntas que forman al árbol de decisión se conocen como nodos. Estos pueden verse como puntos en el árbol de decisiones donde se forma una división. Cada una de estas subdivisiones es una pregunta de si o no.
- Hojas: Debido a la presencia de nodos, hay muchas rutas alternativas que se crean en cualquier árbol de decisión, aun cuando tengan pocas capas de nodos. Los puntos finales de cada una de estas rutas se conocen como hojas. Estas representan el valor o la clase final que va a predecir para los datos de entrada.

En la figura 2.7 se puede observar un ejemplo de cómo se vería un árbol de decisiones que se utiliza para clasificar 3 clases: bicicleta, motocicleta y automóvil. Los recuadros blancos son los nodos y los recuadros verdes son las hojas.

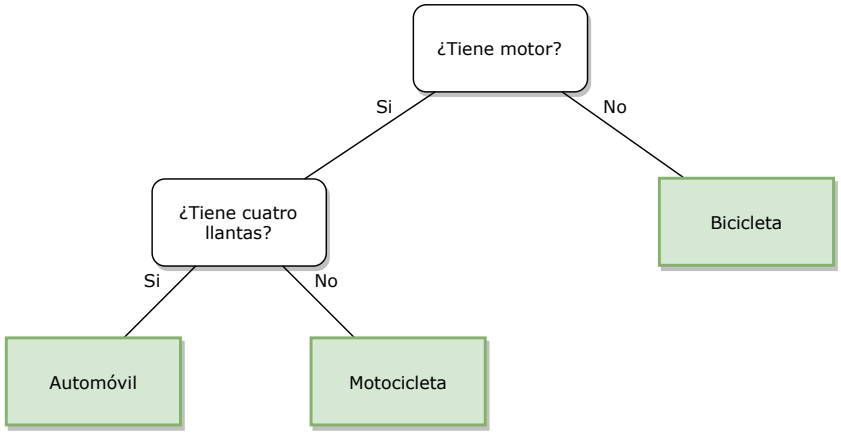

FIGURA 2.7: Árbol de decisión con tres clases (bicicleta, automóvil y motocicleta).

Es el algoritmo de aprendizaje automático de árbol de decisiones durante el entrenamiento, el que se encarga de analizar las tres clases y formular las preguntas necesarias para diferenciar entre ellas.

Una vez que el árbol es entrenado y las preguntas fueron formuladas, podemos predecir a qué clase pertenecen nuestros datos, simplemente respondiendo a estas preguntas de si o no.

2.3.4. Bosques aleatorios

Aun cuando los árboles de decisión tienen un buen funcionamiento en la clasificación, existe un límite en la precisión con la que un árbol de decisión individual puede realizar una clasificación. Por lo tanto, en lugar de utilizar uno solo, con mucha frecuencia tendemos a crear un conjunto de árboles de decisión que realizan dicha distribución. Este modelo se conoce como bosques aleatorios y ofrecen una mejor precisión que la de un árbol de decisión individual.

En el aprendizaje automático un modelo de bosques aleatorio consta de varios árboles de decisión. La razón por la que se denominan aleatorios es porque cada árbol se entrena mediante un subconjunto de datos de entrenamiento.

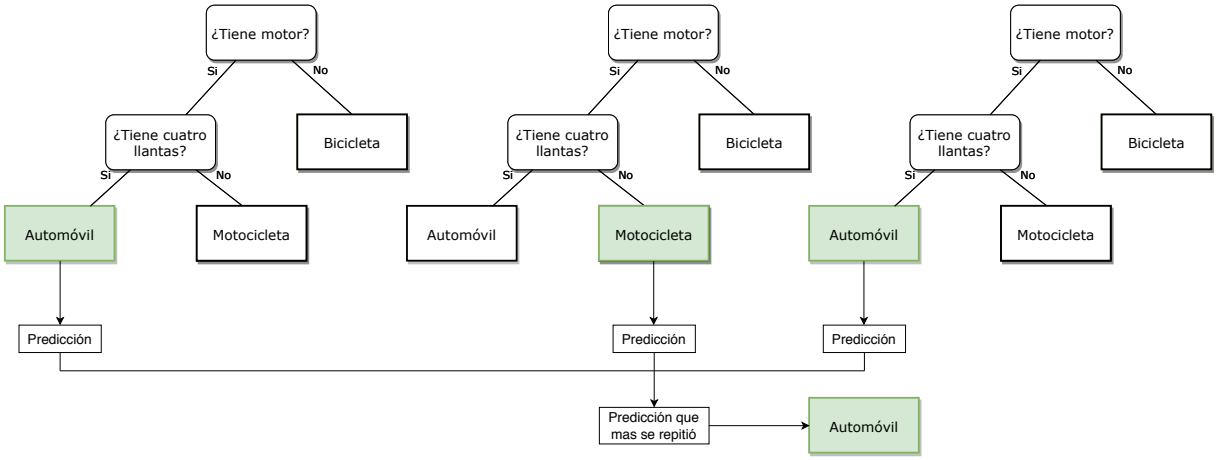

FIGURA 2.8: Ejemplo de bosque aleatorio con tres arboles de decisión.

En los modelos típicos de bosques aleatorios, un subconjunto muestreado al azar de los datos sobre los que se va a realizar la clasificación se pasa a través de cada uno de los árboles de decisión en el bosque para entrenar ese árbol. Al momento de realizar la clasificación de un conjunto de datos, cada árbol de decisión que forma al bosque aleatorio realiza una predicción, la cual se decide midiendo qué clase fue predicha por una mayor cantidad de árboles de decisión. En la figura 2.8 se muestra un ejemplo de un bosque aleatorio conformado por tres arboles de decisión.

Una de las principales razones por las que los clasificadores de bosques aleatorios son tan efectivos, es porque cada árbol no está relacionado en gran medida con los demás, cuya condición se logra gracias al muestreo aleatorio de datos. Esto provoca que algunos de los árboles arrojen predicciones incorrectas, pero dado que se utiliza la clase predicha por la mayor cantidad de árboles, se logra obtener una precisión bastante alta.

Otra ventaja que ofrece el uso de bosques aleatorios para la clasificación, es que ayudan a eliminar los problemas que pueden causar los valores atípicos de los datos. Con la mayoría de los modelos de aprendizaje automático, estos valores atípicos tienen un impacto bastante grande en el modelo y reducen la precisión general. Con los clasificadores de bosque aleatorio, se puede reducir significativamente el problema, dado

que se está tomando una muestra aleatoria de los datos para cada árbol. Aunque habrá algunos que se entrenarán con estos valores atípicos, la mayoría de los árboles no y, por tanto, la predicción general que recibimos seguirá siendo correcta.

Capítulo 3

Trabajos relacionados

En esta sección se presentan distintos trabajos relacionados con la medición de carga cognitiva implementando datos oculares y clasificadores entrenados con datos obtenidos a través de rastreadores oculares. La clasificación que realizan difiere entre ellos, esto es porque hasta el momento de la elaboración de esta tesis, no hay clasificadores entrenados con datos oculares que estén siendo utilizados para medir la dificultad de una tarea dada.

3.1. Medición de carga cognitiva con datos oculares

Los parpadeos están influenciados por los procesos cognitivos. La frecuencia de parpadeos durante tres tareas visuales diferentes, categorizadas por nivel de dificultad, se analizaron en sujetos jóvenes y revelaron diferencias estadísticamente significativas en la frecuencia de parpadeo (Marandi y col., 2018). En otros estudios, la frecuencia de parpadeo se vieron afectados por los cambios entre las tareas auditivas en entrevistas (Frosina y col., 2018b) y preguntas de si o no (George y col., 2017). Otros enfoques para caracterizar el efecto de la carga cognitiva sobre los parpadeos incluyen los realizados durante tareas como conducir, done la frecuencia de parpadeo disminuye cuando aumenta la complejidad del entorno de conducción (Faure, Lobjois y Benguigui, 2016).

Se puede decir que la frecuencia de parpadeo esta relacionada con cambios de la carga cognitiva, sin embargo, para no perder el registro de parpadeos rápidos, la frecuencia del dispositivo debe ser de 100 Hz o más.

Las fijaciones de los ojos es otra variable extraída por rastreadores oculares que puede ser utilizada para la medición de carga cognitiva. Donde la velocidad máxima (Di Stasi y col., 2010), los movimientos (Marandi y col., 2018) de los saltos sacádicos y la duración de las fijaciones (Andrzejewska y Skawińska, 2020) pueden describir la carga cognitiva. Aunque para el cálculo correcto de fijaciones y movimientos sacádicos se necesita un rastreador ocular remoto a 60 Hz o más.

Desde 1964, los investigadores de pupilometría saben que el tamaño de la pupila cambia en respuesta a la actividad mental (Hess y Polt, 1964). Actualmente, los datos sobre cambios en el diámetro de la pupila, medidos con tecnología de seguimiento ocular, se utilizan ampliamente para estudiar tareas de carga cognitiva como conducir un vehículo mientras se escucha un diálogo (Kun y col., 2013a), interactuar con interfaces para la toma de decisiones (Lallé y col., 2015) , realizando ejercicios matemáticos, memorizando números y percibiendo estímulos visuales (Beatty, 1982) y, asimismo, realizando operaciones aritméticas mentales (Chen, Epps y Chen, 2011), entre otros. En un estudio de (Beatty, 1982) se informó una diferencia de 0,5 mm en términos de tamaño de la pupila al pasar de 3 a 7 dígitos memorizados. Además, otro estudio (Papesh, Goldinger y Hout, 2012b) reveló que el diámetro medio de la pupila aumentaba con la recuperación de la memoria a largo plazo de las palabras habladas y, esto mismo, inducía un aumento mayor en comparación con la creación de recuerdos (codificación). Para las tareas auditivas, la dilatación de la pupila presentó un diámetro mayor en respuestas difíciles para verdadero o falso (Webb y col., 2009), entrevistas (Nugroho, Nasrun y Setianingsih, 2017) y múltiples opciones (Nurçin y col., 2017), lo que indica que la dilatación máxima podría ser útil para medir la carga cognitiva .

La línea base de la pupila se utiliza para obtener el valor promedio del tamaño de

la pupila cuando al participante no se le está exigiendo un uso de carga cognitiva extra (Klingner, Tversky y Hanrahan, 2011) antes de la tarea de ejecución. Varios autores utilizan el cambio del diámetro de la pupila y, a partir de esta línea base, se mide la carga cognitiva; específicamente, el cambio medio del diámetro de la pupila en condiciones de conducción (Kun y col., 2013a; Palinko y Kun, 2012). Incluyendo el cambio porcentual promedio en el tamaño de la pupila para tareas de lectura, razonamiento de operaciones matemáticas y búsqueda de objetos (Iqbal y col., 2005; Lallé y col., 2015). Ambas medidas se analizan en este estudio.

El tiempo también se estudia en el proceso de memorización. La dilatación de la pupila alcanzó el punto más alto entre el final de la etapa de codificación y el inicio de la recuperación de la memoria (Siegle, Ichikawa y Steinhauer, 2008), revelando una posible asociación entre el tiempo de codificación de la memoria y la recuperación con el tamaño de la pupila; este fenómeno puede expresarse en una nueva medida de carga cognitiva.

3.2. Clasificadores relacionados con datos oculares

Para hacer una mejor selección de los clasificadores que fueron entrenados con datos extraídos de rastreadores oculares, se investigaron trabajos donde los autores utilizaron estos datos para clasificar diferentes situaciones, no necesariamente carga cognitiva. Las variables que utilizan estos sistemas pueden ser de parpadeos, saltos sacádicos, fijaciones y pupilares. En la tabla 3.1 se muestran los trabajos identificados a través de una revisión literaria, mostrando las clases, variables de entrenamiento, tipos de clasificadores implementados y sus precisiones.

Como podemos observar en la tabla 3.1, SVM es el clasificador más utilizado además de mostrar una precisión mayor en la mayoría de los casos. Biedert y col., 2012

TABLA 3.1: Trabajos de clasificadores relacionados con datos oculares

Autor	Clasificación	Tipo de características	Variables	Clasificador	Precisión
Biedert et al. (2012)	2 clases de lectura (lectura y skimming).	Movimientos oculares	T1={Características de Fijaciones} T2={Características de saltos sacádicos} T3={Características de fijaciones y saltos sacádicos}	SVM (RBF)	80.0(T_1)
					79.0(T_2)
					88.0(T_3)
Kollmorgen & Holmqvist (2007)	2 clases de lectura (lectura y no lectura).	Movimientos oculares	Duración de las fijaciones, Tamaño de saltos sacádicos	HMM	91.0
				HMM simulado con línea de condición	88.0
				HMM no supervisado	87.0
				Neural Network	88.0
Kunze et al. (2013)	5 clases tipos de lecturas (novela, manga, revista, periódico y libro de texto	Movimientos oculares	8 características extraídas de fijaciones y sacádicos	Árbol de decisión	74.0
Eivazi & Bednarik(2011)	5 patrones cognitivos (Cognición, evaluación, planeación, intención y movimiento concurrente)	Movimientos oculares	T_1={Duración media de fijación, suma de la duración de las fijaciones, distancia promedio entre fijaciones, suma total de las distancias entre fijaciones, numero de fijaciones, tasa de fijaciones} T_2={T1, tiempo medio de acción en los enunciados}	SVM	53.25(T_1)
	3 clases de rendimiento (bajo, medio y alto).	Tiempos en la tarea			66.48(T_2)[1]
					87.5(T_2)[2]
Li et al. (2020)	3 clases de fatiga mental (bajo, medio y alto)	Parpadeos Pupila Movimientos oculares	T1={21 características de pupila y parpadeo} T2={56 características de parpadeo y mirada} T3={63 características de movimientos oculares} T4={70 características de movimientos oculares y duración de la tarea}	SVM	85.0(T_1) 81.3(T_2) 79.5T(T_3) 84.1(T_4)
				Árbol de decisión	78.4(T_1) 79.4(T_2) 79.7(T_3) 87.1(T_4)
				Árbol mejorado	81.0(T_1) 71.5(T_2) 73.6(T_3) 75.4(T_4)
				KNN	76.5(T_1) 63.4(T_2) 63.9(T_3) 73.9(T_4)
		Tiempos de la tarea		LDA	86.0(T_1) 81.2(T_2) 80.5(T_3) 78.8(T_4)
Deng et al. (2019)	3 clases de los comportamientos de conducción (cambio de carril izquierdo / derecho y el mantenimiento del carril)	Entorno	T_1={51 características de saltos sacádicos, parpadeos, mirada} T_2={Velocidad del vehículo, distancia al vehículo en frente, distancia al vehículo en la parte delantera izquierda, distancia al vehículo en la parte delantera derecha, distancia al vehículo dejado atrás, tiempo para colisión, número del carril actual} T_3={T1,T2}	SVM	70.32(T_1) 92.45(T_2) 79.79(T_3)
				HMM	64.94(T_1) 93.66(T_2) 94.37(T_3)
				CNN	86.19(T_1) 89.25(T_2) 90.35(T_3)
		Movimientos oculares Parpadeos		RF	93.66(T_1) 99.14(T_2) 99.92(T_3)

[1]3 clases de rendimiento (bajo, medio y alto), [2]2 clases de rendimiento (bajo y alto)

entrenó un clasificador SVM para detectar cuando los participantes realizaban una lectura completa o una lectura tipo "skimming", este clasificador fue entrenado con tres conjuntos de características: el primero estuvo constituido por variables de fijaciones, el segundo eran variables de saltos sacádicos y, por último, una combinación de variables de fijaciones y saltos sacádicos. Aplicando un kernel RBF consiguieron una precisión del 80 % en el primer conjunto, 79 % en el segundo y 88 % en el tercero.

Los clasificadores SVM también han sido utilizados para inferir el rendimiento y el estado cognitivo de una persona al realizar una tarea (Eivazi y Bednarik, 2011). Mientras los participantes resolvían un rompecabezas (Eivazi y Bednarik, 2011) recolectó los datos oculares para entrenar un clasificador que fuera capaz de detectar estados cognitivos como la planificación o la cognición. Implementaron un clasificador SVM con kernel RBF, entrenado con variables obtenidas de las fijaciones de los usuarios. El resultado final fue un clasificador con una precisión de 53.25 % para 5 clases diferentes de estados cognitivos. Además, usaron variables obtenidas de la duración de la tarea junto con las fijaciones para entrenar dos clasificadores de rendimiento, uno con tres clases (bajo, medio y alto) y el otro con dos (bajo y alto), consiguiendo una precisión del 66.48 % y 87.5 %, respectivamente.

En el área de la construcción, (Li y col., 2020) clasificó el nivel de fatiga mental en los operadores utilizando diferentes clasificadores. SVM fue el clasificador que mejores resultados obtuvo al implementar cuatro conjuntos de entrenamiento diferentes. Estos consistían en medidas obtenidas de pupila y parpadeos, parpadeos y mirada en pantalla, todas las medidas oculares juntas y, por último, un conjunto basado en las medidas oculares y medidas obtenidas del rendimiento en la tarea que los participantes realizaron. Los resultados obtenidos fueron 85 % para el primer conjunto de características, 81.3 % para el segundo, 79.5 % para el tercero y 84.1 % para el cuarto.

Por último, Deng y col., 2019 entrenó un clasificador SVM de tres clases para predecir comportamientos dentro de la conducción (cambio de carril izquierdo/derecho y

mantenerse en el mismo carril). Utilizaron tres conjuntos de entrenamiento, el primer conjunto de variables obtenidas del rastreador ocular (saltos sacádicos, parpadeos y mirada), un segundo grupo con variables ambientales como la velocidad del vehículo y, por último, una combinación de ambos conjuntos. Con estos datos de entrenamiento obtuvieron una precisión promedio del 70.32 %, 92.45 % y 79.79 %, respectivamente.

Las máquinas de soporte vectoriales han sido usadas repetidamente con datos extraídos de rastreadores oculares, su rendimiento mantiene un porcentaje promedio por encima del 80 % al clasificar hasta tres clases. Además, han mostrado un buen desempeño al usar diferentes conjuntos de medidas oculares, por ello podemos esperar una buena clasificación con variables extraídas de fijaciones, saltos sacádicos, pupilas y parpadeos. En la literatura actual, el kernel más utilizado es el RBF, sería interesante probar el rendimiento de otros kernels al trabajar con los datos extraídos de rastreadores oculares.

Los árboles de decisiones son clasificadores que también han sido empleados con datos extraídos de rastreadores oculares. Kunze y col., 2013 hizo una clasificación de los tipos de lecturas que los participantes realizaron mientras sus datos oculares eran registrados. Con variables extraídas de las fijaciones y saltos sacádicos, entrenaron un árbol de decisiones para detectar qué tipo de lectura estaban realizando. En total clasificaron 5 clases (novela, manga, revista, periódico y libro de texto), alcanzando una precisión del 74 %.

El mismo algoritmo de clasificación se utilizó para clasificar 3 niveles de fatiga mental (Li y col., 2020). Para esto, se usaron cuatro conjuntos de entrenamiento, donde combinaban variables de parpadeos, pupila, mirada y el ambiente de la tarea. Las precisiones que se obtuvieron con los cuatro conjuntos variaron entre 78.4 % y 87.1 %.

Estos trabajos demuestran que utilizar un árbol de decisiones con variables oculares puede llegar a un buen rendimiento en la clasificación. Se ha conseguido una precisión de 74 % con cinco clases y más de 80 % con tres. Para este trabajo, se espera que el

algoritmo llegue a una precisión por encima del 70 % al clasificar entre 3 clases.

En la tabla 3.1 nos encontramos con los bosques aleatorios, estos clasificadores son una variación de árboles de decisiones que genera un determinado número de árboles aleatoriamente, lo podemos ver como la búsqueda del mejor de ellos en un conjunto finito. Con este sistema, lograron tener precisiones por encima del 90 % (Deng y col., 2019). Esta precisión se consiguió al clasificar 3 clases de comportamientos en la conducción (girar a la izquierda/derecha y mantenerse en el carril). El entrenamiento consistió en una combinación de variables ambientales y oculares. Primero entrenaron al clasificador con un conjunto de características de los saltos sacádicos, parpadeos y mirada, consiguiendo una precisión del 93.66 %. Después, entrenaron el mismo método utilizando variables del entorno, con este entrenamiento se obtuvo una precisión del 99.14 %. Posteriormente, se entrenó con ambos conjuntos alcanzando una precisión del 99.92 %.

La precisión conseguida por Deng y col., 2019 está cerca del 100 % al clasificar entre 3 clases. Esto se consiguió mediante datos de rastreadores oculares, aunque solo se ha implementado en un trabajo dentro de esta revisión literaria, es importante considerarlo para nuestro trabajo debido a sus altos índices de exactitud.

Al igual que la máquina de soporte vectorial, el análisis discriminante lineal se aplicó en la clasificación del nivel de fatiga mental con tres niveles mientras trabajadores operaban maquinaria Li y col., 2020. Se utilizaron cuatro tipos de conjuntos de entrenamientos en donde estaban presentes variables extraídas de la pupila, mirada, parpadeos y duración de la tarea. Sus precisiones fueron de 78.8 % con todas las variables y al 86 % con variables de pupila y parpadeos.

Cuando comparamos los resultados de LDA con los otros 4 clasificadores que se usaron para clasificar entre tres clases de fatiga mental, nos encontramos con que fue el segundo con mejor rendimiento. Además, el que mejor resultado en precisión obtuvo con 3 de los 4 conjuntos de entrenamiento.

Los modelos ocultos de Markov son otro algoritmo que fue utilizado con buenos resultados. Kollmorgen y Holmqvist, 2007 diferenció entre dos clases con tres algoritmos distintos de los modelos ocultos de Markov, dos supervisados y uno no supervisado. Estos fueron entrenados con variables extraídas de las fijaciones y los saltos sacádicos, llegando a una precisión del 87 % con el modelo no supervisado y, mientras tanto, 88 % y 91 % con los supervisados.

Deng y col., 2019 utilizó los modelos ocultos de Markov para clasificar en tres clases los comportamientos durante la conducción. Para esto, emplearon 3 entrenamientos distintos. El primero consistió en variables extraídas a partir de un rastreador ocular, este obtuvo una precisión del 93.66 %. En el segundo, se usaron variables del entorno con las que se consiguió una exactitud del 99.14 %. Mientras el último, fue una combinación de los primeros dos, con el cual alcanzaron una precisión del 99.92 %, siendo esta la más alta.

A pesar del buen desempeño de los modelos ocultos de Markov al ser entrenados con variables oculares, en este trabajo de tesis no será implementado. Esto a causa de la arquitectura con la que operan, ya que los modelos ocultos de Markov funcionan como máquinas de estados que van cambiando en función del tiempo, donde tenemos dos patrones: uno visible, el cual se puede observar y, también, uno oculto cuyo cambio está directamente relacionado al primero. Para trabajar con este clasificador se tendría que utilizar otro tipo de variables que nos permitan modelar los cambios en la carga cognitiva como si se tratara de una máquina de estados.

Capítulo 4

Clasificación de dificultad en tareas de memoria auditiva con datos oculares

El esfuerzo mental se mide ampliamente en estudios de seguimiento ocular con datos como dilataciones de pupila, parpadeos, sacadas y fijaciones. Durante entrevistas, las respuestas engañosas necesitan más esfuerzo mental que las verdaderas y, este mismo, se le impone a la memoria de trabajo. En este experimento, se exploran medidas oculares asociadas con el uso de la memoria de trabajo en adolescentes mientras realizan tareas de memoria auditiva mediante la contestación de preguntas durante una entrevista, ya sea con una verdad (dificultad fácil) o alguna mentira (dificultad difícil).

4.1. Participantes

En esta investigación, 24 participantes (12 hombres y 12 mujeres) entre las edades de 12-13 años, con una edad promedio de M = 12.3 (SD = 2.2), fueron reclutados en una escuela secundaria en México. El director otorgó su consentimiento para realizar este estudio. No obstante, se conservó el anonimato de los participantes, por lo cual no se les solicitó información personal. A los implicados se les preguntó sobre sus condiciones

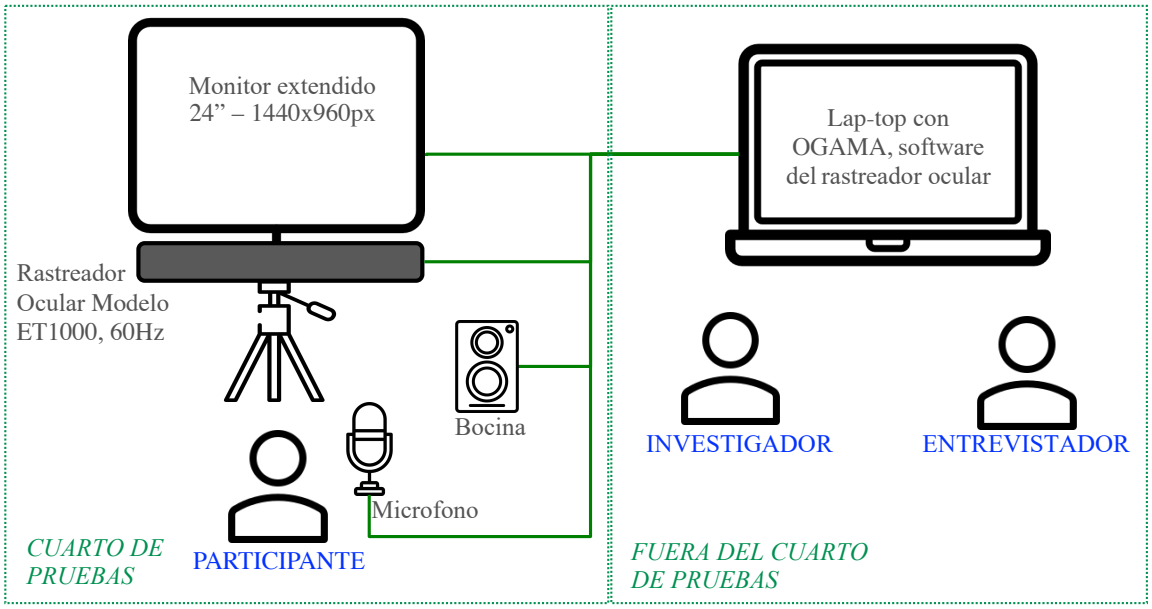

FIGURA 4.1: Configuración de los materiales utilizados durante el experimento de tareas de memoria auditiva

psicológicas (trastornos o síndromes) y visuales (miopía, hipermetropía, astigmatismo y presbicia) para reducir errores en los resultados del seguimiento ocular.

4.2. Materiales

Un dispositivo de seguimiento ocular "EyeTribe" modelo ET1000 con una frecuencia de muestreo de 60Hz y una precisión promedio de alrededor de 0.5 a 1°, fue acoplado a una pantalla (monitor extendido de 24") con resolución de 1440 x 960 píxeles; el rastreador ocular estaba ubicado a una distancia de 50-60 cm de la cara del participante. Se configuraron altavoces y un micrófono, permitiendo a los participantes escuchar y responder al entrevistador. Todos los dispositivos se conectaron a una computadora portátil que tiene el software de seguimiento ocular para la recolección de datos, en la figura 4.1 se puede apreciar un esquema sobre la estructura del experimento.

Se aplicaron dos cuestionarios antes de la recopilación de datos oculares. Los cuestionarios eran sobre temas de interés en México, tales como: aborto, pena de muerte,

matrimonio homosexual, drogas, sexualidad infantil, piratería y prostitución. En el primero, se implementó una escala de calificación de 7 puntos para saber qué tan de acuerdo estaban con los temas. El segundo, por su parte, se trató de una escala de honestidad de 7 puntos para acreditar cada declaración contestada en los distintos tópicos.

Se requieren algunas condiciones experimentales para controlar el tamaño de la pupila ocular. Para garantizar que la apariencia del entrevistador (una persona atractiva, por ejemplo) no actuara como estímulo para ningún participante y afectara el tamaño de su pupila (Burley, Gray y Snowden, 2017), el entrevistador usó un micrófono permaneciendo fuera de la vista de los participantes. Además, para reducir los factores que pueden cambiar el tamaño de la pupila de los integrantes, la luz exterior se controló con cortinas opacas y la ambiental interior con luz blanca.

4.3. Medidas

Para la validación de datos, los registros de Tamaño de pupila izquierda (*LPS, del ingles: left pupil size*) y Tamaño de pupila derecha (*RPS, del ingles: right pupil size*), se eliminaron con los siguientes criterios: LPS == 0 o RPS == 0. Además, el filtro de Hampel se usó para el seguimiento de datos sin procesar, para descartar valores atípicos.

- Línea base de la pupila (*BLPS, del ingles: Base Line Pupil Size*). Para el tiempo de observación de la imagen, previa al estímulo o la imagen de línea base, se estableció un período de 4 segundos y, asimismo, para la estabilización del tamaño de la pupila, se fijaron 2 segundos antes de la pregunta de estímulo. En resumen, el $BLPS$ se calcula como el promedio de las muestras recolectadas de estos últimos 2 segundos.

- Cambio medio del diámetro de la pupila (*MPDC, del ingles: Mean Pupil Diameter Change*) (Kun y col., 2013a; Palinko y col., 2010). Primero, la línea base $BLPS$ se

calculó 2 segundos antes de la pregunta. Durante la respuesta del participante, se recopilan los datos pupilares y el diámetro medio. Al diámetro medio de la pupila se le resta $BLPS$ para obtener este valor.

$$MPDC = AVG(PS_i) - BLPS \tag{4.1}$$

donde PS son los datos pupilares recopilados, AVG representa el promedio, $BLPS$ es la línea de base y N es el número de datos recopilados después de tomar $BLPS$.

- Cambio de porcentaje promedio en el tamaño de la pupila (*APCPS, del íngles: Average Percentage Change Pupil Size*) (Iqbal y col., 2005; Lallé y col., 2015). El cambio porcentual en el tamaño de la pupila (*PCPS, del íngles: Percentage Change Pupil Size*) se calcula como la diferencia entre la medida de la pupila y el $BLPS$, dividido entre $BLPS$ de la pupila. El $APCPS$ es el promedio de esta medida, el cual definiremos como los puntos de datos en el tiempo del intervalo de medición.

$$PCPS = \frac{PS_i - BLPS}{BLPS} \tag{4.2}$$

Donde PS representa el tamaño de la pupila medido por el rastreador ocular y $BLPS$ la línea base de la pupila medida, dos segundos antes de comenzar la prueba.

Para obtener $APCPS$, simplemente se promedia PCPS con el número total de muestras de tamaños de pupila durante la etapa de estudio:

$$APCPS = \frac{\sum_{i=1}^{n} PCPS}{n} \tag{4.3}$$

donde n es el número de mediciones del tamaño de la pupila después de medir el $BLPS$.

- Para calcular la frecuencia de parpadeo (*BF, del ingles: Blink Frequency*), se eligió un rango en la frecuencia de parpadeo entre 100 y 300 ms. Medir esta frecuencia es sencillo: cuando los datos del rastreador ocular son nulos, significa que los párpados del sujeto estaban cerrados, lo cual nos dice que este movimiento es producto de un parpadeo. Para evitar datos erróneos, verificamos que estos tiempos nulos se encuentran dentro del rango de un parpadeo. El BF se calcula dividiendo el número de parpadeos que ocurrieron durante la respuesta del participante entre el tiempo que tardó realizando la tarea en segundos.

$$BF = \frac{NB}{T} \tag{4.4}$$

  Donde NB es el número de parpadeos y T representa el tiempo que duró la respuesta.

- Entropía de fijaciones oculares (H_{eye}, *del ingles: Eye entropy*). El calculo de la entropía de las fijaciones se realizo con las técnicas propuestas por Mitre-Hernandez y col., 2019. Este enfoque es importante ya que además de la posición de las fijaciones toma en cuenta la duración que tuvieron. Con esto se logra evaluar variaciones en el tiempo que el usuario utilizo para las fijaciones en cada sección.

  La entropía se puede describir cualitativamente como una medida de dispersión de energía. El concepto en sí, está relacionado con el desorden: la entropía es una medida del desorden y la naturaleza tiende hacia la entropía máxima para cualquier sistema aislado. La entropía de la información se define como la ecuación 4.5.

$$H = -\sum p_i \log p_i \tag{4.5}$$

Donde p_i es la probabilidad de ocurrencia del i-ésimo símbolo de un alfabeto. Para obtener una métrica basada en la entropía, se superpone a la imagen una cuadrícula que cubre la región de interés. Esta representación se ajusta a una variedad de elementos y estructuras que pueden contener la imagen (por ejemplo, la pregunta en forma de texto). Se implementaron dos estrategias:

Espacio de escala Mitre-Hernandez y col., 2019. El concepto de octava se implementó para obtener una métrica robusta que es invariable a los cambios de tamaño de los objetos. Dado el tamaño de cuadrícula inicial de $n \cdot m$ celdas, cada octava se obtiene multiplicando el tamaño de cuadrícula original por $K = 2^{s-1}$. Es decir, el tamaño de cuadrícula G_s (ecuación 4.6) para la i-ésima octava es (ecuación 4.7).

$$G_s = m_s \cdot n_s \tag{4.6}$$

$$K \cdot G_s = 2^{s-1}m \cdot 2^{s-1}n \tag{4.7}$$

Por ejemplo, dada la cuadrícula de tamaño original $G_1 = 2 \cdot 3$ que se muestra en la Fig. 4.2(a), la cuadrícula de la octava 2 es $G_2 = 4 \cdot 6$ (Fig. 4.2(b)) y la cuadrícula para la octava 3 es $G_3 = 8 \cdot 12$ (Fig. 4.2(c)).

El tamaño de celda para una imagen de $x \cdot y$ píxeles y una cuadrícula de m_s filas y n_s columnas es:

$$S_x = \frac{x}{n_s} \tag{4.8}$$

$$S_y = \frac{y}{m_s} \tag{4.9}$$

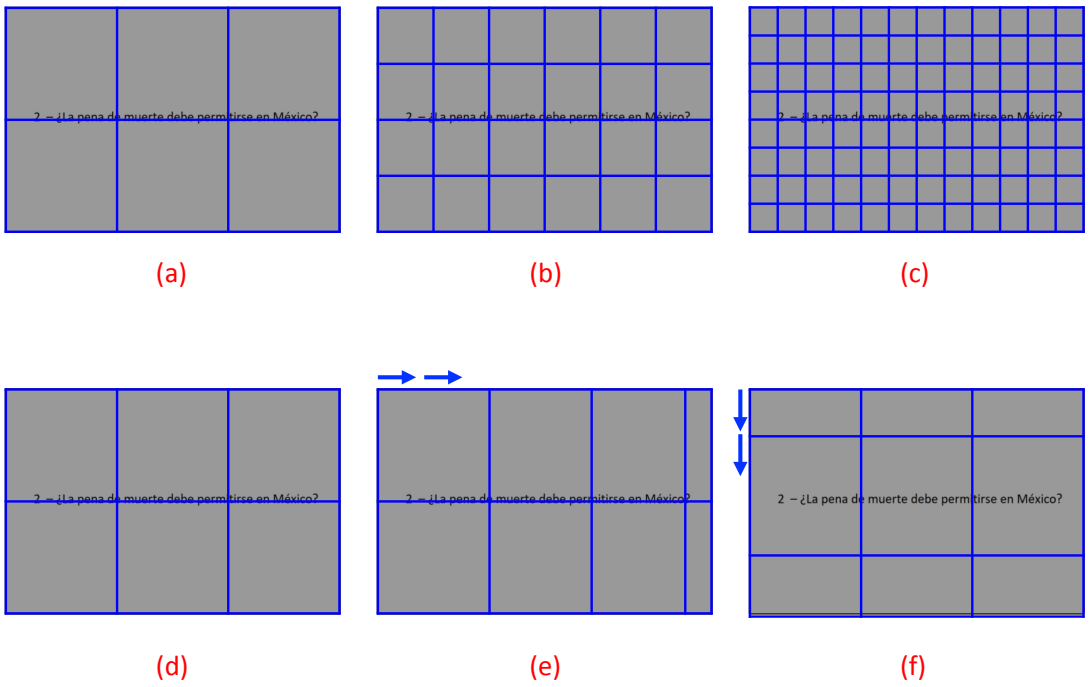

FIGURA 4.2: Ilustración de la cuadrícula superpuesta a la pregunta de la imagen para calcular la métrica basada en entropía. La estrategia *Espacio de escala* involucra: (a) cuadrícula de tamaño original $G_1 = 2 \cdot 3$, (b) segunda octava, $G_2 = 4 \cdot 6$, (c) tercera octava $G_3 = 8 \cdot 12$. La estrategia *ventana deslizante* involucra: (d) cuadrícula en la posición inicial, (e) cuadrícula desplazada horizontalmente, (f) cuadrícula desplazada verticalmente.

Ventana deslizante Mitre-Hernandez y col., 2019. Este concepto se utiliza porque se desconoce la posición de los objetos/texto. La cuadrícula se mueve sucesivamente en ambas direcciones mediante incrementos de:

$$\Delta_x = \frac{S_x}{p} \tag{4.10}$$

$$\Delta_y = \frac{S_y}{p} \tag{4.11}$$

donde p es el número predefinido de pasos. Como se muestra en la Fig. 4.2(d,e,f)

cada píxel de la región de interés siempre está cubierto por una sola celda. De ahora en adelante, G_s^{hv} denota una cuadrícula de tamaño $G_s = m_s \cdot n_s$ moviendo h y v pasos en direcciones horizontal y vertical, respectivamente.

Dado el tiempo dedicado a observar la imagen mientras el sujeto responde al entrevistador, el rastreador ocular recoge sus fijaciones oculares. Para una cuadrícula G_s^{hv}, la entropía se calcula como:

$$H(G_s^{hv}) = -\sum_{i=1}^{m_s}\sum_{j=1}^{n_s} p(A_{ij}) \log p(A_{ij}) \tag{4.12}$$

dónde A_{ij} es la celda de la cuadrícula en la fila i, la columna j y $p(A_{ij})$ es:

$$p(A_{ij}) = \frac{t_{ij}}{\sum_i \sum_j t_{ij}} \tag{4.13}$$

Aquí t_{ij} es el tiempo total en que el sujeto observó la celda A_{ij}. La entropía del ojo para la escala s es:

$$H_{eye}(s) = \frac{1}{p^2} \sum_{h=0}^{p-1}\sum_{v=0}^{p-1} \frac{H(G_s^{hv})}{H_s'} \tag{4.14}$$

donde $H_s^{'} = -\log \frac{1}{m_s m_s}$ es la entropía máxima para una cuadrícula de tamaño $n_s \cdot m_s$.

- Velocidad pico sacádica (SPV). Es el valor máximo del conjunto de muestras de velocidad de los saltos sacádicos (px/s) tomadas de la respuesta del participante.

$$SPV = max\{SV_1, SV_2, ..., SV_k\} \tag{4.15}$$

- *Ms* (milisegundos). Es el tiempo de respuesta del participante.

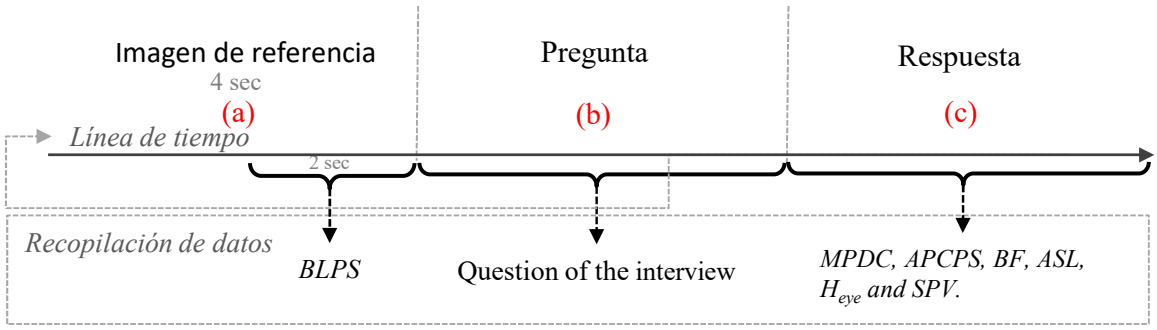

FIGURA 4.3: Pasos en la ejecución del experimento.

4.4. Procedimiento

Los participantes calificaron los cuestionarios sobre los temas de interés en México. Más tarde realizaron la misma actividad, pero sobre la honestidad de sus respuestas a cada tema. El investigador seleccionó dos oraciones para calificar qué tan de acuerdo y honestos fueron a responder (ambas con una escala de 1 a 7, siendo 1 totalmente en desacuerdo/deshonesto, mientras 7 es totalmente de acuerdo/honesto), esto con la finalidad de catalogar sus respuestas como mentira (L, *del íngles: Lie*) y verdad (T, *del íngles: Truth*). Estas oraciones, se consideran las más honestas y con las que el participante está más de acuerdo y en desacuerdo.

Las declaraciones seleccionadas se presentaron a cada participante con el siguiente orden: antes de la recopilación de datos oculares, en orden aleatorio, el investigador tomó la declaración etiquetada como T o L y le proporcionó al participante la instrucción de que él/ella debería dar argumentos de verdad o engaño al entrevistador; luego, el entrevistador pregunta al participante y, este mismo, proporciona la respuesta mientras el software de seguimiento ocular recopila los datos del ojo (tamaño de la pupila, fijaciones, saltos sacádicoss y parpadeos), tal como se aprecia en la figura 4.3.

4.5. Análisis de datos

El propósito de este análisis fue seleccionar las variables con diferencias notables entre respuestas, las cuales se ordenaron en dos grupos categóricos: verdad (T) como fácil y mentira (L) como difícil. Las variables dependientes $MPDC$, $APCPS$, BF, ASL, H_{eye} y SPV, se midieron a un nivel continuo. Los datos fueron probados para normalidad; en este caso, la prueba de Shapiro-Wilk fue la prueba más apropiada para muestras pequeñas. Los datos obtenidos de la prueba indicaron que las siguientes variables se consideraron distribuidas normalmente: H_{eye} L, $MPDC$ T, $MPDC$ L, $APCPS$ T, $APCPS$ L, ASL L y SPV L. La prueba T fue la más adecuada para estas variables, siendo la prueba de rango con signo de Wilcoxon la más conveniente para el resto de las variables dependientes.

La prueba de rango con signo de Wilcoxon indicó que la mediana de H_{eye} L ($Mdn = 2.34$) fue estadísticamente más alta en comparación con la mediana de H_{eye} T ($Mdn = 2.19$), $Z = 2.353$, $p < 0.019$. En las respuestas de los sujetos en condición L, la entropía de las fijaciones oculares presentó una puntuación más alta en comparación con las respuestas clasificadas como T. Las respuestas L, necesitaban más esfuerzo mental en comparación con aquellas que fueron clasificadas como T.

Una prueba de Wilcoxon de rango con signo mostró que la mediana de ASL L ($Mdn = 264.29$) fue estadísticamente más alta que la mediana de ASL T ($Mdn = 226.74$), $Z = 2.118$, $p < 0.034$. Las respuestas de los participantes indican una mayor longitud promedio de saltos sacádicos en una L en comparación con una T.

Las variables ASL y H_{eye} confirmaron una mayor amplitud de saltos sacádicos y una entropía de distribución espacial del tiempo de fijación, están asociadas con un alto esfuerzo mental. Pero, la velocidad de los saltos sacádicos no diferenció entre fáciles y difíciles, SPV L ($Mdn = 0.702$) y T ($Mdn = 0.188$), prueba de Wilcoxon, $P = 0.767$, $Z = -0.296$.

Se realizó una prueba T pareada para comparar la media de la variable independiente $APCPS$ y $MPDC$ en las condiciones L y T de los sujetos. No se presentaron diferencias significativas en los puntajes para las condiciones $APCPS$ T ($M = 0.125$, $SD = 0.063$) y $APCPS$ L ($M = 0.106$, $SD = 0.052$); $t(12) = 1.197$, $p = 0.255$. Este fenómeno también se presentó en $MPDC$ T ($M = 3.440$, $SD = 2.063$) y L ($M = 2.965$, $SD = 1.594$; $t(12) = 1.201$, $p = 0.253$).

Una prueba de rango con signo de Wilcoxon indicó que las respuestas clasificadas como T mostraron ($Mdn = 0.143$) en BF, exhibiendo una diferencia no estadística en comparación con L ($Mdn = 0.190$), $Z = 1.643$, $p = 0.100$. Finalmente, los resultados del tiempo de respuesta (Ms) presentaron diferencias significativas en las condiciones L ($Mdn = 61051$) y T ($Mdn = 48080$), prueba de Wilcoxon, $P = 0.013$, $Z = -2.486$. Mentir requiere más tiempo para presentar argumentos engañosos.

H_{eye}, ASL y Ms fueron las características seleccionadas para entrenar a los clasificadores SVM y QDA.

4.6. Entrenamiento y resultados

Después de obtener las características, toda la información se integró en un conjunto de datos $\delta = \{(Xi, yi), i = 1, .., n)\}$, donde Xi corresponde al vector de longitud uniforme que contiene las características $Xi = (H_{eyei}, ASL_i, Ms_i)$, y y_i corresponde a la etiqueta asociada a la respuesta difícil (L) y fácil (T). δ se dividió en subconjuntos de entrenamiento (80 %) y pruebas (20 %) de datos oculares. En los resultados, se obtuvo la mejor precisión en SVM con 80 % y el modelo QDA obtuvo el 40 % de precisión.

Capítulo 5

Clasificación de dificultad en tareas de memoria visual con datos oculares

El recuerdo a corto plazo de una secuencia de dígitos (también conocida como la tarea de intervalo de dígitos) es la tarea experimental más popular en la pupilometría cognitiva. Reportado por primera vez por Beatty y Kahneman, 1966, la tarea también se usó para investigar los procesos relacionados a recuerdos (Beatty y Kahneman, 1966), agrupación (Kahneman, Onuska y Wolman, 1968) y ensayos (Kahneman y Wright, 1971) a largo plazo. Peavler, 1974 mostró que la pupila alcanza una dilatación máxima de aproximadamente 0.5 mm alrededor de la presentación del séptimo dígito. Granholm y col., 1996 replicaron este hallazgo, confirmando que el promedio de dilatación de la pupila se puede usar para estimar tanto la carga momentánea como la capacidad máxima de la memoria de trabajo.

Todas las investigaciones previas de dilataciones de la pupila provocadas por esta tarea presentaron la secuencia de dígitos de forma auditiva. Este experimento realizado por Klingner, Tversky y Hanrahan, 2011 es una réplica del estudio original de Beatty y Kahneman, 1966, con la diferencia en que el estímulo (secuencia de digitos) es presentada visualmente a los participantes.

Los datos crudos obtenidos en el experimento de Klingner, Tversky y Hanrahan,

2011 fueron compartidos. En este trabajo se utilizaron para el análisis y clasificación de los datos de acuerdo a la dificultad de la tarea que los participantes realizaron.

5.1. Participantes

Veinticuatro estudiantes universitarios de Stanford con visión normal participaron en el experimento, se excluyeron aquellos con lentes de contacto o anteojos, y también aquellos que presentan astigmatismo o corrección refractiva de más de 10 dioptrías. Para aumentar el tamaño de la pupila ocular, los autores introdujeron un incentivo monetario para los participantes, $15 para los participantes con los puntajes más bajos y alrededor de $35 para los más altos.

5.2. Materiales

Un dispositivo de seguimiento ocular tipo Tobii 1750, rastreador ocular remoto a 50 Hz con una pantalla de computadora LCD estándar (resolución de pantalla de 1280x1024, 17 pulgadas en diagonal, relación de 5:4), también con luces infrarrojas y cámara infrarroja de alta resolución. El rastreador ocular admite el movimiento de la cabeza, pero para los datos de la pupila ocular, el movimiento de la velocidad debe ser inferior a 10 cm/s dentro de una caja de 30x15x20 cm con una distancia inicial de 60 cm desde la pantalla.

Los autores colocaron el rastreador ocular en la parte superior de la pantalla, esta fue una buena decisión para evitar datos oculares nulos cuando el participante selecciona los dígitos recordados al tocar la pantalla.

5.3. Procedimiento

Antes de iniciar, se explicó la tarea a los participantes, luego se les permitió practicar hasta que estuvieran familiarizados y cómodos con la aplicación de la tarea.

Todos los ensayos fueron iniciados por los participantes, quienes primero fijaron un objetivo pequeño en el centro de la pantalla (pre-estímulo) antes de comenzar el ensayo haciendo clic en un botón del mouse. La mirada de los participantes permaneció así en el centro de la pantalla durante la duración de cada prueba y durante la mayoría de los cortos intervalos de tiempo entre las pruebas. Una serie de pruebas para una sola tarea, generalmente tomó alrededor de 5 min. Se les dijo a los participantes que podían tomar descansos en cualquier momento entre los ensayos para descansar los ojos; 2 lo hicieron.

Se pidió a los participantes que escribieran sus respuestas en un teclado en pantalla de bajo contraste. Esto se hizo para automatizar la recopilación de datos y evitar los reflejos pupilares al brillo variable causado por apartar la vista de la pantalla.

Cada prueba comenzó con un período de estabilización de la pupila en el que se midió la linea base del tamaño de la pupila. Posteriormente, se presentó una secuencia de dígitos a razón de uno por segundo en la pantalla. Después de una breve pausa de retención, los participantes informaron la secuencia utilizando un teclado en pantalla. La longitud de la secuencia de números exhibida para cada ensayo fue variada aleatoriamente independientemente entre tres y ocho dígitos. Las mediciones que se hicieron en los diferentes intervalos de tiempo se pueden observar en la figura 5.1.

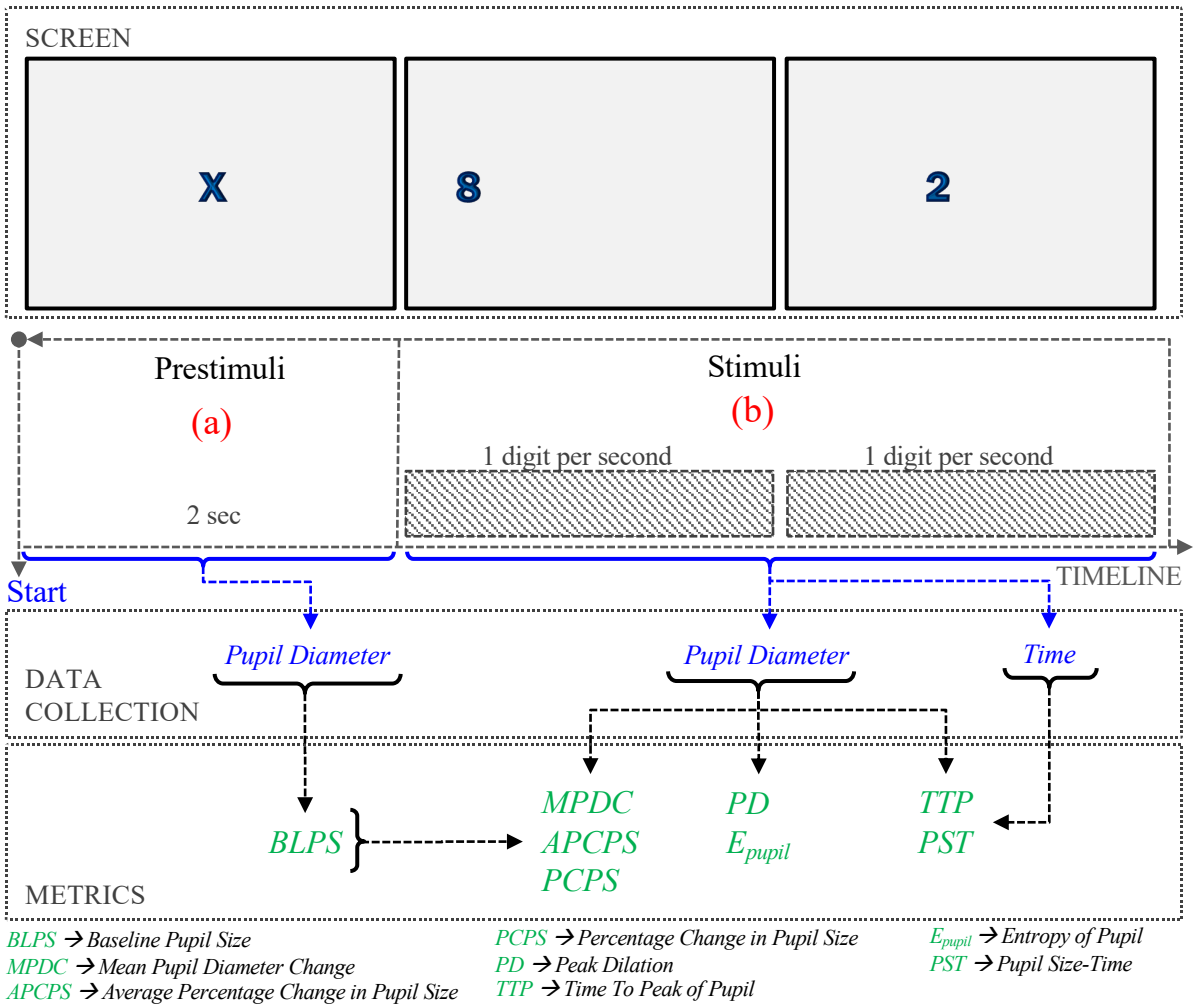

FIGURA 5.1: Procedimiento del experimento.

5.4. Mediciones

- *Línea base del tamaño de la pupila (BLPS)*. La BLPS se usa para establecer un valor cuando la pupila se encuentra estable. Klingner, Tversky y Hanrahan, 2011 utilizaron una línea base en cada prueba basada en el diámetro promedio de la pupila medido en más de 20 muestras en 400 ms al final de un período previo al estímulo (estabilización de la pupila). Este cálculo se realizó en esta investigación.

- *Cambio medio del diámetro de la pupila (MPDC)* (Kun y col., 2013a; Palinko y col., 2010; Palinko y Kun, 2012). Primero, la línea de base *BLPS* se calcula 2 segundos

antes de la prueba. Durante la respuesta del sujeto, se recopilan los datos pupilares, se obtiene el promedio y el $BLPS$ que se obtuvo en el período previo a la prueba se resta a este valor.

$$MPDC = AVG(PS_i) - BLPS \tag{5.1}$$

Donde PS son los datos pupilares recopilados, AVG representa el promedio, $BLPS$ es la línea de base y N es el número de datos recopilados después de tomar $BLPS$.

- *Cambio de porcentaje promedio en el tamaño de la pupila (APCPS)* (Iqbal, Zheng y Bailey, 2004; Lallé y col., 2015; Iqbal y col., 2005). El cambio porcentual en el tamaño de la pupila (PCPS, por sus siglas en inglés) se calcula como la diferencia entre el tamaño medido de la pupila y la línea base de la pupila, dividido entre el tamaño dela línea base de la pupila.

$$PCPS = \frac{PS_i - BLPS}{BLPS} \tag{5.2}$$

Donde PS representa el tamaño de la pupila medido por el rastreador ocular y $BLPS$ la línea base de la pupila medida dos segundos antes de comenzar la prueba.

Para obtener $APCPS$, solamente se promedia PCPS con el número total de muestras de tamaños de la pupila durante la etapa de prueba, como sigue:

$$APCPS = \frac{\sum_{i=1}^{n} PCPS}{n} \tag{5.3}$$

donde n es el número de mediciones del tamaño de la pupila después de medir $BLPS$.

- *Dilatación máxima*. La dilatación máxima (PD) es la diferencia entre BLPS y el tamaño máximo de la pupila durante la etapa de prueba.

$$PD = max\{PS_1, PS_2, ..., PS_n\} - BLPS \tag{5.4}$$

Donde PS representa el tamaño de la pupila y BLPS es la línea de base.

- *Entropía de la pupila (E_{pupil})*. La entropía se puede describir cualitativamente como una medida de dispersión de energía. El concepto en sí, está relacionado con el desorden: la entropía es una medida del desorden, y la naturaleza tiende hacia la entropía máxima para cualquier sistema aislado. La entropía de de la pupila se define como la siguiente ecuación:

$$E = -\sum_i PS_i \log PS_i \tag{5.5}$$

Donde PS_i es la probabilidad de aparición en la medición de la pupila i de un conjunto de datos tomados a partir de una prueba.

- *Tiempo para alcanzar el tamaño máximo de pupila (TTP)*. En los resultados de (Siegle, Ichikawa y Steinhauer, 2008) se observa una relación entre la carga cognitiva y el tiempo en que la pupila llega a su tamaño máximo. Esta métrica ocupa el tiempo después de tomar la linea base de la pupila hasta llegar al tamaño máximo, como se muestra a continuación:

$$PPS = max\{Ps_1, Ps_2, ..., Ps_k\} \tag{5.6}$$

$$TTP = Time\{PPS\} \tag{5.7}$$

Donde PPS es el tamaño máximo de la pupila y k es el número total de tamaños de pupila hasta el valor más alto.

- *Pupil Size-Time (PST)*. En Siegle, Ichikawa y Steinhauer, 2008 se identificó que su experimento se conformaba en tres secciones importantes: codificación de memoria, almacenamiento de memoria y recuperación de memoria. En sus resultados, la dilatación de la pupila llegó al punto más alto entre la finalización de la etapa de codificación de memoria y el inicio del almacenamiento de memoria. Asimismo, fueron proporcionales en los tres niveles de dificultad de tareas de memoria que utilizó durante su experimento (entre mayor dificultad, mayor era el tamaño máximo de la pupila). Con esta métrica se busca tener una relación entre el tiempo y la dilatación de la pupila durante la etapa de codificación de memoria, ya que esta puede ser un buen indicador de la carga mental al realizar una tarea. Para encontrar esta relación, se utilizó el método de mínimos cuadrados y, así obtener la pendiente que relaciona al tiempo con el tamaño máximo de la pupila, como sigue:

$$m = \frac{\sum T * PS - \frac{(\sum T)(\sum PS)}{n}}{\sum T^2 - \frac{(\sum T)^2}{n}} \tag{5.8}$$

  Donde T representa el tiempo en que la muestra del tamaño de la pupila fue tomada,PS es el tamaño de la pupila y n es el número de muestras hasta el tamaño máximo.

$$PST = tan^{-1}(m) \tag{5.9}$$

  En 5.9 se calcula el ángulo de inclinación, con la tangente inversa de la pendiente, este ángulo es llamado PST.

El calculo de estas mediciones se realizo en Python utilizando las librerías pandas y numpy. En el apéndice A se encuentra el código utilizado, este lee todos los archivos con la información de las pruebas y los tamaños de las pupilas obtenidos en cada una de

las pruebas con el rastreador ocular, calcula cada una de las mediciones y los resultados son guardados en un archivo csv que se actualiza con cada prueba que se va leyendo de los participantes.

5.5. Análisis de datos

Los datos crudos obtenidos del rastreador ocular fueron preprocesados antes de hacer el cómputo de las características ($MPDC$, $APCPS$, PD, E_{pupil}, TTP y PST). Primero se identificaron las mediciones nulas dentro de los datos crudos, en este caso fueron guardados con -1 por el rastreador ocular. Tomando en cuenta esto, se identificaron los datos nulos seguidos que en total tienen una duración entre 100 y 400ms, estos fueron catalogados como parpadeos. La información de los parpadeos fue borrada del dataset para hacer el calculo de las características pupilares. El siguiente paso, fue promediar los tamaños de la pupila donde ambos ojos cuenten con esta información, cuando solo uno de los ojos cuenta con la información de su tamaño, esta es tomada sin necesidad de promediar. Cuando ambos ojos no cuentan con información todos los datos de esas muestras fueron eliminados.

Además, se verificó que la cantidad de datos eliminados no sobrepasara el 10 % del total de datos registrados en cada prueba. En ninguna de las pruebas este limite se excedió, por lo tanto no fue necesario eliminar la prueba de ningún sujeto en este punto.

Para el análisis y clasificación de los datos, solo se consideraron tres niveles de dificultad que dependen del número de dígitos presentados al participante. El nivel bajo consiste en memorizar números de tres dígitos, nivel medio para números de cinco dígitos y nivel alto para números de ocho dígitos.

Los niveles de dificultad que se les mostraron a los sujetos, se presentaron en orden

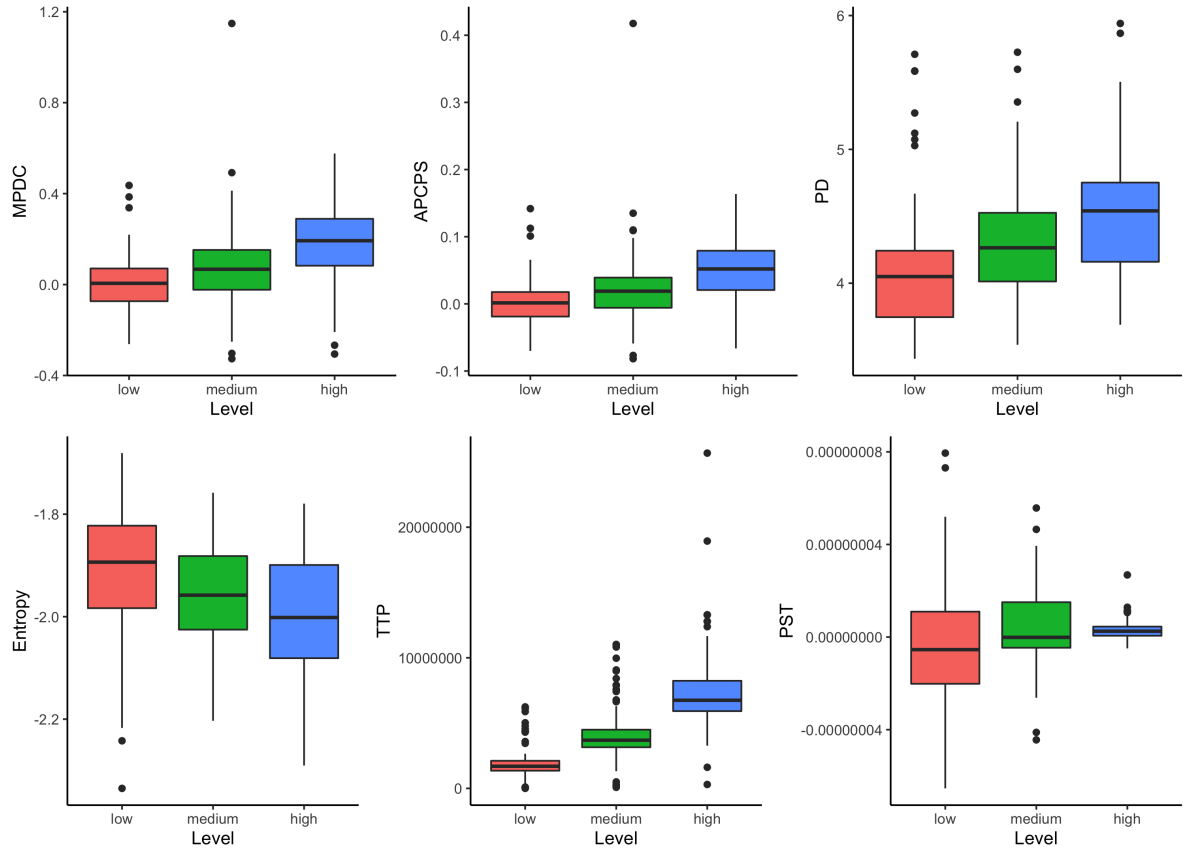

FIGURA 5.2: Medidas con respecto al nivel de dificultad.

aleatorio. Por esta razón, se obtuvo un número diferente de mediciones para cada sujeto en cada nivel de dificultad. Todos ellos tienen un mínimo de tres mediciones y un máximo de siete, diez y ocho mediciones respectivamente para los niveles bajo, medio y alto.

5.5.1. Análisis estadístico

Es interesante investigar si existen diferencias significativas entre las variables de estudio ($MPDC$, $APCPS$, PD, E_{pupil}, TTP y PST) con respecto al nivel de dificultad. En la figura 5.2 se aprecian los diagramas de cajas para cada variable con respecto al nivel de la tarea.

Se propone el uso de un modelo como el siguiente:

$$y_{ij} = \mu + l_k + s_j + (sl)_{jk} + \epsilon_{ij}, \tag{5.10}$$

Donde

y_{ij} = Variable de respuesta (MPDC, APCPS, PD, Entropía, TTP o PST) para la i-ésima observación del j-ésimo sujeto

μ = intersección,

l_k = k-ésimo efecto de tratamiento (donde la i-ésima observación ha utilizado el nivel de dificultad k)

s_j = efecto del sujeto j-ésimo,

$(sl)_{jk}$ = k-ésimo efecto del nivel en el j-ésimo sujeto, o efecto del sujeto jk,

ϵ_{ij} = término de error (residual) para la i-ésima observación del j-ésimo sujeto.

Llamamos a *nivel l* un efecto fijo, y ϵ es nuestro *término de error* que representa desviaciones de nuestras predicciones debido a factores *aleatorios* que no podemos controlar experimentalmente. Sin embargo, se tomaron varias medidas por sujeto en cada nivel de dificultad y eso contradice el supuesto de independencia de un modelo lineal. Por otro lado, cada individuo tiene una capacidad de carga cognitiva diferente y, por tanto, este será un factor característico que afectará todas las respuestas del mismo sujeto, contribuyendo así, que estas respuestas sean interdependientes en lugar de independientes, ver figura 5.3. La forma en que abordamos esta situación es agregando un efecto aleatorio al sujeto y a la interacción a nivel de sujeto. Esto nos permite resolver esta falta de independencia asumiendo una intersección y pendiente diferente para

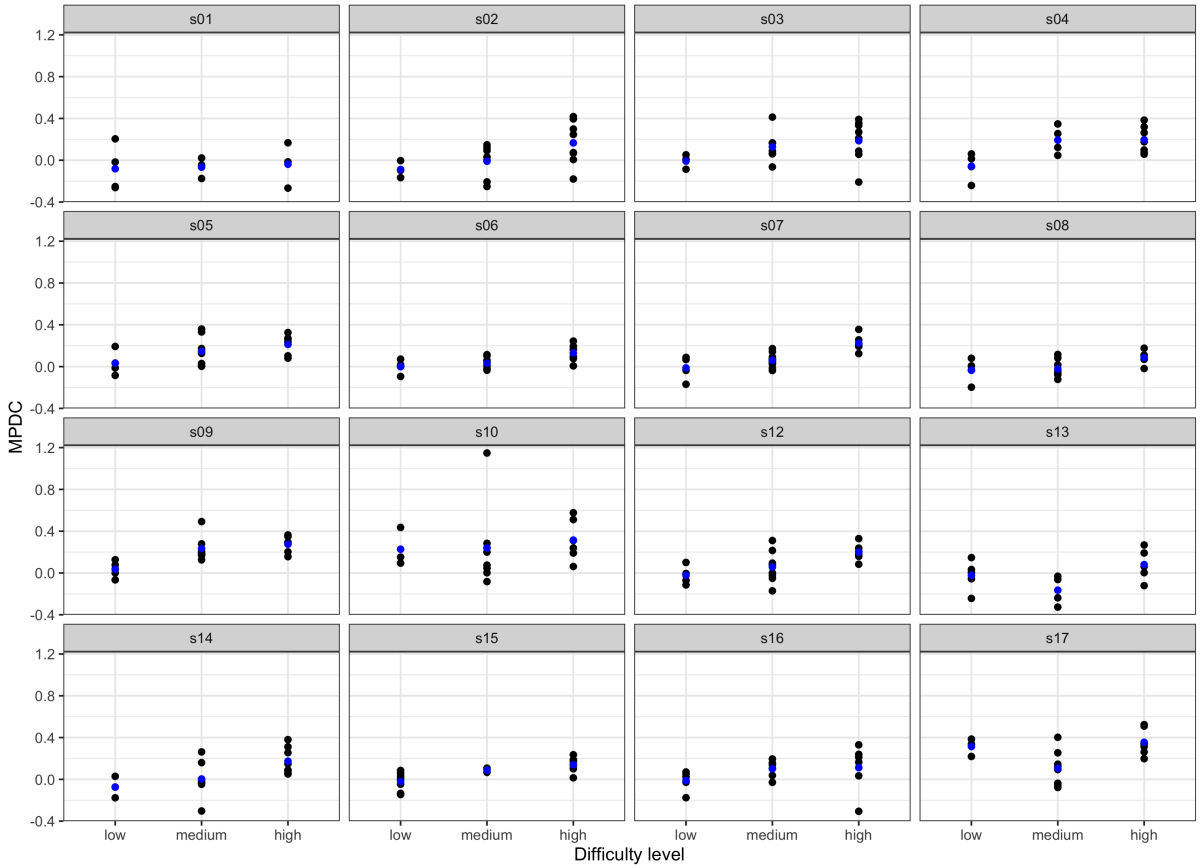

FIGURA 5.3: Resultados de la medición de MPDC versus nivel de dificultad por individuo (Valor medio en azul).

cada sujeto. Y, por último, suponemos que los efectos residuales y de nivel del sujeto, son relaciones de distribuciones separadas, todas con cero significa:

$$\epsilon_{ij} \sim N(0, \sigma^2),$$
$$s_j \sim N(0, \sigma_s^2),$$
$$(sl)_{jk} \sim N(0, \sigma_{sl}^2).$$

Por lo tanto, s_j y $(sl)_{jk}$ ahora son efectos aleatorios, mientras μ y l_k son efectos fijos.

Para las 6 variables mencionadas anteriormente, se ajusta un modelo mixto lineal. En particular, el modelo se estima usando **máxima verosimilitud restringida** (REML) en un enfoque Patterson y Thompson, 1971; Harville, 1977 usando estadística R system

(R Core Team, 2020) R Core Team, 2020, especialmente usando el paquete *lme4* (versión 1.1-23) Bates y col., 2015. Como efectos fijos, ingresamos niveles de dificultad en el modelo. Los valores P se obtuvieron mediante pruebas de razón de probabilidad del modelo completo con el efecto en cuestión contra el modelo sin el efecto en cuestión, ver tabla 5.1. Como resultado, se obtiene que todos fueron significativos a un nivel de 0.001, excepto PST que resultó con una significación mayor a 0.10. Después de ajustar un modelo adecuado, las comparaciones post hoc se realizan utilizando la prueba Tukey HSD, utilizando el paquete R *multcomp* (versión 1.4-13) Hothorn, Bretz y Westfall, 2008. Las diferencias fueron significativas a un nivel de $p < 0.05$. Sin embargo, para $MPDC$ y $APCPS$ se encontró una observación influyente, para PD y TTP hubo dos y cuatro para PST. También se observó heterosedasticidad de los residuos del modelo para PD y TTP. Cada modelo se ajustó una vez más sin observaciones influyentes y utilizando una transformación de Box-Cox para los modelos PD y TTP. Las pruebas Tukey HSD se realizaron nuevamente y los resultados son consistentes con los obtenidos con los modelos originales.

TABLA 5.1: Razón de verosimilitud (*likelihood ratio*)

	$\chi^2(2)$	$p - value$
MPDC	32.5	$p < .001$
APCPS	29.4	$p < .001$
PD	27.2	$p < .001$
E_{pupil}	30.1	$p < .001$
TTP	44.5	$p < .001$
PST	2.65	0.27

Los resultados del análisis del modelo $MPDC$ y E_{pupil} se muestran en detalle a continuación.

Con el fin de comparar estadísticamente el efecto del nivel de dificultad (bajo, medio y alto) en la medición de MPDC, se realizó un modelo de efectos mixtos, cuyo resultado fue significativo $\chi^2(2) = 32.5$, $p < .001$. Las comparaciones post hoc de la prueba

Tukey HSD dan como resultado que la media de MPDC para el nivel de dificultad bajo (M = 0.00397, SD = 0.139, N = 65) fue significativamente diferente del nivel de dificultad medio (M = 0.0777, SD = 0.178, N = 107). Con respecto a este último, se mostró representativamente diferente con respecto al nivel de dificultad alto (M = 0.185, SD = 0.157, N = 100). Sin embargo, existe un punto influyente, por lo que el análisis se realizó nuevamente sin este punto. Y aún así, persiste un efecto significativo del nivel de dificultad en la medición de MPDC ($\chi^(2) = 32.5$, $p < .001$). La comparación post hoc usando la prueba Tukey HSD resulta en $p = 0.058$ significancia para la diferencia entre niveles de dificultad bajo y medio, $p < .001$ para la diferencia entre niveles de dificultad bajo y alto y, también para la diferencia entre dificultad media y alta.

Estos resultados sugieren que el nivel de dificultad tiene un efecto en la medición de MPDC. Precisamente, cuando el nivel de dificultad aumenta, la medición de MPDC también. si bien, vale la pena señalar que el nivel de dificultad debe ser alto para ver un efecto, ya que de dificultad media no parece aumentar significativamente la medición de MPDC.

Con respecto a la medición de la entropía, el modelo de efectos mixtos fue significativo a un nivel de $p < 0.05$. Hubo un efecto principal significativo del nivel de dificultad en la medición de entropía, $\chi^2(2) = 30.1$, $p < .001$. Las comparaciones post hoc empleando la prueba Tukey HSD, indicaron que la medición de entropía media para el nivel de dificultad bajo (M = -1.92, SD = 0.131, N = 65) fue significativamente distinto del nivel de dificultad medio (M = -1.96, SD = 0.115 , N = 107). Y el nivel de dificultad medio fue notoriamente distinto del nivel de dificultad alto (M = -2.00, SD = 0.123, N = 100). La comparación post hoc usando la prueba Tukey HSD resulta en $p = .004$ significancia para la diferencia entre niveles de dificultad bajo y medio, $p < .001$ entre bajo y alto y, también, para medio y alto.

En resumen, estos resultados sugieren que el nivel de dificultad tiene un efecto en la medición de entropía. De tal manera que, cuando el nivel de dificultad aumenta, la

medida de entropía disminuye.

TABLA 5.2: Prueba HSD de Tukey

Null hypothesis	Initial model			Final model		
	Estimate		Std. Error	Estimate		Std. Error
MPDC						
Medium-Low=0	0.0616	*	-0.026	0.0526	†	(0.0278)
High-Low=0	0.1728	***	-0.0237	0.1711	***	(0.0222)
High-Medium=0	0.1112	***	-0.0217	0.1185	***	(0.0217)
APCPS						
Medium-Low=0	0.01672	*	-0.00736	0.01322	†	(0.00795)
High-Low=0	0.04559	***	-0.00682	0.04484	***	(0.00608)
High-Medium=0	0.02887	***	-0.00627	0.03162	***	(0.00596)
PD						
Medium-Low=0	0.1253	**	-0.0399	0.002028	***	(0.000540)
High-Low=0	0.3772	***	-0.0478	0.005204	***	(0.000636)
High-Medium=0	0.2519	***	-0.0397	0.003176	***	(0.000408)
E_{pupil}						
Medium-Low=0				-0.02543	**	-0.00883
High-Low=0				-0.06768	***	-0.00895
High-Medium=0				-0.04225	***	-0.00616
TTP						
Medium-Low=0	0.1253	**	-0.0399	0.002028	***	(0.000540)
High-Low=0	0.3772	***	-0.0478	0.005204	***	(0.000636)
High-Medium=0	0.2519	***	-0.0397	0.003176	***	(0.000408)
PST						
Medium-Low=0	0.000000006176		-0.000000003755	0.00000000868	***	(0.00000000369)
High-Low=0	0.000000005297		-0.000000004213	0.0000000072	***	(0.00000000384)
High-Medium=0	-0.000000000879		-0.000000002561	-0.00000000148	***	(0.00000000268)

Note. Estimates and standard errors (in parentheses)
† $p < .10$, * $p < .05$, ** $p < .01$, *** $p < .001$

Para la mayoría de las variables, los supuestos de distribución del modelo mixto no se cumplieron. La tabla 5.2 muestra los resultados de las pruebas post hoc Tukey-HSD de los modelos con los datos originales y con datos donde se excluyeron observaciones influyentes y / o utilizando la transformación Box-Cox cuando era necesario. Como se puede ver, en general, las conclusiones se mantienen con las transformaciones a excepción de PST. Antes de la transformación no hay diferencia significativa, pero aplicándola sí. Con la variable entropía no hubo necesidad de transformar los datos; se cumplieron los supuestos.

5.6. Clasificación de datos

La clasificación de la dificultad en la tarea se realizó entrenando 5 clasificadores supervisados diferentes. Su elección se hizo en base a una revisión literaria, donde se eligieron los más utilizados y con mejores resultados. Estos fueron entrenados con características extraídas de datos crudos de rastreadores oculares. Los clasificadores elegidos fueron máquina de soportes vectoriales (con cinco kernel diferentes) (Biedert y col., 2012; Eivazi y Bednarik, 2011; Li y col., 2020; Deng y col., 2019), análisis discriminante lineal (Li y col., 2020), árbol de decisiones (Kunze y col., 2013; Li y col., 2020), y bosque aleatorio (Deng y col., 2019).

Se utilizaron 6 características (MPDC, APCPS, PD, E_{pupil}, TTP, PST) para el entrenamiento de los clasificadores, que fueron etiquetadas con 3 niveles de dificultad de la tarea (bajo, medio y alto). Para esto, se tomaron en cuenta todas las combinaciones de conjuntos de entrenamiento posibles que se pueden tener. Con un conjunto de n características de entrenamiento se pueden realizar $2^n - 1$ distintos, donde n es el número de características. Es decir, podemos realizar entrenamientos empleando una sola característica, combinaciones de dos características, hasta llegar a la combinación de las n características. Para este caso, con 6 características de entrenamiento tenemos un total de 63 combinaciones posibles.

Cada uno de los clasificadores fue entrenado con el 80 % de los datos y evaluado con él 20 %. Se entrenaron con las 63 combinaciones de conjuntos de entrenamientos posibles. Los resultados de estos entrenamientos se plasmaron en una tabla por cada clasificador.

Para el entrenamiento de los clasificadores se utilizo Scikit-learn, que es una biblioteca de aprendizaje automático para el lenguaje de programación Python. Se decidió utilizarla ya que incluye los algoritmos de clasificación que se decidieron aplicar en este trabajo. En el apéndice B se encuentra el código utilizado para entrenar y evaluar los

Modelo	Caracteristicas de entrenamiento	Precisión			
		Bajo	Medio	Alto	Promedio
SVM (poly)	MPDC/APCPS/PD	1	0.49	1	0.83
SVM (RBF)	MPDC/APCPS/Entropy/PST	1	0.62	0.75	0.79
árbol de decisiones	MPDC/PD/Entropy/TTP	0.67	0.83	0.76	0.75
bosque aleatorio	MPDC/APCPS/PD/TTP	0.77	0.76	0.71	0.75
SVM (Linear)	MPDC/APCPS/PST	1	0.5	0.72	0.74
LDA	MPDC/PD/Entropy/TTP/PST	0.75	0.61	0.84	0.73
SVM (LinearSVC)	MPDC/APCPS/PST	1	0.53	0.62	0.72
SVM (sigmoid)	APCPS/TTP	0.6	0.68	0.69	0.66

TABLA 5.3: Pasos en la ejecución del experimento.

clasificadores SVM, en el se puede observar que se evalúan 5 kernel diferentes (RBF, Poly, Linear, LinearSVC y Sigmoid) de los cuales sus resultados son mostrados en pantalla. En los apéndices C, C y F se adjuntan los códigos para LDA, árbol de decisión y bosques aleatorios respectivamente.

5.7. Resultados

En la tabla 5.3 se muestran los clasificadores que mejor desempeño presentaron al ser evaluados. La tabla contiene el modelo entrenado, sus características y la precisión con que se clasificó cada uno de los niveles de dificultad y su precisión promedio.

Como podemos observar en la tabla 5.3, el clasificador con mejor desempeño fue SVM con un kernel polinomial, seguido del modelo SVM con kernel RBF. Esto concuerda con lo expuesto en trabajos relacionados, donde encontramos que los clasificadores que alcanzaban mejor desempeño al trabajar con datos extraídos de rastreadores oculares eran SVM (Biedert y col., 2012; Li y col., 2020; Deng y col., 2019).

La mejor precisión promedio se consiguió al entrenar un modelo SVM con un kernel polinomial y un vector de entrenamiento de 3 (MPDC, APCPS, PD) de las 6 características disponibles. Con esta configuración se consiguió una precisión del 100 % de los niveles bajo y alto al ser evaluado. Sin embargo, el nivel de dificultad medio presentó

una precisión baja del 49 %, lo que nos indica que a este modelo se suele confundir haciendo pasar por falsos positivos niveles bajos y altos. La baja precisión en el nivel medio puede ser el reflejo de los resultados en los análisis de las características MPDC y APCPS, donde el p-valor fue menor a 0.1 para ambas características al evaluar el nivel medio contra el bajo, mientras que el p-valor al evaluar el nivel alto contra bajo y alto contra medio fue menor a 0.001.

El modelo SVM con kernel RBF que obtuvo la segunda mejor precisión fue entrenado con cuatro características (MPDC, APCPS, E_{pupil}, PST), con las que se consiguió una precisión promedio del 79 %. Su precisión para el nivel bajo de dificultad es del 100 %, seguido de un 75 % para el nivel alto y 62 % para el nivel medio. A diferencia del modelo entrenado con un kernel polinomial, este disminuye su precisión para detectar el nivel alto de dificultad y aumentó en 13 % la detección del nivel medio. También se puede observar en este modelo contra el anterior que el p-valor de E_{pupil} y PST en el análisis de datos es menor a 0.001, lo que pude ayudar a diferenciar de una manera más equilibrada entre los niveles medio y alto.

Tanto el árbol de decisiones como bosque aleatorio consiguieron la misma precisión promedio, con un 75 %. Ambos fueron entrenados con cuatro características de entrenamiento, la diferencia es que el árbol de decisiones utiliza entre su conjunto la característica E_{pupil}, mientras que el bosque aleatorio descarta esta característica y utiliza APCPS. La precisión con la que clasifican cada nivel se puede observar más equilibrada en el bosque aleatorio, donde se consigue una precisión del 77 %, 76 % y 71 % para los niveles de dificultad bajo, medio y alto, respectivamente. El árbol de decisiones fue el modelo que mejor precisión obtuvo al clasificar los niveles medio de dificultad, alcanzando una precisión del 80 %, para el nivel bajo 67 % y 76 % para alto.

SVM con un kernel lineal alcanzó una precisión del 74 %. Esto se logró con 3 características (MPDC, APCPS, PST) en su entrenamiento, en donde el nivel bajo obtuvo una precisión del 100 %, esto quiere decir que no confundió a los otros niveles con el bajo. El

nivel medio alcanzó un 50 % y el alto 72 % de precisión, indicándonos que otros niveles si se llegan a confundir con estos dos. Al igual que los demás modelos SVM, el nivel medio de dificultad es el que más falsos positivos llega a tener.

El modelo LDA fue el que alcanzó una precisión más alta aplicando más características. En este caso 73 % utilizando 5 características (MPDC, PD, E_{pupil}, TTP, PST) de entrenamiento. Su mayor precisión al clasificar fue en el nivel alto con un 84 %, seguido de un 75 % en el nivel bajo y 61 % en el nivel medio. Aún cuando otros modelos obtuvieron una precisión promedio mejor, al ver su matriz de confusión se puede observar que sus falsos positivos en general son menos que en el clasificador SVM con kernel polinomial a causa de la baja precisión en el nivel medio. Además, se puede observar que el nivel con que más se confundió es el nivel bajo, lo que coincide con que MPDC obtuvo un p-valor menor a 0.1 en el análisis estadístico de los niveles bajos vs medio, a diferencia de las demás variables que su p-valor se encuentra por debajo de 0.001.

SVM con el kernel sigmoide fue el modelo que consiguió la precisión más baja, obteniendo su clasificación más alta al aplicar los 63 conjuntos de entrenamiento, siendo esta de 66 % al ser entrenado con dos características (APCPS, TTP). La precisión con que clasificó cada uno de los niveles de dificultad se mantuvo menor al 70 %, en el caso de nivel bajo obtuvo un 60 %, con el medio un 68 % y el alto un 69 %.

Capítulo 6

Conclusiones y trabajo futuro

En este capítulo se concluye lo investigado en este trabajo, así como los trabajos a futuro que pueden mejorar o dar otra perspectiva a la propuesta.

6.1. Conclusión

En este trabajo se propuso emplear datos oculares obtenidos mientras los usuarios realizaban una tarea de memoria para medir la carga cognitiva que se estaba produciendo y utilizarlos para la clasificación de la dificultad que produce la tarea. Los datos oculares se extrajeron utilizando un rastreador ocular remoto, lo que permite que el análisis no sea invasivo para el usuario.

Se realizaron dos experimentos, en el primero se clasificaron tareas de memoria auditiva y en el segundo tareas de memoria visual. Para el primer caso, se usaron características obtenidas a partir de la pupila, saltos sacádicos y parpadeos. En el segundo, solo se manejaron variables pupilares.

En el experimento con tareas de memoria auditiva se entrenaron dos clasificadores: SVM y QDA. La mejor precisión se consiguió con SVM llegando a un 80 %, por otro lado, la clasificación de QDA fue baja 40 % para identificar entre dos niveles de dificultad, fácil y difícil. Esto concuerda con las investigaciones expuestas en los trabajos

relacionados, donde SVM era uno de los clasificadores con mejor desempeño y QDA no era utilizado para hacer clasificaciones con datos obtenidos a través de rastreadores oculares.

Para el segundo experimento, los clasificadores fueron entrenados solamente con características pupilares, se hizo la diferenciación de tres niveles de dificultad: fácil, medio y difícil. En total se entrenaron cuatro clasificadores SVM, árbol de decisiones, bosque aleatorio y LDA. El primer y segundo mejor resultado se obtuvo con el algoritmo de clasificación SVM utilizando un kernel polinomial y RFB respectivamente, con esta configuración se alcanzó una precisión del 83 % y 79 %. En tercer lugar, con el árbol de decisión y bosque aleatorio, ambos consiguieron un 75 % de precisión.

El enfoque que se implementó para seleccionar las características para los entrenamientos, fue el uso de medidas que se han utilizado con anterioridad y cuentan con un respaldo en la literatura para la medición de carga cognitiva. Además, en el segundo estudio se añadieron dos medidas que no habían sido utilizadas PST y TTP. Ambas fueron lo suficientemente relevantes para estar presentes en los entrenamientos de los clasificadores que obtuvieron mejores resultados.

Como conclusión, tenemos que los datos obtenidos con rastreadores oculares pueden ser utilizados para la medición de carga cognitiva y ser clasificados según el nivel de dificultad que presenten los usuarios cuando están realizando una tarea.

6.2. Trabajo futuro

En este trabajo se mostraron los primeros clasificadores de dificultad durante tareas de memoria auditivas y visuales por medio de la medición de carga cognitiva con medidas oculares. Anteriormente se habían publicado estudios donde se utilizaban medidas oculares para la medición de carga cognitiva, pero estas no habían sido aplicadas para clasificar la dificultad de la tarea que los participantes estuvieran realizando.

Este primer enfoque a la clasificación de la dificultad es importante para futuras aplicaciones donde el control de la dificultad de una tarea sea esencial. Por ejemplo, si en un futuro logramos controlar la dificultad en contenidos de aprendizaje, el rendimiento de los participantes sería mayor (Yang y col., 2020). Cuando estos contenidos tienen una alta complejidad (sobrecarga cognitiva) los alumnos entran en un estado de frustración que bloquea el aprendizaje. Por otra parte, cuando el contenido es sencillo (genera baja carga cognitiva) lleva a los alumnos a un estado de aburrimiento que también bloquea el aprendizaje (DeFalco y col., 2017).

Igualmente, se puede buscar mejorar los resultados de los clasificadores que se presentaron. Una opción es utilizar rastreadores oculares con una mayor tasa de muestreo y mejores precisiones al momento de capturar la información ocular. Mejores rastreadores permitirían calcular medidas para el entrenamiento que se ha comprobado que tienen relación con el aumento o disminución de la carga cognitiva.

Además, para mejorar el rendimiento de los clasificadores se podría diseñar un experimento con una combinación de tareas auditivas y visuales que sea ejecutado en un número considerable de participantes para tener datos suficientes para el entrenamiento de los clasificadores.

Apéndice A

Medidas.py

Código en python 3 para el cálculo de las métricas oculares utilizadas en esta tesis. Dentro del código se define una función para el filtrado de los datos y una función para cada una de las métricas. Al final del cálculo las métricas son guardadas en un archivo CSV para su posterior análisis.

```
1  import os
2  from xml.dom import minidom
3  from xml.dom.minidom import parse
4  import pandas as pd
5  import numpy as np
6
7  def filtrarDatos(datos):
8          contador = 0
9          tamInicial = len(datos)
10         diameter_pupil_lefteye = np.array(datos.diameter_pupil_lefteye)
11         diameter_pupil_righteye = np.array(datos.diameter_pupil_righteye)
12         timestamp_microsec = np.array(datos.timestamp_microsec)
13         pupilsize = np.zeros(len(diameter_pupil_lefteye))
14
```

```
15
16          tiempo = 0
17          inicioBlink = 0
18          numeroDeBlinks = 0
19          for x in range(0,len(diameter_pupil_lefteye)):
20                  if (diameter_pupil_lefteye[x] == -1 and inicioBlink == 0):
21                          inicioBlink = 1
22                  elif (diameter_pupil_lefteye[x] == -1 and inicioBlink ==
                    ↪  1):
23                          if timestamp_microsec[x]>timestamp_microsec[x-1]:
24                                  tiempo = tiempo +
                                    ↪  (timestamp_microsec[x]-timestamp_microsec[x-1])
25                          else:
26                                  tiempo = tiempo + (1000000 -
                                    ↪  timestamp_microsec[x-1])
27                                  tiempo = tiempo + timestamp_microsec[x]
28                  elif (diameter_pupil_lefteye[x] != -1 and inicioBlink ==
                    ↪  1):
29                          if (tiempo>=100000 and tiempo<=400000):
30                                  numeroDeBlinks = numeroDeBlinks + 1
31                                  inicioBlink = 0
32                                  tiempo = 0
33                          else:
34                                  inicioBlink = 0
35                                  tiempo = 0
36                  else:
```

```
37                      inicioBlink = 0
38                      tiempo = 0
39
40
41
42          while 1:
43                  if contador==len(diameter_pupil_lefteye):
44                          break;
45
46                  if (diameter_pupil_lefteye[contador]!=-1 and
                     ↪  diameter_pupil_righteye[contador]!=-1):
47                          pupilsize[contador] =
                             ↪  (diameter_pupil_lefteye[contador] +
                             ↪  diameter_pupil_righteye[contador])/2
48                          contador = contador + 1
49
50                  elif (diameter_pupil_lefteye[contador]==-1 and
                     ↪  diameter_pupil_righteye[contador]!=-1):
51                          pupilsize[contador] =
                             ↪  diameter_pupil_righteye[contador]
52                          contador = contador + 1
53
54                  elif (diameter_pupil_lefteye[contador]!=-1 and
                     ↪  diameter_pupil_righteye[contador]==-1):
55                          pupilsize[contador] =
                             ↪  diameter_pupil_lefteye[contador]
```

```
56                          contador = contador + 1
57                  else:
58                          timestamp_microsec = np.delete(timestamp_microsec,
                            ↪  contador)
59                          diameter_pupil_lefteye =
                            ↪  np.delete(diameter_pupil_lefteye, contador)
60                          diameter_pupil_righteye =
                            ↪  np.delete(diameter_pupil_righteye, contador)
61                          pupilsize = np.delete(pupilsize, contador)
62
63          tamFinal = len(pupilsize)
64          porcent = (100/tamInicial)*tamFinal
65          return pupilsize, timestamp_microsec, porcent, numeroDeBlinks
66
67  def hampel(x,k,thr=3):
68          arraySize = len(x)
69          idx=np.arange(arraySize)
70          newX=x.copy()
71          omadIdx=np.zeros_like(x)
72          for i in range(arraySize):
73                  mask1=np.where( idx>= (idx[i]-k) ,True, False)
74                  mask2=np.where( idx<= (idx[i]+k) ,True, False)
75                  kernel= np.logical_and(mask1,mask2)
76                  med0=np.median(x[kernel])
77                  s0=1.4826*np.median(np.abs(x[kernel]-med0))
78                  if np.abs(x[i]-med0)>thr*s0:
```

```
79                          omadIdx[i]=1
80                          newX[i]=med0
81          return newX,omadIdx.astype(bool)
82
83  def baseLine(pupilsize, timestamp_microsec):
84          stop = False
85          tiempo = 0
86          tamaño = 0
87          i = 1
88          while not stop:
89                  if tiempo<=2000000:
90                          if timestamp_microsec[i]>timestamp_microsec[i-1]:
91                                  tiempo = tiempo + (timestamp_microsec[i] -
                                    ↪  timestamp_microsec[i-1])
92                                  i = i + 1
93                          else:
94                                  tiempo = tiempo + (1000000 -
                                    ↪  timestamp_microsec[i-1])
95                                  tiempo = tiempo + timestamp_microsec[i]
96                                  i = i + 1
97                  else:
98                          i = i-1
99                          tiempo = tiempo - (timestamp_microsec[i] -
                            ↪  timestamp_microsec[i-1])
100                         stop = True
101
```

```
102
103         tamaño = np.sum(pupilsize[(i-19):(i+1)])
104
105         blps = tamaño/(20)
106         return blps, i
107
108 def meanPupilDiameterChange(pupilsize,BLPS,i):
109         pupil = np.sum(pupilsize[i:])
110         MPDC = (pupil/len(pupilsize[i:]))-BLPS
111         return MPDC
112
113 def avaragePercentageChangePupil(pupilsize,BLPS,i):
114         pcps = 0
115         pupil = pupilsize[i:]
116         for x in range(len(pupil)):
117                 pcps = pcps + ((pupil[x] - BLPS)/BLPS)
118         APCPS = pcps/len(pupil)
119         return APCPS
120
121 def peakDilation(pupilsize,i):
122         PD1 = np.max(pupilsize[i:])
123         return PD1
124
125 def inform(subject, archivos):
126         informacion = pd.read_csv("InformacionPruebasVisual.csv")
127         name = archivos[0:-4]
```

```
128         posicion = -99
129         for x in range(0,len(informacion)):
130                 if (name == informacion.Name[x] and
                    ↪  informacion.Subject[x]==subject):
131                         posicion = x
132                         break
133         if posicion == -99:
134                 nivel = 0
135         elif str(informacion.Stimulus_sequence[posicion])=='nan':
136                 nivel = 0
137         else:
138                 #3=5, 4=7, 5=9
139                 nivel =
                    ↪  len(informacion.Stimulus_sequence[posicion])-((len(
                    ↪  informacion.Stimulus_sequence[posicion])-1)/2)-2
140
141         training = informacion.Training[posicion]
142
143
144         return subject, name, training, nivel
145
146 def pupilEntropy(pupilsize,i):
147         A = pupilsize[i:]
148         pA = A / A.sum()
149         Shannon2 = -np.sum(pA*np.log2(A))
150         return Shannon2
```

```
151
152 def timeToPeakPupilSize(pupilsize,i, pd1,timestamp_microsec):
153         stop = False
154         tiempo = 0
155         tamaño = 0
156         i = i
157
158         while not stop:
159                 if pupilsize[i]!=pd1:
160                         if timestamp_microsec[i]>timestamp_microsec[i-1]:
161                                 tiempo = tiempo + (timestamp_microsec[i] -
                                    ↪ timestamp_microsec[i-1])
162                                 i = i + 1
163                         else:
164                                 tiempo = tiempo + (1000000 -
                                    ↪ timestamp_microsec[i-1])
165                                 tiempo = tiempo + timestamp_microsec[i]
166                                 i = i + 1
167                 else:
168                         i = i-1
169                         tiempo = tiempo - (timestamp_microsec[i] -
                            ↪ timestamp_microsec[i-1])
170                         stop = True
171         return tiempo
172
173 def pupilSizeTime(pupilsize,i,timestamp_microsec):
```

```
174         pupilsize=np.array(pupilsize)
175         tiempo = np.array(timestamp_microsec)
176         for x in range(0,len(timestamp_microsec)):
177                 if x==0:
178                         tiempo[x] = 0
179                 else:
180                         if timestamp_microsec[x]>timestamp_microsec[x-1]:
181                                 tiempo[x] = tiempo[x-1] +
                                    ↪ (timestamp_microsec[x]-timestamp_microsec[x-1])
182                         else:
183                                 tiempo[x] = tiempo[x-1] + (1000000 -
                                    ↪ timestamp_microsec[x-1])
184                                 tiempo[x] = tiempo[x] +
                                    ↪ timestamp_microsec[x]
185
186         m =
            ↪ ((np.sum(tiempo*pupilsize))-((np.sum(pupilsize)*np.sum(tiempo))/len(pupils
            ↪ ))/((np.sum(tiempo*tiempo))-((np.sum(tiempo)**2)/len(tiempo)))
187         PST = np.arctan(m)
188         return PST, tiempo[len(tiempo)-1]
189
190 def blinkFrequency(numeroDeBlinks, tiempo):
191         BF = numeroDeBlinks/(tiempo/1000000)
192         return BF
193
194 def filtrarDatosGaze(gaze):
```

```
195         contador = 0
196         tamInicial = len(gaze)
197         x_gazepos_lefteye = np.array(gaze[12])
198         x_gazepos_righteye = np.array(gaze[13])
199         y_gazepos_lefteye = np.array(gaze[16])
200         y_gazepos_righteye = np.array(gaze[17])
201         timestamp_microsec = np.array(gaze[6])
202
203         diameter_pupil_lefteye = np.array(gaze[2])
204         diameter_pupil_righteye = np.array(gaze[3])
205         pupilsize = np.zeros(len(diameter_pupil_lefteye))
206
207
208         while 1:
209                 if contador==len(x_gazepos_lefteye):
210                         break;
211
212                 if (x_gazepos_lefteye[contador]==-1 or
                    ↪  x_gazepos_righteye[contador]==-1 or
                    ↪  y_gazepos_lefteye[contador]==-1 or
                    ↪  y_gazepos_righteye[contador]==-1):
213                         x_gazepos_lefteye = np.delete(x_gazepos_lefteye,
                            ↪  contador)
214                         x_gazepos_righteye = np.delete(x_gazepos_righteye,
                            ↪  contador)
```

```
215                         y_gazepos_lefteye = np.delete(y_gazepos_lefteye,
                             ↪  contador)
216                         y_gazepos_righteye = np.delete(y_gazepos_righteye,
                             ↪  contador)
217                         timestamp_microsec = np.delete(timestamp_microsec,
                             ↪  contador)
218                 else:
219                         contador = contador + 1
220
221         return x_gazepos_lefteye, x_gazepos_righteye, y_gazepos_lefteye,
             ↪  y_gazepos_righteye, timestamp_microsec
222
223 def tiempoInicio(timestamp_microsec):
224         stop = False
225         tiempo = 0
226         i = 1
227         while not stop:
228                 if tiempo<=2000000:
229                         if timestamp_microsec[i]>timestamp_microsec[i-1]:
230                                 tiempo = tiempo + (timestamp_microsec[i] -
                                     ↪  timestamp_microsec[i-1])
231                                 i = i + 1
232                         else:
233                                 tiempo = tiempo + (1000000 -
                                     ↪  timestamp_microsec[i-1])
234                                 tiempo = tiempo + timestamp_microsec[i]
```

```
235                                 i = i + 1
236                 else:
237                         i = i-1
238                         tiempo = tiempo - (timestamp_microsec[i] -
                            ↪ timestamp_microsec[i-1])
239                         stop = True
240
241         return i
242
243 def fijaciones(i,x_gazepos_lefteye, x_gazepos_righteye, y_gazepos_lefteye,
    ↪ y_gazepos_righteye, timestamp_microsec, missing=0.0, maxdist=25,
    ↪ mindur=50):
244         """Detects fixations, defined as consecutive samples with an
            ↪ inter-sample
245         distance of less than a set amount of pixels (disregarding missing
    ↪ data)
246
247         arguments
248         x                -        numpy array of x positions
249         y                -        numpy array of y positions
250         time                -        numpy array of EyeTribe timestamps
251         keyword arguments
252         missing        -        value to be used for missing data (default
    ↪ = 0.0)
253         maxdist        -        maximal inter sample distance in pixels
    ↪ (default = 25)
```

```
254         mindur        -        minimal duration of a fixation in
    ↪ milliseconds; detected
255                       fixation cadidates will be disregarded if they are
    ↪ below
256                       this duration (default = 100)
257 
258         returns
259         Sfix, Efix
260                 Sfix        -        list of lists, each containing
    ↪ [starttime]
261                 Efix        -        list of lists, each containing
    ↪ [starttime, endtime, duration, endx, endy]
262         """
263 
264         # empty list to contain data
265         Sfix = []
266         Efix = []
267         x_gazepos_lefteye = np.array(gaze[12])
268         y_gazepos_lefteye = np.array(gaze[16])
269         timestamp_microsec = timestamp_microsec/1000
270 
271         # loop through all coordinates
272         si = 0
273         fixstart = False
274         for i in range(1,len(x_gazepos_lefteye)):
```

```
275                 # calculate Euclidean distance from the current fixation
                    ↪ coordinate
276                 # to the next coordinate
277                 dist = ((x_gazepos_lefteye[si]-x_gazepos_lefteye[i])**2 +
                    ↪ (y_gazepos_lefteye[si]-y_gazepos_lefteye[i])**2)**0.5
278                 # check if the next coordinate is below maximal distance
279                 if dist <= maxdist and not fixstart:
280                         # start a new fixation
281                         si = 0 + i
282                         fixstart = True
283                         Sfix.append([timestamp_microsec[i]])
284                 elif dist > maxdist and fixstart:
285                         # end the current fixation
286                         fixstart = False
287                         # only store the fixation if the duration is ok
288                         if timestamp_microsec[i-1]-Sfix[-1][0] >= mindur:
289                                 Efix.append([Sfix[-1][0],
                                    ↪ timestamp_microsec[i-1],
                                    ↪ timestamp_microsec[i-1]-Sfix[-1][0],
                                    ↪ x_gazepos_lefteye[si],
                                    ↪ y_gazepos_lefteye[si]])
290                         # delete the last fixation start if it was too
                            ↪ short
291                         else:
292                                 Sfix.pop(-1)
293                         si = 0 + i
```

```
294                 elif not fixstart:
295                         si += 1
296
297         return Sfix, Efix
298
299
300 def archivo(subject, name, training, nivel, blps, mpdc, apcps, pd1,
    ↪  entropy, TTP, PST, BF, numeroDeBlinks):
301         if os.path.isfile('VariablesVisual.csv')==False:
302                 data = pd.DataFrame(columns=('Subject', 'Name',
                    ↪  'Training', 'Nivel','BLPS','MPDC','APCPS','PD',
                    ↪  'Entropy', 'TTP','PST', 'BF', 'numeroDeBlinks'))
303                 data.loc[len(data)]=[subject, name, training, nivel, blps,
                    ↪  mpdc, apcps, pd1, entropy, TTP, PST, BF,
                    ↪  numeroDeBlinks]
304                 data.to_csv('VariablesVisual.csv', index = None,
                    ↪  header=True)
305         else:
306                 data = pd.read_csv('VariablesVisual.csv')
307                 data.loc[len(data)]=[subject, name, training, nivel, blps,
                    ↪  mpdc, apcps, pd1, entropy, TTP, PST, BF,
                    ↪  numeroDeBlinks]
308                 data.to_csv('VariablesVisual.csv', index = None,
                    ↪  header=True)
309
310 participantes = 17
```

```
311
312 for x in range(1,participantes+1):
313         if x<=9:
314                 origen = "D:/GoogleDriveCIMAT/CIMAT/Tesis/Memoria de
                    ↪ trabajo/Datos/visual-data/s0"+str(x)
315         else:
316                 origen = "D:/GoogleDriveCIMAT/CIMAT/Tesis/Memoria de
                    ↪ trabajo/Datos/visual-data/s"+str(x)
317
318         for carpetas in os.listdir(origen):
319                 if (carpetas[0]=='S' or carpetas[0]=='s'):
320                         for archivos in
                            ↪ os.listdir(os.path.join(origen,carpetas)):
321                                 if archivos[-1]=='v':
322                                         csv =
                                            ↪ origen+'/'+carpetas+'/'+archivos
323                                         datos = pd.read_csv(csv)
324                                         print("####"+csv+"###")
325                                         [pupilsize, timestamp_microsec,
                                            ↪ porcent, numeroDeBlinks] =
                                            ↪ filtrarDatos(datos)
326                                         pupilsize = hampel(pupilsize,10)
327                                         [blps,i] = baseLine(pupilsize,
                                            ↪ timestamp_microsec)
328                                         mpdc =
                                            ↪ meanPupilDiameterChange(pupilsize,blps,i)
```

```
329                         apcps =
                             ↪ avaragePercentageChangePupil(pupilsize,blp
330                         pd1 = peakDilation(pupilsize,i)
331                         entropy =
                             ↪ pupilEntropy(pupilsize,i)
332                         TTP =
                             ↪ timeToPeakPupilSize(pupilsize,i,pd1,timest
333                         [PST,tiempo] =
                             ↪ pupilSizeTime(pupilsize,i,timestamp_micros
334                         BF =
                             ↪ blinkFrequency(numeroDeBlinks,
                             ↪ tiempo)
335                         [subject, name, training, nivel] =
                             ↪ inform(origen[-3:],archivos)
336                         print(nivel)
337                         archivo(subject, name, training,
                             ↪ nivel, blps, mpdc, apcps, pd1,
                             ↪ entropy, TTP, PST,BF,
                             ↪ numeroDeBlinks)
338                         #archivo(subject, name, training,
                             ↪ stimulus_sequence, duration)
339                         #######Borrar todo lo que tenga
                             ↪ que ver con fijaciones
340                     elif (archivos[-1]=='t' and
                         ↪ archivos[0]=='P'):
```

```
341                 txt =
                    ↪  origen+'/'+carpetas+'/'+archivos
342                 gaze = pd.read_csv(txt,
                    ↪  header=None, sep='  ')
343                 [x_gazepos_lefteye,
                    ↪  x_gazepos_righteye,
                    ↪  y_gazepos_lefteye,
                    ↪  y_gazepos_righteye,
                    ↪  timestamp_microsec] =
                    ↪  filtrarDatosGaze(gaze)
344                 i =
                    ↪  tiempoInicio(timestamp_microsec)
345                 [Sfix, Efix] = fijaciones(i,
                    ↪  x_gazepos_lefteye,
                    ↪  x_gazepos_righteye,
                    ↪  y_gazepos_lefteye,
                    ↪  y_gazepos_righteye,
                    ↪  timestamp_microsec)
```

Apéndice B

SVM.py

Código en python 3 para el entrenamiento de los diferentes modelos de SVM utilizados. Como primer paso se preparan los datos con los que se realizará el entrenamiento y la evaluación, seguido de la definición de los modelos y por último su evaluación. Para evaluarlos se imprime en pantalla un reporte y la matriz de confusión.

```
1  import pandas as pd
2  from sklearn.model_selection import train_test_split
3  from sklearn import svm
4  from sklearn.metrics import classification_report, confusion_matrix
5
6  # Se carga archivo csv con las medidas calculadas para cada prueba.
7  datos = pd.read_csv('D:/GoogleDriveCIMAT/CIMAT/Tesis/Memoria de
   ↪  trabajo/Analisis/analisis-tareas-de-memoria-de-trabajo/Entrenamiento.csv')
8  datos=datos.astype(float)
9
10 # y=Nivel de dificultad de la prueba, x=Características
11 y=datos.Nivel
12 # Se eliminan de la lista las características con las que queremos
   ↪  entrenar el clasificador.
```

```
13 X=datos.drop(["Nivel","BLPS","Entropy","TTP","PST","BF","numeroDeBlinks"],axis=1)
14
15 print('Cantidad de pruebas por nivel:')
16 print(datos['Nivel'].value_counts())
17
18 # Se separan los datos para entrenamiento (80%) y evaluación (20%).
19 X_train, X_test, y_train, y_test = train_test_split(X, y,
20                                                     test_size=0.2,
21                                                     random_state=42,
22                                                     stratify=y)
23
24 # Preparación de los datos para el entrenamiento.
25 from sklearn.preprocessing import StandardScaler
26 sc = StandardScaler()
27 X_train_array = sc.fit_transform(X_train.values)
28 X_train = pd.DataFrame(X_train_array, index=X_train.index,
    ↪ columns=X_train.columns)
29 X_test_array = sc.transform(X_test.values)
30 X_test = pd.DataFrame(X_test_array, index=X_test.index,
    ↪ columns=X_test.columns)
31
32 print("**********************Diferentes kernel SVM*****************")
33
34 C = 1.0
35 models = (svm.SVC(kernel='linear', C=C),
36           svm.LinearSVC(C=C, max_iter=100),
```

```
37            svm.SVC(kernel='rbf', gamma='auto', C=C),
38            svm.SVC(kernel='sigmoid', C=C),
39            svm.SVC(kernel='poly', degree=3, gamma='auto', C=C))
40
41  for x in models:
42          print(x)
43          # Se crea y entra el modelo
44          x.fit(X_train,y_train)
45          y_pred = x.predict(X_test)
46          # Matriz de confusión y reporte de estadísticas.
47          print(confusion_matrix(y_test,y_pred))
48          print(classification_report(y_test,y_pred))
49          print("********************")
```

Apéndice C

LDA.py

Código en python 3 para el entrenamiento del modelo de clasificación LDA. Como primer paso se preparan los datos con los que se realizará el entrenamiento y la evaluación, seguido de la definición de los modelos y por último su evaluación. Para evaluarlos se imprime en pantalla un reporte y la matriz de confusión.

```
1  import pandas as pd
2  from sklearn.model_selection import train_test_split
3  from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as
   ↪  LDA
4  from sklearn.metrics import classification_report, confusion_matrix
5
6  # Se carga archivo csv con las medidas calculadas para cada prueba.
7  datos = pd.read_csv('D:/GoogleDriveCIMAT/CIMAT/Tesis/Memoria de
   ↪  trabajo/Analisis/analisis-tareas-de-memoria-de-trabajo/Entrenamiento.csv')
8  datos=datos.astype(float)
9
10 # y=Nivel de dificultad de la prueba, x=Características
11 y=datos.Nivel
```

```
12 # Se eliminan de la lista las características con las que queremos
   ↪ entrenar el clasificador.
13 X=datos.drop(['Nivel','BLPS','APCPS','BF','numeroDeBlinks'],axis=1)
14
15 print('Cantidad de pruebas por nivel:')
16 print(datos['Nivel'].value_counts())
17
18 # Se separan los datos para entrenamiento (80%) y evaluación (20%).
19 X_train, X_test, y_train, y_test = train_test_split(X, y,
20                                                     test_size=0.2,
21                                                     random_state=42,
22                                                     stratify=y)
23
24 # Preparación de los datos para el entrenamiento.
25 from sklearn.preprocessing import StandardScaler
26 sc = StandardScaler()
27 X_train_array = sc.fit_transform(X_train.values)
28 X_train = pd.DataFrame(X_train_array, index=X_train.index,
   ↪ columns=X_train.columns)
29 X_test_array = sc.transform(X_test.values)
30 X_test = pd.DataFrame(X_test_array, index=X_test.index,
   ↪ columns=X_test.columns)
31
32 # Se crea y entra el modelo
33 clf = LDA(solver='svd')
34 clf = clf.fit(X_train, y_train)
```

```
35
36 # Matriz de confusión y reporte de estadísticas.
37 y_pred = clf.predict(X_test)
38 print(confusion_matrix(y_test,y_pred))
39 print(classification_report(y_test,y_pred))
```

Apéndice D

tree.py

Código en python 3 para el entrenamiento del modelo de árbol de decisión. Como primer paso se preparan los datos con los que se realizará el entrenamiento y la evaluación, seguido de la definición de los modelos y por último su evaluación. Para evaluarlos se imprime en pantalla un reporte y la matriz de confusión.

```
1  import pandas as pd
2  from sklearn.model_selection import train_test_split
3  from sklearn import tree
4  from sklearn.metrics import classification_report, confusion_matrix
5
6  # Se carga archivo csv con las medidas calculadas para cada prueba.
7  datos = pd.read_csv('D:/GoogleDriveCIMAT/CIMAT/Tesis/Memoria de
   ↪  trabajo/Analisis/analisis-tareas-de-memoria-de-trabajo/Entrenamiento.csv')
8  datos=datos.astype(float)
9
10 # y=Nivel de dificultad de la prueba, x=Características
11 y=datos.Nivel
12 # Se eliminan de la lista las características con las que queremos
   ↪  entrenar el clasificador.
```

```
13 X=datos.drop(['Nivel','BLPS','Entropy','TTP','PST','BF','numeroDeBlinks'],axis=1)
14
15 print('Cantidad de pruebas por nivel:')
16 print(datos['Nivel'].value_counts())
17
18 # Se separan los datos para entrenamiento (80%) y evaluación (20%).
19 X_train, X_test, y_train, y_test = train_test_split(X, y,
20                                                     test_size=0.2,
21                                                     random_state=42,
22                                                     stratify=y)
23
24 # Preparación de los datos para el entrenamiento.
25 from sklearn.preprocessing import StandardScaler
26 sc = StandardScaler()
27 X_train_array = sc.fit_transform(X_train.values)
28 X_train = pd.DataFrame(X_train_array, index=X_train.index,
    ↪ columns=X_train.columns)
29 X_test_array = sc.transform(X_test.values)
30 X_test = pd.DataFrame(X_test_array, index=X_test.index,
    ↪ columns=X_test.columns)
31
32 # Se crea y entra el modelo
33 clf = tree.DecisionTreeClassifier(criterion='gini', splitter='best')
34 clf = clf.fit(X_train, y_train)
35
36 # Matriz de confusión y reporte de estadísticas.
```

```
37 y_pred = clf.predict(X_test)
38 print(confusion_matrix(y_test,y_pred))
39 print(classification_report(y_test,y_pred))
```

Apéndice E

randomforest.py

Código en python 3 para el entrenamiento del modelo de bosque aleatorio. Como primer paso se preparan los datos con los que se realizará el entrenamiento y la evaluación, seguido de la definición de los modelos y por último su evaluación. Para evaluarlos se imprime en pantalla un reporte y la matriz de confusión.

```
1  import numpy as np
2  import pandas as pd
3  from matplotlib import pyplot as plt
4  from sklearn.ensemble import RandomForestClassifier
5  from sklearn.model_selection import train_test_split
6  from sklearn.metrics import classification_report, confusion_matrix
7
8  # Se carga archivo csv con las medidas calculadas para cada prueba.
9  data=pd.read_csv('D:/GoogleDriveCIMAT/CIMAT/Tesis/Memoria de
   ↪  trabajo/Analisis/analisis-tareas-de-memoria-de-trabajo/Entrenamiento.csv')
10 data=data.astype(float)
11
12 # y=Nivel de dificultad de la prueba, x=Características
13 df_y=data.Nivel
```

```
14 # Se eliminan de la lista las características con las que queremos
   ↪ entrenar el clasificador.
15 df_x=data.drop(['Nivel','BLPS','MPDC','APCPS','PD','Entropy','TTP','PST','BF',
   ↪ 'numeroDeBlinks'],axis=1)
16
17 print('Cantidad de pruebas por nivel:')
18 print(datos['Nivel'].value_counts())
19
20 # Se separan los datos para entrenamiento (80%) y evaluación (20%).
21 x_train, x_test, y_train, y_test = train_test_split(df_x, df_y,
   ↪ test_size=0.2, random_state=4)
22
23 # Se crea y entra el modelo
24 rf=RandomForestClassifier(n_estimators=100)
25 rf.fit(x_train,y_train)
26
27 # Matriz de confusión y reporte de estadísticas.
28 y_pred = rf.predict(x_test)
29 print(confusion_matrix(y_test,y_pred))
30 print(classification_report(y_test,y_pred))
```

Apéndice F

Estancias y artículos

Artículos

Mitre-Hernandez, H, Sanchez-Rodriguez, J, Zatarain-Cabada, R, Barron-Estrada, L. Assessing cognitive load using oculometrics toidentify deceit during interviews. En:Applied Cognitive Psychology 33.2 (2019),págs. 312-321.

Estancia de inestigación (Septiembre 2019 - Diciembre 2019)

Sánchez Rodríguez, J, Díaz Piedra, C, Di Stasi, L, (2019). Evaluación de la respuesta ocular y termográfica ante respuestas sinceras y falsas. Estancia de investigación, Centro de Investigación Mente, Cerebro y Comportamiento, Universidad de Granada.

Bibliografía

[1] Magdalena Andrzejewska y Agnieszka Skawińska. «Examining Students' Intrinsic Cognitive Load During Program Comprehension–An Eye Tracking Approach». En: *International Conference on Artificial Intelligence in Education*. Springer. 2020, págs. 25-30.

[2] Alan Baddeley. «Working memory». En: *Science* 255.5044 (1992), págs. 556-559.

[3] Ryan S.J.d. Baker y col. «Better to be frustrated than bored: The incidence, persistence, and impact of learners' cognitive-affective states during interactions with three different computer-based learning environments». En: *International Journal of Human-Computer Studies* 68.4 (2010), págs. 223 -241. ISSN: 1071-5819. DOI: http://dx.doi.org/10.1016/j.ijhcs.2009.12.003.

[4] Douglas Bates y col. «Fitting Linear Mixed-Effects Models Using lme4». En: *Journal of Statistical Software* 67.1 (2015), págs. 1-48. DOI: 10.18637/jss.v067.i01.

[5] J Beatty y D Kahneman. «Pupil diameter and load on memory(Pupillary diametric size as indicator of load on human memory)». En: *Science* 154 (1966), págs. 1583-1585.

[6] Jackson Beatty. «Task-evoked pupillary responses, processing load, and the structure of processing resources.» En: *Psychological bulletin* 91.2 (1982), págs. 276-292. DOI: http://dx.doi.org/10.1037/0033-2909.91.2.276.

[7] Ralf Biedert y col. «A robust realtime reading-skimming classifier». En: *Proceedings of the Symposium on Eye Tracking Research and Applications*. 2012, págs. 123-130.

[8] Daniel T Burley, Nicola S Gray y Robert J Snowden. «As far as the eye can see: Relationship between psychopathic traits and pupil response to affective stimuli». En: *PLoS One* 12.1 (2017), e0167436.

[9] Vinod Kumar Chauhan, Kalpana Dahiya y Anuj Sharma. «Problem formulations and solvers in linear SVM: a review». En: *Artificial Intelligence Review* 52.2 (2019), págs. 803-855.

[10] Siyuan Chen, Julien Epps y Fang Chen. «A Comparison of Four Methods for Cognitive Load Measurement». En: *Proceedings of the 23rd Australian Computer-Human Interaction Conference*. OzCHI '11. ACM, 2011, págs. 76-79. URL: 10.1145/2071536.2071547.

[11] Corinna Cortes y Vladimir Vapnik. «Support-vector networks». En: *Machine learning* 20.3 (1995), págs. 273-297.

[12] Jeanine A. DeFalco y col. «Detecting and Addressing Frustration in a Serious Game for Military Training». En: *International Journal of Artificial Intelligence in Education* (sep. de 2017). ISSN: 1560-4306. DOI: 10.1007/s40593-017-0152-1. URL: https://doi.org/10.1007/s40593-017-0152-1.

[13] Qi Deng y col. «Prediction performance of lane changing behaviors: a study of combining environmental and eye-tracking data in a driving simulator». En: *IEEE Transactions on Intelligent Transportation Systems* (2019).

[14] Leandro L Di Stasi y col. «Saccadic peak velocity sensitivity to variations in mental workload». En: *Aviation, space, and environmental medicine* 81.4 (2010), págs. 413-417.

[15] Shahram Eivazi y Roman Bednarik. «Predicting problem-solving behavior and performance levels from visual attention data». En: *Proc. Workshop on Eye Gaze in Intelligent Human Machine Interaction at IUI*. 2011, págs. 9-16.

[16] Vérane Faure, Regis Lobjois y Nicolas Benguigui. «The effects of driving environment complexity and dual tasking on drivers' mental workload and eye blink behavior». En: *Transportation research part F: traffic psychology and behaviour* 40 (2016), págs. 78-90. DOI: https://doi.org/10.1016/j.trf.2016.04.007.

[17] P Frosina y col. «The effect of cognitive load on nonverbal behavior in the cognitive interview for suspects». En: *Personality and Individual Differences* 130 (2018), págs. 51-58.

[18] P Frosina y col. «The effect of cognitive load on nonverbal behavior in the cognitive interview for suspects». En: *Personality and Individual Differences* 130 (2018), págs. 51-58. DOI: https://doi.org/10.1016/j.paid.2018.03.012.

[19] Kyosuke Fukuda. «Eye blinks: new indices for the detection of deception». En: *International Journal of Psychophysiology* 40.3 (2001), págs. 239-245.

[20] S. George y col. «Eye blink count and eye blink duration analysis for deception detection». En: *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. Sep. de 2017, págs. 223-229. DOI: 10.1109/ICACCI.2017.8125844.

[21] L Gila, A Villanueva y R Cabeza. «Fisiopatología y técnicas de registro de los movimientos oculares». En: *Anales del sistema sanitario de Navarra*. Vol. 32. SciELO Espana. 2009, págs. 9-26.

[22] Eric Granholm y col. «Pupillary responses index cognitive resource limitations». En: *Psychophysiology* 33.4 (1996), págs. 457-461.

[23] Sandra G. Hart. «Nasa-Task Load Index (NASA-TLX); 20 Years Later». En: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 50.9 (2006), págs. 904-908. DOI: 10.1177/154193120605000909.

[24] David A. Harville. «Maximum Likelihood Approaches to Variance Component Estimation and to Related Problems». En: *Journal of the American Statistical Association* 72.358 (1977), págs. 320-338. DOI: 10.1080/01621459.1977.10480998. eprint: https://www.tandfonline.com/doi/pdf/10.1080/01621459.1977.10480998. URL: https://www.tandfonline.com/doi/abs/10.1080/01621459.1977.10480998.

[25] Eckhard H Hess y James M Polt. «Pupil size in relation to mental activity during simple problem-solving». En: *Science* 143.3611 (1964), págs. 1190-1192.

[26] Torsten Hothorn, Frank Bretz y Peter Westfall. «Simultaneous Inference in General Parametric Models». En: *Biometrical Journal* 50.3 (2008), págs. 346-363.

[27] Shamsi T. Iqbal, Xianjun Sam Zheng y Brian P. Bailey. «Task-evoked Pupillary Response to Mental Workload in Human-computer Interaction». En: *CHI '04 Extended Abstracts on Human Factors in Computing Systems*. CHI EA '04. ACM, 2004, págs. 1477-1480. URL: 10.1145/985921.986094.

[28] Shamsi T. Iqbal y col. «Towards an Index of Opportunity: Understanding Changes in Mental Workload During Task Execution». En: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '05. ACM, 2005, págs. 311-320. URL: 10.1145/1054972.1055016.

[29] David E Irwin y Laura E Thomas. «Eyeblinks and cognition.» En: (2010).

[30] Daniel Kahneman, Linda Onuska y Ruth E Wolman. «Effects of grouping on the pupillary response in a short-term memory task». En: *Quarterly Journal of Experimental Psychology* 20.3 (1968), págs. 309-311.

[31] Daniel Kahneman y Patricia Wright. «Changes of pupil size and rehearsal strategies in a short-term memory task». En: *The Quarterly journal of experimental psychology* 23.2 (1971), págs. 187-196.

[32] Jeff Klingner, Barbara Tversky y Pat Hanrahan. «Effects of visual and verbal presentation on cognitive load in vigilance, memory, and arithmetic tasks». En: *Psychophysiology* 48.3 (2011), págs. 323-332. DOI: 10.1111/j.1469-8986.2010.01069.x.

[33] Sepp Kollmorgen y Kenneth Holmqvist. «Automatically detecting reading in eye tracking data». En: *Lund University Cognitive Studies* (2007).

[34] Andrew L. Kun y col. «On the feasibility of using pupil diameter to estimate cognitive load changes for in-vehicle spoken dialogues». En: *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*. International Speech y Communication Association, 2013, págs. 3766-3770.

[35] Andrew L Kun y col. «On the feasibility of using pupil diameter to estimate cognitive load changes for in-vehicle spoken dialogues.» En: *INTERSPEECH*. 2013, págs. 3766-3770.

[36] Kai Kunze y col. «I know what you are reading: recognition of document types using mobile eye tracking». En: *Proceedings of the 2013 International Symposium on Wearable Computers*. 2013, págs. 113-116.

[37] Sébastien Lallé y col. «Prediction of Users' Learning Curves for Adaptation While Using an Information Visualization». En: *Proceedings of the 20th International Conference on Intelligent User Interfaces*. IUI '15. ACM, 2015, págs. 357-368. URL: 10.1145/2678025.2701376.

[38] Jue Li y col. «Identification and classification of construction equipment operators' mental fatigue using wearable eye-tracking technology». En: *Automation in Construction* 109 (2020), pág. 103000.

[39] Fu-Ren Lin y Chien-Min Kao. «Mental effort detection using EEG data in E-learning contexts». En: *Computers & Education* 122 (2018), págs. 63-79.

[40] Ramtin Zargari Marandi y col. «Reliability of Oculometrics During a Mentally Demanding Task in Young and Old Adults». En: *Ieee Access* 6 (2018), págs. 17500-17517. DOI: 10.1109/ACCESS.2018.2819211.

[41] Richard E Mayer y Roxana Moreno. «Nine ways to reduce cognitive load in multimedia learning». En: *Educational psychologist* 38.1 (2003), págs. 43-52.

[42] Hugo Mitre-Hernandez y col. «Assessing cognitive load using oculometrics to identify deceit during interviews». En: *Applied Cognitive Psychology* 33.2 (2019), págs. 312-321.

[43] Nargess Nourbakhsh y col. «Detecting users' cognitive load by galvanic skin response with affective interference». En: *ACM Transactions on Interactive Intelligent Systems (TiiS)* 7.3 (2017), págs. 1-20.

[44] R. H. Nugroho, M. Nasrun y C. Setianingsih. «Lie detector with pupil dilation and eye blinks using hough transform and frame difference method with fuzzy logic». En: *2017 International Conference on Control, Electronics, Renewable Energy and Communications (ICCREC)*. Sep. de 2017, págs. 40-45. URL: 10.1109/ICCEREC.2017.8226697.

[45] Fatih Veysel Nurçin y col. «Lie detection on pupil size by back propagation neural network». En: *Procedia Computer Science* 120 (2017), págs. 417-421. DOI: https://doi.org/10.1016/j.procs.2017.11.258.

[46] Oskar Palinko y Andrew L. Kun. «Exploring the Effects of Visual Cognitive Load and Illumination on Pupil Diameter in Driving Simulators». En: *Proceedings of the Symposium on Eye Tracking Research and Applications*. ETRA '12. ACM, 2012, págs. 413-416. URL: http://doi.acm.org/10.1145/2168556.2168650.

[47] Oskar Palinko y col. «Estimating Cognitive Load Using Remote Eye Tracking in a Driving Simulator». En: *Proceedings of the 2010 Symposium on Eye-Tracking Research*

& Applications. ETRA '10. ACM, 2010, págs. 141-144. DOI: 10.1145/1743666.1743701. URL: http://doi.acm.org/10.1145/1743666.1743701.

[48] Megan H Papesh, Stephen D Goldinger y Michael C Hout. «Memory strength and specificity revealed by pupillometry». En: *International Journal of Psychophysiology* 83.1 (2012), págs. 56-64.

[49] Megan H Papesh, Stephen D Goldinger y Michael C Hout. «Memory strength and specificity revealed by pupillometry». En: *International Journal of Psychophysiology* 83.1 (2012), págs. 56-64. DOI: https://doi.org/10.1016/j.ijpsycho.2011.10.002.

[50] H. D. Patterson y R. Thompson. «Recovery of inter-block information when block sizes are unequal». En: *Biometrika* 58.3 (dic. de 1971), págs. 545-554. ISSN: 0006-3444. DOI: 10.1093/biomet/58.3.545. eprint: https://academic.oup.com/biomet/article-pdf/58/3/545/635437/58-3-545.pdf. URL: https://doi.org/10.1093/biomet/58.3.545.

[51] W Scott Peavler. «Pupil size, information overload, and performance differences». En: *Psychophysiology* 11.5 (1974), págs. 559-566.

[52] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2020. URL: https://www.R-project.org/.

[53] Greg J Siegle, Naho Ichikawa y Stuart Steinhauer. «Blink before and after you think: blinks occur prior to and following cognitive load indexed by pupillary responses». En: *Psychophysiology* 45.5 (2008), págs. 679-687. DOI: 10.1111/j.1469-8986.2008.00681.x.

[54] John Sweller. «Cognitive load during problem solving: Effects on learning». En: *Cognitive science* 12.2 (1988), págs. 257-285.

[55] Yuu Tanaka y Kiyoshi Yamaoka. «Blink activity and task difficulty». En: *Perceptual and motor skills* 77.1 (1993), págs. 55-66.

[56] Aldert Vrij y col. «Saccadic eye movement rate as a cue to deceit». En: *Journal of Applied Research in Memory and Cognition* 4.1 (2015), págs. 15-19.

[57] Yingxu Wang. «Formal description of the cognitive process of memorization». En: *Transactions on Computational Science V*. Springer, 2009, págs. 81-98.

[58] Andrea K Webb y col. «Eye movements and pupil size reveal deception in computer administered questionnaires». En: *International Conference on Foundations of Augmented Cognition*. Vol. 5638. 2009, págs. 553-562. DOI: https://doi.org/10.1007/978-3-642-02812-0_64.

[59] Qi-Fan Yang y col. «Balancing cognitive complexity and gaming level: Effects of a cognitive complexity-based competition game on EFL students' English vocabulary learning performance, anxiety and behaviors». En: *Computers & Education* (2020), pág. 103808.

[60] B Yekkehkhany y col. «A comparison study of different kernel functions for SVM-based classification of multi-temporal polarimetry SAR data». En: *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 40.2 (2014), pág. 281.