

Sistema de autenticación continua basado en el comportamiento mediante técnicas de streaming

Arturo Silvelo Pallín

Tutores: José Carlos Dafonte Vázquez e Daniel Garabato Míguez

Curso 2020/2021

Arturo Silvelo Pallín

Sistema de autenticación continua basado en el comportamiento mediante técnicas de streaming

Trabajo Fin de Máster. Curso 2020/2021

Tutores: José Carlos Dafonte Vázquez e Daniel Garabato Míguez

Máster Inter-Universitario en Ciberseguridad

Universidad de A Coruña

Facultad de Informática

Campus de Elviña

15071, A Coruña

Abstract

Nowadays, authentication systems allow to verify the user's identities at a specific point in the system, usually at login, while leaving the user with the responsibility of logging off each time they stop using their computer. Because of this step, in multiuser and professional environments, users often leave their sessions open so that other users can access and perform unauthorized operations on their behalf. These actions can result in access to confidential data that could cause legal problems for both the organization and the legitimate user, that may result in financial penalties.

To avoid these problems, new authentication systems have been developed, such as fingerprint schemes, but this entails a cost in price due to the installation of a reader for each workstation. Additionally, operating systems can implement restriction policies that close inactive sessions and force users to change their passwords in short periods of time. However, such measures may disturb the user and lead to the use of weak passwords, causing a false sense of security.

For this purpose, an authentication system has been proposed in this project as a second factor mechanism, in which the user's identities are periodically verified according to their behavior during their sessions. Thus, data related to mouse movement is gathered in order to build unique user profiles by means of Artificial Intelligence techniques that were combined with streaming processing methods to achieve a real-time authentication process. As a result, a scalable system able of authenticating users in a transparent manner to them was successfully developed and deployed.

Keywords — Mouse dynamics, Behavioral biometrics, Security monitoring

Resumen

Los sistemas de autenticación actuales permiten verificar la identidad de los usuarios en un momento concreto del sistema, por lo general al inicio, lo que deja al usuario la responsabilidad de cerrar la sesión cada vez que abandone su ordenador. Debido a este paso, en entornos multiusuario y en el ámbito profesional los usuarios suelen dejar sus sesiones abiertas permitiendo que otros usuarios accedan y realicen operaciones en su sesión. Estas acciones pueden derivar en acceso a datos confidenciales que provocarían problemas legales tanto a la organización como al usuario legítimo, resultando en posibles sanciones económicas.

Para evitar estos problemas se han implementado nuevos sistemas de autenticación, como los esquemas basados en la huella dactilar, pero esto conlleva un coste en precio debido a la necesidad de instalar un lector por cada puesto de trabajo. Por otra parte, los sistemas operativos permiten configurar políticas de restricción que cierran sesiones inactivas y obligan a los usuarios a cambiar sus contraseñas en cortos periodos de tiempo. Sin embargo, estas medidas pueden provocar malestar en los usuarios y llevar al uso de contraseñas débiles, lo que acaba provocando una falsa sensación de seguridad.

Para este propósito, en este proyecto se ha planteado un sistema de autenticación como segundo factor, en el que se verifica la identidad del usuario periódicamente a partir de su comportamiento en el uso del dispositivo. Para ello, se recopilan los datos relativos al movimiento del ratón para la creación de perfiles de usuario únicos con técnicas de Inteligencia Artificial que combinado con procesamiento *streaming* realice el proceso de autenticación en tiempo real. De esta manera, se ha implementado y desplegado un sistema escalable capaz de autenticar a los usuarios de manera transparente para ellos.

Palabras clave — Movimientos de ratón, Biometría del comportamiento, Monitorización de seguridad

Agradecimientos

A mis padres

Índice general

1	Introducción	1
1.1	Motivación	1
1.2	Objetivos	4
1.3	Organización de la memoria	4
2	Fundamentos teóricos	7
2.1	Seguridad Informática	7
2.1.1	Conceptos previos	7
2.1.2	Objetivos de la seguridad informática	8
2.1.3	Autenticación de usuario	8
2.2	Características del uso del ratón	10
2.3	Evaluación del rendimiento	12
2.4	Inteligencia Artificial	14
2.4.1	Tipo de aprendizaje	15
2.4.2	Redes Neuronales Artificiales	16
	Perceptrón Multicapa	17
2.4.3	Máquinas de soporte vectorial	19
	Tipos de problemas	20
2.4.4	Árboles de Decisión	21
2.4.5	Métodos Ensambladores	22
	Bootstrapping	22
	Bagging	22
	Boosting	23
	Stacking	23
2.4.6	Bosques Aleatorios	23
	Ventajas	24
	Desventajas	24
2.4.7	Clasificador por Votación	24
	Tipos de votación	25
	Ventajas	25
	Desventajas	25
2.5	Sistemas de gestión de colas	25

3	Estado del arte	29
4	Planificación y Metodología	31
4.1	Recursos y actividades	31
4.2	Planificación Inicial	34
4.3	Seguimiento del proyecto	34
4.4	Análisis de riesgos	36
4.5	Metodología	36
4.5.1	Desarrollo incremental	38
5	Tecnologías	39
5.1	Estructura final	39
5.2	Arquitectura	40
5.3	Capa de Presentación	40
5.3.1	Aplicación de captura de datos	40
5.3.2	Aplicación de administración	41
	Angular	42
5.4	Capa de Negocio	44
5.4.1	Docker	45
5.4.2	Sistemas de gestión de colas	46
5.4.3	Machine Learning	47
5.4.4	Orquestador	47
5.4.5	Capa de Datos	48
	ReplicaSet	50
6	Inteligencia Artificial	51
6.1	Selección de características	51
6.1.1	Tratamiento de datos	53
6.2	Selección de algoritmos e hiperparámetros	54
7	Kafka: Sistema de colas	57
7.1	Envío de datos	57
7.2	Diseño del Escenario	59
7.2.1	Escenario 1	59
7.2.2	Escenario 2	60
7.2.3	Escenario 3	60
7.2.4	Escenario 4	61
8	Resultados	63
8.1	Datos de <i>Random Forest</i>	63

8.2 Datos de <i>Streaming</i>	63
9 Conclusiones	69
10 Trabajo Futuro	71
Bibliografía	73
Índice de figuras	77
Índice de cuadros	79

Introducción

En este capítulo se expondrá una visión generalizada del proyecto, abordando la idea original del mismo, cómo surge y qué objetivos se pretenden conseguir con la realización de este.

1.1 Motivación

Desde los inicios de la informática, la ciberseguridad [22, 45] ha sido una de las áreas de estudio que mayor interés ha despertado y, hoy en día, ha adquirido una especial relevancia en la vida cotidiana de los usuarios en prácticamente cualquier ámbito [18, 19], debido a la enorme proliferación de diferentes tipos de dispositivos, tales como ordenadores personales, tabletas o teléfonos inteligentes (smartphones), así como el uso habitual de diferentes plataformas y aplicaciones informáticas, como pueden ser productos de ofimática tradicionales, redes sociales, aplicaciones bancarias o plataformas de salud en línea. Actualmente, la ciberseguridad ha tomado una especial relevancia debido al creciente aumento de ataques informáticos a diferentes infraestructuras y plataformas, llegando a generar una gran preocupación no solo para las empresas responsables de dichos servicios, sino también para el usuario final de tales herramientas.

La seguridad informática abarca una gran variedad de técnicas y métodos para tratar de dotar a los sistemas de información de una cierta protección ante posibles ataques, incluyendo los sistemas físicos, los sistemas software o servicios, así como también la información almacenada en ellos. Entre estos mecanismos, se encuentran los sistemas de autenticación, un componente esencial en cualquier modelo de seguridad que actúa como un primer elemento de defensa de los sistemas de información mediante la verificación de la identidad de los usuarios. De este modo, una vez comprobada la identidad los sistemas de control de acceso podrán otorgar los permisos pertinentes a un usuario legítimo o bien denegar el acceso al sistema a un usuario ilegítimo, registrando y notificando dicha incidencia a los administradores del sistema.

Para llevar a cabo el proceso de autenticación, se emplean dos fases: en primer lugar, el usuario debe proporcionar una identificación para que, en un segundo paso,

el sistema pueda llevar a cabo una comprobación para determinar si existe una correspondencia entre dicha identificación y la identidad del usuario. Actualmente existen diferentes maneras de realizar este procedimiento, en función del factor que se utilice para completarlo [9]:

- El esquema de autenticación más habitual está basado en un factor de conocimiento por parte del usuario y consiste en el uso de una determinada información que, teóricamente, solo deberían conocer el usuario legítimo y el sistema de autenticación, como pueden ser el uso de pares usuario/contraseña o números de identificación personal (PIN).
- Otro de los esquemas ampliamente utilizados, especialmente para llevar a cabo el control de acceso a edificios o áreas, es el factor de posesión, donde el usuario presenta al sistema autenticador un objeto o token para llevar a cabo la verificación de identidad. El uso de documentos acreditativos de identidad como el DNI o incluso las tarjetas bancarias son claros ejemplos de este tipo de sistemas, así como también podrían serlo el uso de tarjetas inteligentes de propósito específico o incluso la utilización de dispositivos móviles.
- Durante los últimos años también se han convertido en sistemas de autenticación habituales aquellos basados en factores biométricos, es decir, los que emplean ciertas propiedades inherentes a los usuarios que, además, son invariables en el tiempo para llevar a cabo la comprobación de la identidad, tales como la huella dactilar, el iris o incluso sistemas de reconocimiento facial [43].
- Uno de los sistemas más novedosos que está tomando cada vez una mayor relevancia se basa en realizar el proceso de autenticación apoyándose en la ubicación del usuario, empleando desde las direcciones MAC e IP del equipo utilizado, hasta el posicionamiento GPS a partir de la geolocalización de un dispositivo móvil [49].
- Otro tipo de sistemas que está tomando cada vez una mayor relevancia son los basados en factores de comportamiento biométrico [2], donde la verificación de la identidad de los usuarios se lleva a cabo a través de la monitorización de la interacción entre el propio usuario y el sistema, como por ejemplo el tipo de aplicación utilizada, los sitios Web frecuentados, la manera de teclear o incluso la utilización del ratón [12, 23].

A lo largo de los últimos años, los sistemas de autenticación han buscado reforzar el proceso añadiendo pasos complementarios en el proceso, requiriendo la presencia de múltiples factores para autenticar al usuario. Un claro ejemplo de este tipo de procedimientos son los sistemas de verificación en dos pasos [12], ampliamente

extendidos hoy en día, donde se lleva a cabo una autenticación mediante un factor primario, habitualmente basado en un esquema usuario/contraseña, solicitando posteriormente al usuario un código de confirmación enviado a su dispositivo móvil, bien sea un SMS o a través de una aplicación específica.

Mediante la utilización de los métodos anteriormente señalados, es importante resaltar que la identidad del usuario solamente se verifica en el momento inicial y, por tanto, podría darse una usurpación de una sesión abierta. Ante este tipo de circunstancias, empleando mecanismos de autenticación convencionales únicamente sería posible prevenir este tipo de situaciones mediante la repetición periódica de todo el proceso de autenticación, lo que dificultaría la experiencia del usuario al tratarse generalmente de procesos invasivos que interrumpirían la sesión. Por tanto, para llevar a cabo este proceso de forma transparente al usuario será necesario recurrir a factores de autenticación no invasivos, un perfil en el que encajan perfectamente los factores basados en el comportamiento del usuario. De este modo, sería posible monitorizar la actividad del usuario de forma constante, completando el proceso de autenticación de forma continuada y permitiendo la detección de posibles suplantaciones de identidad [13].

El trabajo que se ha realizado en este proyecto se centra precisamente en estos sistemas de autenticación de segunda fase basados en el comportamiento del usuario frente al dispositivo que emplea, en este caso orientado a explotar la interacción mediante un dispositivo convencional, como es el ratón. Para ello, el usuario debe completar el proceso de autenticación inicial para la apertura de sesión mediante un factor primario (usuario/contraseña) y, posteriormente, la verificación continua de su identidad se llevará a cabo de forma recurrente y totalmente transparente para el usuario.

Aquí nos planteamos un sistema de monitorización en tiempo real y sobre entornos de trabajo reales. De esta forma, las capturas asociadas al comportamiento se realizarán en un entorno no controlado, es decir, entornos en los que el usuario tendrá libertad de movimientos y no será un escenario guiado. Estas nos proporcionarán eventos que posteriormente convertiremos en características. Aplicando sobre estas características diferentes técnicas de Inteligencia Artificial, podremos encontrar patrones que permitan autenticar al usuario. Todos los datos del sistema se procesarán mediante una plataforma de *streaming* de datos, que permitirá realizar el proceso de autenticación en tiempo real.

1.2 Objetivos

El proyecto debe cumplir los siguientes objetivos, para garantizar la calidad y funcionalidad del mismo:

1. Implementar una aplicación de escritorio que permita la captura de los eventos del usuario para su posterior envío. El lenguaje de programación elegido deberá permitir generar un aplicativo multiplataforma.
2. Implementar un servidor que permita la distribución de los mensajes. Estudiando las propuestas y estándares disponibles y seleccionando una plataforma de distribución de mensajes que permita un gran volumen de tráfico. Además, la opción elegida deberá contar con la opción de incrementar sus funcionalidades base mediante *plugins* o librerías.
3. Realizar una fase de recogida de información con la aplicación de escritorio creada, con el fin de obtener un conjunto de datos que contenga una muestra representativa de la población, es decir, personas de diferentes edades y géneros.
4. Analizar la información recopilada, planteando, desarrollando y evaluando diferentes mecanismos de autenticación basados en el estudio de patrones de comportamiento de los usuarios mediante múltiples técnicas de Inteligencia Artificial, como pueden ser redes neuronales [48], máquinas de soporte vectorial [17] o árboles aleatorios [15]. Por otro lado, también será necesario estudiar las diferentes características del comportamiento de los usuarios y determinar aquellas de mayor relevancia para acometer el proceso de autenticación, generando perfiles únicos por usuario.
5. Implementar un servicio que permita la automatización del proceso de creación de modelos únicos para cada usuario.
6. Implementar una aplicación web que permita visualizar los datos generados y el estado del sistema de una manera sencilla. Este desarrollo se debe realizar con un *framework* que agilice su programación y mantenimiento.

1.3 Organización de la memoria

La memoria se estructura basándose en una serie de capítulos:

- **Fundamentos teóricos:** En este capítulo se abordan los diferentes términos para comprender mejor el dominio.

- **Situación actual:** En este capítulo se muestra algunas de las alternativas disponibles y artículos encontrados relacionados con el tema que se trata en el proyecto.
- **Planificación y evaluación de costes:** En este capítulo se exponen las fases del desarrollo del proyecto, su realización, costes, recursos y riesgos asociados.
- **Tecnologías:** En este capítulo se analizan las herramientas utilizadas para el desarrollo del proyecto y las decisiones tomadas para su elección.
- **Metodología:** En este capítulo se muestra la información relativa al desarrollo del proyecto.
- **Inteligencia Artificial:** En este capítulo se muestra el procedimiento realizado para el proceso de creación de los modelos de los usuarios.
- **Kafka - Sistemas de colas:** En este capítulo se muestra el proceso de implementación y creación del sistema de gestión de colas.
- **Resultados:** En este capítulo se exponen y discuten los resultados obtenidos.
- **Conclusiones:** En el último capítulo de la memoria se comprobará el grado de cumplimiento de los requisitos y su impacto.
- **Trabajo Futuro:** En este capítulo se expondrá las diferentes ramas de desarrollo que se han tenido en cuenta para la continuación del proyecto.

Fundamentos teóricos

En este capítulo se presentarán las principales temáticas relacionadas con el dominio del proyecto, con el objetivo de comprenderlo mejor.

2.1 Seguridad Informática

La seguridad informática forma parte de un término más genérico como es la seguridad de la información, y tiene como objetivo prevenir y detectar el uso no autorizado de un sistema informático.

2.1.1 Conceptos previos

- **Atacante:** Sujeto o entidad que pone en riesgo un sistema.
- **Ataque:** Consiste en cualquier acción hecha por individuos u organizaciones que roban, alteran o destruyen a un blanco específico.
 - **Ataque pasivo:** Son ataques difíciles de detectar, puesto que el atacante solo observa o monitoriza la información.
 - **Ataque activo:** Son aquellos que implican algún tipo de modificación de la información con el fin de dañar al objetivo.
- **Intrusión:** Conjunto de acciones que intentan comprometer la integridad, confidencialidad o disponibilidad de un recurso. Es decir, la intrusión no solo consiste en el acceso no autorizado, sino también en la denegación del acceso a otros usuarios o la manipulación de la información.
- **Vulnerabilidad [4]:** Debilidad o fallo en un sistema de información que pone en riesgo la seguridad de la información, permitiendo que un atacante pueda comprometer la integridad, disponibilidad o confidencialidad de la misma.

- **Amenaza [4]:** Acción que aprovecha una vulnerabilidad para atentar contra la seguridad de un sistema de información. Es decir, que podría tener un potencial efecto negativo sobre algún elemento de nuestro sistema.
- **Riesgo [4]:** El riesgo es la probabilidad de que se produzca un incidente de seguridad, materializándose una amenaza y causando pérdidas o daños.
- **Política de Seguridad [21]:** Conjunto de requisitos definidos por los responsables de un sistema que indican en términos generales qué está y qué no está permitido en el área de la seguridad durante el uso del sistema.

2.1.2 Objetivos de la seguridad informática

Los sistemas de información guardan, distribuyen o generan información para usuarios, empresas, procesos o aplicaciones, y esta información debe garantizar ciertas características para evitar fraudes. La seguridad informática intenta asegurar estas garantías sobre la información [25]:

- **Confidencialidad:** Es la capacidad de que solo los usuarios autorizados puedan acceder a nuestros recursos, datos e información. Este es uno de los principales problemas a los que se enfrentan las empresas.
- **Integridad:** Es la capacidad de asegurar que los datos sean legítimos, es decir, asegurar que los datos recibidos sean los generados inicialmente y que nada, ni nadie ajeno pueda modificar dichos datos.
- **Disponibilidad:** Esta característica es de las más importantes y asegura que los datos estén accesibles siempre que el usuario, proceso o sistema lo necesite.

2.1.3 Autenticación de usuario

En el ámbito de la seguridad informática es muy importante demostrar que un usuario o una aplicación es realmente quien dicha persona o aplicación asegura ser. Para esta verificación se pueden usar varias técnicas [5]:

1. **Sistemas basados en algo conocido:** Es el modelo de autenticación más básico y consiste en decidir si un usuario dice ser quien es simplemente basándonos en una prueba que a priori solo ese usuario puede saber. Esta aproximación es la más barata y también la más vulnerable a todo tipo de ataques. Las entidades que participan en la autenticación acuerdan una clave,

que mantendrán en secreto. Cuando una de las partes necesita autenticarse solo tiene que mostrar la clave secreta que han acordado para poder acceder a los recursos.

2. **Sistemas basados en algo poseído:** Este modelo de autenticación es más complejo y se puede complementar con otros sistemas como el anterior de una manera fácil, pero también implica un mayor coste. Por lo general, este modelo se caracteriza por el uso de una tarjeta con un chip integrado el cual incorpora la información necesaria para la autenticación del usuario. Este dispositivo fue patentado por **Roland Moreno** en 1970, y consistía en una tarjeta de plástico con un chip integrado. A día de hoy este tipo de tarjeta está presente en multitud de aplicaciones como tarjeta bancarias, tarjetas de acceso, etc.
3. **Sistemas de autenticación biométrica:** Hoy en día existen otros mecanismos que se basan en las cualidades del usuario. Este tipo de sistemas basados en el reconocimiento de cualidades físicas del usuario, utilizan características únicas del individuo para su identificación. El proceso general para este tipo de sistemas es el siguiente:
 1. Capturar una muestra de los datos del usuario.
 2. Extraer las características que sean únicas de ese usuario.
 3. Comparar dichas características con las extraídas en un primer momento.
 4. Basándonos en esa comparación se decide si el usuario es quien dice ser.

Los sistemas de autenticación biométricos más comunes son:

- **Verificación de huella dactilar:** La huella de un ser humano es un rasgo de identificación único (salvo alguna excepción), pero los sistemas de autenticación miden ciertas características que se han demostrado únicas en todos los individuos [31].
- **Verificación de patrones oculares:** Estos modelos de reconocimiento mediante patrones oculares, por lo general miden rasgos en el iris o en la retina. Han demostrado ser los más fiables con una probabilidad de coincidencia cercana a cero.

Otros sistemas de autenticación biométricos basados en el comportamiento humano, como el que se trata en este proyecto, son:

- **Reconocimiento de voz:** El reconocimiento de voz intenta detectar características típicas de la voz del usuario para identificarlo. La técnica aplicada para lograr este fin es similar a los sistemas de reconocimiento de canciones como *shazam* [47].
- **Verificación de escritura:** Estos sistemas intentan verificar la firma manuscrita de una persona. En ella se verifican los patrones y características del trazo realizado.

2.2 Características del uso del ratón

Los datos en bruto generados por un ratón [Tabla 2.1] deben ser procesados para extraer aquellas características relevantes para el problema y que pueden ser tratadas adecuadamente por las técnicas de aprendizaje automático.

Para este propósito se analizaron diferentes propuestas de la literatura que utilizan características de distinta naturaleza, principalmente basadas en propiedades del movimiento [7, 23, 32] como velocidades, aceleraciones y variaciones de ángulo. En este proyecto, nos hemos centrado en las características basadas en el movimiento, ya que el objetivo es obtener un sistema de autenticación basado en el comportamiento. En consecuencia, hemos seleccionado como referencia las características de [23] y hemos incluido nuevas características que se derivan de estas aplicando las siguientes técnicas:

- Valores absolutos: Obtenidos a partir de las características originales.
- Descomposición en características básicas: Algunas características se descomponen en valores más primitivos, como la velocidad que se divide en horizontal y vertical.
- Hemos decidido agrupar los eventos, tomando el primer evento como origen y calculando las características respecto a él.

La totalidad de las características extraídas son detalladas en la Tabla 2.2.

Característica	Definición
x	Posición del ratón en el eje horizontal de la pantalla
y	Posición del ratón en el eje vertical de la pantalla
timestamp	Instante de tiempo del movimiento (formato UNIX)

Tab. 2.1: Valores de entrada de un movimiento

Característica	Definición
Horizontal Velocity	$hv = \frac{x_i - x_j}{t_i - t_j}$
Absolute Horizontal Velocity	$hv_abs = hv $
Horizontal Velocity Left	$hv_l = \begin{cases} hv_abs, & \text{if } hv > 0 \\ 0, & \text{otherwise} \end{cases}$
Horizontal Velocity Right	$hv_r = \begin{cases} hv_abs, & \text{if } hv < 0 \\ 0, & \text{otherwise} \end{cases}$
Vertical Velocity	$vv = \frac{y_i - y_j}{t_i - t_j}$
Absolute Vertical Velocity	$vv_abs = vv $
Vertical Velocity Left	$vv_l = \begin{cases} vv_abs, & \text{if } vv > 0 \\ 0, & \text{otherwise} \end{cases}$
Vertical Velocity Right	$vv_r = \begin{cases} vv_abs, & \text{if } vv < 0 \\ 0, & \text{otherwise} \end{cases}$
Tangencial Velocity	$tv = \sqrt{hv^2 + vv^2}$
Tangencial Acceleration	$ta = \frac{tv_i - tv_j}{t_i - t_j}$
Absolute Tangencial Acceleration	$ta_abs = ta $
Tangencial Jerk	$tj = \frac{ta_i - ta_j}{t_i - t_j}$
Absolute Tangencial Jerk	$tj_abs = tj $
Tangencial Jerk Left	$tj_l = \begin{cases} tj_abs, & \text{if } tj > 0 \\ 0, & \text{otherwise} \end{cases}$
Tangencial Jerk Right	$tj_r = \begin{cases} tj_abs, & \text{if } tj < 0 \\ 0, & \text{otherwise} \end{cases}$
Origin Distance	$d_o = \sqrt{x^2 + y^2}$
Distance	$d = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$
Slope Angle Tangent	$ts_angle = \text{atan}_2(y, x)$
Origin Slope Angle Tangent	$ts_origin_angle = \text{atan}_2(y_i - y_o, x_i - x_o)$
Curvature	$c = \frac{ts_angle_i - ts_angle_j}{t_i - t_j}$

<i>Absolute Curvature</i>	$c_{abs} = c $
<i>Curvature Rate</i>	$cr = \frac{c_i - c_j}{d_j - d_i}$
<i>Absolute Curvature Rate</i>	$cr_{abs} = cr $
<i>Origin Curvature</i>	$c_{origin} = \frac{ts_{origin_angle_o} - ts_{origin_angle_j}}{t_o - t_j}$
<i>Absolute Origin Curvature</i>	$c_{origin_abs} = c_{origin} $
<i>Origin Curvature Rate</i>	$cr_{origin} = \frac{c_o - c_j}{t_o - t_j}$
<i>Absolute Origin Curvature Rate</i>	$cr_{origin_abs} = cr_{origin} $

donde x es la posición del ratón, t es el *timestamp* del evento, o es el registro origen, i es el registro actual y j el registro anterior.

Tab. 2.2: Lista de características transformadas

2.3 Evaluación del rendimiento

Para analizar los resultados obtenidos y comprobar la fiabilidad de los algoritmos usados, necesitamos emplear unas métricas que evalúen su rendimiento y permitan comparar diferentes aproximaciones.

El conjunto de datos utilizados para la evaluación del rendimiento consistirá en un *dataset* con los datos distribuidos en dos grupos, los datos del usuario legítimo, que son los eventos obtenidos de la sesión de ese usuario, frente a los datos del usuario ilegítimo, que son los eventos capturados de otros usuarios. La distribución de este conjunto se realizará en una proporción 1:1.

Para evaluar el rendimiento de los algoritmos se empleará la matriz de confusión [Figura 2.1], la cual categoriza los datos en cuatro grupos:

- **Verdaderos Negativos (TN):** Es la cantidad de eventos de usuarios no legítimos que fueron clasificados correctamente.
- **Falsos Positivos (FP):** Es la cantidad de eventos de usuarios no legítimos que fueron clasificados incorrectamente como legítimos.
- **Falsos Negativos (FN):** Es la cantidad de eventos del usuario legítimo que fueron clasificados incorrectamente como no legítimos.

- **Verdaderos Positivos (TP):** Es la cantidad de eventos del usuario legítimo que fueron clasificados correctamente.

		NEGATIVOS	POSITIVOS
		PREDICCIÓN	
OBSERVACIÓN	NEGATIVOS	VERDADEROS NEGATIVOS	FALSOS POSITIVOS
	POSITIVOS	FALSOS NEGATIVOS	VERDADEROS POSITIVOS

Fig. 2.1: Esquema matriz de confusión.

A partir de los datos obtenidos de una matriz de confusión, se pueden calcular medidas que permiten obtener una representación analítica de los resultados. Por otra parte, durante el entrenamiento de los algoritmos también se han calculado los tiempos de cómputo.

- **Recall (RC):** Es la relación entre las predicciones positivas correctas y el total de observaciones positivas.

$$\frac{TP}{TP + FN} \quad (2.1)$$

- **Precision (PS):** Es la relación entre las predicciones positivas correctas y el total de predicciones positivas.

$$\frac{TP}{TP + FP} \quad (2.2)$$

- **F1:** Es la media armónica de los valores anteriores.

$$\frac{2}{\frac{1}{precision} + \frac{1}{recall}} = 2 * \frac{precision * recall}{precision + recall} = \frac{TP}{TP + \frac{FN+FP}{2}} \quad (2.3)$$

- **Accuracy:** Es la relación entre las predicciones positivas y el total de casos.

$$\frac{TP}{TP + FP + FN + TN} \quad (2.4)$$

A partir de los datos de la matriz de confusión y un umbral (*threshold*) es posible calcular ciertos ratios que permiten parametrizar de una manera más objetiva el rendimiento del algoritmo. Estos ratios son los definidos a continuación:

- **Threshold:** Es un valor que se elige para considerar si el modelo de predicción es correcto o no. Este valor indica el grado de sensibilidad para rechazar casos falsos positivos y casos de falsos negativos.
- **False Acceptance Rate (FAR) [Figura 2.2]:** Representa el ratio de casos en el que un usuario no legítimo es clasificado como legítimo. En el *threshold* 0, la probabilidad de esta identificación es del 100% e irá disminuyendo cuando se baja el *threshold*.

$$\frac{FP}{FP + TN} \quad (2.5)$$

- **False Reject Rate (FRR) [Figura 2.2]:** Representa el ratio de casos en el que un usuario legítimo es clasificado como no legítimo. En el *threshold* 0, el usuario legítimo será identificado correctamente el 100% de los casos e irá disminuyendo cuando se baja el *threshold*.

$$\frac{FN}{FN + TP} \quad (2.6)$$

- **Equal Error Rate (EER) [Figura 2.2]:** Representa el punto óptimo de *threshold* donde las curvas de *FRR* y *FAR* se cortan, este valor suele ser mejor cuanto más cercano a cero se encuentre.

$$\frac{FAR + FRR}{2} \quad (2.7)$$

2.4 Inteligencia Artificial

En 1956, **John McCarthy** [29], conocido como el padre de la Inteligencia Artificial, acuñó el término Inteligencia Artificial, en adelante *IA*, como la ciencia y la ingeniería de hacer máquinas inteligentes, especialmente programas informáticos inteligentes.

En la *IA* existen dos grupos definidos, fuerte y débil. La *IA* fuerte es aquella que tiene las mismas características que un humano inteligente. La *IA* débil es aquella que muestra inteligencia en un área en concreto, pero carece de la misma en otras.

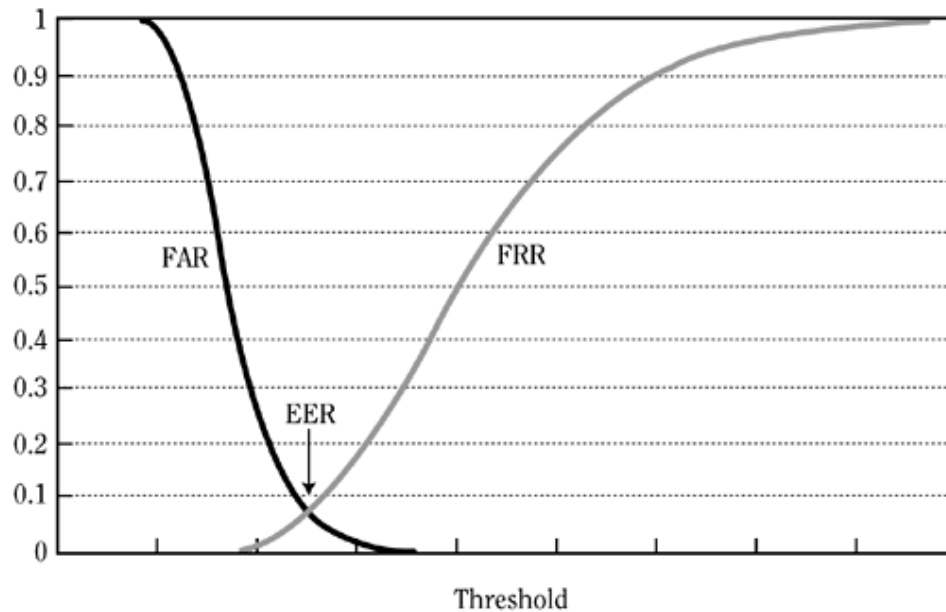


Fig. 2.2: Relación de EER con FAR y FRR

Para conseguir dotar de inteligencia a una máquina se suele utilizar el aprendizaje máquina o *machine learning*. El aprendizaje máquina, en adelante *ML*, es una de las áreas más extendidas de la *IA*. En 1959, **Arthur Samuel** definió *ML* como el campo de estudio que brinda a las computadoras la capacidad de aprender sin estar programado explícitamente.

2.4.1 Tipo de aprendizaje

Dentro del *ML* se pueden distinguir dos tipos de aprendizaje:

- **Aprendizaje supervisado:** Los algoritmos son entrenados basándose en un conjunto de datos de los que conocemos su respuesta correcta. De esta manera lo que se intenta es que el algoritmo obtenga una respuesta aceptable a partir de las características disponibles.
 - **Problemas de regresión:** Son usados para evaluar las relaciones que existen entre las variables y obtener un valor de esa estimación.
 - **Problemas de clasificación:** Son utilizados para dividir un conjunto de datos de entrada en distintas clases según sus características.

- **Aprendizaje no supervisado:** En este modelo, los datos no contienen una respuesta correcta. Este tipo de aprendizaje intenta buscar ciertos patrones para obtener una respuesta aceptable.

La finalidad de todos estos tipos de aprendizajes es la de generalizar, es decir, que puedan resolver problemas que no han visto previamente. Cuando entrenamos modelos computacionales con un conjunto de datos de entrada estamos haciendo que el algoritmo sea capaz de generalizar un concepto para que al consultarle por un nuevo conjunto de datos desconocido, este sea capaz de comprenderlo y proporcionarnos un resultado fiable [Figura 2.3].

Si nuestros datos de entrenamiento son muy pocos o poco representativos nuestra máquina no será capaz de generalizar el conocimiento y estará incurriendo en *underfitting* [Figura 2.3].

Si sobreentrenamos (*overfitting*) nuestro modelo lo que ocurrirá es que nuestra máquina solo se limitará a memorizar los casos particulares que le enseñamos y será incapaz de reconocer nuevos datos de entrada, perdiendo toda capacidad de generalización.

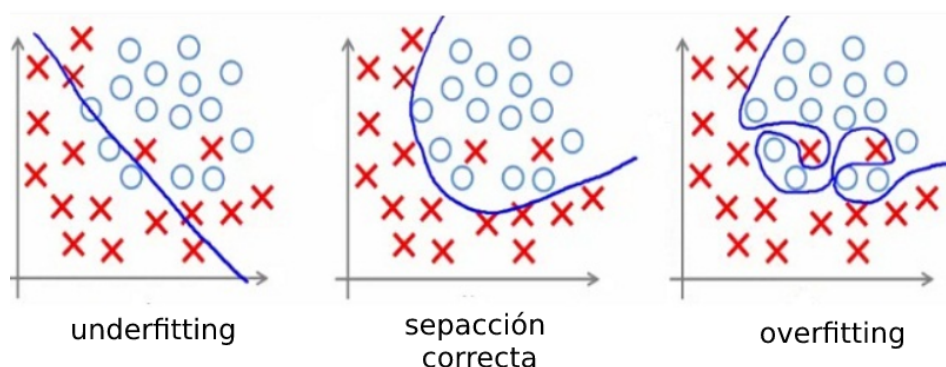


Fig. 2.3: Problemas de entrenamiento

2.4.2 Redes Neuronales Artificiales

Las redes neuronales artificiales [42], en adelante *RNA*, son un modelo computacional inspirado en la estructura del sistema nervioso de los seres humanos. La unidad elemental de una *RNA* es la neurona artificial [Figura 2.4] y generalmente están organizadas en capas. Poseen una capa de entrada con n entradas y una capa de salida con m salidas, que se calcula normalmente realizando una suma ponderada de las entradas con sus pesos [Ecuación (2.8)].

$$\sum_{i=1}^n w_i x_i + w_0 = \begin{cases} \geq 0 & y = 1 \\ < 0 & y = 0 \end{cases} \quad (2.8)$$

, donde x_i son las entradas, w_i es el peso de cada entrada y w_0 es el valor de *bias*.

Este resultado es modificado por una función de activación y el valor obtenido se transmite directamente al siguiente elemento. Normalmente para conseguir esta transformación se emplean las funciones comentadas en la Tabla 2.3.

Función	Ecuación
Sigmoidal	$f(x) = \frac{1}{1+e^{-x}}$
Tangente hiperbólica	$f(x) = \tanh(x)$
Gaussiana	$f(x) = e^{-\frac{x^2}{2}}$

Tab. 2.3: Funciones de activación

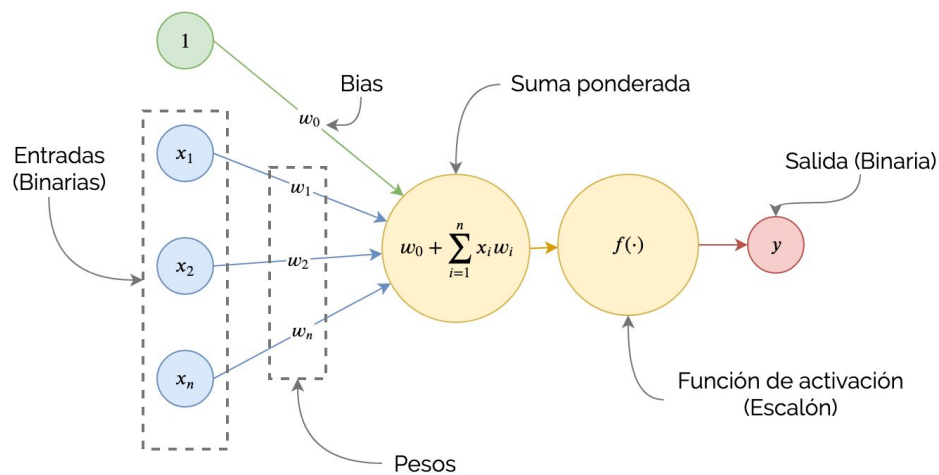


Fig. 2.4: Esquema de una neurona artificial

Perceptrón Multicapa

En 1958, **Rosenblatt** [39] diseñó y desarrolló el perceptrón. Este modelo implementa el funcionamiento de una sola neurona que es capaz de resolver problemas lineales. En 1969, **Minsky y Papert** escribieron un libro [30] donde demostraron que un solo perceptrón era incapaz de aprender la función exclusiva (XOR), es decir, problemas cuya resolución no es lineal. En este mismo libro se expone un nuevo paradigma de RNA llamado perceptrón multicapa también conocido como *MLP* por

sus siglas en inglés (*Multi-Layer Perceptron*). Este modelo es una combinación de varios perceptrón, que permiten aproximar cualquier problema, aunque no sea lineal. Este se caracteriza por tener sus neuronas agrupadas en capas de diferentes niveles, por lo general tres [Figura 2.5].

- **Capa de entrada:** Esta capa conecta la red con el exterior, cada neurona se corresponde con cada una de las variables de entrada a la red.
- **Capas ocultas:** Es un conjunto de capas que cuyas entradas son las salidas de la capa anterior y cuya salida pasan a la capa sucesora.
- **Capa de salida:** Conecta las capas ocultas con la salida de la red que proporciona los resultados.

Además, sus conexiones están dirigidas hacia adelante, es decir, las neuronas de una capa se conectan con las neuronas de la siguiente capa y generalmente todas las neuronas de una capa se encuentran enlazadas con las de la siguiente capa.

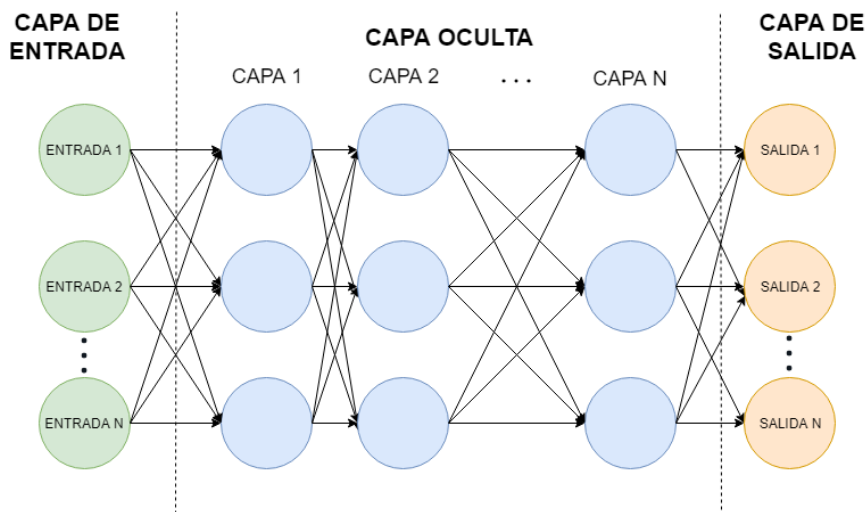


Fig. 2.5: Esquema del MLP

Inicialmente, este planteamiento no se pudo materializar porque no existía un mecanismo que ajustase automáticamente los pesos de la capa oculta. En 1986, **Rummelhart, Hinton y Williams** desarrollan la *Regla Delta Generalizada* [40] para adaptar los pesos propagando los errores hacia atrás. De esta manera se demuestra que el *MLP* es capaz de resolver problemas lineales y no lineales.

En la Figura 2.6 se puede ver el funcionamiento del *MLP*. En ella se muestra el esquema de un *MLP* que contiene una capa de entrada, dos capas ocultas y la salida. Cada neurona está representada visualmente con una gráfica que muestra la función

aplicada a los datos, por ejemplo, en la capa de entrada, las variables de entrada son dos funciones que dividen los datos en vertical (X_1) y horizontal (X_2). También se pueden observar las distintas conexiones que existen entre las neuronas e incluso el peso de las conexiones representadas por un color y grosor diferentes.

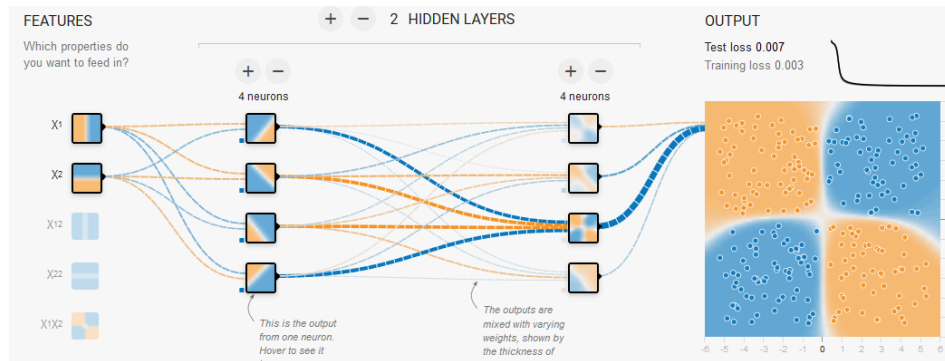


Fig. 2.6: Ejemplo de una MLP

Este tipo de arquitectura es muy utilizada debido a su capacidad de aproximación universal. No obstante, necesitan un largo proceso de aprendizaje para problemas complejos que requieren de un gran número de variables y una arquitectura compleja.

2.4.3 Máquinas de soporte vectorial [11]

Las máquinas de soporte vectorial también conocidas como *SVM* de sus siglas en inglés (*Support Vector Machines*), es un conjunto de algoritmos desarrollados por **Vladimir Vapnik**, capaces de realizar clasificaciones, regresiones e incluso detectar valores atípicos.

Las *SVM* generan un hiperplano¹ para intentar clasificar los datos [Figura 2.7]. En dicho hiperplano se forma una *calle* para separar los datos, donde la línea continua separa el conjunto y las líneas discontinuas indican el margen de error. Las *SVM* intentan maximizar el margen de error, lo que ocasiona una *calle* más grande y por lo tanto generalizará mejor el problema.

¹Es un plano de una dimensión inferior al origen, que divide el espacio en dos mitades

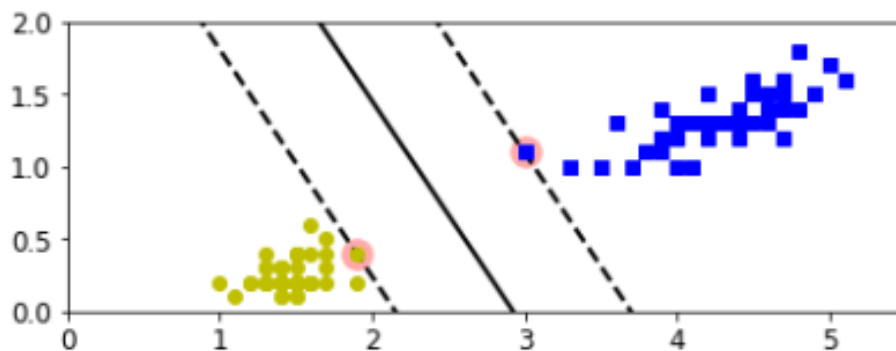


Fig. 2.7: Ejemplo de Clasificación SVM

Tipos de problemas

El conjunto de datos utilizados por estos algoritmos suelen ser multidimensionales, por lo tanto, en ciertas ocasiones puede ser complicado separar los datos. Basándose en esto podemos encontrarnos con dos tipos de escenarios:

- **Lineales [Figura 2.7]:** La solución de este tipo de problemas genera un hiperplano que es capaz de separar el conjunto de datos perfectamente.
- **No lineales:** Son aquellos en los que el conjunto de entrada no es posible separarlos con un hiperplano, pero el hecho de que no sean separables en el espacio original, no significa que no lo sean en un espacio de dimensiones distinto. Para transformar el espacio de entradas a una dimensión diferente se emplean las funciones *kernel* comentadas en la Tabla 2.4.

Lineal donde c es una constante.	$K(x, y) = xy + c$
Polinómica donde d es el orden del polinomio y a es una constante.	$K(x, y) = (a + xy)^d$
Función de base radial Gaussiana [Figura 2.8] donde σ es la anchura del <i>kernel</i> y $\ x - y\ $ es la distancia Euclídea entre x e y	$K(x, y) = \exp\left(-\frac{\ x - y\ ^2}{2\sigma^2}\right)$
Función sigmoide donde α es la pendiente y c es una constante.	$K(x, y) = \tanh(\alpha xy + c)$

Tab. 2.4: Funciones *kernel*

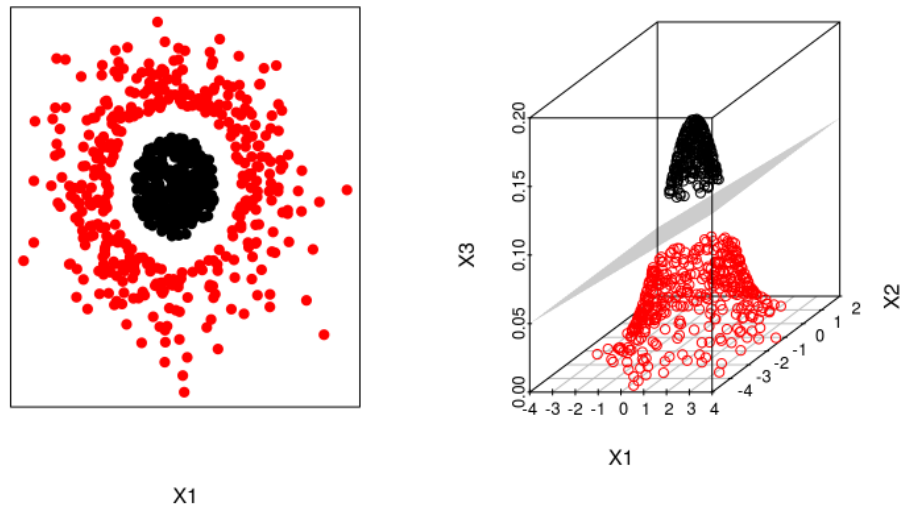


Fig. 2.8: Ejemplo de Clasificación SVM no lineal, usando función Gaussiana

2.4.4 Árboles de Decisión [16]

Generan modelos de clasificación o regresión usando árboles como estructuras internas [Figura 2.9]. En dicha estructura cada nodo representa una característica del problema, cada rama representa una decisión de esa característica y los nodos hoja contienen el valor de la predicción o clase.

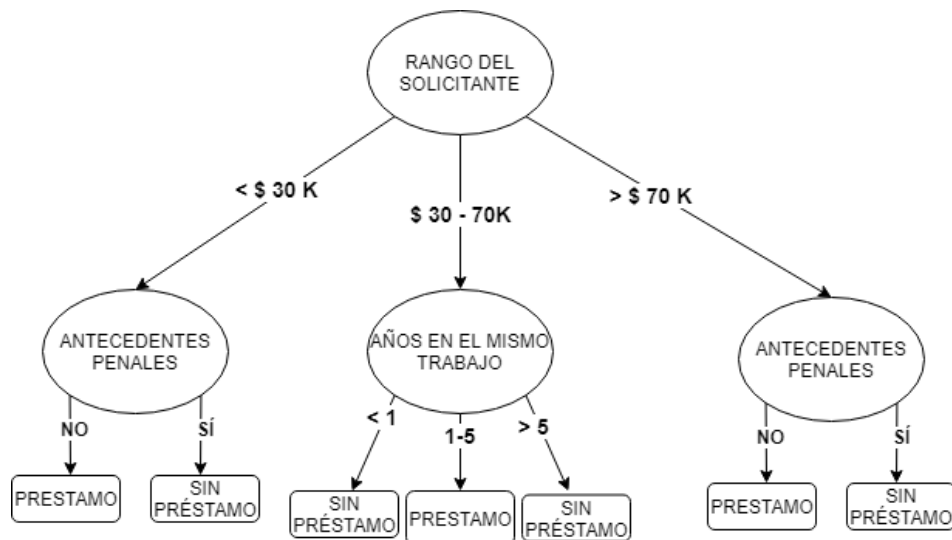


Fig. 2.9: Ejemplo de árbol de decisión para conceder un préstamo

2.4.5 Métodos Ensambladores [13, 14, 28]

Los métodos de tipo ensamblador están formados por un grupo de modelos predictivos que permiten alcanzar una mejor precisión y estabilidad del modelo. Estos utilizan diferentes técnicas para mejorar los resultados de un algoritmo, ya sea combinándolo con otros o utilizando varias instancias del mismo.

Algunas de estas técnicas son *Stacking*, *Bagging* y *Boosting*. Estas dos últimas utilizan *Bootstrapping* como método de muestreo de datos.

Bootstrapping

Es una técnica de muestreo. De las n muestras disponibles, se escogen k con reemplazo. Luego ejecutamos nuestros algoritmos utilizando esas muestras. Se utiliza el reemplazo para asegurar que las muestras sean aleatorias. Si se realizase sin reemplazo, las muestras extraídas dependerán de las anteriores.

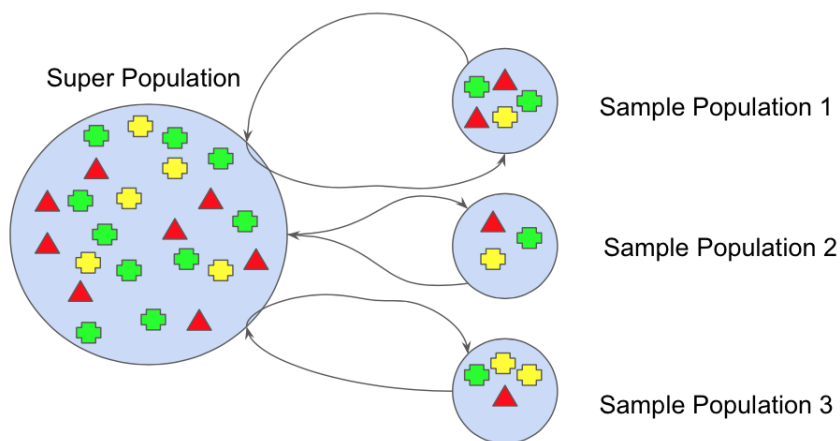


Fig. 2.10: Ejemplo de Bootstrapping

Bagging [13]

Este método genera múltiples instancias de un mismo modelo predictivo para conseguir una mejora en la precisión de la predicción. Generalmente esta técnica puede ser usada para reducir una alta varianza.

Boosting

Esta técnica emplea un conjunto de algoritmos que utilizan promedios ponderados para convertir aprendizajes débiles en fuertes. Cada modelo ejecutado dicta en que características se centrará el siguiente modelo.

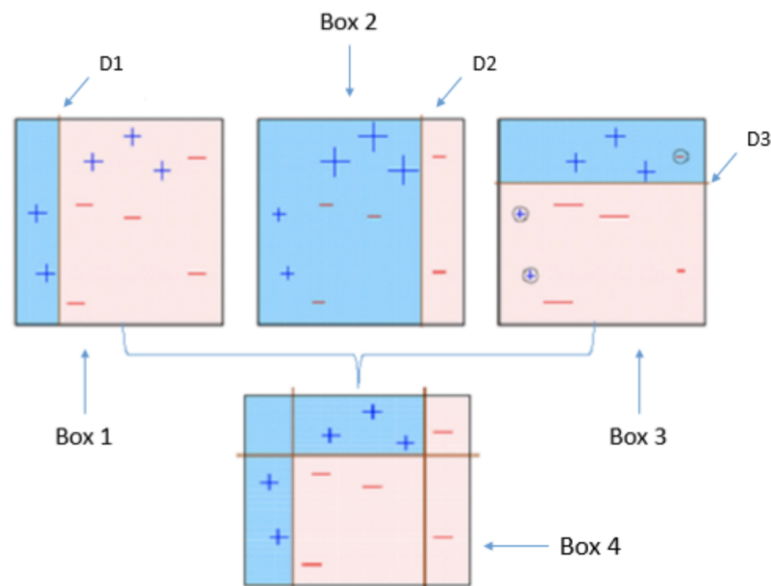


Fig. 2.11: Ejemplo de Boosting

Stacking

Esta técnica combina múltiples modelos de clasificación o regresión. Los modelos son entrenados individualmente utilizando un conjunto de entrenamiento y sus resultados son combinados para obtener una predicción final.

2.4.6 Bosques Aleatorios [15]

Los bosques aleatorios también conocidos como *Random Forest*, fueron desarrollados por **Leo Breiman** y **Adele Cutler**. Este algoritmo utiliza la técnica de *Bagging* para realizar las predicciones.

Son un conjunto de árboles de decisión en el que cada árbol depende de los valores de un vector aleatorio probado independientemente y con la misma distribución para cada uno de los árboles del bosque [Figura 2.12].

Ventajas

Las ventajas de los *Random Forest* son:

- Es uno de los algoritmos de aprendizaje más fiables.
- Funciona bien con conjunto de datos muy grandes.
- Puede manejar cientos de variables de entrada.
- Guarda la información sobre las variables más importantes.

Desventajas

- No funcionan bien cuando hay variables categóricas.
- Puede sobreajustar en ciertos grupos de datos con mucho ruido.

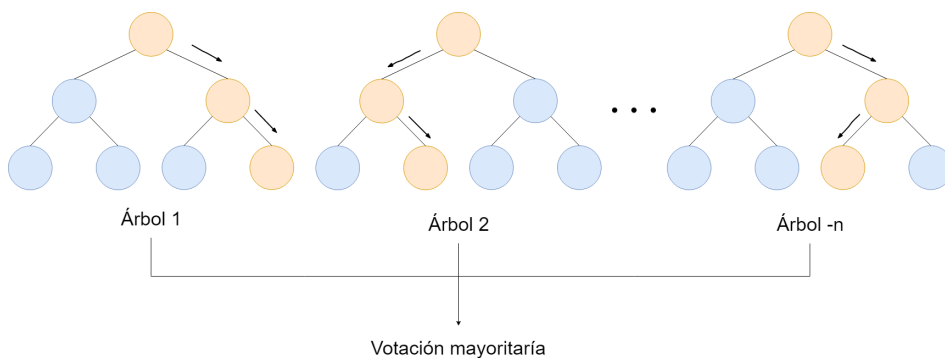


Fig. 2.12: Ejemplo de Random Forest

2.4.7 Clasificador por Votación [41]

La clasificación por votación es un meta-clasificador, es decir, no implementa un algoritmo de clasificación sino que evalúa las predicciones de otros algoritmos para obtener una nueva [Figura 2.13]. Este algoritmo se basa en la técnica de *Stacking* para obtener las predicciones.

Tipos de votación

- **Votación Dura/Mayoritaria:** Es un caso de selección por mayoría simple. La predicción se hace basándose en el mayor número de votos por parte de los algoritmos utilizados.
- **Votación Blanda:** Esta técnica calcula el mejor resultado obteniendo la media de las probabilidades calculadas por los algoritmos individualmente.

Ventajas

- Por norma general suelen proporcionar mejores resultados, si se rigen por ciertas condiciones, como que los clasificadores sean totalmente independientes [41].

Desventajas

- No todos los algoritmos son válidos para esta técnica, especialmente cuando se usa el método de votación blanda, ya que no todos los algoritmos estiman las probabilidades de las salidas.

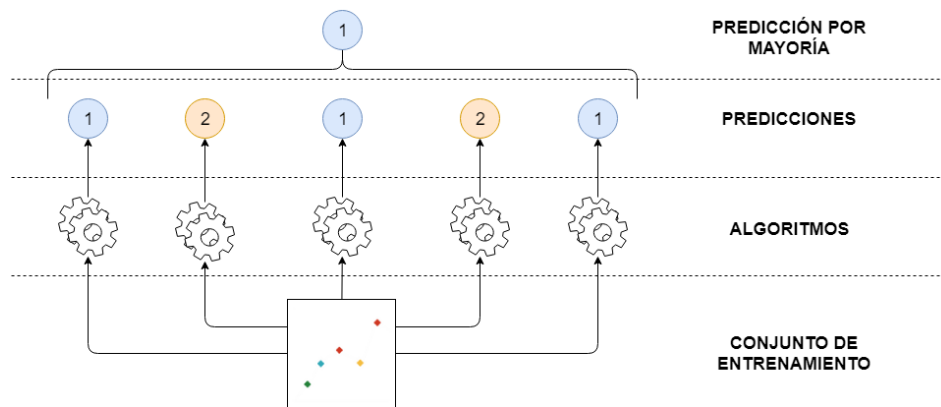


Fig. 2.13: Esquema clasificación por votación

2.5 Sistemas de gestión de colas

En 1999, **Andy Stanford-Clark** y **Arlen Nipper** inventaron el protocolo *Message Queueing Telemetry Transport (MQTT* [20]). Ellos necesitan un protocolo con el

mínimo consumo de batería y ancho de bando posible para enviar la información del estado de las tuberías de petróleo.

El desarrollo del protocolo permaneció bajo licencia de IBM hasta el 2010, donde fue liberado como proyecto de código abierto, y en 2014, fue oficialmente aprobado por la organización OASIS como un estándar.

El protocolo fue definido con un patrón de publicación/suscripción e incorpora una serie de elementos que son comunes en todas sus implementaciones:

- **Topic:** Es una cola sobre un tema particular, identificada por un nombre.
- **Particiones:** División de un *topic* en múltiples colas, que permite un mayor rendimiento.
- **Mensaje:** Es cada elemento que se almacena en un *topic*.
- **Broker:** En un nodo identificado por un ID dentro de un *clúster*, actúa como servidor permitiendo replicar y balancear el *clúster*, para que sea escalable y tolerante a fallos.
- **Productor:** Agente que permite a una aplicación la capacidad de publicar mensajes en una cola.
- **Consumidor:** Agente que permite a una aplicación la capacidad de suscribirse a una cola para consumir los mensajes.

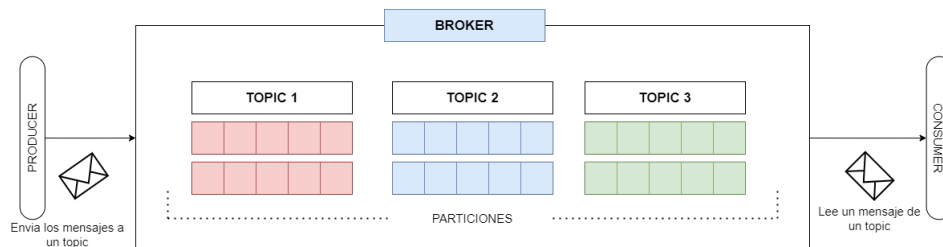


Fig. 2.14: Esquema de gestión de colas

Debido a las ventajas de este protocolo se utiliza mucho en dispositivos *Internet of Things* (IoT) y muchas empresas han implementado su propio software basando en este protocolo, una de las más destacadas es la implementación de *Eclipse Mosquitto* [24].

En 2003, **John O'Hara** (J.P. Morgan) creó el *Advance Message Queue Protocol* (AMQP) [34, 46]. Durante los siguientes años se fueron incorporando empresas para continuar el desarrollo y la documentación del protocolo.

La versión 1.0 de *AMQP* fue lanzado en 2011, en una conferencia en Nueva York. En 2014 fue aprobada y recibió la designación como ISO/IEC 19464. La implementación sigue la misma estructura que la de *MQTT* y su principal diferencia es que *AMQP* sigue el patrón de cliente/servidor. El software más popular que implementa dicho protocolo es *RabbitMQ* [34, 46]. Este software inicialmente implementaba solo el soporte de *AMQP*, pero que con el paso de los años también han dado soporte a *MQTT*.

En 2010, el equipo de LinkedIn desarrollo *Kafka* [27] para poder procesar la gran cantidad de mensajes que se enviaban dentro de la plataforma. El equipo de trabajo tuvo en mente utilizar los protocolos *MQTT* y *AMQP*, pero estos protocolos no estaban pensados para realizar un procesamiento de estos mensajes y esto era algo crítico para LinkedIn.

En 2011, LinkedIn reportó que su sistema de gestión de colas soportaba el procesamiento de mil millones de mensajes al día. Actualmente el desarrollo de *Kafka* es mantenido por la *Apache Foundation*.

Estado del arte

Se pueden encontrar diferentes trabajos [8, 12] que han realizado estudios preliminares para analizar la capacidad de las características de interacción del usuario con fines de autenticación, pero no abordan la cuestión de la autenticación continua. Estos trabajos han analizado la interacción sobre los principales tipos de dispositivos habituales: teclado, ratón y pantalla táctil.

La autenticación basada en ratón ha sido abordada por diferentes estudios en los últimos años. En [3], se analizó la autenticación a través de dígrafos y trígrafos de pulsaciones de teclas y diferentes acciones del ratón (movimiento del ratón, arrastrar y soltar, apuntar y hacer clic, sin movimiento) mediante redes neuronales obteniendo resultados prometedores. Los mismos autores continuaron su trabajo anterior [6] realizando un análisis más profundo de las características de comportamiento del ratón para la autenticación mediante redes neuronales, que obtuvo resultados consistentes.

En [35], se propuso la reautenticación de usuarios basada en los movimientos del ratón, recogiendo los movimientos en bruto y extrayendo características como las distancias, los ángulos o la velocidad, que se utilizaron para alimentar árboles de decisión. Sin embargo, dicho trabajo seguía un enfoque inusual al crear un modelo de usuario específico por aplicación, lo cual no es escalable, ya que el número de modelos necesarios para un sistema de autenticación de propósito general se volvería rápidamente inmanejable.

El trabajo presentado en [44] aborda una cuestión ligeramente diferente, la autenticación estática de usuarios, es decir, la autenticación primaria durante la fase de inicio de sesión (contraseña gráfica). Para ello, utilizaron un modelo de red neuronal que fue alimentado con características de comportamiento del ratón basadas en gestos, de manera que el usuario dibujó una serie de gestos durante la fase de inscripción que se utilizaron para el entrenamiento y luego, durante la fase de verificación de la identidad del usuario (*login*), se le pedía que replicara esos gestos y se los comparaba con el modelo de referencia. Los resultados mostraron que estos métodos podrían ser también adecuados para los procesos de autenticación estática.

En [50], se analizaron métricas basadas en el ángulo de los movimientos del ratón para verificar la identidad de los usuarios a través de máquinas de soporte vectorial.

Sin embargo, la mayoría de ellos se limitan solo a escenarios controlados o incluso a condiciones particulares, como el seguimiento de la interacción solo durante la fase de inicio de sesión o la autenticación primaria mediante una contraseña gráfica. En este proyecto se destaca la importancia de un sistema de autenticación funcional, que opere de forma autónoma y completamente online, bajo entornos no controlados.

Planificación y Metodología

Todo proyecto debería de disponer de una planificación inicial para establecer un guión a seguir, asegurando el cumplimiento de los plazos, costes y calidad del producto final. A lo largo de este capítulo se detallará la planificación realizada para llevar a cabo el proyecto.

En un primer punto se detallarán los recursos que han participado y las tareas del las tareas del proyecto. Con estos datos podemos establecer la planificación inicial creando para ello su línea base¹. A la finalización del proyecto se compararán los resultados estimados con los reales, con el fin de obtener los puntos críticos e intentar evitarlos en futuros proyectos.

En la última sección del capítulo se realizará un análisis de riesgos, exponiendo que riesgos podrían presentarse y, para los más críticos realizar una planificación, para que en el caso de ocurrir podamos reducir o evitar su impacto.

4.1 Recursos y actividades

Para la elaboración de cualquier proyecto debemos conocer que recursos participarán en él y cual es su disponibilidad. En la Tabla 4.1 podemos observar los recursos que participarán en el proyecto.

Una vez definidos los recursos procedemos a describir las tareas y desglosarlas en tareas más pequeñas con el fin de gestionarlas mejor. A continuación se muestran las tareas y subtareas con una breve descripción:

1. Estudio de viabilidad del proyecto

- **Estudio de estándares y propuestas:** Se pretende analizar la situación actual, estudiando estándares y propuestas similares disponibles para identificar las ventajas que podremos aplicar a nuestro proyecto.

¹Es una foto fija de la planificación efectos de comparación

Recurso	Coste
Programador Front End	€16/hora
Programador Back End	€16/hora
Programador Data Science	€20/hora
Escritor	€14/hora
Analista	€22/hora
Diseñador	€22/hora
Director 1	€25/hora
Director 2	€25/hora
Servidor	€0.013/hora
Servidores para entrenamiento IA	€0.7/hora
Ordenador	€650

Tab. 4.1: Tabla de recursos [1]

- **Análisis de las diferentes tecnologías existentes:** Se estudiará el estado del arte de las tecnologías actuales con el fin de determinar aquellas que mejor se ajustan a las características de nuestro proyecto.
- **Documentación del proyecto:** Elaborar la memoria del proyecto y otra documentación sobre instalación y configuración del proyecto.

2. Gestión del proyecto

- **Análisis de requisitos:** Se estudiarán los requisitos del software a desarrollar, atendiendo a las diferentes necesidades del proyecto y estableciendo una línea de trabajo que nos garantice la finalización del proyecto.
- **Análisis de riesgos:** Se llevará a cabo una identificación de aquellos riesgos que puedan surgir durante el desarrollo del proyecto, estableciendo un plan para su prevención y mitigación, de forma que se garantice el cumplimiento de los objetivos planteados.
- **Especificación del proyecto:** Se asentarán los conocimientos adquiridos durante las fases previas, definiendo la arquitectura, los componentes y las tecnologías que emplearemos durante el desarrollo del proyecto para su consecución.
- **Seguimiento del proyecto y gestión de riesgos:** Este proyecto se realizará siguiendo una metodología de desarrollo iterativa incremental, en la que se establecerán diferentes fases y se llevará a cabo un seguimiento periódico del mismo.

3. Desarrollo del software de captura de datos

- **Desarrollo de la aplicación captura:** Uno de los aspectos más importantes de la experimentación consiste en la obtención de una muestra de

datos representativa que lo posibilite. Durante esta fase, se desarrollará un sistema que permita recopilar la información procedente del ratón.

- **Distribución del software:** Se realizará una campaña de distribución del software desarrollado en la fase previa para poder recopilar los datos de aquellos usuarios a los que se les ha proporcionado.
- **Obtención de datos:** Durante este período se recopilará un volumen de datos suficiente para poder llevar a cabo la elaboración de los diferentes modelos de autenticación en la siguiente fase. La calidad de la muestra recopilada determinará, en gran medida, el proceso a seguir en las siguientes fases.

4. Planteamiento e implementación de los algoritmos

- **Implementación de un conjunto de algoritmos de Inteligencia Artificial:** Se analizarán diferentes técnicas de Inteligencia Artificial que permitan llevar a cabo el proceso de autenticación para verificar la identidad de los usuarios. Para ello, será necesario realizar un adecuado tratamiento de los datos recopilados, de forma que nos permita definir una serie de características identificativas de los usuarios que puedan ser procesadas por la técnica o técnicas escogidas.
- **Entrenamiento del conjunto de algoritmos:** Se entrenarán con las técnicas seleccionadas, elaborando un perfil de autenticación único para cada usuario que permita verificar unívocamente si el comportamiento detectado se corresponde con el comportamiento del usuario autenticado al inicio de la sesión.
- **Optimización de los algoritmos de clasificación:** El sistema ha de ser capaz de manejar un gran volumen de información procedente de los eventos asociados al ratón. Por tanto, será necesario optimizar el proceso de entrenamiento y evaluación de estos modelos de forma que permitan llevar a cabo el proceso de autenticación en tiempo real y de forma continuada durante las sesiones.

5. Planteamiento e implementación sistema de colas

- **Creación y configuración de la infraestructura para el alojamiento del sistema:** Para llevar a cabo el proceso de monitorización continua del comportamiento de los usuarios es necesario disponer de un sistema que permita procesar en tiempo real el gran volumen de información generado. Para ello, se requiere de una infraestructura hardware y software capaz de soportar tal carga de trabajo y que, además, cumpla con los requisitos para implementar un sistema de colas en la siguiente etapa.

- **Implementación y configuración del sistema de colas:** Los eventos asociados al uso del ratón se procesarán mediante un sistema de colas que permita obtener unos tiempos de latencia bajos, propios de un sistema en tiempo real, para llevar a cabo el proceso de autenticación dinámicamente.
- **Integración de los sistemas de Inteligencia Artificial:** El proceso de autenticación se realizará dentro del propio sistema de colas sirviéndose de los perfiles de usuario basados en modelos de IA elaborados anteriormente, de forma que los eventos entrantes de la sesión activa se puedan autenticar en tiempo real.

6. Desarrollo del software de gestión

- **Desarrollo de una aplicación web para la administración del sistema:** El mecanismo de autenticación desarrollado en las fases previas precisa de un sistema que permita configurar adecuadamente los parámetros del mismo, como pueden ser la respuesta ante una autenticación negativa (bloqueo de la sesión de usuario, notificación al administrador del sistema responsable, etc.), así como también debe ofrecer la posibilidad de analizar el estado general del sistema de autenticación (tasa de éxitos, tasa de fallos, etc). Para ello se propone un entorno web que permita desarrollar estas tareas de manera deslocalizada.
- **Integración con el software de autenticación:** La aplicación web descrita anteriormente debe comunicarse de forma efectiva con el software de autenticación utilizado, tarea que se abordará en este punto.

4.2 Planificación Inicial

La planificación inicial estima como deberá de ejecutarse todo el proyecto, las relaciones entre las tareas y los recursos asignados a estas. En la Figura 4.1 se pueden ver todas las tareas del proyecto, sus dependencias y los recursos asignados.

4.3 Seguimiento del proyecto

En esta sección se detallará el seguimiento realizado sobre el proyecto a su finalización, ya que durante su transcurso se produjeron algunos retrasos.

Modo de	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
1	Hito: Inicio Proyecto	1 día?	01/02/2021	01/02/2021		
2	PT 2.4: Seguimiento del proyecto y gestión de riesgos	267.5 días	02/02/2021	10/02/2022		Servidor
3	PT 1.3: Documentación del proyecto	267.5 días	02/02/2021	10/02/2022		Escritor
4	PT 1: Estudio de viabilidad del proyecto	30 días	02/02/2021	15/03/2021		
5	T1.1: Estudio de estándares y propuestas	15 días	02/02/2021	22/02/2021	1	Analista[90%]
6	T1.2: Análisis de las diferentes tecnologías existentes	15 días	23/02/2021	15/03/2021	5	Analista[80%], Director 1[20%], Director 2[20%]
7	PT 2: Gestión del proyecto	45 días	16/03/2021	17/05/2021		
8	T2.1: Análisis de requisitos	15 días	16/03/2021	05/04/2021	6	Analista[80%], Director 1[20%], Director 2[20%]
9	T2.2: Análisis de riesgos	15 días	06/04/2021	26/04/2021	8	Analista[80%], Director 1[20%], Director 2[20%]
10	T2.3: Especificación del proyecto	15 días	27/04/2021	17/05/2021	9	Analista[80%], Director 1[20%], Director 2[20%]
11	PT 3: Desarrollo del software de captura de datos	87.5 días	18/05/2021	16/09/2021		
12	T3.1: Desarrollo de la aplicación captura	20 días	18/05/2021	14/06/2021	10	Programador Front End[80%]
13	T3.2: Distribución del software	30 días	15/06/2021	26/07/2021	12	Programador Front End[80%]
14	T3.3: Obtención de datos	60 días	24/06/2021	16/09/2021	13CC+25%	
15	PT 4: Planteamiento e implementación de los algoritmos	60 días	05/08/2021	28/10/2021		Servidores para entrenamiento IA
16	T4.1: Implementación de un conjunto de algoritmos de Inteligencia Artificial	20 días	05/08/2021	02/09/2021	14CC+50%	Analista[80%]
17	T4.2: Entrenamiento del conjunto de algoritmos	30 días	19/08/2021	30/09/2021	16CC+50%	Analista[80%]
18	T4.3: Optimización de los algoritmos de clasificación	20 días	30/09/2021	28/10/2021	17	Analista[80%]
19	PT 5: Planteamiento e implementación sistema de colas	50 días	28/10/2021	06/01/2022		
20	T5.1: Creación y configuración de la infraestructura para el alojamiento del sistema	20 días	28/10/2021	25/11/2021	18	Programador Back End[80%]
21	T5.2: Implementación y configuración del sistema de colas	20 días	25/11/2021	23/12/2021	20	Programador Back End[80%]
22	T5.3: Integración de los sistemas de Inteligencia Artificial	10 días	23/12/2021	06/01/2022	21,17FF	Programador Back End[80%]
23	PT 6: Desarrollo del software de gestión	25 días	06/01/2022	10/02/2022		
24	T6.1: Desarrollo de una aplicación web para la administración del sistema	20 días	06/01/2022	03/02/2022	22	Programador Front End[80%], Diseñador[20%]
25	T6.2: Integración con el software	5 días	03/02/2022	10/02/2022	24	Programador Front End[80%]
26	T1.3: Documentación del proyecto	10 días	10/02/2022	24/02/2022	25	Escritor[25%]
27	Hito: Fin Proyecto	1 día?	24/02/2022	25/02/2022	26	

Fig. 4.1: Lista de tareas planificadas

La Tabla 4.2 muestra la desviación entre la planificación prevista inicialmente y los datos reales alcanzados al final del proyecto, donde se puede observar un retraso de 50 días. Este retraso se debió en gran medida a que los recursos asignados al proyecto tuvieron que dedicarle tiempo a otro proyecto simultáneamente, y por lo tanto solo se observa un retraso en el tiempo y no en costes directos.

	Estimación	Real
Fecha de Inicio	01/02/2021	01/02/2021
Fecha de Fin	17/12/2022	25/02/2022
Trabajo	6,744 horas	6,744 horas
Duración	229.5 días	279.5 días
Costo	€69,279.00	€69,279.00
Variación		50 días

Tab. 4.2: Costes y tiempos del proyecto

En la Figura 4.2 se puede ver el diagrama de Gantt del proyecto una vez finalizado.

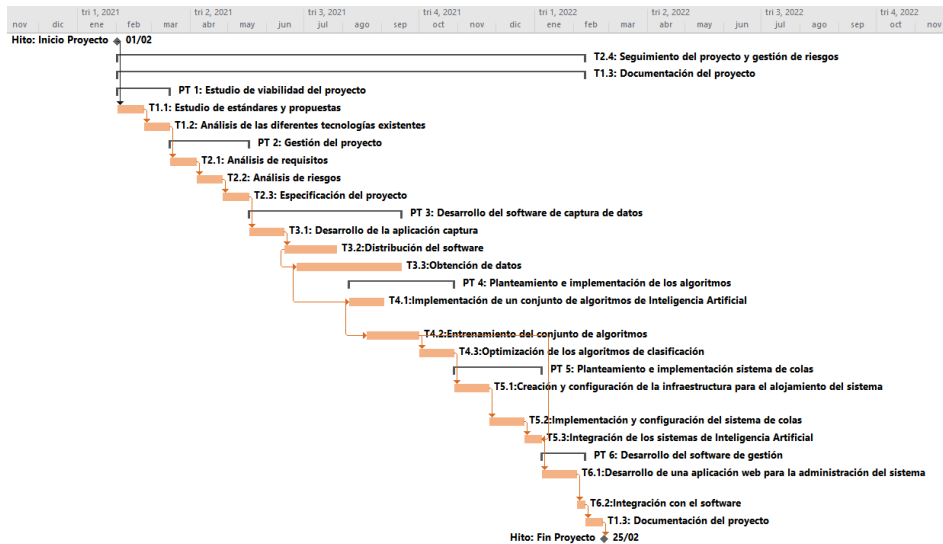


Fig. 4.2: Diagrama de Gantt

4.4 Análisis de riesgos

A la hora de planificar un proyecto hay que tener en cuenta las posibles situaciones de riesgo que pueden afectar a la planificación prevista para el proyecto, tanto en términos de tiempo como de coste.

A la hora de preparar un plan de riesgos se siguen ciertos pasos ya establecidos:

- **Identificación:** Elaborar una lista de posibles riesgos.
- **Valoración:** Cuantificar los riesgos para conocer el impacto que tendrían.
- **Análisis:** Estudiar alternativas y crear planes de prevención y contención.

Para este proyecto se han identificado algunos riesgos [Tabla 4.3] que en caso de suceder podrían retrasar el proyecto.

Para algunos de ellos se ha elaborado un plan de contingencia para minimizar su impacto en caso de que ocurran [Tabla 4.4].

4.5 Metodología

Con el fin de seguir la planificación planteada y conseguir alcanzar los objetivos definidos hemos elegido una metodología de desarrollo iterativa incremental. Esta

Nombre	Descripción	Prob.	Impacto
Conocimiento del dominio	Parte del dominio del proyecto es desconocido para el autor.	Alto	Medio
Configuración del sistema	La configuración de cada una de las partes del servidor puede llegar a ser bastante complicada en ciertos puntos y generar conflictos.	Alto	Medio
Caídas de servidores	Los servidores pueden sufrir problemas.	Media	Bajo
Análisis de Datos	Buscar características que sean identificativas del usuario.	Alto	Alto

Tab. 4.3: Tabla de riesgos

Riesgos	Plan de Gestión
Configuración del sistema	Tener una copia de seguridad actualizada con toda la configuración y servicios necesario, para poder volver a una configuración estable.
Caída de servidores	Tener automatizado el despliegue del servidor para desplegar un nuevo servidor, pudiendo continuar el desarrollo en local o en otro servidor.

Tab. 4.4: Tabla de gestión de riesgos

metodología ágil nos permitirá optimizar la realización de un proyecto sobre tecnologías poco conocidas y con cambios frecuentes debido al análisis que requieren ciertas partes del proyecto.

4.5.1 Desarrollo incremental

Este método consiste en una serie de iteraciones, en ventanas de tiempo, en las que al final de las mismas tendremos una parte del producto que el cliente podrá revisar y posteriormente mejorar y/o corregir [Figura 4.3]

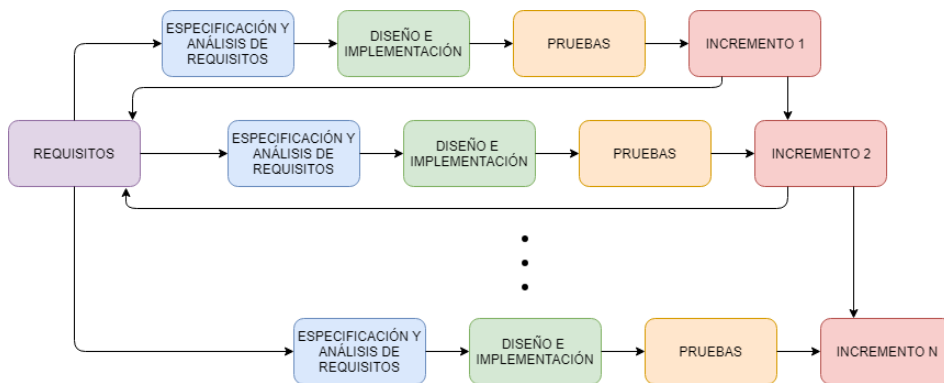


Fig. 4.3: Fases del modelo incremental

Esta metodología exige tener dos grupos de usuarios:

- **Cliente²:** Se encarga de revisar el producto al final de las iteraciones.
- **Desarrollador:** Se encarga de desarrollar el producto.

Las principales razones por las que se ha elegido esta metodología frente a otras existentes son:

- **El cliente no sabe exactamente lo que necesita:** Al inicio del proyecto se tenía una idea general del proyecto pero al tratarse de un proyecto con tantas partes diferenciadas los cambios, tecnologías, conexiones ... podrían sufrir cambios que implicasen redefinir ciertos conceptos.
- **Obtener un producto usable:** Este proyecto tiene una parte crítica, la de obtener el conjunto de datos lo más rápido posible. Esta parte implica tener una aplicación y el sistema de guardado de información lo antes posible para no bloquear el proyecto.

²Al ser un proyecto de fin de máster, no existe un cliente como tal, por lo que se ha decidido que los directores del proyecto tomen el rol de clientes.

Tecnologías

En este capítulo se detallará la estructura, tecnologías y decisiones tomadas en el desarrollo del proyecto.

5.1 Estructura final

Las decisiones de las que se hablarán en este capítulo tienen como finalidad obtener la estructura final del sistema. Esta estructuración tiene como finalidad generar un sistema escalable tanto horizontalmente, es decir, que cada uno de los módulos puede ser redimensionado sin afectar al resto, p. ej. distribuyendo cada uno de los módulos en un servidor, y verticalmente, p. ej. aumentando los recursos de un servidor concreto para mejorar el rendimiento.

La estructura final y los detalles de las conexiones entre los diferentes subsistemas que se pretende conseguir se pueden ver en la Figura 5.1.

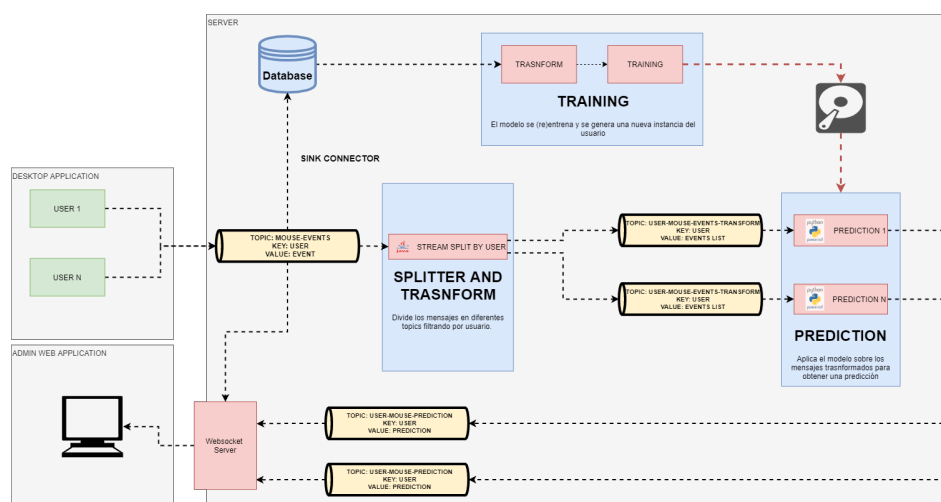


Fig. 5.1: Esquema detallado del sistema de streaming

5.2 Arquitectura

En el desarrollo de un proyecto de software es necesario definir una arquitectura que nos permita crear un sistema robusto y escalable.

En este proyecto se ha utilizado una arquitectura de tres capas [Figura 5.2] que se distribuye de la siguiente manera:

- **Capa de presentación:** Es la encargada de presentar y capturar la información del usuario.
- **Capa de negocio:** Es donde residen los programas que se ejecutan, recibe las peticiones del usuario.
- **Capa de datos:** Es donde residen los datos y es la encargada de acceder a los mismos.



Fig. 5.2: Arquitectura de tres capas

5.3 Capa de Presentación

En esta sección se comentará las diferentes tecnologías usadas para desarrollar las dos aplicaciones que conforman esta capa.

5.3.1 Aplicación de captura de datos

Esta aplicación permite capturar los datos del ratón en todo el escritorio del usuario y enviarlos al sistema de colas. Para el desarrollo de la aplicación se decidió utilizar *Java* por las siguientes razones:

- **Multiplataforma:** Se puede ejecutar en la mayoría de sistemas operativos y dispositivos, con una única base de código.

- **Java Native Interface:** Permite ejecutar librerías nativas para interactuar con elementos de bajo nivel.

La aplicación se ejecuta en segundo plano capturando los datos y enviándolos al servidor. Además, la aplicación cuenta con un menú contextual activar y desactivar el servicio de captura [Figura 5.3].

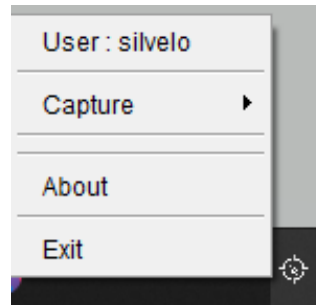


Fig. 5.3: Menú de la aplicación de escritorio.

5.3.2 Aplicación de administración

Esta aplicación permite mostrar el estado del sistema en tiempo real. Para el desarrollo de la aplicación se decidió usar el framework *Angular* por las siguientes razones:

- **Estructuración:** Proporciona un sistema por módulos que ayuda a organizar la aplicación separándolas por componentes.
- **Mantenimiento:** Permite modificar o reemplazar componentes con nuevas mejoras de una forma ágil.

Para las conexiones que se realizan contra el servidor se utilizan *WebSockets* en vez de *HTTP*, debido a las ventajas que ofrece en este tipo de sistemas:

- **Alto rendimiento:** Proporciona un canal de comunicación bidireccional, lo que nos permite enviar y recibir gran cantidad de datos de forma más eficiente.
- **Ahorro recursos:** Solo es necesario establecer una conexión y mantenerla abierta a la espera de recibir los datos. El establecimiento de la conexión tiene un coste más bajo que una conexión *HTTP*.

Los mensajes que se obtienen por *WebSockets* [37] se almacenan de manera local utilizando el patrón *Redux* [Figura 5.4]. Este patrón mantiene un objeto inmutable global en formato *JSON* y solo es modificable mediante las acciones predefinidas.

A pesar de que el uso de este patrón tiene un impacto negativo en el rendimiento, proporciona algunas ventajas a tener en cuenta:

- **Acceso global:** Es accesible desde cualquier punto de la aplicación. Esto permite crear componentes que pueden obtener los datos sin depender de otros.
- **Inmutable:** Solo se puede modificar con las acciones creadas por el desarrollador, dando un control exhaustivo sobre que modificaciones se pueden llevar a cabo.
- **Precarga:** El objeto creado permanece en el tiempo, haciendo que en visitas posteriores los datos se encuentren precargados y listos para mostrar.

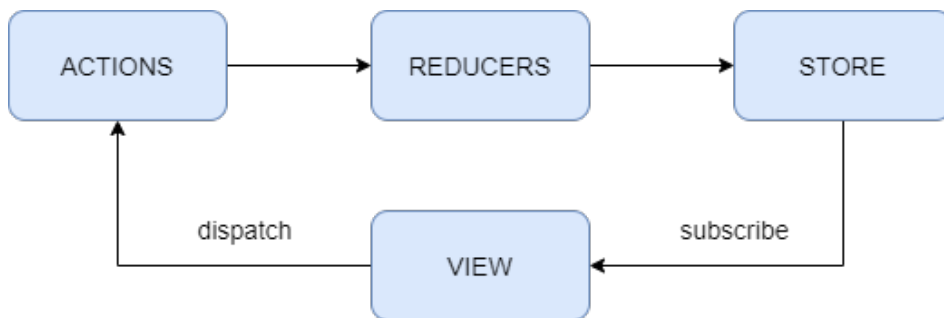


Fig. 5.4: Patrón Redux

Por defecto, *JavaScript* utiliza un único hilo y este se utiliza para el procesamiento de los datos y dibujar de la interfaz. Esto implica que al procesar una gran cantidad de datos, el hilo y la interfaz de usuario quedan bloqueados durante ese tiempo. Por ello, se planteó el uso de *WebWorker* que permite crear un hilo para ejecutar el procesamiento de los datos, dejando el hilo principal para la interfaz de usuario. Este planteamiento mejora el rendimiento y ofrece una experiencia fluida en la interfaz de usuario.

Angular

Este framework sigue el patrón Modelo-Vista-Vista-Modelo [Figura 5.5], el cual es una modificación del patrón Modelo-Vista-Controlador que se produce por el *two-way-binding* que se da en *Angular*.

La aplicación desarrollada permite ver la información del sistema en tiempo real. La Figura 5.6 muestra la pantalla principal donde se puede ver una gráfica con los eventos que llegan al servidor a tiempo real.

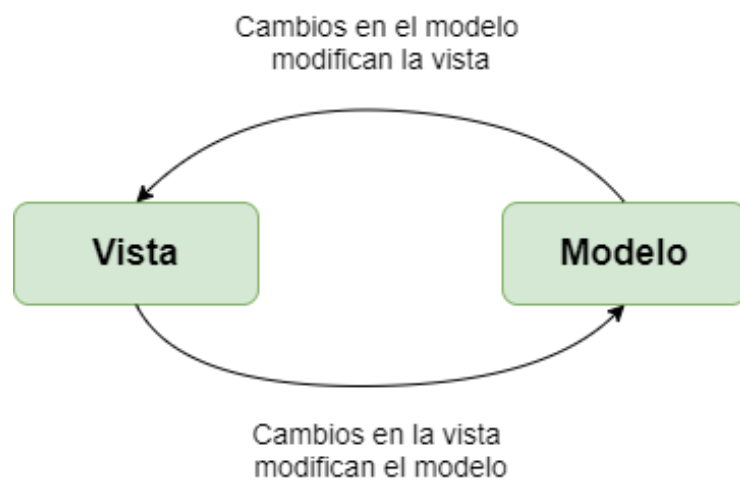


Fig. 5.5: Patrón modelo-vista-vista-modelo

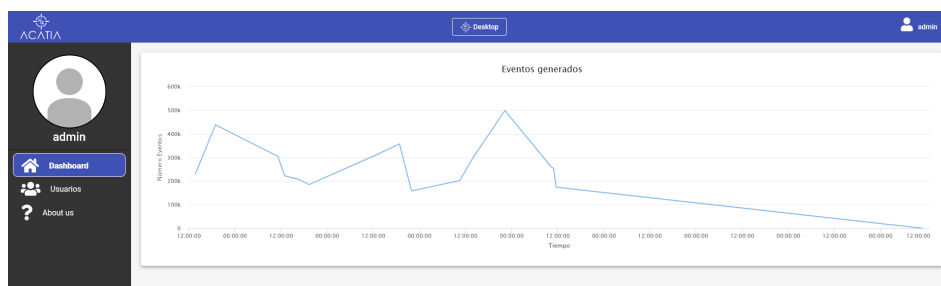
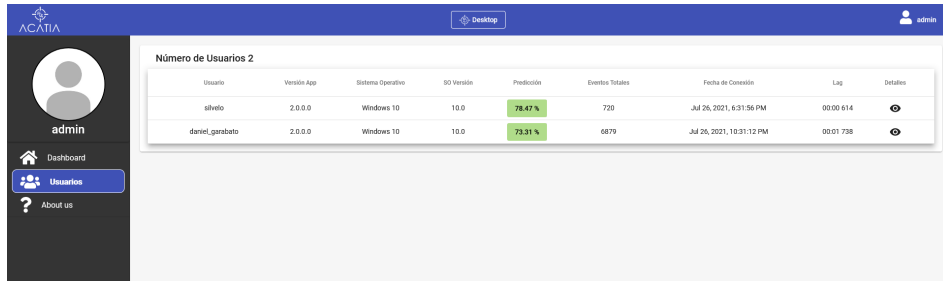


Fig. 5.6: Ventana con la gráfica de eventos generados

En otra pantalla de la aplicación se listan los usuarios que hay en el sistema, pudiendo observar de un solo vistazo cual es el estado de cada uno de ellos y acceder al detalle del usuario [Figura 5.8]. Esta ventana de detalle muestra una conjunto de gráficos con el ratio de autenticaciones satisfactorias y el tiempo de latencia de los eventos. Esta información ayuda al administrador del sistema a responder en tiempo real ante una incidencia.



Usuario	Versión App	Sistema Operativo	SO Versión	Predicción	Eventos Totales	Fecha de Conexión	Lag	Detalles
silvelo	2.0.0.0	Windows 10	10.0	78.47 %	720	Jul 26, 2021, 6:31:56 PM	00:00:614	
daniel_gonzalez	2.0.0.0	Windows 10	10.0	73.31 %	6879	Jul 26, 2021, 10:31:12 PM	00:01:738	

Fig. 5.7: Ventana de la lista de usuarios

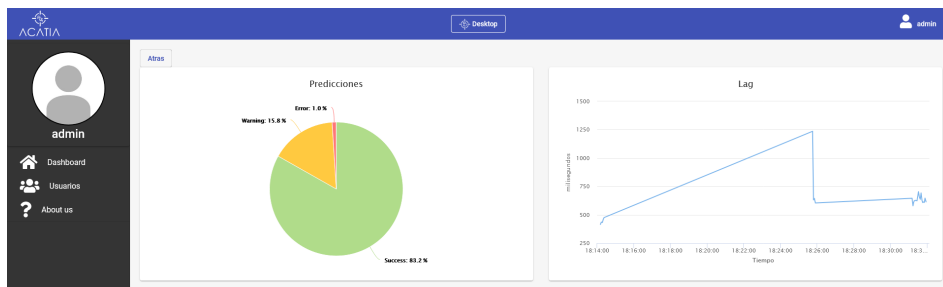


Fig. 5.8: Ventana de estadísticas del usuario

5.4 Capa de Negocio

La capa de lógica de negocio es la encargada de procesar los eventos y obtener las predicciones. Esta capa esta compuesta de un conjunto de tres servicios:

- **Sistema de gestión de colas:** Es el encargado de manejar los eventos y distribuirlos entre los diferentes servicios del sistema. El servidor encargado de manejar los eventos y distribuirlos entre los diferentes aplicativos finales.
- **Machine Learning:** Se el encarga de manejar todos los procesos relacionados con la Inteligencia Artificial, como la generación de modelos y la obtención de predicciones.
- **Orquestador:** Este servicio permite automatizar los procesos de generación de modelos y creación de máquinas.

El esquema general de este sistema se puede ver en la Figura 5.9, en él se observan las conexiones entre los diferentes servicios y capas, además se puede ver la composición de las máquinas de cada servicio.

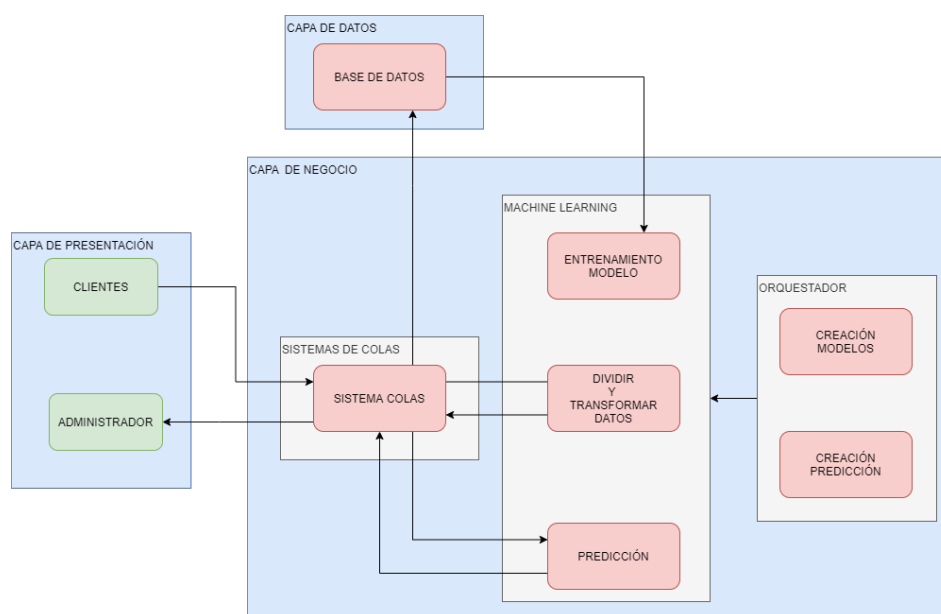


Fig. 5.9: Esquema de la capa de negocio

Esta estructuración se realizó pensando en la estabilidad del sistema, permitiéndonos aumentar el número de nodos de un servicio independientemente de otro.

5.4.1 Docker

Es una plataforma de virtualización a nivel de sistema operativo que nos permite crear una aplicación y empaquetarla junto con sus dependencias y librerías en un contenedor. Esta virtualización nos ofrece una serie de ventajas a la hora de crear y desplegar las aplicaciones:

- **Autocontenido:** El contenedor creado contiene todas las dependencias necesarias para la ejecución del programa, lo que implica que los servidores solo deben tener instalado *Docker*.
- **Mayor consistencia:** Las pruebas se hacen dentro de un contenedor y el despliegue es el mismo. Esto implica que el entorno es idéntico en ambos casos.
- **Ligeras:** Las instancias creadas con *Docker* son más ligeras que un sistema de virtualización tradicional.

- **Actualizaciones:** La actualización de los contenedores se realiza de forma sencilla, actualizando solo la versión de la imagen creada.
- **Aislamiento:** *Docker* garantiza que cada contenedor se encuentre aislado por defecto, garantizando que tenga sus propios recursos.

5.4.2 Sistemas de gestión de colas

Este servicio se encarga de distribuir los mensajes entre los distintos aplicativos del sistema. Para ello se decidió utilizar el patrón publicación-suscripción [Figura 5.10] que nos permite manejar una gran cantidad de datos en tiempo real.

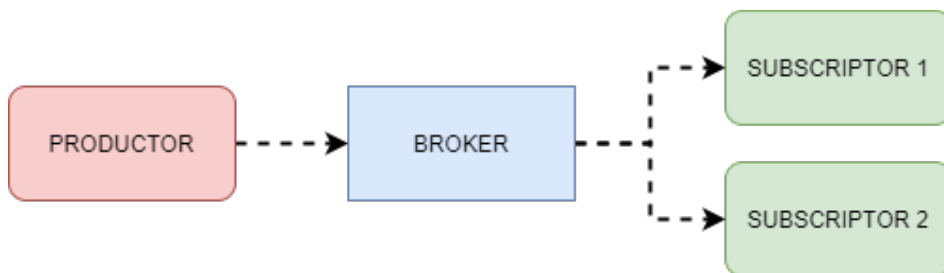


Fig. 5.10: Esquema Publicador/Suscriptor

En el mercado existen diferentes productos que implementan este patrón y permiten desplegar un servidor, de los que destacan:

- **RabbitMQ [34, 46]:** Es un proyecto de código abierto que implementa el estándar de mensajería *AMQP* para casos de uso de colas de baja latencia.
- **Kafka [27]:** Es una plataforma de transmisión de eventos distribuida de código abierto y uno de los proyectos más activos de Apache.

	Kafka	RabbitMQ
Rendimiento máximo	605 MB/s	38 MB/s
Latencia	5 ms	1 ms ¹

Tab. 5.1: Latencias y rendimiento de los distintos productos [33]

De las opciones disponibles se optó por *Kafka* por las siguientes razones:

- **Flujo de datos:** Nuestra aplicación genera 400 mensajes por usuario y segundo, ambas opciones deberían soportar bien este flujo de datos, pero *Kafka* ofrece un mejor rendimiento.

¹Las latencias de RabbitMQ se degradan significativamente a velocidades superiores a los 30 MB/s

- **Streams:** *Kafka Streams* es una tecnología que nos permite procesar los datos realizando transformaciones y divisiones.
- **Consumidores:** Nuestra aplicación necesita que algunos datos sean consumidos por varios procesos y en este caso *Kafka* nos permite hacerlo de forma nativa.

5.4.3 Machine Learning

Este proceso se encargará de toda la parte de Inteligencia Artificial que engloba los siguientes procesos:

- **Entrenamiento modelo:** Genera el modelo de Inteligencia Artificial de un usuario.
- **Transformación de datos:** Realiza la transformación de los datos en bruto.
- **Predicción:** Obtiene la predicción del modelo.

Para este sistema se utilizó el lenguaje de programación *Python* debido a la versatilidad que ofrece en el análisis de datos y *Java* para el proceso de transformación dada su compatibilidad con *Kafka Streams*.

5.4.4 Orquestador

Este proceso se encarga de automatizar el proceso de creación de modelos y máquinas de predicción haciendo uso de las API de *Docker* y *Kafka*. La Figura 5.11 muestra la estructura y en ella podemos como el proceso de orquestación obtiene los datos de las API y se encarga de realizar las llamadas correspondientes a los subprocessos de generación de modelos y creación de la máquina de predicción:

- **Generación de modelos:** Accede a la base de datos para obtener los eventos del usuario, generar el modelo correspondiente a ese usuario y guardar ese modelo en disco.
- **Máquina de predicción:** Configura una nueva máquina basándose en los parámetros del usuario y modelo, desplegándola automáticamente sin interacción por parte de un administrador.

Este sistema ejecuta un proceso que mapea las máquinas y modelos existentes con los usuarios, obteniendo en el proceso la relación de usuarios que no disponen de modelos o máquinas de predicción. El flujo de trabajo de este proceso se puede

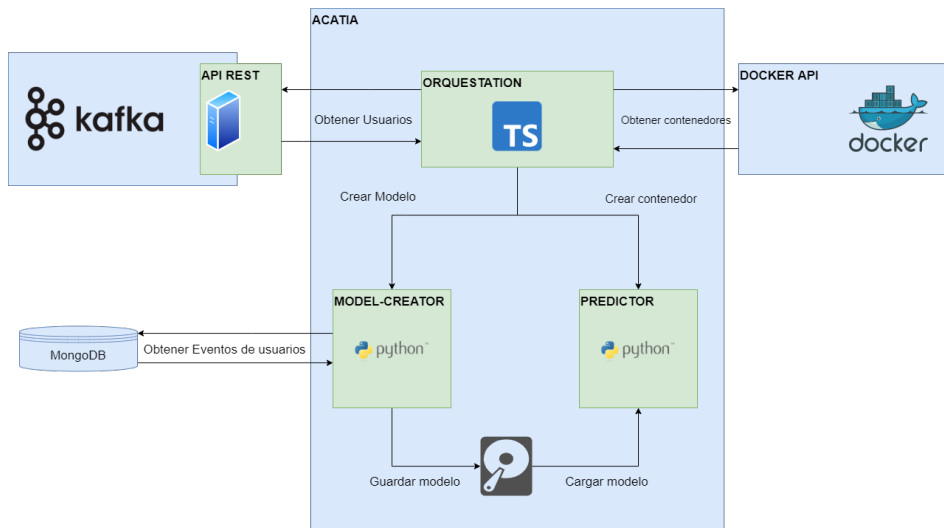


Fig. 5.11: Workflow del sistema de orquestación

ver en detalle en las Figuras 5.12 y 5.13, donde se puede observar la lógica de la creación de un modelo y las diferentes comprobaciones que se llevan a cabo para que dicho modelo tenga una calidad aceptable.

5.4.5 Capa de Datos

Esta capa se utilizará para almacenar los datos de entrada de manera persistente, para realizar operaciones en el futuro como creación o actualización de nuevos modelos, búsqueda de nuevas características o técnicas de inteligencia artificial.

Como base de datos se ha decidido usar *MongoDB*, una base de datos NoSQL. Los motivos por los que se ha elegido este tipo de base de datos frente a otras (relacional) son:

- **Velocidad:** Una de las principales ventajas de esta base de datos es la prioridad en manejar lecturas, y en este caso es perfecto para nuestro sistema, debido a que la mayor parte de accesos serán de lectura.
- **Volumen:** La aplicación de captura genera una gran cantidad de datos y por lo tanto, necesitamos una base de datos que funcione en sistemas de *Big Data*.
- **Variabilidad:** La aplicación manejará datos, que pueden no ser consistentes e incluso algunos de ellos pueden modificarse con el tiempo. Este tipo de base de datos no necesitan un esquema para trabajar por lo que las hace ideales para este tipo de casos.

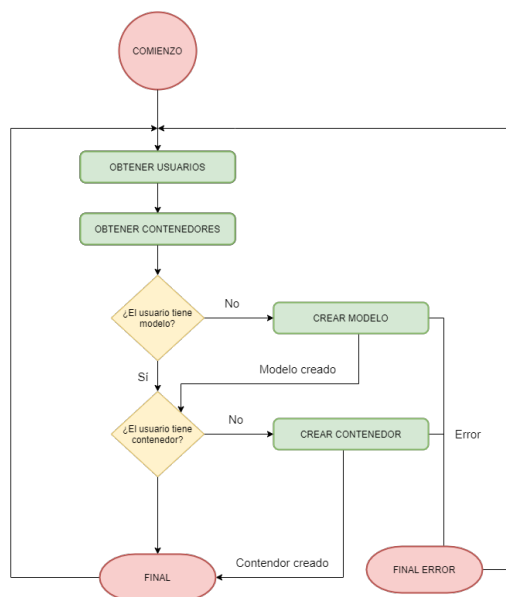


Fig. 5.12: Flujo del sistema de orquestación

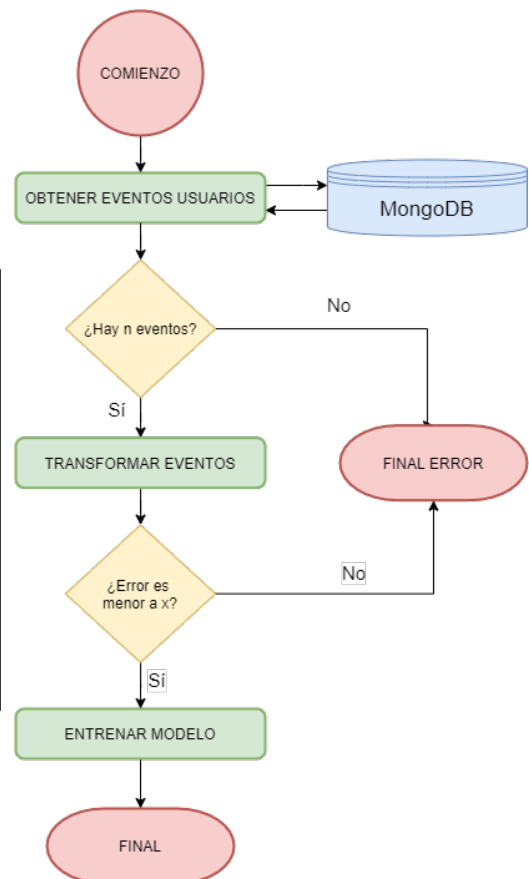


Fig. 5.13: Flujo de creación de modelos

ReplicaSet

Un *ReplicaSet* en *MongoDB* [Figura 5.14] es un grupo de instancias que mantienen el mismo conjunto de datos. Este sistema permite tener una alta disponibilidad y redundancia de datos.

En el proyecto se crearán tres instancias de *MongoDB* donde la aplicación se conectará a la primaria y las otras instancias se sincronizarán con ella. Una de las ventajas que presenta este sistema es que permite trabajar de manera distribuida, de modo que si un nodo se cae el resto puede asumir el trabajo y el sistema no se ve afectado.

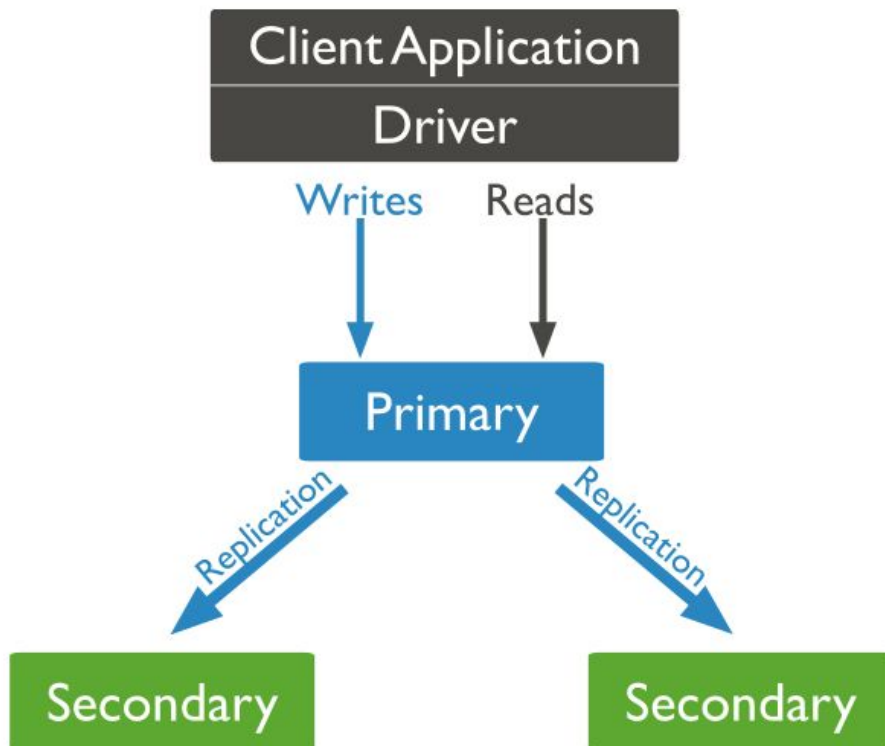


Fig. 5.14: Estructura replicaSet [38]

Inteligencia Artificial

En este capítulo se detallarán los pasos seguidos respecto a la parte de Inteligencia Artificial, el análisis y procesamiento de los datos.

6.1 Selección de características

Una vez obtenidos los suficientes datos de los usuarios, se han analizado y tratado con el fin de encontrar posibles valores incorrectos y buscar las mejores características para la predicción.

Cada *input* de un usuario contiene las características relativas a la posición del ratón [Tabla 2.1]. Los datos son procesados para obtener el conjunto de características más complejas comentadas en la Tabla 2.2 y que se usarán para obtener los modelos finales.

Con las características obtenidas se ha procedido a seleccionar las mejores para intentar optimizar el rendimiento de los algoritmos. Para ello se han utilizado varias técnicas:

- **Correlación:** Consiste en obtener la relación entre cada característica y eliminar aquellas que poseen un alto nivel [Figura 6.1]. En nuestro caso se ha tomado 0.8 como el límite máximo de esa relación.
- **RFE (Recursive Feature Elimination):** Esta técnica elimina en cada iteración las características menos importantes, obteniendo en cada una de ellas la predicción [Figura 6.2] y el número de características óptimas para nuestro algoritmo. En nuestro caso se han obtenido los valores optimizándolos para las métricas *precision* y *recall*, empleando el algoritmo *Random Forest*, debido a su capacidad de proporcionar un valor de importancia para cada característica.

Una vez aplicadas las dos técnicas se obtienen los resultados de rendimiento para cada caso, y de esta forma, analizar los resultados obtenidos en función de las métricas de *precision* y *recall* [Tabla 6.1].

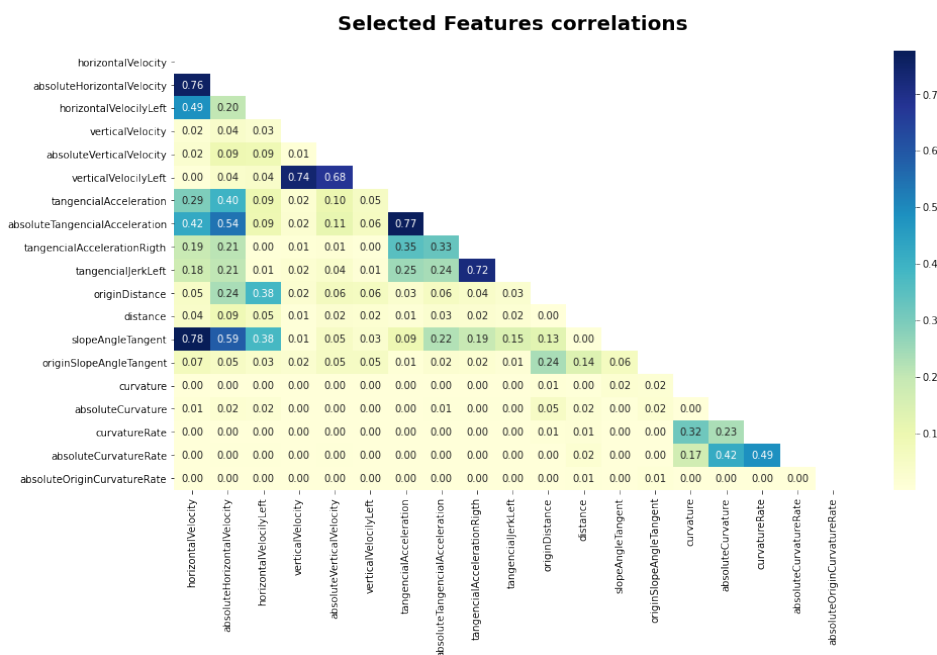


Fig. 6.1: Matriz de correlación de la selección de características

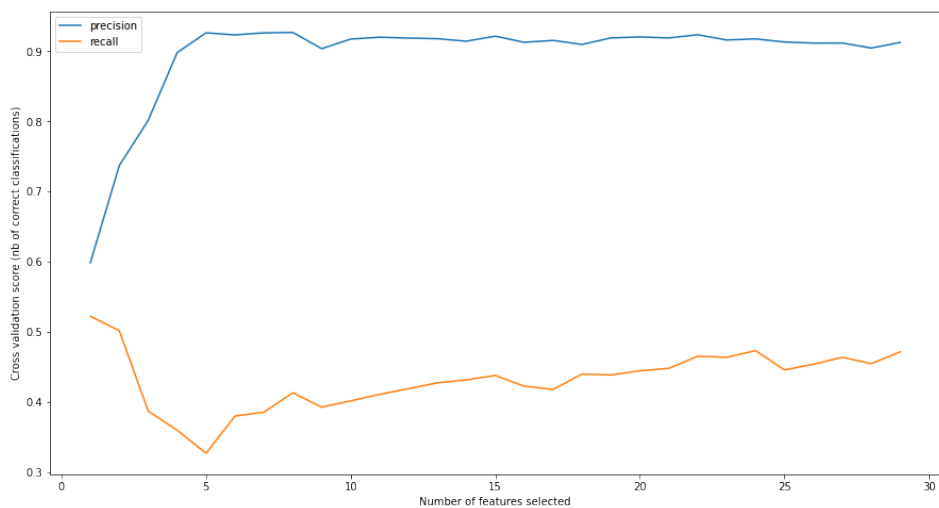


Fig. 6.2: Gráfica de eliminación recursiva de características

Tipo	# características	Precision	Recall	Tiempo
Correlation	20	0.914 (± 0.085)	0.435 (± 0.085)	3.443 (± 0.090)
RFE Precision	8	0.903 (± 0.109)	0.432 (± 0.109)	4.399 (± 0.291)
RFE Recall	1	0.920 (± 0.074)	0.450 (± 0.074)	4.294 (± 0.139)

Tab. 6.1: Resultados usando Random Forest

Los resultados muestran valores similares en todas las métricas. Donde más variación existe es en el número de características, y por ello, descartamos la opción de *RFE Recall*, ya que solo tiene en cuenta una característica (*distance*) para obtener la predicción.

De las dos opciones restantes observamos que con más características obtenemos unos mayores valores en ambas métricas, una menor desviación y un tiempo más bajo. Por estas razones se utilizarán las 20 características obtenidas del proceso de correlación.

6.1.1 Tratamiento de datos

El escalado de características a través de la estandarización puede ser un paso muy importante para muchos algoritmos de Inteligencia Artificial. La normalización implica reescalar las características para obtener las propiedades de una distribución normal con una media de cero y una desviación estándar de uno.

Para realizar el escalado se pueden emplear múltiples técnicas como:

- *StandardScaler*: Normaliza los datos eliminando la media y escalando los datos de forma que su varianza sea igual a 1.
- *RobustScaler* [Figura 6.3]: Escala los datos de acuerdo al primer y tercer cuartil.
- *PowerTransformer*: Aplica una transformación de potencia a cada característica para que los datos tengan una apariencia Gaussiana, estabilizando la varianza y minimizando la asimetría.

En este proyecto se escalaron los datos utilizando la técnica de *RobustScaler*, ya que ofreció mejores resultados para tratar con los valores atípicos (*outliers*).

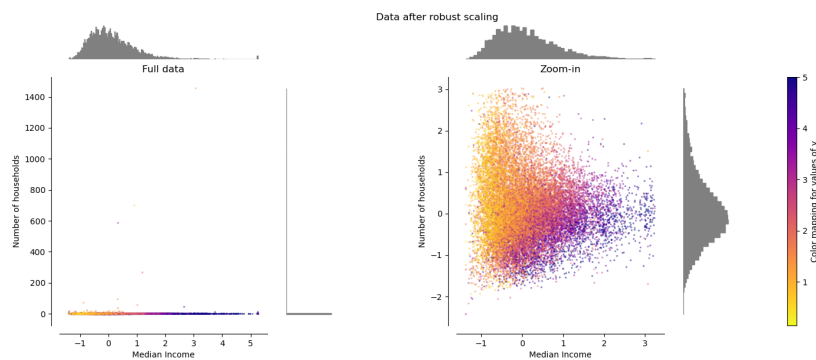


Fig. 6.3: Ejemplo de escalado de datos

6.2 Selección de algoritmos e hiperparámetros

Existen diferentes algoritmos que nos permiten clasificar los datos en diferentes categorías. En este proyecto se realiza una clasificación binaria, es decir, se clasifica basándose en dos etiquetas, si el usuario es legítimo o no legítimo.

Muli Layer Perceptron [39], *Support Vector Machines* [11] y *Random Forest* [15] son técnicas que han demostrado un buen rendimiento en este tipo de problemas y es por ello que serán los utilizados para resolver el problema del proyecto.

Para cada uno de los algoritmos seleccionados se procedió a realizar una búsqueda de los hiperparámetros. Estos valores permiten optimizar el ajuste del modelo a los datos con el fin de obtener unos resultados más precisos para el proceso de autenticación.

La cantidad de parámetros y sus valores son distintos en función del algoritmo, por esto se utilizan dos técnicas de búsqueda:

- **Grid Search:** Es una búsqueda exhaustiva de los mejores parámetros de una combinación dada.
- **Random Search:** Es una búsqueda aleatoria sobre una combinación dada.

En este proyecto se ha usado la técnica de *Grid Search* como la idónea debido al beneficio/rendimiento sobre ambos procedimientos [10]. Para las técnicas de *Multilayer Percetron* y *Random Forest* se configuran una serie de parámetros que modifican su estructura interna. En el *Multilayer Percetron* algunos de estos valores son el número de capas ocultas, número de neuronas por capa y la función de activación. Para *Random Forest* algunos valores son la profundidad máxima del árbol, número de muestras mínimo por hoja y número mínimo de muestras para crear

una nueva hoja. En ambos casos, la modificación de estos parámetros afecta a la creación del algoritmo, decidiendo su estructura interna. Por el contrario, la técnica *SVM* permite la configuración de parámetros que modifican la tolerancia de error. Estos valores son C , que controla el margen de error, y γ , que modifica el comportamiento de la función *kernel*.

La búsqueda de parámetros y posteriores pruebas se han realizado utilizando *Cross Validation* [26, 36] con cinco muestras. Esta técnica tiene un gran impacto a la hora de evitar el *overfitting* debido a una distribución inteligente de los datos.

Como se puede ver en la Figura 6.4 este procedimiento divide el conjunto inicial de datos en n subconjuntos. Esto permite que el entrenamiento que se realiza obtenga resultados más consistentes y reduciendo el *overfitting*, con un coste mayor en tiempo.

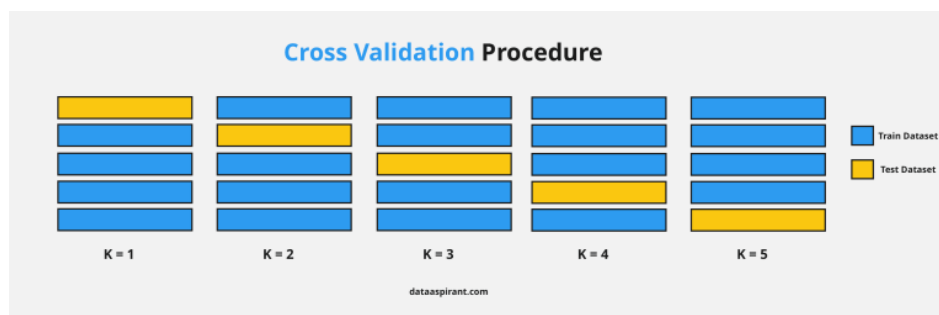


Fig. 6.4: Funcionamiento de *Cross Validation*

Los parámetros obtenidos como los mejores para el problema a resolver se muestran en la Tabla 6.2 y sus valores medios para las métricas *recall* y *precision* de cada algoritmo se muestran en la Tabla 6.3 y la Figura 6.5.

Los datos muestran que los tres algoritmos se comportan muy bien a la hora de obtener una predicción, pero debido a la complejidad que podría suponer generar y agrupar tres modelos por usuario, se ha decidido seleccionar *Random Forest* como el algoritmo para el proceso de autenticación final, ya que además de ofrecer un alto rendimiento, también se aprecia una mayor estabilidad a través de la desviación estándar obtenida.

Multilayer Percetron				
<i>alpha</i>	<i>hidden_layer_sizes</i>	<i>learning_rate</i>	<i>learning_rate_init</i>	<i>max_iter</i>
0.001	(30, 30, 30)	<i>adaptive</i>	0.01	1000
Support Vector Machines				
<i>C</i>		<i>gamma</i>	<i>kernel</i>	
1000000		0.00001	<i>rbf</i>	
Random Forest				
<i>max_depth</i>	<i>max_features</i>	<i>min_samples_leaf</i>	<i>min_samples_split</i>	<i>n_estimators</i>
110	3	5	12	1000

Tab. 6.2: Mejores hiperparámetros para cada algoritmo

Algoritmo	Métrica	Media
RF	<i>precision</i>	0.965 (± 0.005)
	<i>recall</i>	0.965 (± 0.005)
MLP	<i>precision</i>	0.855 (± 0.048)
	<i>recall</i>	0.849 (± 0.052)
SVM	<i>precision</i>	0.941 (± 0.006)
	<i>recall</i>	0.941 (± 0.006)

Tab. 6.3: Media de los resultados de los algoritmos

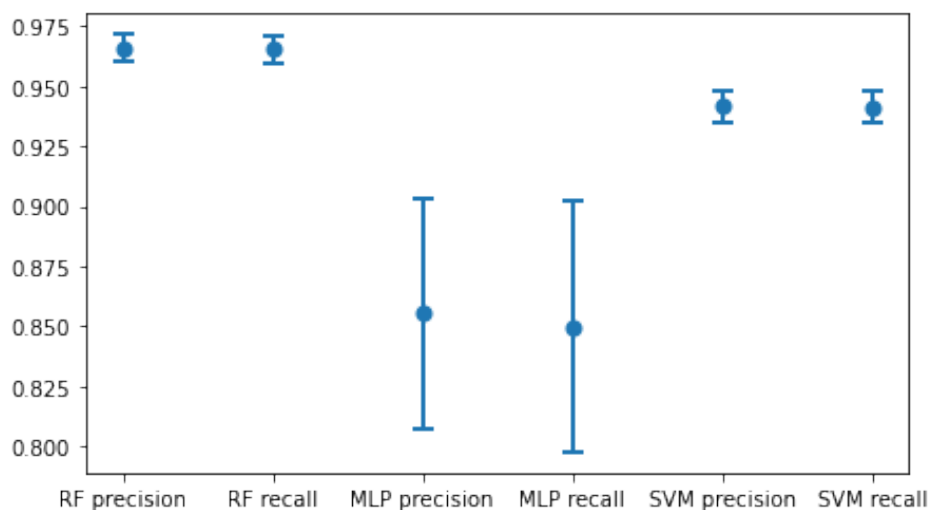


Fig. 6.5: Media de resultados de los algoritmos con su desviación

Kafka: Sistema de colas

En este capítulo se detallarán las medidas realizadas en *Kafka* para distribuir los datos entre las aplicaciones optimizando la estabilidad del sistema y el tiempo de respuesta.

7.1 Envío de datos

Los datos enviados por el cliente al sistema de colas es el resultado de la combinación de la *metadata* [Tabla 7.1] y los datos de los eventos [Tabla 2.1].

Características	Descripción
Usuario	Nombre del usuario del sistema
Versión aplicación	Versión de la aplicación
Sistema Operativo	Nombre del sistema operativo
Versión del SO	Versión del sistema operativo
Timestamp	Fecha del envío de los datos (Formato UNIX)

Tab. 7.1: Características de la metadata

Esta definición de los mensajes implica que deben ser producidos y consumidos de manera secuencial. *Kafka* asegura que los mensajes se consumen de manera secuencial solamente cuando existe una partición por cola, esto no permite escalar de forma correcta nuestro sistema, y por lo tanto es necesario definir de nuevo los datos enviados.

La solución obtenida parte por generar bloques de eventos [Figura 7.1] que proporcionan independencia con otros bloques pero que a su cada uno contiene la información necesaria para ser analizada por los modelos. Esta solución también nos proporciona otras mejoras:

- **Generamos menos tráfico:** La *metadata* solo se envía una vez por grupo.

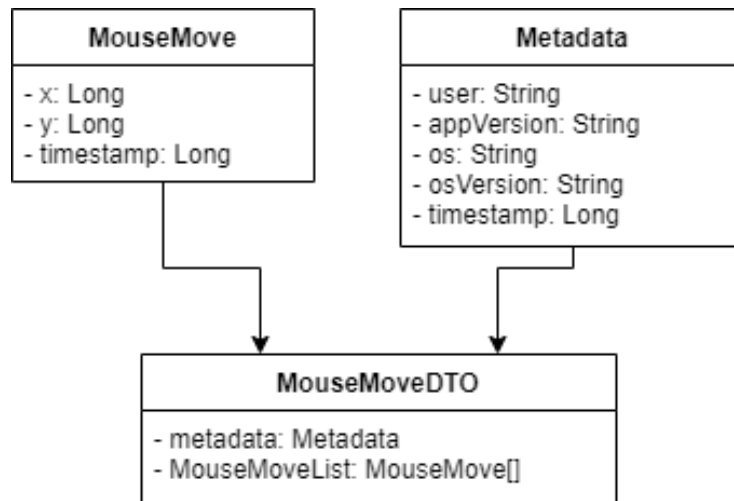


Fig. 7.1: Diagrama de clases de los datos

- **Realizan menos peticiones de red:** Con esta solución se envía una petición por cada bloque.

Para decidir el tamaño del bloque se han tenido en cuenta dos limitaciones de nuestro sistema:

- **MongoDB:** El tamaño máximo que se puede guardar en *MongoDB* es de 16 MB
- **Kafka:** La configuración por defecto del tamaño de mensaje de *Kafka* es de 1 MB.

Con esta información hemos calculado los tamaños de los distintos bloques, de esta manera, saber el número de eventos a enviar por bloque. En la Tabla 7.2 podemos observar los tamaños de evento y de bloque elegidos. La tabla muestra que con 1000 eventos obtenemos un tamaño de bloque inferior a 1 MB, que es el valor límite establecido por defecto en *Kafka*. Además, el tiempo en obtener estos 1000 eventos es de 2 segundos, que determinará el tiempo mínimo necesario para obtener la primera predicción, por lo que es un dato que se debería mantener lo más bajo posible.

Por otra parte, una de las ventajas que se observa al utilizar el bloque como forma de enviar datos entre las aplicaciones, es la reducción del tamaño. Esto implica un consumo de ancho un 72% menor para el usuario y un 58% para el servidor.

Datos	Evento Único		Lista (1000 Eventos)	
	Solo	Con Metadata	Solo	Con Metadata
Evento	44 Bytes	263 Bytes	258 KB	72 KB
Evento Transformado	953 Bytes	1,23 KB	1239 KBB	921 KB

Tab. 7.2: Tamaño de los eventos enviados uno a uno y agrupados

7.2 Diseño del Escenario

Una vez definida la estructura de datos dentro del sistema, queda definir las conexiones entre las diferentes aplicaciones. Para ello se plantearon cuatro escenarios distintos, en los cuales se pueden encontrar cuatro procesos:

- **Splitter:** Es el proceso que divide los datos por usuario.
- **Transform:** Es el encargado de transformar los datos de los eventos del ratón a las características finales.
- **Splitter and Transform:** Es la unión de los dos procesos anteriores, realizando ambos (transformación y división) en una sola máquina.
- **Predictor:** Es el encargado de obtener la predicción para un usuario.

7.2.1 Escenario 1

En este escenario [Figura 7.2] se plantean tres máquinas, donde la primera máquina (**Splitter**) recibe los eventos de todos los usuarios y los divide en una cola por usuario, que son enviados primero a la máquina de transformación y después a la de predicción.

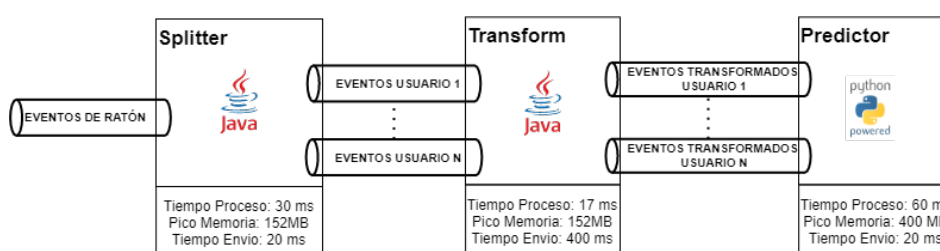


Fig. 7.2: Escenario 1

Este planteamiento presenta una gran desventaja y es la necesidad de tener una máquina de transformación por cada usuario, lo que nos conlleva un alto requisito de recursos.

7.2.2 Escenario 2

Este escenario [Figura 7.3] se plantean tres máquinas, pero intercambiando las máquinas de **Splitter** y **Transform** con respecto a la versión del escenario anterior.

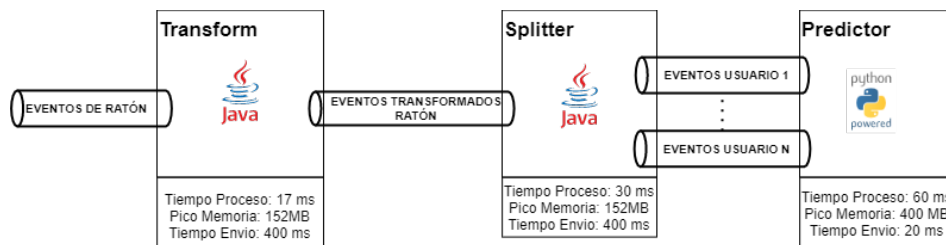


Fig. 7.3: Escenario 2

Esta versión evita la desventaja del escenario anterior pero añade un problema y es que incrementamos el tamaño de las colas significativamente lo que implica un aumento de los tiempos de envío.

7.2.3 Escenario 3

Este escenario [Figura 7.4] cuenta con dos máquinas, delegando el proceso de división a la aplicación de escritorio.

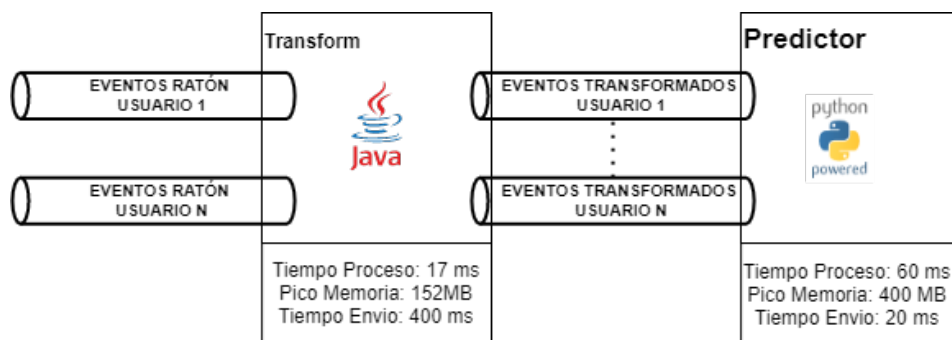


Fig. 7.4: Escenario 3

Este planteamiento soluciona los problemas de los escenarios anteriores, pero tiene un gran inconveniente y es el almacenamiento de los datos del usuario en la base de datos de *MongoDB*. El proceso de guardado se vuelve más complejo y añade una alta demanda de procesador (uno por usuario) por lo que vuelve un escenario con una escalabilidad baja.

7.2.4 Escenario 4

El último escenario [Figura 7.5] cuenta con dos máquinas, agrupando el **Splitter** y **Transform** en una sola.

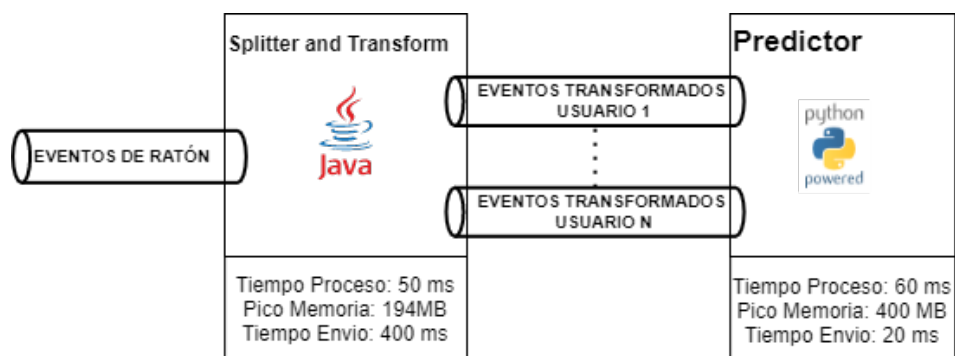


Fig. 7.5: Escenario 3

Este escenario soluciona los problemas comentados en el punto anterior, y también los de almacenamiento que teníamos en el escenario anterior ofreciendo un único punto de conexión para almacenar los datos con un coste de 33 ms latencia asumible por el beneficio que aporta.

Resultados

En este capítulo se presentarán los resultados obtenidos aplicando las técnicas mencionadas en los capítulos anteriores.

Primero se analizarán de manera *offline* y acto seguido se analizarán los procesos de creación y predicción de los modelos a lo largo del proceso de *streaming*.

8.1 Datos de *Random Forest*

En la Tabla 8.1 se pueden observar los resultados obtenidos para diez usuarios con el algoritmo e hiperparámetros seleccionados en la Sección 6.2. Los datos fueron obtenidos usando la técnica de CV con el fin de evitar *overfitting*.

Los resultados muestran la media obtenida para cada usuario sobre las cinco iteraciones del CV. En todos los casos se puede observar valores positivos, que superan el 90% y con una desviación inferior al 1%.

Estos datos se repiten tanto para la métrica de *precision* como para *recall*, demostrando la fiabilidad del sistema.

Con el fin de comprobar la robustez del sistema se obtuvo el *EER*. En la Figura 8.1 se puede observar la intersección de las curvas de *FRR* y *FAR* con el fin de obtener de *EER* y cuyos valores se pueden consultar en la Tabla 8.2. En ella se puede observar que el ratio es 0.079 con una mínima desviación. Esto implica que el sistema tiene una tasa de fallo inferior al 8%, lo que ofrece una alta fiabilidad.

8.2 Datos de *Streaming*

Uno de los principales problemas del sistema online es el tiempo que tarda en conseguir los datos necesarios para generar el modelo. En el proyecto se utilizan 20000 eventos, 10000 del usuario legítimo y 10000 de otros usuarios, para crear el modelo. Por lo tanto, necesitamos calcular el tiempo que se tarda en obtener 20000

Usuario	Precision	Recall	Time
1	0.863 (± 0.012)	0.945 (± 0.012)	4.943 (± 0.033)
2	0.914 (± 0.007)	0.925 (± 0.007)	4.914 (± 0.093)
3	0.885 (± 0.007)	0.908 (± 0.007)	5.091 (± 0.239)
4	0.883 (± 0.013)	0.924 (± 0.013)	4.907 (± 0.084)
5	0.870 (± 0.010)	0.880 (± 0.010)	4.919 (± 0.039)
6	0.911 (± 0.017)	0.937 (± 0.017)	4.882 (± 0.003)
7	0.950 (± 0.005)	0.920 (± 0.005)	4.898 (± 0.150)
8	0.860 (± 0.014)	0.930 (± 0.014)	4.904 (± 0.129)
9	0.961 (± 0.008)	0.963 (± 0.008)	4.929 (± 0.086)
10	0.956 (± 0.006)	0.931 (± 0.006)	4.860 (± 0.041)
11	0.977 (± 0.005)	0.972 (± 0.005)	4.789 (± 0.122)
12	0.906 (± 0.011)	0.938 (± 0.011)	4.920 (± 0.100)
Media	0.911 (± 0.010)	0.931 (± 0.010)	4.913 (± 0.093)

Tab. 8.1: Resultados de predicción obtenidos

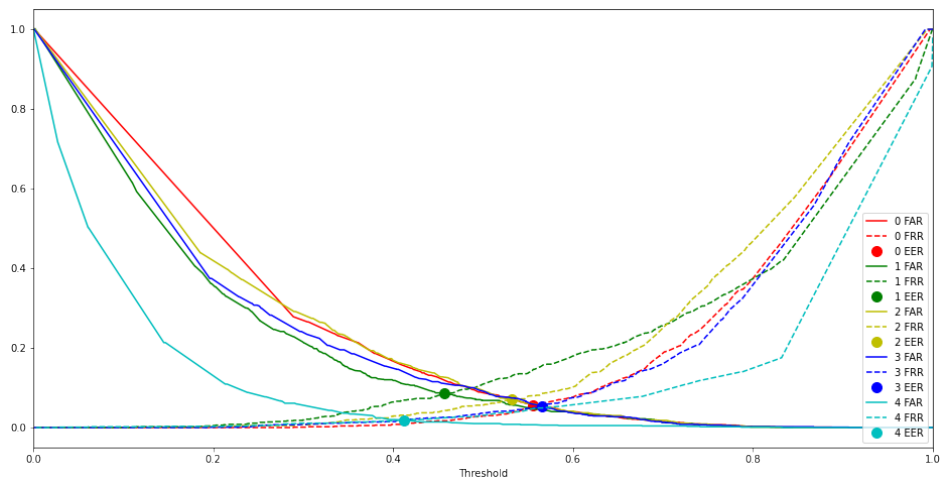


Fig. 8.1: FAR y FRR de un conjunto de 5 usuarios

Usuario	Equal Error Rate (EER)
1	0.098 (± 0.011)
2	0.080 (± 0.007)
3	0.102 (± 0.005)
4	0.100 (± 0.005)
5	0.126 (± 0.013)
6	0.077 (± 0.012)
7	0.061 (± 0.008)
8	0.111 (± 0.012)
9	0.038 (± 0.005)
10	0.051 (± 0.005)
11	0.025 (± 0.005)
12	0.080 (± 0.001)
Media	0.079 (± 0.008)

Tab. 8.2: Valores de *EER*

eventos, debido a que los eventos de múltiples usuarios son generados de manera simultánea. En la Tabla 8.3 se pueden ver los tiempos obtenidos en diez pruebas. En ella se observa la cantidad máxima de eventos generados por un ratón a los diez segundos en diferentes iteraciones. Con estos resultados se ha calculado el tiempo para generar un bloque de 1000 eventos (2.025 segundos) y de 10 bloques.

Iteración	Movimientos en 10 segundos	1 Bloque	10 Bloques
1	5001.0	1.999	19.996
2	4999.0	2.000	20.004
3	4994.0	2.002	20.024
4	4984.0	2.006	20.064
5	4999.0	2.000	20.004
6	4993.0	2.002	20.028
7	4605.0	2.171	21.715
8	4997.0	2.001	20.012
9	4983.0	2.006	20.068
10	4858.0	2.058	20.584
Media	4941.3	2.025	20.250

Tab. 8.3: Tiempo de obtención los eventos

Por lo tanto, estos datos nos indican que el tiempo mínimo para que el sistema empiece a realizar predicciones sería de 20.25 segundos.

A estos datos también hay que añadir el tiempo que el sistema tarda en generar el modelo. La Tabla 8.4 muestra el tiempo necesario para obtener los datos de la base de datos, la creación del modelo y la creación de la máquina para realizar predicciones. El tiempo obtenido 29.802 segundos de media. Para todos los modelos, la mayor parte de este tiempo se centra en su creación, en la cual se utiliza solo un núcleo de la máquina. De este modo, podríamos escalar la máquina verticalmente (es decir, incrementando sus recursos) y reducir los tiempos linealmente.

Usuarios	Tiempos			
	Obtener datos	Creación modelo	Creación máquina	Total
1	4.143	21.386	1.029	26.558
2	5.561	20.511	1.919	27.991
3	5.811	22.602	1.904	30.317
4	5.322	24.545	0.868	30.735
5	4.335	20.296	1.012	25.643
6	4.276	23.355	1.717	29.348
7	4.168	21.288	1.022	26.478
8	5.193	25.638	1.055	31.886
9	4.651	24.155	1.211	30.017
10	4.880	20.578	1.756	27.214
11	4.710	24.501	1.061	30.272
12	4.286	25.306	1.783	31.375
Media	4.778	23.168	1.856	29.802

Tab. 8.4: Tiempos de creación de los modelos en el sistema

Una vez creado el modelo, el sistema empieza a realizar las predicciones de manera inmediata con los datos que se encuentran en las colas. En la Tabla 8.5 se muestra el tiempo que tarda un administrador en ver la predicción desde que el bloque se envía al sistema. Como se puede observar, tanto el tiempo de procesado de los eventos como el de predicción son ínfimos en comparación con el tiempo de transporte, que es externo al sistema y depende de factores ajenos como la conexión a internet. Aún con este tiempo, la media se mantiene inferior a los 7 segundos, un tiempo aceptable para ser considerado un sistema de procesamiento en tiempo real en este ámbito de aplicación.

En la Figura 8.2 se muestra la ventana de administración con los datos obtenidos a lo largo del desarrollo del proyecto. En esta prueba los usuarios utilizaron su ordenador con el sistema de autenticación activado durante su jornada laboral. Los resultados muestran que el sistema de autenticación detecta correctamente a cuatro de los seis usuarios mostrados. Los otros dos usuarios (en color rojo) obtienen un

Usuarios	Tiempo			
	Procesado	Predicción	Envío	Total
1	0.4320	0.922399	8.8620	10.216399
2	0.4050	0.920659	3.3790	4.704659
3	0.4470	0.923645	5.2140	6.584645
4	0.3910	0.924681	4.1190	5.434681
5	0.4330	0.919187	6.5520	7.904187
6	0.4216	0.921780	5.6252	6.968580
7	0.4320	0.917038	8.8620	10.211038
8	0.4050	0.820295	3.3790	4.604295
9	0.4470	0.820450	5.2140	6.481450
10	0.3910	0.919727	4.1190	5.429727
11	0.4330	0.918583	6.5520	7.903583
12	0.4216	0.920441	5.6252	6.967241
Media	0.904074	0.4320	5.6252	6.961274

Tab. 8.5: Tiempos de obtención de predicciones

peor porcentaje debido a que durante el transcurso del proyecto cambiaron su ratón por uno ergonómico, demostrando la necesidad de actualizar el modelo utilizado para estos usuarios. Dicho procedimiento se comentará en el Capítulo 10 como una de las líneas de trabajo futuro.

The screenshot shows the ACATIA user management interface. It features a sidebar with navigation links: Dashboard, Usuarios (selected), and About us. The main content area displays three tables, each titled 'Usuarios (2)', showing user authentication data. Each table has columns for Usuario, Versión App, Sistema Operativo, SO Versión, Predicción, Eventos Totales, Fecha de Conexión, Lag, and Detalles. The first table shows users 'sofia' and 'raul' with prediction percentages of 46.14% and 41.36% respectively. The second table shows users 'lara' and 'silvelo' with prediction percentages of 56.75% and 66.89% respectively. The third table shows users 'iker' and 'dafonte' with prediction percentages of 82.85% and 92.67% respectively. Each table includes a pagination control at the bottom right.

Usuario	Versión App	Sistema Operativo	SO Versión	Predicción	Eventos Totales	Fecha de Conexión	Lag	Detalles
sofia	2.0.0.0	Windows 10	10.0	46.14 %	25253	Mar 8, 2022, 1:31:06 PM	00:03 319	
raul	2.0.0.0	Windows 10	10.0	41.36 %	10495	Mar 8, 2022, 1:52:16 PM	00:02 020	

Usuario	Versión App	Sistema Operativo	SO Versión	Predicción	Eventos Totales	Fecha de Conexión	Lag	Detalles
lara	2.0.0.0	Windows 10	10.0	56.75 %	10558	Mar 8, 2022, 1:48:45 PM	00:01 312	
silvelo	2.0.0.0	Windows 10	10.0	66.89 %	16347	Mar 8, 2022, 1:53:44 PM	00:00 872	

Usuario	Versión App	Sistema Operativo	SO Versión	Predicción	Eventos Totales	Fecha de Conexión	Lag	Detalles
iker	2.0.0.0	Windows 10	10.0	82.85 %	14282	Mar 8, 2022, 1:54:55 PM	00:01 755	
dafonte	2.0.0.0	Windows 10	10.0	92.67 %	11778	Mar 8, 2022, 1:54:22 PM	00:42 655	

Fig. 8.2: Ventana de administración con datos de autenticación

Conclusiones

Los resultados obtenidos del proyecto desarrollado demuestran que es posible un sistema de autenticación, como segundo factor, basado en el comportamiento del usuario y utilizando técnicas de *streaming* para obtener un sistema que es capaz de trabajar en tiempo real a la hora de predecir la legitimidad del usuario.

Para la obtención de datos se creó una aplicación multiplataforma que permitió capturar los eventos generados por el usuario para su posterior análisis. Esta aplicación se instaló en las máquinas de 12 usuarios con el fin de capturar los eventos generados por el ratón.

Con estos datos recopilados se aplicaron diferentes técnicas de Inteligencia Artificial (Capítulo 6) y los resultados obtenidos después de los diferentes procedimientos obtuvieron un porcentaje de acierto superior al 90%.

Se implementaron múltiples contenedores con *docker* que nos permitieron automatizar todo el proceso de generación de los modelos de predicción y la obtención de las predicciones. Al usar los contenedores de *docker*, pudimos aislar los distintos servicios, configurando que solo sean visibles por red entre ellos y que no tengan acceso hacia el exterior, lo que nos ofreció una mayor seguridad a la hora de implantar el sistema en producción.

Para el almacenamiento de los datos se creó un *clúster* de *MongoDB*, lo que nos permitió tener acceso y redundancia de los datos. Esta configuración final realizada tiene una mayor tolerancia a fallos y facilita el acceso a los datos en caso de fallo en algún nodo.

Para el control y envío de estos datos se implementó una plataforma de gestión de colas (*Kafka*). Esta plataforma envía los datos de los eventos de ratón a la Base de Datos y alimenta el resto de aplicativos con los datos necesarios. Esto nos permite que los procesos de transformación de datos y predicción se realicen con una baja latencia.

Por último, se diseñó y creó una aplicación web que presentase el estado del sistema, mostrando en todo el momento la información relevante sobre los usuarios. Esta

presentación permite a un administrador del sistema tener una visión clara del estado actual del sistema.

Como conclusión a título personal, en proyecto se emplearon diferentes tecnologías de diversos ámbitos, lo que me ofreció la posibilidad de investigar sobre dichas tecnologías que a priori no conocía, ampliando de esta manera el conocimiento sobre ciertas áreas como el procesamiento de datos empleando un sistema de colas o la gestión y automatización de una red virtual de ordenadores con docker.

Trabajo Futuro

En este capítulo se comentarán algunos de los posibles desarrollos que se podrían llevar a cabo en un futuro.

- **Refinar el proceso de extracción y selección de características:** Partiendo de la implementación actual, se propone explorar nuevas características en la búsqueda de una optimización de los resultados.
- **Actualización de los modelos:** Este tipo de sistemas suelen estar entrenados con un conjunto de datos iniciales que es persistente en el tiempo, pero el comportamiento de una persona puede cambiar con el paso del tiempo, debido a circunstancias como enfermedades o lesiones. Por lo tanto, un aprendizaje que evolucionase con el usuario permitiría obtener una mayor fiabilidad al adaptarse a cambios en los hábitos y comportamiento de los usuarios.
- **Explorar nuevas técnicas:** Las técnicas elegidas a lo largo del proyecto son una buena base de referencia para el proyecto, pero creemos que diseños basados en tecnologías de *deep-learning* pueden mejorar los resultados obtenidos.
- **Combinar técnicas:** Las técnicas usadas se han utilizado de forma independiente, una futura rama de investigación sería combinar las técnicas y obtener de esta forma una predicción por votación.
- **Nuevas agrupaciones:** En el proyecto se emplea una agrupación de 1000 eventos con el fin de optimizar el envío y procesamiento de datos. Se plantea realizar un estudio más detallado sobre el tamaño de bloque, buscando de esta forma encontrar un equilibrio entre la mejora de resultados y la degradación en los tiempos de procesamiento.
- **Mejorar la web de administración:** La aplicación web permite mostrar la información del estado actual del sistema para realizar las acciones que se consideren oportunas. Este tipo de acciones como un mensaje de aviso o cerrar la sesión del usuario podrían ser implementadas para que se realicen desde la propia aplicación.

Bibliografía

- [1](Vid. pág. 32).
- [2]Ahmed Awad E Ahmed e Issa Traore. “Anomaly intrusion detection based on biometrics”. En: *Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop*. IEEE. 2005, págs. 452-453 (vid. pág. 2).
- [3]Ahmed Awad E. Ahmed e Issa Traoré. “Detecting Computer Intrusions Using Behavioral Biometrics”. En: *PST*. 2005 (vid. pág. 29).
- [4]*Amenaza vs Vulnerabilidad, ¿sabes en qué se diferencian?* Abr. de 2017 (vid. págs. 7, 8).
- [5]*Autenticación de usuarios*. Jul. de 2002 (vid. pág. 8).
- [6]Ahmed Awad e Issa Traore. “A New Biometric Technology Based on Mouse Dynamics”. En: *Dependable and Secure Computing, IEEE Transactions on* 4 (ago. de 2007), págs. 165-179 (vid. pág. 29).
- [7]B. Sayed y col. “Biometric Authentication Using Mouse Gesture Dynamics”. En: *IEEE Systems Journal* 7.2 (2013), págs. 262-274 (vid. pág. 10).
- [8]Mohammadreza Barkadehi, Mehrbaksh Nilashi, Othman Ibrahim, Ali Fardi y Sarminah Samad. “Authentication Systems: A Literature Review and Classification”. En: *Telematics and Informatics* 35 (mar. de 2018) (vid. pág. 29).
- [9]Mohammadreza Hazhirpasand Barkadehi, Mehrbaksh Nilashi, Othman Ibrahim, Ali Zakeri Fardi y Sarminah Samad. “Authentication systems: A literature review and classification”. En: *Telematics and Informatics* 35.5 (2018), págs. 1491-1511 (vid. pág. 2).
- [10]James Bergstra y Yoshua Bengio. “Random search for hyper-parameter optimization.” En: *Journal of machine learning research* 13.2 (2012) (vid. pág. 54).
- [11]R Berwick. “An idiots guide to support vector machines”. En: *Advanced Computer Vision. University of Central Florida* (2003) (vid. págs. 19, 54).
- [12]Monika Bhatnagar, Raina K Jain y Nilam S Khairnar. “A survey on behavioral biometric techniques: mouse vs keyboard dynamics”. En: *Int. J. Comput. Appl* 975 (2013), pág. 8887 (vid. págs. 2, 29).
- [13]Leo Breiman. “Bagging predictors”. En: *Machine learning* 24.2 (1996), págs. 123-140 (vid. pág. 22).
- [14]Leo Breiman. “Pasting small votes for classification in large databases and on-line”. En: *Machine learning* 36.1 (1999), págs. 85-103 (vid. pág. 22).
- [15]Leo Breiman. “Random Forests”. En: *Machine Learning* (2001) (vid. págs. 4, 23, 54).

- [16]Leo Breiman, Jerome H Friedman, Richard A Olshen y Charles J Stone. *Classification and regression trees*. Routledge, 2017 (vid. pág. 21).
- [17]Corinna Cortes y Vladimir Vapnik. “Support-vector networks”. En: *Machine learning* 20.3 (1995), págs. 273-297 (vid. pág. 4).
- [18]Eurostat. *Individuals - computer use*. 2021. URL: https://ec.europa.eu/eurostat/databrowser/product/view/ISOC_CI_CFP_CU?lang=en (vid. pág. 1).
- [19]Eurostat. *Use of computers and the internet by employees*. 2021. URL: https://ec.europa.eu/eurostat/databrowser/product/view/ISOC_CI_CM_PN2?lang=en (vid. pág. 1).
- [20]Jeremy G Frey, Jamie Robinson, Andrew Stanford-Clark, Andrew Reynolds y Bharat Bendi. “Sensor Networks and Grid Middleware for Laboratory Monitoring”. En: (2005) (vid. pág. 25).
- [21]*Gestión de la seguridad*. Jul. de 2002 (vid. pág. 8).
- [22]Barbara Guttman y E Roback. *An Introduction to Computer Security: the NIST Handbook*. en. 1995-10-02 de 1995 (vid. pág. 1).
- [23]H. Gamboa y col. “An Identity Authentication System Based On Human Computer Interaction Behaviour”. En: *Pattern Recognition in Information Systems, Proceedings of the 3rd International Workshop on Pattern Recognition in Information Systems, PRIS 2003, In conjunction with ICEIS 2003, Angers, France, April 2003*. ICEIS Press, 2003, págs. 46-55 (vid. págs. 2, 10).
- [24]Gaston C Hillar. *MQTT Essentials-A lightweight IoT protocol*. Packt Publishing Ltd, 2017 (vid. pág. 26).
- [25]*Introducción y conceptos previos*. Jul. de 2002 (vid. pág. 8).
- [26]Ron Kohavi y col. “A study of cross-validation and bootstrap for accuracy estimation and model selection”. En: *Ijcai*. Vol. 14. 2. Montreal, Canada. 1995, págs. 1137-1145 (vid. pág. 55).
- [27]Jay Kreps, Neha Narkhede, Jun Rao y col. “Kafka: A distributed messaging system for log processing”. En: *Proceedings of the NetDB*. Vol. 11. 2011, págs. 1-7 (vid. págs. 27, 46).
- [28]Gilles Louppe y Pierre Geurts. “Ensembles on random patches”. En: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2012, págs. 346-361 (vid. pág. 22).
- [29]John McCarthy. “What is artificial intelligence?” En: (2007) (vid. pág. 14).
- [30]Marvin Minsky y Seymour A Papert. *Perceptrons: An introduction to computational geometry*. MIT press, 2017 (vid. pág. 17).
- [31]Mseg (vid. pág. 9).

- [32]N. Zheng y col. "An Efficient User Verification System via Mouse Movements". En: *Proceedings of the 18th ACM Conference on Computer and Communications Security*. Ed. por ACM. New York, NY, USA: Association for Computing Machinery, 2011, págs. 139-150 (vid. pág. 10).
- [33]Alok Nikhil y Vinoth Chandar. *Benchmarking Kafka vs. Pulsar vs. RabbitMQ: Which is Fastest?* Ago. de 2020 (vid. pág. 46).
- [34]John O'Hara. "Toward a Commodity Enterprise Middleware: Can AMQP Enable a New Era in Messaging Middleware? A Look inside Standards-Based Messaging with AMQP". En: *Queue* 5.4 (mayo de 2007), págs. 48-55 (vid. págs. 26, 27, 46).
- [35]Maja Pusara y Carla E. Brodley. "User Re-Authentication via Mouse Movements". En: *Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security*. VizSEC/DMSEC '04. Washington DC, USA: Association for Computing Machinery, 2004, págs. 1-8 (vid. pág. 29).
- [36]R Bharat Rao, Glenn Fung y Romer Rosales. "On the dangers of cross-validation. An experimental evaluation". En: *Proceedings of the 2008 SIAM international conference on data mining*. SIAM. 2008, págs. 588-596 (vid. pág. 55).
- [37]JOSEPH REA. *Consuming messages out of apache kafka in a browser*. Mar. de 2019 (vid. pág. 41).
- [38]*Replication* (vid. pág. 50).
- [39]Frank Rosenblatt. "Perceptron simulation experiments". En: *Proceedings of the IRE* 48.3 (1960), págs. 301-309 (vid. págs. 17, 54).
- [40]David E Rumelhart, Geoffrey E Hinton, Ronald J Williams y col. "Learning representations by back-propagating errors". En: *Cognitive modeling* 5.3 (1988), pág. 1 (vid. pág. 18).
- [41]Dymitr Ruta y Bogdan Gabrys. "Classifier selection for majority voting". En: *Information fusion* 6.1 (2005), págs. 63-81 (vid. págs. 24, 25).
- [42]Rodrigo Salas. "Redes neuronales artificiales". En: *Universidad de Valparaso. Departamento de Computación* 1 (2004), págs. 1-7 (vid. pág. 16).
- [43]Pouya Samangouei, Vishal M Patel y Rama Chellappa. "Facial attributes for active authentication on mobile devices". En: *Image and Vision Computing* 58 (2017), págs. 181-192 (vid. pág. 2).
- [44]Bassam Sayed, Issa Traore, Isaac Woungang y Mohammad Obaidat. "Biometric Authentication Using Mouse Gesture Dynamics". En: *IEEE Systems Journal* 7 (jun. de 2013), págs. 262-274 (vid. pág. 29).
- [45]William Stallings, Lawrie Brown, Michael D Bauer y Michael Howard. *Computer security: principles and practice*. Vol. 2. Pearson Upper Saddle River, 2012 (vid. pág. 1).
- [46]Steve Vinoski. "Advanced Message Queuing Protocol". En: *IEEE Internet Computing* 10.6 (2006), págs. 87-89 (vid. págs. 26, 27, 46).

- [47]Avery Wang. “An Industrial Strength Audio Search Algorithm.” En: ene. de 2003 (vid. pág. 10).
- [48]Paul J Werbos. “Foreword: ADP-The Key Direction for Future Research in Intelligent Control and Understanding Brain Intelligence.” En: *IEEE Trans. Syst. Man Cybern. Part B* 38.4 (2008), págs. 898-900 (vid. pág. 4).
- [49]Feng Zhang, Aron Kondoro y Sead Muftic. “Location-based authentication and authorization using smart phones”. En: *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*. IEEE. 2012, págs. 1285-1292 (vid. pág. 2).
- [50]Nan Zheng, Aaron Paloski y Haining Wang. “An Efficient User Verification System via Mouse Movements”. En: *Proceedings of the 18th ACM Conference on Computer and Communications Security*. CCS '11. Chicago, Illinois, USA: Association for Computing Machinery, 2011, págs. 139-150 (vid. pág. 30).

Índice de figuras

2.1	Esquema matriz de confusión.	13
2.2	Relación de EER con FAR y FRR	15
2.3	Problemas de entrenamiento	16
2.4	Esquema de una neurona artificial	17
2.5	Esquema del MLP	18
2.6	Ejemplo de una MLP	19
2.7	Ejemplo de Clasificación SVM	20
2.8	Ejemplo de Clasificación SVM no lineal, usando función Gaussiana	21
2.9	Ejemplo de árbol de decisión para conceder un préstamo	21
2.10	Ejemplo de Bootstrapping	22
2.11	Ejemplo de Boosting	23
2.12	Ejemplo de Random Forest	24
2.13	Esquema clasificación por votación	25
2.14	Esquema de gestión de colas	26
4.1	Lista de tareas planificadas	35
4.2	Diagrama de Gantt	36
4.3	Fases del modelo incremental	38
5.1	Esquema detallado del sistema de streaming	39
5.2	Arquitectura de tres capas	40
5.3	Menú de la aplicación de escritorio.	41
5.4	Patrón Redux	42
5.5	Patrón modelo-vista-vista-modelo	43
5.6	Ventana con la gráfica de eventos generados	43
5.7	Ventana de la lista de usuarios	44
5.8	Ventana de estadísticas del usuario	44
5.9	Esquema de la capa de negocio	45
5.10	Esquema Publicador/Suscriptor	46
5.11	Workflow del sistema de orquestación	48
5.12	Flujo del sistema de orquestación	49
5.13	Flujo de creación de modelos	49
5.14	Estructura replicaSet [38]	50

6.1	Matriz de correlación de la selección de características	52
6.2	Gráfica de eliminación recursiva de características	52
6.3	Ejemplo de escalado de datos	54
6.4	Funcionamiento de <i>Cross Validation</i>	55
6.5	Media de resultados de los algoritmos con su desviación	56
7.1	Diagrama de clases de los datos	58
7.2	Escenario 1	59
7.3	Escenario 2	60
7.4	Escenario 3	60
7.5	Escenario 3	61
8.1	FAR y FRR de un conjunto de 5 usuarios	64
8.2	Ventana de administración con datos de autenticación	67

Índice de cuadros

2.1 Valores de entrada de un movimiento	10
2.2 Lista de características transformadas	12
2.3 Funciones de activación	17
2.4 Funciones <i>kernel</i>	20
4.1 Tabla de recursos [1]	32
4.2 Costes y tiempos del proyecto	35
4.3 Tabla de riesgos	37
4.4 Tabla de gestión de riesgos	37
5.1 Latencias y rendimiento de los distintos productos [33]	46
6.1 Resultados usando Random Forest	53
6.2 Mejores hiperparámetros para cada algoritmo	56
6.3 Media de los resultados de los algoritmos	56
7.1 Características de la metadata	57
7.2 Tamaño de los eventos enviados uno a uno y agrupados	59
8.1 Resultados de predicción obtenidos	64
8.2 Valores de <i>EER</i>	65
8.3 Tiempo de obtención los eventos	65
8.4 Tiempos de creación de los modelos en el sistema	66
8.5 Tiempos de obtención de predicciones	67

