UNIVERSIDAD AUTÓNOMA DE CIUDAD JUÁREZ

Instituto de Ingeniería y Tecnología

Departamento de Ingeniería Eléctrica y Computación

ANÁLISIS DE DATOS OCULOMÉTRICOS FRENTE A PROCESAMIENTO VISUAL DE CÓDIGO FUENTE

Reporte Técnico de Investigación presentado por:

Mario Amador Román 159545

Requisito para la obtención del título de:

INGENIERO DE SOFTWARE

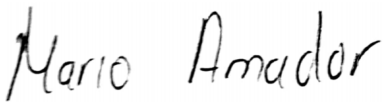
Asesor:

Dr. Francisco López Orozco

Ciudad Juárez, Chihuahua 15 de diciembre de 2020

Declaración de Originalidad

Yo, Mario Amador Román, declaro que el material contenido en esta publicación fue elaborado con la revisión de los documentos que se mencionan en el capítulo de Bibliografía, y que la solución obtenida es original y no ha sido copiada de ninguna otra fuente, ni ha sido usada para obtener otro título o reconocimiento en otra institución de educación superior.



Mario Amador Román

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Planteamiento del problema

En el capítulo uno se abarca los antecedentes, la definición del problema, objetivos, hipótesis y la justificación, conformando una parte vital del desarrollo y entendimiento del proyecto.

1.1 Antecedentes

Estudiar el comportamiento humano al momento de realizar distintas tareas por medio de una computadora para conocer sus estímulos, toma gran relevancia cuando se contempla que en este tipo de estudios se puede encontrar y aprender nuevas maneras para realizar dichas actividades y tener una nueva perspectiva de éstas.

Cuando se trata de obtener datos, el Eye Tracking se vuelve una alternativa viable ya que no acapara la atención del usuario y se enfoca exclusivamente en la tarea que los investigadores desean obtener información [1].

Por ello los trabajos sobre Eye Tracking tienen gran relevancia en tareas que se enfocan principalmente en el área de computación, ya que solo con observar los movimientos visuales se puede deducir y concluir distintos comportamientos sobre el ser humano. Un ejemplo es el trabajo hecho por Joseph Tao-yi Wang, Michael Spezio y Colin F. Camerer donde se puede percibir si los usuarios mienten o dicen la verdad sobre la remuneración económica que obtienen [2].

Actualmente existe un grupo llamado *Eye Movements in Programming* el cual se especializa en investigar el comportamiento ocular de programadores al realizar y analizar código fuente para consolidar nuevas teorías.

En "*Analyzing Individual Performance of Source Code Review Using Reviewers' Eye Movement*" [3] se analiza el rendimiento individual de los participantes mientras revisaban el

código fuente, esto con motivo de caracterizar a los individuos en función de sus habilidades obteniendo como resultado un patrón llamado "*Scan*" haciendo énfasis en que la lectura del código no es una lectura normal u ordinaria a la lectura sobre documentos como periódicos, revistas u otro tipo de documentos.

Un trabajo muy similar es presentado por Roman Bednarik y Markku Tukiainen en que se realiza una investigación sobre la comprensión de los programas, es decir, reconocer la habilidad que se tiene para entender los programas escritos con anterioridad. En dicho estudio utilizan un total de 18 participantes con 3 programas cortos escritos en java, que rondan entre 15 a 38 líneas de código. Este estudio tiene la particularidad de que se utilizó un programa llamado "*Jeliot*" el cual traza el código fuente línea por línea teniendo una perspectiva visual del procesamiento del programa. Como resultado de este trabajo se obtiene que los programadores más experimentados tenían mayores fijaciones en el código fuente [4].

En cuanto al estudio del comportamiento visual de los programadores mediante *Eye Traking* se tiene que una de las personas más destacadas es Bonita Sharif quien junto a Jonathan Maletic realizaron una investigación sobre la convención de nombres en la programación, viendo la comprensión que tenían los programadores sobre *camelCase* y *under_score*. La importancia de este estudio es que tiene la posibilidad determinar cuál es el método más rápido en la identificación de la declaración de variables llegando a la conclusión que el estilo del identificador afecta en tiempo y esfuerzo visual de los programadores, siendo *under_score* más rápido [5].

Todas estas investigaciones han sido realizadas con el propósito de llegar a nuevas teorías sobre cómo los movimientos oculares se ven reflejados en el análisis de código fuente, esto con motivo de encontrar mejores prácticas y diferenciadores de programadores expertos y programadores novatos, en estos casos cada investigación tiene un propósito en particular.

Uno de los más recientes trabajos es dado por la *Eye Movements in Programming* donde realizó una investigación en la cual tomaron programadores con diferente nivel de experiencia a analizar código fuente a través de un *Eye Tracker* esta información fue almacenada en un conjunto de datos disponible para su análisis [6].

Con el análisis de datos se desea crear información con el principal propósito de aportar hacia la comunidad de *Eye Movements in Programming*, utilizando algoritmos para obtener conclusiones sobre el mismo conjunto de datos y generar nuevas hipótesis o teorías.

1.2 Definición del Problema

Actualmente en el campo científico existe una inmensa variedad de herramientas para recopilar datos de los usuarios llevando a cabo tareas frente a un computador. El *Eye Tracking* es una de las tecnologías que ha demostrado su importancia ya que una de sus mayores

fortalezas es la obtención de datos sin interrumpir a los usuarios que están prestando sus habilidades para la realización de los experimentos.

Si no fuera por éste, la persona encargada de la investigación debe preguntar a los sujetos sobre lo que están viendo o pensando en ciertos momentos, registrando solo el resultado final y dejando por alto datos de vital importancia, ya que los sujetos pueden simplemente olvidar algunas variables por las cuales eligieron tal respuesta o hicieron tal acción.

Una manera de solucionar esto, es pidiendo a los sujetos que tomen nota y registren cada una de sus acciones mientras están realizando el experimento, sin embargo, esto interrumpe el trabajo y le quita el protagonismo a la tarea principal.

La programación al ser una tarea que requiere concentración no puede ser interrumpida por diferentes estímulos, es por ello que los métodos tradicionales sobre obtención de datos quedan descartados, razón por la cual utilizar *Eye Traker* ayuda significativamente a la investigación ya que es un aparato silencioso y poco vistoso, es decir, no capta la atención de los programadores y esto ayuda a que los programadores se enfoquen solo en la tarea de programar [2].

A partir del año 2013 se han realizado *workshop* sobre *Eye Tracking* con enfoque al desarrollo de software, en los cuales participan todo tipo de programadores, desde los más experimentados (*Senior*) hasta los que van iniciando (*Junior*), esto implica una alta recolección de datos y con ello, se puede encontrar diferentes características que los programadores ejercen a través del movimiento de sus ojos [6].

La importancia de los movimientos oculares en la programación se ha vuelto innegable ya que se ha demostrado su veracidad científicamente, el número de artículos académicos sigue en crecimiento consolidando nuevas teorías, sin embargo, este es un área que aún sigue en aumento.

Personas de alto renombre como Bonita Sharif, Norman Peitek y Marjaana Puurtinen, entre otros, se encargan de organizar *workshops* con un designio exploratorio y científico, con diferentes trabajos enfocados a la programación [5].

Con el fin de aportar a esta comunidad, se pretende hacer un análisis de datos sobre los movimientos oculares en la programación utilizando un conjunto de datos por parte de *Eye Movements in Programming* en búsqueda de estilos, estrategias o patrones que realizan programadores experimentados y los novatos, por consiguiente, descubrir nuevas teorías y afirmar las ya existentes para un correcto aprendizaje de la programación y saber el por qué algunas personas son más diestras al momento de programar.

1.3 Objetivos

A continuación, se enuncian los objetivos de este proyecto.

Objetivo General: Analizar y explicar el procesamiento visual de programadores frente a código fuente para generar conocimiento sobre los movimientos oculares en tareas de programación.

Objetivos Específicos: Los objetivos específicos nos llevan a completar el general, teniendo un total de cuatro.

1. Seleccionar un conjunto de datos validado sobre movimientos oculares en tareas de programación para su análisis.
2. Identificar las secciones del código en las cuales los programadores tienen mayor atención visual.
3. Deducir los factores por los cuales los programadores fijan su atención visual a ciertas secciones de código.
4. Considerar y aplicar técnicas de *Machine Learning* para caracterizar a programadores en función de su procesamiento visual de código fuente.

1.4 Hipótesis

El comportamiento visual de los programadores frente al código fuente se relaciona con el nivel de experiencia que tienen sobre el lenguaje de programación, los novatos tienen una mayor cantidad de fijaciones que los programadores expertos.

1.5 Justificación

La obtención de datos sobre los movimientos oculares en la programación se ve reflejada en la considerable cantidad de estudios que han surgido, aproximadamente desde que se puede hacer captura de los movimientos oculares frente al computador. Por ejemplo, en "*How do we Read Algorithms?*" de Martha E. Crosby y Jan Stelovsky con fecha de enero de 1990 se captura el movimiento de los ojos en programación sobre el lenguaje Pascal. También se nos dice que el monitoreo de los ojos surge para explorar la atención de los individuos en cuestión [7].

Desde entonces, en el análisis de la programación se sigue estudiando y verificando ideas y teorías sobre los movimientos oculares para llegar a conclusiones. *Eye Movements in Programming* lleva a cabo diferentes experimentos en los cuales ponen a personas de distintas etnias a examinar código.

Son datos limpios y bien estructurados, dicho de otra manera, no contiene datos crudos y se puede utilizar para este análisis, esto es debido a que recolectar datos es una tarea que consume tiempo el cual puede ser empleado en el análisis de datos ya que el propósito no es recolectarlos, sino analizarlos.

Para la generación del conjunto de datos, se le presentaron dos códigos a cada programador en los cuáles ellos debían comprender el algoritmo, esto fue hecho sin ninguna restricción de tiempo y al final de la prueba responder unas preguntas para validar su compresión sobre el código, antes de esto, se realiza unas preguntas a los participantes, como nivel de inglés, lenguajes de programación, etcétera [6].

Se plantea que una de las preguntas que tiene relación con las fijaciones que los programadores hacen, es el nivel de experiencia en el lenguaje en el cual se desarrolla la prueba, ya que los novatos tienden a mirar más el código y repetir sus fijaciones.

Con el análisis de este conjunto de datos, se pretende realizar un aporte hacia la enseñanza de la programación, como ya se sabe, es *Eye Movements in Programming* quien se encarga de dar acceso público a este conjunto de datos para que sea la comunidad misma quien siga aportando a la investigación.

Por ende, podría ser de gran ayuda para el sector educativo, encontrando así las diferencias del procesamiento visual entre los programadores expertos, intermedios y los novatos especialmente para aquellos que van iniciando con algún lenguaje de programación, estos diferenciadores pueden a ayudar a los novatos a comprender en qué elementos es más factible detenerse a observar y analizar y en cuáles se deben tomar más a la ligera, para así enfocarse en lo primordial y lograr una comprensión sobre el código más efectiva.

Con esto, las debilidades de los programadores pueden ser disminuidas, permitiendo explorar nuevas maneras en las que se instruye o enseña programación y comenzar nuevos cuestionamientos sobre si la manera típica de enseñanza sobre esta rama es la correcta y cuáles son las otras alternativas para aprender.

Por consiguiente, generar nuevos conocimientos sobre los métodos de análisis de código es de gran ayuda para el campo científico y para aquellas personas que se dedican a la programación, de esta manera se genera información que puede ser utilizada más adelante en proyectos futuros.

Capítulo 2

Marco Referencial

Se enlistan el marco teórico y tecnológico, necesarios para la comprensión de la terminología utilizada a lo largo de la investigación y aquellas herramientas necesarias para el análisis.

2.1 Marco teórico

En esta sección se da contexto al proyecto mediante la definición de conceptos y áreas de vital importancia para el correcto entendimiento de esta investigación. Se divide en *Eye Tracking*, aprendizaje automático y técnicas de validación.

Eye Tracking

Por su traducción literal, el *Eye Tracking* "registro o seguimiento visual" es la obtención y captura de los movimientos oculares de los usuarios realizando una acción. Para ello se utilizan aparatos tecnológicos con este propósito conocidos como *Eye Trackers*, existiendo distintos aparatos con diferente arquitectura [8].

Definido por Yusef Hassan y Víctor Herrero [9] como el "Conjunto de tecnologías que permiten monitorizar y registrar la forma en la que una persona mira una determinada escena o imagen, en concreto en qué áreas fija su atención, durante cuánto tiempo y qué orden sigue en su exploración visual", dando especial importancia a los aparatos tecnológicos necesarios para la recopilación de dichos movimientos dados por los usuarios.

Mediante estas descripciones, se puede apreciar que una de las ventajas de realizar *Eye Tracking* es la obtención de datos sobre los movimientos oculares de los usuarios, que por lo regular son almacenados en archivos los cuales requieren una lectura y análisis para posteriormente poder interpretar los resultados. Con gran frecuencia, estos archivos se pueden abrir mediante programas esenciales con los que cualquier ordenador cuenta, sin embargo,

para realizar un buen análisis de los datos se requiere software orientado a ello.

Eye Trackers

Como se ha mencionado anteriormente, los aparatos llamados *Eye Trackers* son los que recopilan toda la información ocular de los usuarios existiendo gran variedad de estos, por lo que se pueden adecuar según las necesidades de los distintos proyectos de investigación sin tener que hacer gastos que excedan lo necesario.

Por mencionar algunos ejemplos, se tienen las siguientes categorías:

Potenciales eléctricos

En este método se utilizan potenciales eléctricos, consiste en colocar electrodos a manera de parches alrededor de los ojos, calculando la diferencia entre la retina y la córnea, es un procedimiento tedioso que debe tener especial cuidado al momento de colocar el equipo [10].

Sensado invasivo

Este tipo de *Eye Tracker* se incorpora de una manera muy similar a un lente de contacto, es la razón por la que se llama invasiva ya que puede llegar a incomodar al usuario, brindando una experiencia insatisfactoria, a pesar de brindar una precisión casi exacta es poco utilizado y se encuentran pocos trabajos con este tipo [11].

Sensado no invasivo

A diferencia de los otros tipos, este es el más agradable al usuario, ya que de cierta manera no interactúa directamente con él y no interrumpe las actividades necesarias para el análisis dado que no existe un contacto directo, esta es la razón que debe su éxito y es de los dispositivos más utilizados y del que más versiones existen. Este tipo de *Eye Tracker* utilizan una luz infrarroja reflejada en los ojos para captar su movimiento tomando la pupila como eje principal, teniendo como único defecto los factores externos como la iluminación [12]. La figura 2.1 es un ejemplo de un *Eye Tracker* no invasivo [13].

Figura 2.1: SMI RED250 mobile

Existen dos tipos de movimientos oculares, las siguientes definiciones son dadas por “”Eye Tracking in Computing Education [1]:

Fijaciones: Se refiere al tiempo en que la mirada se centra en un objeto o parte especifica de la pantalla durante un corto periodo de tiempo.

Sacadas: Son los movimientos rápidos de los ojos de un punto a otro, esto puede ser representado como líneas que se encuentran entre las fijaciones. Además de esto, las fijaciones

cuentan con un camino ordenado o rutas de exploración que los usuarios hacen a través de las sacadas.

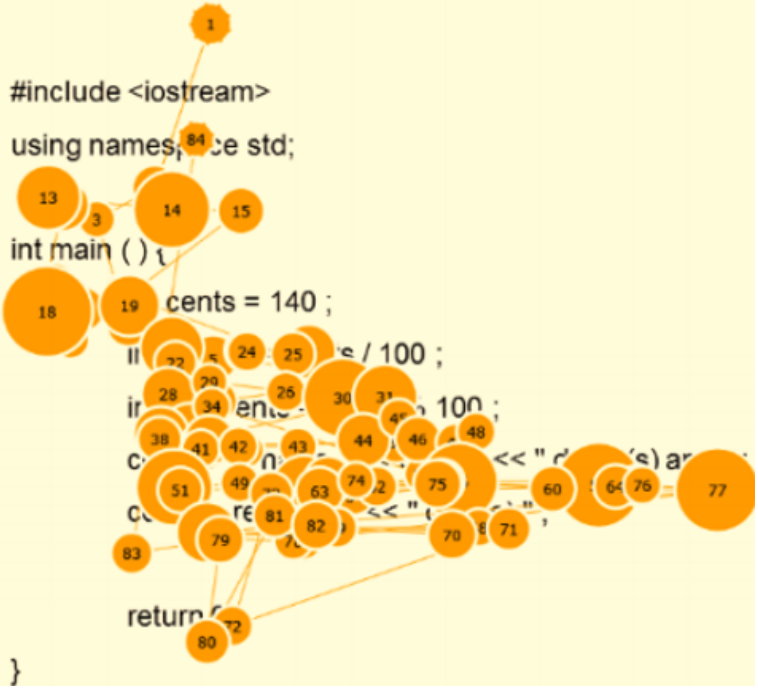


Figura 2.2: Fijaciones y sacadas

Como se puede apreciar en la figura 2.2 [14], los círculos de color naranja corresponden a las fijaciones dadas por los usuarios, es decir, es cuando el usuario enfocó su vista durante un lapso de tiempo, entre mayor sea la extensión del círculo mayor es la fijación, dentro de los círculos se tiene un número que simbolizan el orden de las fijaciones, mientras que las líneas color naranja son las sacadas, pequeños caminos que se encuentran entre cada fijación.

Métricas de Eye Tracking

Con respecto a lo anterior se puede deducir una necesidad de establecer un estándar en las métricas sobre *Eye Tracking*, dado que el área es grande se generan estudios para conocer los procesos cognitivos además de los esfuerzos requeridos por los usuarios al momento de realizar tareas que involucren Ingeniería de Software. Es preciso e importante llamar de la misma manera a todas las métricas que se utilizan, esto es porque distintos autores tienden a nombrar de manera distinta a las métricas establecidas o incluso dar a entender que no son métricas las que se evalúan al momento de realizar investigaciones, por consiguiente, Zohreh Sharafi y otros autores [15] organizaron las métricas de la siguiente manera:

Métricas sobre fijaciones

Conteo de fijaciones: El número total de fijaciones en cada área de interés, AOI por sus siglas en inglés *Area of Interest*.

Tasa de fijaciones: Se calcula dividiendo el área de la mirada abreviado comúnmente como AOG que significa *Area of Glance*, entre el número total de fijaciones sobre las áreas de interés (AOI), mientras que FR es *Fixation Rate*

$$E = \frac{Nmerototaldefijacionesen AOI}{Nmerototaldefijacionesen AOG} \tag{2.1}$$

Densidad espacial de las fijaciones: Denominado como *Fixation Spatial Density* y abreviado como SD, es igual al número de celdas que contienen al menos una fijación, dividido por el número total de celdas y está dado por la siguiente ecuación:

$$SD = \frac{\sum_{i=1}^{n} C_i}{n} \tag{2.2}$$

Donde n es el número de celdas en la cuadrícula, C_i es igual a 1 si el número de celda i es visitado, de otra manera se representa como 0.

Área del casco convexo: El área del conjunto convexo más pequeño de fijaciones que contiene todas las fijaciones de un participante para mostrar las partes preferidas del estímulo visual.

Promedio de la duración de las fijaciones: La suma de todas las fijaciones entre el número de fijaciones. En inglés *Average Fixation Duration* (AFD).

$$AFD(AOI) = \frac{\sum_{i=1}^{n}(ET(F_i) - ST(F_i))enAOI}{n} \tag{2.3}$$

Donde $ET(F_i)$ y $ST(F_i)$ son el tiempo final e inicial para la fijación (F_i) y n es el número total de fijaciones dadas por el área de interés.

Relación de ON-Target: Comúnmente mencionado como *Ratio of ON-target:All-target Fixation Time (ROAFT)* es la suma de las duraciones de todas las fijaciones en un área de interés, dividida por la duración total de todas las fijaciones para el área de la mirada (AOG).

$$ROAFT = \frac{\sum_{i=1}^{n}(ET(F_i) - ST(F_i))enAOI}{\sum_{j=1}^{n}(ET(F_j) - ST(F_j))enAOG} \tag{2.4}$$

Donde $ET(F_i)$ y $ST(F_i)$ son el tiempo final e inicial para la fijación (F_i) y n es el número total de fijaciones dadas por el área de interés, dividido por esto mismo, pero en áreas de mirada.

Tiempo de fijación: Es la suma de las duraciones de todas las fijaciones en un área de interés (AOI).

Promedio de duración de fijaciones relevantes: Se define como el total de la duración de las fijaciones relevantes, el cual es el resultado de la división de la duración de las fijaciones relevantes en áreas de interés entre el número total de áreas de interés relevantes. En inglés se llama Average Duration of Relevant Fixations, abreviado comúnmente como ADRF y es dado por la siguiente ecuación:

$$ADRF = \frac{Duracindefijacionesrelevantesen AOIs}{NmerototaldeAOIsrelevantes} \tag{2.5}$$

Taza normalizada de fijaciones relevantes: Permite comparar dos o más estímulos juntos, dicho en inglés *Normalised Rate of Relevant Fixations* abreviado como NRRF.

$$ADRF = \frac{ADRF}{\frac{Duracindefijacionesentodoslos AOIs}{NmerodetodoslosAOIs}} \tag{2.6}$$

Donde ADRF es el promedio de duración de fijaciones relevantes dado por la ecuación 2.5, dividido entre el número de fijaciones de todas las áreas de interés sobre el número total de las áreas de interés.

Métricas basadas en sacadas

Duración de sacadas: El número total de sacadas en el área de interés. Tasa de regresiones: Indica el porcentaje de retroceso de sacadas de cualquier longitud. Los buenos lectores tienen menos regresiones.

Métricas basadas en las rutas de exploración

Frecuencia de cambios: Mide la atención visual utilizando el número de cambios entre un área de interés por minuto.

Matriz de transición: *Transitional Matrix* (TM) es una representación tabular entre las transiciones de las áreas de interés, está dado por la siguiente ecuación.

$$TM = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} C_{i,j}}{n^2} \tag{2.7}$$

Donde n es el número total de fijaciones en áreas de interés (para una celda) mientras que C_i es igual a 1 si el área de interés es visitada, de otra manera es igual a 0. Para comparar 2 matrices de transición, la densidad de una matriz de transición es el número de celdas de matrices distintas de cero dividido por el número total de celdas.

Editar distancia: Utiliza el algoritmo de Levenshtein el cual calcula el costo mínimo de

edición para transformar una cadena en otra utilizando 3 operaciones, esto utilizando únicamente la locación de las fijaciones.

Minería secuencial de patrones *(SPAM)*: Usa un algoritmo de profundidad para extraer y comparar las rutas de exploración considerando las ubicaciones de las fijaciones y la duración.

ScanMatch: Compara las rutas de exploración basadas en el algoritmo de Needleman-Wunsch, esta métrica calcula la similitud de dos rutas de exploración.

Linealidad: Determina la estrategia de búsqueda de un estímulo, representando que tan cerca los lectores leen el código de manera natural a un texto.

Tamaño de la pupila y frecuencia de parpadeo

Pocas tasas de parpadeo indican mayor carga de atención, mientras que las tasas más altas se asocian con la fatiga.

Eye Tracking en Ingeniería de Software

La investigación en *Eye Tracking* cuenta con diferentes especializaciones como por ejemplo la publicidad [16], medicina [17], en la psicología [18] e incluso en el comercio electrónico [19] además de contar con un enfoque a la programación, el cual es motivo de este proyecto de titulación. La programación es la manera por la cual el ser humano da instrucciones a un computador para realizar determinada acción, puesto que existe una gran cantidad de lenguajes de programación, los estilos varían dependiendo del lenguaje, el paradigma y por supuesto, el desarrollador en cuestión.

Dado que la programación tiene un crecimiento considerable se buscan nuevos métodos para un mejor aprendizaje, está razón es de gran peso para *Eye Movements in Programming*, dado que ellos realizan talleres denominados *workshop* con la intención de recopilar datos de programadores con distinto nivel de conocimiento sobre el lenguaje a analizar código, esto con el propósito de ver particularidades sobre su visión con respecto al código y poder llegar a distintas conclusiones, teorías, hipótesis, etcétera y plantear nuevas preguntas que ayuden al área enfocada a la enseñanza de la programación.

Con la información recabada, *Eye Movements in Programming* genera conjuntos de datos con la estructura necesaria para que sean utilizados por la misma comunidad para realizar análisis de datos y generar información con los análisis realizados.

Aprendizaje automático

Definido por Mitchel [20] como el desarrollo de técnicas computacionales sobre el aprendizaje y construcción de sistemas capaces de adquirir conocimientos automáticamente, el cual viene a ser un subcampo de las ciencias de la computación, son algoritmos que otorgan a los sistemas la capacidad de aprender. Se coloca como un tema muy popular en la actualidad dado que todas las grandes empresas utilizan algoritmos de aprendizaje automático para conocer las preferencias de sus usuarios y generar más ingresos con esos conocimientos, sin

embargo y como se puede observar en lo citado anteriormente no es un área nueva del todo ya que tiene sus inicios en los 50's y desde entonces se han logrado avances significativos en el área.

Dicho de otra manera, *"Machine Learning"* como se conoce popularmente, trata de hacer que las máquinas aprendan por sí mismas mediante algoritmos previamente programados e información recopilada con anterioridad.

Según [21] se puede llevar el aprendizaje automático de dos maneras:

Aprendizaje automático supervisado: Donde hay entradas de datos con sus respectivas salidas se presentará a los datos un algoritmo utilizando los datos de entrenamiento, después de su procesamiento, la salida obtenida deberá ser los datos esperados obtenidos con anterioridad. Como lo dice su nombre, se tiene un control sobre los datos así que desde un principio se sabe con precisión los resultados que deberá arrojar el algoritmo.

Aprendizaje automático no supervisado: El conjunto de datos no se encuentra etiquetado, es decir, no se sabe con exactitud cuáles son los conjuntos de prueba y se debe agrupar según características similares de los datos para poder entrenar los datos correctamente y esperar los resultados según las propiedades dadas con anterioridad.

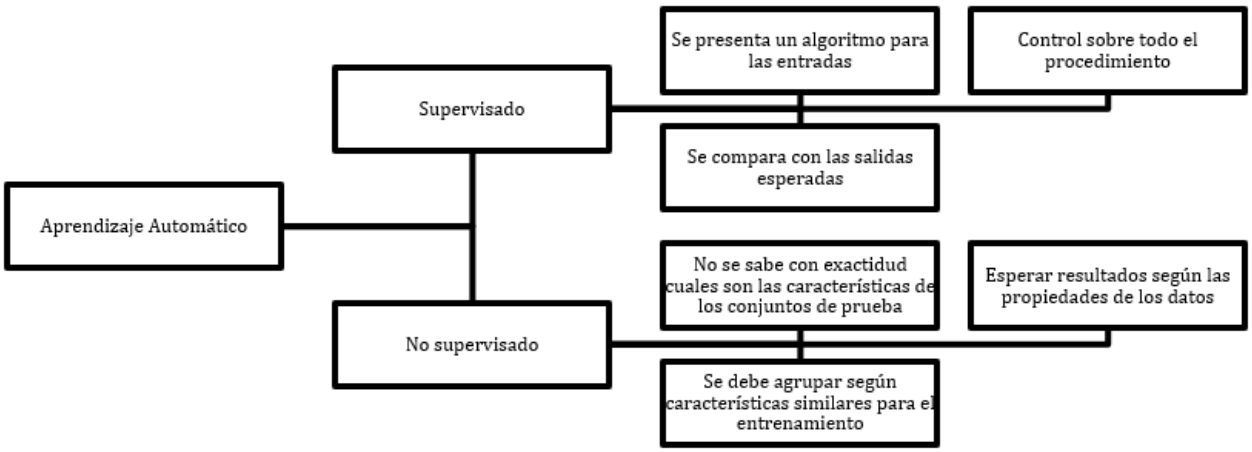


Figura 2.3: Aprendizaje autómatico

Técnicas de aprendizaje automático supervisado

Partiendo de este conjunto de datos se pretende probar un algoritmo capaz de averiguar características de los demás sujetos de prueba en el mismo conjunto de datos, teniendo en cuenta solo algunos de los atributos del individuo descartando los demás.

Con esto en mente se pueden utilizar los siguientes algoritmos [22]:

Árboles de decisión: Los árboles de decisión utilizan una estrategia de formación de

divide y vencerás. Para definir un nodo se selecciona la característica más rica en información, después de eso se sigue una división univariante, es decir, una característica única de los datos con respecto a una constante predeterminada para la característica seleccionada. Cada instancia se divide de manera iterativa y el procedimiento se repite para cada nodo hijo hasta que se asignan todas las instancias. Los nodos terminales se llaman nodos hojas y es lo que sugiere la clasificación de todas las instancias que lo alcanzan.

Clasificación de Naïve Bayes: Es un método sencillo, que utiliza métodos de distribución de probabilidad estándar para aprender las frecuencias relativas de diferentes clases y valores de características en los datos de entrenamiento para estimar la probabilidad de la clase y la distribución de probabilidad condicional de una clase dados los valores de la característica.

Técnicas de validación

Después de seleccionar los métodos de predicción adecuados, se pueden utilizar diferentes estrategias de validación, por ejemplo, cruzada para hacer selección de un método de clasificación empíricamente. El método de validación cruzada puede conducir a un desempeño promedio más alto que la aplicación de cualquier estrategia de clasificación única, esto reduce el riesgo de un desempeño deficiente y verifica que se cumplan las condiciones necesarias para obtener respuestas acertadas [23].

La validación cruzada permite estimar las distintas estrategias para identificar los errores. Entre los métodos más conocidos de validación cruzada se encuentran los siguientes [24]:

Validación simple Básicamente consiste en dividir el conjunto de datos en dos grandes partes, esto sin tener en cuenta qué tipos de datos se van a cada grupo ya que la partidura se realiza de manera aleatoria. La primera parte se utiliza para entrenar el modelo mientras que la segunda tiene como propósito ser evaluada.

Dejar una validación En inglés *"Leave One Out Cross-Validation"* trata sobre un método iterativo, es decir, realiza repeticiones a través de los datos, pero en cada diferente iteración, deja un solo elemento fuera, para de esta manera evaluar todos los posibles datos y con ello calcular los errores. Esto ajusta el modelo en cada nueva iteración dando un promedio de cada error calculado. El coste computacional juega un rol importante a la hora de realizar esta validación, más en aquellos conjuntos de datos de tamaños considerablemente grandes.

K-Fold Cross-Validation Al igual que el método de dejar una validación se trata de un proceso iterativo, en el cual se dividen los datos para prueba y otros para entrenar el modelo, dependiendo del análisis, estos pueden ser seleccionados de manera aleatoria y puede llegar a ser más robusto que los métodos anteriores.

2.2 Marco tecnológico

En esta sección se tiene en cuenta las herramientas necesarias para encarar los objetivos, se da una descripción sobre Python, Anaconda, Jupyter Notebook y TensorFlow.

Python

El lenguaje de programación Python tiene sus orígenes en la década de los 80, siendo el año 1989 el año de su salida y teniendo como predecesor el lenguaje llamado ACB, desde entonces se ha consolidado como uno de los lenguajes de programación más atractivos a la hora de aprender e introducirse a esta tecnología [25].

Python cuenta con algunas características un tanto destacables, las cuales son mencionadas por Ashwin Pajankar en "Introduction to Python [26] y por ello ha sido considerado para la realización de esta investigación:

Simple: Python es un lenguaje que se vuelve fácil de entender y comprender, dado a que cuenta solo con lo necesario, este minimalismo vuelve a Python tan sencillo de leer como si de un texto se tratase.

Fácil de aprender y leer: Dado a la sencillez de su sintaxis, aprender a programar en Python es una tarea sumamente fácil comparada a aprender programación con algún otro lenguaje, ya que solo incluye lo necesario para programar, es decir, programar en Python es olvidarse de librerías, problemas de declaraciones, detalles del lenguaje, para centrarse exclusivamente en el código. La facilidad de lectura juega un rol importante en la compresión del código, esto ayuda de igual manera al mantenimiento.

Código abierto: Esto significa que el código puede ser visto y editado por los programadores, teniendo acceso al código fuente sin ningún permiso especial o pago de licencias.

Lenguaje de alto nivel: En la actualidad se vería prácticamente imposible programar en lenguajes de bajo nivel, ya que trabajar con sentencias similares al lenguaje máquina quitaría mucho tiempo de desarrollo.

Portable: El código es portable, es decir, se puede utilizar entre diferentes plataformas, ahorrando esfuerzo en programar la misma lógica para distintos sistemas operativos.

Interpretado: No es necesario el pre-procesamiento que hacen los compiladores, esto maximiza la eficiencia de los programas al ser interpretados en tiempo real desde el código fuente.

Manejo de memoria: A diferencia de lenguajes de programación como C y C++, en Python no es responsabilidad del programador encargarse del manejo de memoria, esto es debido a que es el mismo interprete quien se encarga de liberar la memoria.

Librerías y soporte de la comunidad: Cuenta con una extensa variedad de librerías que día con día reciben actualizaciones por parte de la misma comunidad, esto es de gran

ayuda, ya que al tener personas que respalden los proyectos, siempre se puede encontrar soluciones a los problemas presentados.

Anaconda

Anaconda junto con Python y otras herramientas, se vuelve uno de los instrumentos perfectos para el análisis de datos, mediante anaconda se tiene un entorno dedicado a ello de una manera rápida y organizada que, combinado con algunas paqueterías que se pueden encontrar en este lenguaje de programación, brinda una de las mejores experiencias en cuanto a análisis e investigación se refiere.

Ofrece una interfaz sencilla y fácil de utilizar, tal y como se muestra en la figura 2.4:

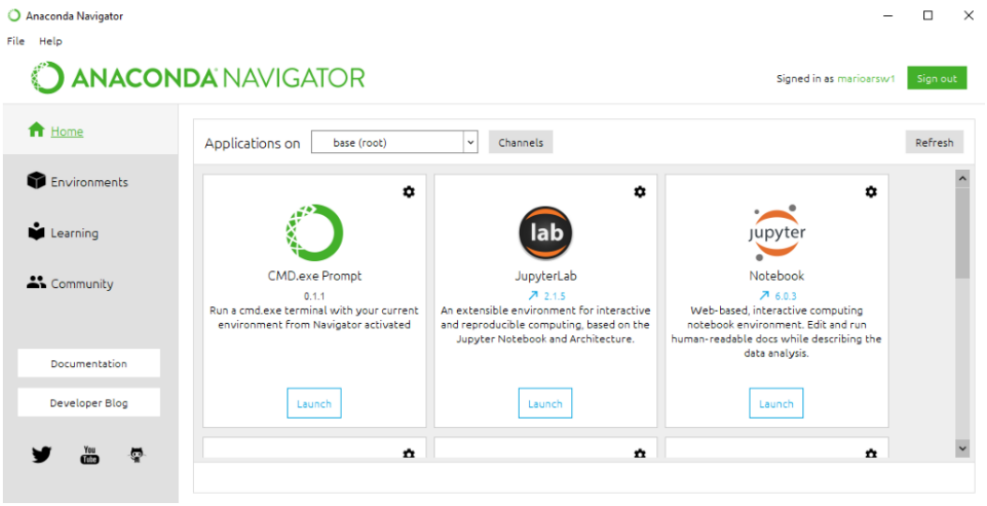


Figura 2.4: Anaconda Navigator

En su sitio web oficial se puede encontrar toda la documentación adecuada para conocer el entorno. Dentro de la misma documentación se puede encontrar información acerca de más de 7500 paqueterías, además de ser una plataforma gratuita y fácil de instalar, brinda una comunidad lista para dar soporte hacia aquellos usuarios con dudas con respecto a distintos temas. Anaconda cuenta con paqueterías por defecto, pero en caso de necesitar alguna extra o necesaria para un propósito en específico, se puede instalar más paqueterías de una manera sencilla [27].

Jupyter Notebook

Jupyter Notebook ofrece la facilidad de programar y documentar todo en un mismo cuaderno. Al ser utilizado en un navegador web, vuelve relativamente sencillo su uso, dicho de otra manera, programar con Jupyter Notebook es escribir el código en un cuaderno y estar haciendo apuntes a la vez, tal como si fuera una presentación lo cual favorece al ámbito educativo. La ejecución de código mediante simples segmentos hace más identificable los

procesos que realizan esas líneas, es decir, se tiene una línea de código, se ejecuta y es ahí mismo donde se muestra el resultado, favoreciendo la comprensión del código. En la figura 2.5 se puede observar la interfaz de Jupyter, en el cual se puede trabajar de manera sencilla con respecto a la edición de código.

Figura 2.5: Jupyter Notebook

El uso de herramientas de este tipo ayuda a mejorar la productividad de los profesores y estudiantes, dando flexibilidad y otorgando la oportunidad de apoyo colaborativo. Por ejemplo, realizar cursos donde se utilice Jupyter ayuda a agilizar todos los procesos de enseñanza, desde un mismo medio se puede enseñar programación, compartir materiales educativos mediante texto, imágenes y códigos en un solo documento a través de un navegador web [28].

TensorFlow

TensorFlow fue desarrollado como un sistema para experimentar con nuevos modelos y entrenarlos con grandes conjuntos de datos admitiendo entrenamientos a grande escala. Para ello se utilizan de manera eficiente cientos de servidores de gran potencia para ejecutar un entrenamiento rápido y modelos entrenados para inferencia en producción en varias plataformas, teniendo en cuenta desde grandes clústeres distribuidos en un centro de datos hasta los dispositivos móviles. A la vez, TensorFlow es lo suficientemente flexible para respaldar información e investigación de nuevos modelos de aprendizaje automático y optimizaciones a nivel sistema [29].

Como se puede apreciar, todas estas herramientas son las más adecuadas para la investigación y elaboración de modelos mediante técnicas de aprendizaje automático debido a sus paqueterías, su gran comunidad y documentación. Por ende, se puede decir que están preparadas de manera natural para realizar estas actividades.

Capítulo 3

Producto esperado y validación

En esta sección se ve una descripción del producto esperado, así como las delimitaciones y limitaciones formando parte importante para la comprensión del propósito general de este proyecto, además de la forma de validación.

3.1 Producto esperado

Al finalizar este proyecto, se espera generar conocimiento de manera tal, que pueda ser utilizado en proyectos futuros con una índole exploratoria. *Eye Movements in Programming* a través de sus estudios e investigaciones, genera conjuntos de datos que son accesibles a todo aquel que quiera analizarlos con la finalidad de generar aportes para la comunidad y descubrir y hacer nuevas hipótesis y preguntas de investigación capaces de acrecentar el área y mejorar los campos educativos relacionados, aumentando las oportunidades de aprendizaje en cuanto a programación se refiere.

tacos tacos *tacos*

Uno de los propósitos es realizar análisis para obtener conclusiones con respecto a los datos, además de técnicas de clasificación supervisada utilizando como lenguaje de programación Python y otras herramientas como Anaconda, Jupyter y TensorFlow. Todo esto en un estudio muy específico, llevado a cabo con 216 participantes, de cada uno de ellos se tienen los siguientes campos representados en la figura 3.1 en la que aparecen los atributos de identificador, edad, género, lengua nativa, nivel de inglés, ayuda visual, maquillaje, lenguaje utilizado para el experimento, experiencia con el lenguaje y tiempo que el programador ha estado programado en dicho lenguaje respectivamente.

Dadas los atributos de este conjunto de datos, se pretende agrupar a los sujetos de prueba según sus similitudes para aprender a identificar patrones y lograr así, una caracterización pa-

```
In [23]: dataframe = pd.read_csv('emip_metadata.csv')
         dataframe
```

Out[23]:

	id	age	gender	mother_tongue	english_level	visual_aid	makeup	experiment_language	expertise_experiment_language	time_experiment_language	...
0	1	63	male	Czech	high	glasses	no	Java	high	30.0	...
1	2	24	male	Slovak	medium	glasses	no	Java	medium	2.0	...
2	3	27	female	Hebrew	high	no	no	Java	none	0.0	...
3	4	24	male	Hindi	high	glasses	no	Java	medium	4.0	...
4	5	24	male	Italian	medium	no	no	Java	none	0.0	...

Figura 3.1: Conjunto de datos

ra cada individuo, esto dado a través del análisis de los datos. Algunos ejemplos se encuentran a continuación.

- Agrupar los datos según los años que el sujeto de prueba tiene programando en el lenguaje y buscar una relación con la experiencia que tienen sobre el lenguaje. Como resultado se espera que aquellas personas que tienen más tiempo programando, se vea reflejado con su nivel de experiencia teniendo un nivel alto, mientras que los que llevan poco tiempo tengan un nivel menor.
- Agrupar según la lengua madre y verificar si esto influye en el nivel de experiencia del programador. Ver la relación que tienen los diferentes idiomas con el nivel de entendimiento del lenguaje de programación, se espera que los hablantes nativos del inglés se les facilite más la comprensión.
- Comprobar la influencia que tiene el nivel del idioma inglés sobre el análisis de código. Se espera que las personas que hablan inglés como un segundo idioma presenten alguna dificultad o tengan un nivel menor de experiencia sobre el lenguaje, esto es debido a que una gran mayoría de los lenguajes de programación tienen sus palabras reservadas en el idioma inglés o abreviaciones de este.

En la figura 3.2 se puede ver como agrupar a los sujetos de prueba según sus características (En este caso mayores a 50) para crear un nuevo conjunto de datos.

Como salida se debe obtener un atributo de la clase a la que pertenece, indicando las características que el sujeto de pruebas debe tener en base a las características ingresadas.

```
In [42]: mayores_50 = dataframe['age'] >= 50
         dataframe[mayores_50]
```

Out[42]:

	id	age	gender	mother_tongue	english_level	visual_aid	makeup	experiment_language	expertise_experiment_language	time_experiment_language
0	1	63	male	Czech	high	glasses	no	Java	high	30.0
38	39	68	female	Spanish	medium	glasses	no	Java	low	1.0
122	123	62	male	Finnish	medium	glasses	no	Java	high	19.0
135	136	55	male	English	high	no	no	Java	medium	5.0

Figura 3.2: Clasificación

3.2 Delimitaciones y limitaciones

En este apartado se muestran los alcances y limitaciones del proyecto

Delimitaciones

- El análisis de datos solo contempla un conjunto de datos proporcionado por el grupo *Eye Movements in programming.*

Limitaciones

- El conjunto de datos contiene a 216 participantes donde 41 personas son mujeres y 175 hombres.
- Los lenguajes de programación fueron elegidos por los participantes.
- Como lenguajes de prueba, se utilizaron Java, Python y Scala.

En la sección de apéndices, se puede visualizar parte de los códigos utilizados para la obtención de los datos.

3.3 Forma de validación

Después de un análisis de datos se debe validar los resultados, siendo una de las etapas más importantes para los resultados. Es de vital importancia para decidir si la precisión del modelo es la adecuada o se deben hacer ajustes para mejorar el rendimiento, esto con

propósito de determinar la robustez de este ya que debe ser capaz de contemplar los diferentes tipos de datos y todas las posibles vías que puede tomar para lograr las salidas correctas y llegar a una validación esperada (En caso de ser aprendizaje supervisado).

K-Fold Cross-Validation

Al igual que el método de dejar una validación se trata de un proceso iterativo, la diferencia es que aquí se divide todos los datos de manera aleatoria en diferentes grupos y todos los grupos se utilizan para entrenar el modelo excepto uno que se utiliza como prueba, el proceso se repite, pero en cada iteración cada grupo de prueba es distinto, como se puede ver en la tabla 3.1, se toman datos para la prueba mientras que se dejan otros para ser de entrenamiento, en este caso se hará mediante 3 iteraciones dividiendo el conjunto de datos para realizar esta validación.

Estimación 1	Prueba	Entrenamiento	Entrenamiento
Estimación 2	Entrenamiento	Prueba	Entrenamiento
Estimación 3	Entrenamiento	Entrenamiento	Prueba

Tabla 3.1: Ejemplo de validación

3.4 Metodología de desarrollo

La metodología elegida para este proyecto de titulación es CRISP-DM dado su versatilidad y eficacia, siendo fácil de utilizar para usuarios nuevos, ofrece tan solo 6 fases, con la oportunidad de realizar regresiones entre ellas, es decir, estar en evaluación y regresar a la comprensión del negocio y volver a partir de los siguientes puntos.

Esta metodología surge para aquellos proyectos enfocados a la minería de datos en el año de 1996 y hasta esta fecha, se ha consolidado como una de las más importantes en cuanto a minería se refiere, junto con KDD y SEMMA.

Las fases se listan como sigue [30]:

1. Comprensión del negocio: Siendo la fase inicial, tiene un enfoque en la comprensión de los objetivos del proyecto y requerimientos desde la perspectiva del negocio, después de esto, convertir este conocimiento hacia una definición de minería de datos con un plan para lograr los objetivos.

2. Comprensión de los datos: Implica la recopilación inicial de los datos, continuando con una serie de actividades necesarias para lograr una familiarización con los datos, además de identificar los problemas de calidad de los datos, detectando subconjuntos interesantes para formar hipótesis sobre la información.

3. Preparación de los datos: Esta fase cubre todas las actividades necesarias para crear el conjunto de datos final, a partir de los datos en bruto iniciales. En este caso, el conjunto de datos final ya se ha obtenido, por esta razón, de primera instancia esta fase es omitida.

4. Modelado: En esta fase se debe seleccionar y aplicar distintas técnicas de modelado correspondiente al análisis de datos. Como se ha dicho, se puede aplicar una o más técnicas para el análisis.

5. Evaluación: El modelo o modelos obtenidos se evalúan más a fondo y se revisan los pasos ejecutados para construir el modelo y asegurarse que se logran adecuadamente los objetivos comerciales. En este proyecto se utilizará validación cruzada para evaluar el modelo de análisis de datos.

6. Despliegue: Dependiendo del tipo de proyecto, el despliegue puede no ser el final. El conocimiento generado debe organizarse y presentarse de manera que los interesados en el proyecto sean capaces de utilizarlo.

La figura 3.3 muestra el ciclo de CRISP-DM [31]:

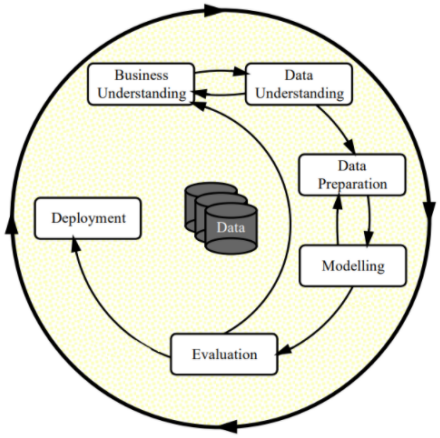


Figura 3.3: CRISP-DM

Cronograma

A continuación, se presenta la siguiente imagen, el cual es el cronograma de actividades a realizar durante Seminario de Tesis 2. (ESTO SOLO ES UNA PRUEBA)

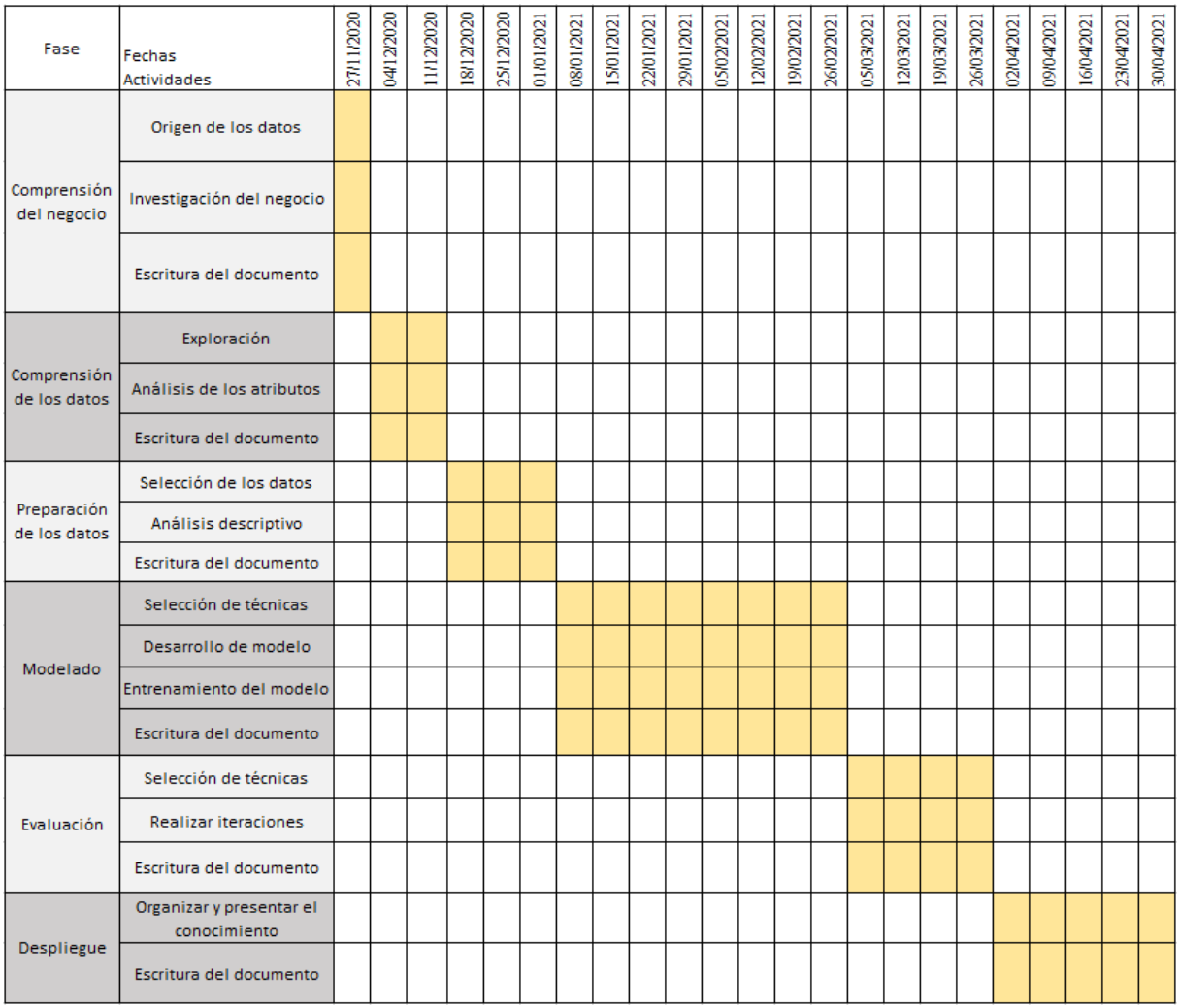


Figura 3.4: Cronograma

3.5 Desarrollo del proyecto

Al realizar un análisis de datos, una de las metodologías que mejor se adaptan a este proyecto de investigación, se va a utilizar CRISP-DM, teniendo como consiguiente las etapas dadas a continuación.

Comprensión del negocio

La recolección de datos en el área de programación por parte de *Eye Movements in Programming* abre la posibilidad de realizar investigaciones e indagar más sobre las posibilidades que se pueden encontrar a través de estos datos, por ello, en este apartado, se indica los objetivos de este análisis.

1. Seleccionar un conjunto de datos validado sobre movimientos oculares en tareas de programación para su análisis.
2. Identificar las secciones del código en las cuales los programadores tienen mayor atención visual.
3. Deducir los factores por los cuales los programadores fijan su atención visual a ciertas secciones del código.
4. Considerar y aplicar técnicas de *Machine Learning* para caracterizar a programadores en función de su procesamiento visual de código fuente.

Para alcanzar estos objetivos, se deben tomar el conjunto de datos para comprender su naturaleza, para realizar esto se debe encontrar software particular con el cual se pueda examinar los datos generados por el *Eye Tracker*, en primera instancia, se tiene en mente *OGAMA* el cual es un programa de código abierto, es decir, está disponible para todo el público, siendo capaces de visualizar el código del mismo y realizar modificaciones del mismo.

Entendimiento de los datos

Esta fase inicia principalmente con la recolección de los datos.

Como ya se ha dicho en anteriores ocasiones, estos datos han sido recolectados por [6], siendo un conjunto de datos relativamente nuevo, el cual lanzado públicamente en el año 2020.

En esta etapa, es crucial que el analista se familiarice con los datos, es aquí donde se encuentran ideas más allá de lo que se tiene planeado en un principio y ayuda a encontrar las distintas posibilidades y dar una limitación correcta hacia el alcance del proyecto, dicho de otra manera, el entendimiento de los datos ayuda a detectar las distintas preguntas de investigación o hipótesis según sea el caso.

Los datos fueron recolectados de 216 participantes, claramente se realizó el experimento con más sujetos de prueba, sin embargo, algunos se tuvieron que quitar de la muestra final, para tener un conjunto de datos coherente, cada uno de los participantes contaba con distinto nivel de experiencia, el experimento fue dado durante dos códigos para medir su habilidad de comprensión. La recopilación de datos fueron a través de una pantalla con una resolución de 1920x1080, este es un dato de gran importancia para poder visualizar correctamente los datos obtenidos conforme a los códigos presentados hacia los usuarios. En esta muestra, 41 personas son mujeres mientras que 175 son hombres. La tabla 3.2 ofrece una descripción de los datos necesarios para la comprensión de el público al cual fue orientado el experimento, en esta tabla se pueden observar algunas particularidades.

De manera más específica, el conjunto de datos se compone de los siguientes 4 apartados:

1. date.txt - Se trata de un archivo que especifica la fecha en la que se subió el conjunto de datos.

2. emip_metadata.csv - Brinda información general de los participantes, además de información sobre las preguntas de comprensión.

3. rawdata - Es una carpeta que contiene los datos de los movimientos oculares sin procesar.

4. stimuli - Es una carpeta que contiene las capturas de pantalla de los experimentos, además de las coordinadas de las áreas de interés sobre los programas.

Tabla 3.2: Descripción de los datos.

Variable	Descripción	Valor
id	Identificador único, se refiere al archivo de datos de la mirada sin procesar	[n]
age	Edad	[años]
gender	Género	[masculino, femenino]
mother_tonge	Lengua madre	[texto-completo]
english_level	Dominio del inglés	[bajo, medio, alto]
visual_aid	El participante utiliza lentes o lentes de contacto	[no, lentes, lentes de contacto]
makeup	El participante utiliza rímel u otro maquillaje en los ojos	[sí, no]

experiment_language	Lenguaje de programación usado en el experimento	[Java, Python, Scala]
expertise_experiment_lg	Experiencia en Java/Python/Scala	[ninguna, bajo, medio, alto]
time_experiment_lg	Cuánto tiempo ha estado programando el participante en Java/Python/Scala	[años]
frequency_experiment_lg	Con qué frecuencia ha estado programando el participante en Java/Python/Scala	[nada, menos de 1h/m, menos de 1h/s, menos de 1h/d, más de 1h/d]
other_languages	Otros lenguajes de programación que el participante conoce	[nivel de experiencia en el lenguaje]
expertise_programming	Experiencia general en programación	[ninguno, bajo, medio, alto]
time_programming	Cuánto tiempo lleva programando el participante	[años]
frequency_other_lg	Con qué frecuencia el participante usa lenguajes de programación distintos de Java / Python / Scala	[nada, menos de 1h/m, menos de 1h/s, menos de 1h/d, más de 1h/d
Para cada estímulo del programa:		
answer [rectangle-vehicle]	Responder a la pregunta de comprensión	[texto-completo]
correct [rectangle—vehicle]	Evaluación de la respuesta	[0,1]
order [rectangle—vehicle]	Orden en que se mostraron los programas de estímulo	[1,2]
stimulus [rectangle—vehicle]	Nombre de archivo de la captura de pantalla en la carpeta “estímulos”	[texto-completo]

mother tongue-time experiment language— time programming— other languages original	Entradas de participantes sin editar	[texto-completo]

Como se puede apreciar en la tabla 3.2 el conjunto de datos que representa trata acerca de los datos principales de los 216 participantes, como lo puede ser la edad, idiomas que hablan, lenguajes en los que programan y algunas otras interesantes que pueden influir en el desarrollo de la programación.

Para cada uno de los 216 participantes, se generó un conjunto de datos que corresponde a cada código, es decir, el código del vehículo y el código del rectángulo. A diferencia del conjunto de datos que brinda las características generales de los participantes, estos datos vienen en formato TSV lo cuál significa que los valores están separados por tabulaciones (BUSCAR UNA REF).

Para comprender el segundo conjunto de datos se tiene la siguiente lista, la cual explica cada uno de los campos y su significado.

- Time: timestamp of the sample
- Type: Sample (SMP) | Message (MSG)
- L Raw X [px]: horizontal pupil position – left eye
- L Raw Y [px]: vertical pupil position – left eye
- R Raw X [px]: horizontal pupil position – right eye
- R Raw Y [px]: vertical pupil position – right eye
- L Dia X [px]: pupil diameter on the X axis in pixels – left eye
- L Dia Y [px]: pupil diameter on the Y axis in pixels – left eye
- L Pupil Diameter [mm]: pupil diameter in mm – left eye
- R Dia X [px]: pupil diameter X axis in pixels – right eye
- R Dia Y [px]: pupil diameter Y axis in pixels – right eye
- R Pupil Diameter [mm]: pupil diameter in mm – right eye
- L CR1 X [px]: horizontal corneal reflex position. One or two CRs can be present – left eye

- L CR1 Y [px]: vertical corneal reflex position. One or two CRs can be present – left eye
- L CR2 X [px]: left eye horizontal corneal reflex position. One or two CRs can be present
- L CR2 Y [px]: left eye vertical corneal reflex position. One or two CRs can be present
- R CR1 X [px]: horizontal corneal reflex position. One or two CRs can be present – right eye
- R CR1 Y [px]: vertical corneal reflex position. One or two CRs can be present – right eye
- R CR2 X [px] : horizontal corneal reflex position. One or two CRs can be present – right eye
- R CR2 Y [px] : vertical corneal reflex position. One or two CRs can be present – right eye
- L POR X [px]: point-of regard in the X axis in pixels – left eye
- L POR Y [px]: point-of regard in the Y axis in pixels – left eye
- R POR X [px]: point-of regard in the X axis in pixels – right eye
- R POR Y [px]: point-of regard in the Y axis in pixels – right eye
- Timing: Quality values
- L Validity: validity – left eye
- R Validity: validity – right eye
- Pupil Confidence
- L Plane: plane number – left eye
- R Plane: plane number – right eye
- L EPOS X: left pupil position from the perspective of the subjective camera, in the X axis in pixels
- L EPOS Y: left pupil position from the perspective of the subjective camera, in the Y axis in pixels
- L EPOS Z: left pupil position from the perspective of the subjective camera, in the Z axis in pixels

- R EPOS X: right pupil position from the perspective of the subjective camera, in the X axis in pixels
- R EPOS Y: right pupil position from the perspective of the subjective camera, in the Y axis in pixels
- R EPOS Z: right pupil position from the perspective of the subjective camera, in the Z axis in pixels
- L GVEC X: left eye gaze vector, from the perspective of the left eye camera, X axis component
- L GVEC Y: left eye gaze vector, from the perspective of the left eye camera, Y axis component
- L GVEC Z: left eye gaze vector, from the perspective of the left eye camera, Z axis component
- R GVEC X: right eye gaze vector, from the perspective of the left eye camera, X axis component
- R GVEC Y: right eye gaze vector, from the perspective of the left eye camera, Y axis component
- R GVEC Z: right eye gaze vector, from the perspective of the left eye camera, Z axis component

Capítulo 4

Resultados y Discusiones

[Sustituye este texto con tu párrafo introductorio. Este capítulo lo puedes dividir como sigue.]

4.1 Resultados

[Los resultados son declaraciones acerca de las observaciones, incluyendo estadísticas, tablas, gráficas, figuras, etc., sobre todo los que responden a los objetivos planteados. Se menciona el rango de variación de la información. Los resultados clave se narran utilizando enunciados claros y objetivos al inicio de los párrafos de esta sección.]

[No se emite un juicio pero sí se presentan suficientes detalles para que los demás pueden realizar sus propias inferencias y construir sus propias explicaciones. Se mencionan resultados positivos y negativos, no se interpretan los resultados eso se hace en las discusiones.]

[Los resultados deberán responder a la pregunta: Cuáles fueron los hallazgos? Estos se presentan tradicionalmente mediante texto, tablas e ilustraciones o figuras. Por lo general, el texto es la forma más rápida y eficiente de presentar pocos datos, las tablas son ideales para presentar datos precisos y repetitivos y las ilustraciones o figuras son ideales para presentar datos que exhiben tendencias o patrones importantes. A menudo, los resultados y la discusión se combinan en una sección de Resultados y Discusión, donde los primeros se presentan y seguidamente se discuten. Si las dos secciones están separadas, es imperativo que la primera se limite a presentar resultados y la segunda a discutirlos. Otro error común es comenzar la sección de resultados con información que pertenece a los materiales y métodos. La sección de resultados se redacta en tiempo pasado ("hemos encontrado[. . .]", "hemos observado[. . .]", etc.). Referencia: [**?**].]

4.2 Interpretaciones o Discusiones

[Las discusiones por lo regular comienzan con unos cuantos enunciados que resumen los resultados más importantes para introducirnos en la discusión. Las discusiones deben ser breves y responder a las siguiente preguntas: ¿Cuáles son los patrones más importantes que observamos? ¿Cuáles son las relaciones, tendencias y generalizaciones entre los resultados? ¿Cuáles son las excepciones o generalizaciones a esos patrones? ¿Cuáles son las causas más probables ? ¿Cuáles son las causas más probables de los patrones resultantes? ¿Hay acuerdo o desacuerdo con trabajos previos?

La discusión puede mencionar someramente los resultados antes de discutirlos, pero no debe repetirlos en detalle. No prolongues la discusión citando trabajos "relacionados" o planteando explicaciones poco probables. Ambas acciones distraen al lector y lo alejan de la discusión realmente importante. La discusión puede incluir recomendaciones y sugerencias para investigaciones futuras, tales como métodos alternos que podrían dar mejores resultados, tareas que no se hicieron y que en retrospectiva debieron hacerse, y aspectos que merecen explorarse en las próximas investigaciones. Referencia: [**?**].

Capítulo 5

Conclusiones

[Sustituye este texto. Estos son los enunciados más importantes y más fuertes que se deben realizar acerca de los resultados y discusiones. 19.5 y 19.5 Las conclusiones deben resumir el contenido y el propósito del proyecto. Se debe hacer énfasis en lo que queremos que se recuerde acerca del proyecto y realizar una síntesis de los resultados que se derivaron de los objetivos específicos trazados inicialmente (y que dieron respuesta a las preguntas de investigación si es que existen). No deben aparecer elementos nuevos o que no fueron discutidos, por ejemplo nuevos resultados observados en otros trabajos. Las conclusiones se refieren única y exclusivamente al proyecto desarrollado.

La forma más simple de presentar las conclusiones es enumerándolas consecutivamente, pero podrías optar por recapitular brevemente el contenido del artículo, mencionando someramente su propósito, los métodos principales, los datos más sobresalientes y la contribución más importante de la investigación. La sección de conclusiones no debe repetir innecesariamente el contenido del resumen. Referencia [**?** **?**].

]

Bibliografía

[1] Teresa Busjahn, Carsten Schulte, Bonita Sharif, Andrew Begel, Michael Hansen, Roman Bednarik, Paul Orlov, Petri Ihantola, Galina Shchekotova, and Maria Antropova, "Eye tracking in computing education", in *Proceedings of the tenth annual conference on International computing education research*, 2014, pp. 3–10.

[2] Joseph Tao-yi Wang, Michael Spezio, and Colin F. Camerer, "Pinocchio's pupil: using eyetracking and pupil dilation to understand truth telling and deception in sender-receiver games", *American Economic Review*, vol. 100, no. 3, pp. 984–1007, 2010, ISBN: 0002-8282.

[3] Hidetake Uwano, Masahide Nakamura, Akito Monden, and Ken-ichi Matsumoto, "Analyzing individual performance of source code review using reviewerséye movement", in *Proceedings of the 2006 symposium on Eye tracking research & applications*, 2006, pp. 133–140.

[4] Roman Bednarik and Markku Tukiainen, "An eye-tracking methodology for characterizing program comprehension processes", in *Proceedings of the 2006 symposium on Eye tracking research & applications*, 2006, pp. 125–132.

[5] Bonita Sharif and Jonathan I. Maletic, "An eye tracking study on camelcase and under_score identifier styles", in *2010 IEEE 18th International Conference on Program Comprehension*. 2010, pp. 196–205, IEEE.

[6] Roman Bednarik, Teresa Busjahn, Agostino Gibaldi, Alireza Ahadi, Maria Bielikova, Martha Crosby, Kai Essig, Fabian Fagerholm, Ahmad Jbara, and Raymond Lister, "EMIP: The eye movements in programming dataset", *Science of Computer Programming*, vol. 198, pp. 102520, 2020, ISBN: 0167-6423 Publisher: Elsevier.

[7] Martha E. Crosby and Jan Stelovsky, "How do we read algorithms? A case study", *Computer*, vol. 23, no. 1, pp. 25–35, 1990, ISBN: 0018-9162 Publisher: IEEE.

[8] Leidy Carolina Martínez Quevedo and Angie Tatiana Pinto Molina, "Estado del arte de la tecnología eye tracking en los campos de la ingeniería industrial".

[9] Yusef Hassan Montero and Víctor Herrero Solana, "Eye-tracking en interacción persona-ordenador", *No solo usabilidad*, , no. 6, 2007.

[10] David A. Robinson, "A method of measuring eye movemnent using a scieral search coil in a magnetic field", *IEEE Transactions on bio-medical electronics*, vol. 10, no. 4, pp. 137–145, 1963, ISBN: 0096-0616 Publisher: IEEE.

[11] Hewitt D. Crane and Carroll M. Steele, "Generation-V dual-Purkinje-image eyetracker", *Applied optics*, vol. 24, no. 4, pp. 527–537, 1985, ISBN: 2155-3165 Publisher: Optical Society of America.

[12] Diederick C. Niehorster and Marcus Nyström, "SMITE: A toolbox for creating Psychophysics Toolbox and PsychoPy experiments with SMI eye trackers", *Behavior Research Methods*, vol. 52, no. 1, pp. 295–304, February 2020.

[13] "Eye Movements in Programming".

[14] Bonita Sharif, "Eye Tracking in Program Eye Tracking in Program Comprehension", February 2018.

[15] Zohreh Sharafi, Timothy Shaffer, Bonita Sharif, and Yann-Gaël Guéhéneuc, "Eye-tracking metrics in software engineering", in *2015 Asia-Pacific Software Engineering Conference (APSEC)*. 2015, pp. 96–103, IEEE.

[16] Pierre Chandon, J. Hutchinson, Eric Bradlow, and Scott H. Young, "Measuring the value of point-of-purchase marketing with commercial eye-tracking data", *INSEAD Business School Research Paper*, , no. 2007/22, 2006.

[17] Katarzyna Harezlak and Pawel Kasprowski, "Application of eye tracking in medicine: A survey, research issues and challenges", *Computerized Medical Imaging and Graphics*, vol. 65, pp. 176–190, 2018, ISBN: 0895-6111 Publisher: Elsevier.

[18] Carmelo Vazquez, Almudena Duque, Ivan Blanco, Teodoro Pascual, Natalia Poyato, Irene Lopez-Gomez, and Covadonga Chaves, "CBT and positive psychology interventions for clinical depression promote healthy attentional biases: An eye-tracking study", *Depression and anxiety*, vol. 35, no. 10, pp. 966–973, 2018, ISBN: 1091-4269 Publisher: Wiley Online Library.

[19] Paweł Ziemba, Jarosław Wątróbski, Artur Karczmarczyk, Jarosław Jankowski, and Waldemar Wolski, "Integrated approach to e-commerce websites evaluation with the use of surveys and eye tracking based experiments", in *2017 Federated Conference on Computer Science and Information Systems (FedCSIS)*. 2017, pp. 1019–1030, IEEE.

[20] Tom M. Mitchell, "Does machine learning really work?", *AI magazine*, vol. 18, no. 3, pp. 11–11, 1997, ISBN: 2371-9621.

[21] Bruno Vicente Alves de Lima, Vinicius Ponte Machado, and Lucas Araújo Lopes, "Automatic labeling of social network users Scientia. Net through the machine learning supervised application", *Social Network Analysis and Mining*, vol. 5, no. 1, pp. 44, 2015, ISBN: 1869-5450 Publisher: Springer.

[22] Pooja Jain, Jonathan M. Garibaldi, and Jonathan D. Hirst, "Supervised machine learning algorithms for protein structure classification", *Computational biology and chemistry*, vol. 33, no. 3, pp. 216–223, 2009, ISBN: 1476-9271 Publisher: Elsevier.

[23] Cullen Schaffer, "Selecting a classification method by cross-validation", *Machine Learning*, vol. 13, no. 1, pp. 135–143, 1993, Publisher: Springer.

[24] Joaquín Amat Rodrigo, "Validación de modelos de regresión: Cross-validation, OneLeaveOut, Bootstrap", November 2016.

[25] Guido van Rossum and Fred L. Drake, *An introduction to Python: release 2.5*, Network Theory Limited, Bristol, 2. print. (rev. for version 2.5) edition, 2006, OCLC: 836605111.

[26] Ashwin Pajankar, "Introduction to Python", in *Python Unit Test Automation*, pp. 1–17. Springer, 2017.

[27] "Anaconda".

[28] Alberto Cardoso, Joaquim Leitão, and César Teixeira, "Using the Jupyter notebook as a tool to support the teaching and learning processes in engineering courses", in *International Conference on Interactive Collaborative Learning*. 2018, pp. 227–236, Springer.

[29] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, and Michael Isard, "Tensorflow: A system for large-scale machine learning", in *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, 2016, pp. 265–283.

[30] Ana Isabel Rojão Lourenço Azevedo and Manuel Filipe Santos, "KDD, SEMMA and CRISP-DM: a parallel overview", *IADS-DM*, 2008.

[31] Rüdiger Wirth and Jochen Hipp, "CRISP-DM: Towards a standard process model for data mining", in *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining*. 2000, pp. 29–39, Springer-Verlag London, UK.