ISSN 1870-4069

Sistema de entrenamiento personalizado para equipos de programación competitiva mediante aprendizaje supervisado

Juventino Aguilar-Correa, Aldonso Becerra-Sánchez,
Roberto Solís-Robles, Gustavo Zepeda-Valles

Universidad Autónoma de Zacatecas,
Unidad Académica de Ingeniería Eléctrica,
México

superjuve@outlook.es,{a7donso, rsolis, gzepeda}@uaz.edu.mx

Resumen. Los concursos de programación competitiva abonan en mejorar las técnicas de los programadores que persiguen perfeccionar su habilidad. En búsqueda de coadyuvar los entrenamientos de participantes en competencias de este tipo, se desarrolló una propuesta de arquitectura de sistema y una aplicación web basada en Django que propicia aprendizaje personalizado a los niveles de destreza del alumno. Esta aplicación emplea algoritmos de redes neuronales y regresión logística como apoyo para predecir y obtener los tipos de problemas particulares que el participante requiere realizar, dependiendo de su nivel de soltura en diferentes áreas de programación. Para realizar esta tarea, el módulo emplea información en tiempo real de los resultados de competencias de programadores de Codeforces (intentos de resolver un problema, dificultad del problema, categorías y veredicto de resolución). Los resultados de entrenamiento de los modelos generan altas expectativa de predicciones, ya que los valores de F1-score, precisión y exactitud son robustos.

Palabras clave: Aprendizaje máquina, aprendizaje supervisado, programación competitiva.

Personalized Training System for Competitive Programming Teams Through Supervised Learning

Abstract. Competitive programming contests help improve the techniques of programmers who seek to perfect their skills. In search of contributing to the training of participants in competitions of this type, a system architecture proposal and a web application based on Django were developed that promote personalized learning at the student's skill levels. This application uses neural network and logistic regression algorithms as support to predict and obtain the types of particular problems that the participant needs to perform, depending on their level of fluency in different programming areas. To perform this task, the module uses real-time information on the results of Codeforces programmer competitions (attempts to solve a problem, problem difficulty, categories, and resolution verdict). The training results of the models generate high prediction expectations, since the F1-score, precision and accuracy values are robust.

Keywords: Competitive programming, machine learning, supervized learning.

1. Introducción

El uso de sistemas de software se ha vuelto cada vez más relevante en la sociedad actual, ya que se utilizan en una amplia variedad de campos y aplicaciones, desde el sector empresarial y gubernamental hasta el entretenimiento y la educación. Donde formalmente el software suele ser concebido como un conjunto de programas informáticos, procedimientos, archivos de configuración, documentación y datos relacionados al funcionamiento de un sistema informático [15], [26]. La fabricación de sistemas de software es un proceso complejo que implica la planificación, diseño, construcción, pruebas y mantenimiento de los sistemas de software.

A medida que los sistemas de software han evolucionado, también lo han hecho las técnicas utilizadas para su desarrollo. Una de las disciplinas que ha tomado auge en el proceso de mejoras de desarrollo de software y programación en general es la programación competitiva, la cual implica una actividad mental en el que los participantes resuelven problemas algorítmicos bajo un límite de tiempo [25].

Esta actividad pone a prueba las habilidades de resolución de problemas, el conocimiento de los algoritmos y la capacidad de escribir código eficiente [24]. La popularidad de la programación competitiva ha crecido significativamente en los últimos años (con cientos de participaciones en 1997, hasta varios miles en la actualidad) [14], gracias al surgimiento de competencias a nivel mundial, como la International Collegiate Programming Contest (ICPC).

El ICPC es un concurso de programación algorítmica para estudiantes universitarios, dividido en etapas, que van desde clasificatorios, regionales y la final mundial. Equipos de tres, en representación de su universidad, trabajan para resolver problemas algorítmicos, promoviendo la colaboración, la creatividad, la innovación y la capacidad de desempeñarse bajo presión [13]. Al igual que el ICPC, existen y han existido otras competiciones organizadas por diferentes compañías como Google [10] y Meta [19], las cuales tienen como objetivo la búsqueda de programadores con gran potencial para reclutarlos dentro de sus equipos.

En la actualidad existen algunos sistemas que ayudan a recomendar problemas para practicar en programación competitiva, los cuales permiten a los equipos sin entrenadores experimentados o clubes de algoritmia, mejorar sus habilidades a través de diferentes técnicas. Algunos de estos sistemas permiten que el usuario defina su propia ruta de aprendizaje a modo de gamificación del entrenamiento [6], mientras que otros se enfocan en recomendar y clasificar problemas según su dificultad y temática [3]. También existen investigaciones que utilizan técnicas de machine learning y deep learning para predecir el desempeño de equipos en futuras competencias [2].

Los equipos de programación competitiva de ciertas universidades en México que no cuentan con apoyo institucional, un club de programación o un entrenador experimentado para su formación, se enfrentan al desafío de no saber cómo entrenar de manera efectiva para mejorar sus habilidades y conocimientos en los distintos temas abordados en este tipo de competencias. Esto los coloca en una situación desventajosa para contender en las competencias regionales. Universidades que podemos destacar que cuentan con este tipo de asociaciones de estudiantes son, la Universidad de Guadalajara, con el Club de Algoritmia CUCEI [5]; el Instituto Tecnológico y de Estudios Superiores de Monterrey Campus Monterrey, con el Club de algoritmia

Tabla 1. Casos de uso de los requisitos funcionales del sistema.

ID	Usuario	Requerimiento	Descripción
CU01	Competidor	Realizar pruebas de diagnóstico.	Conocer el nivel actual en la categoría escogida y establecer una línea de base.
CU02	Competidor	Solicitar recomendaciones.	Recomendaciones de problemas adaptados a su habilidad en una categoría.
CU03	Competidor	Consultar progreso personal.	Progreso en el tiempo en categorías de elección.
CU04	Admin	Gestionar problemas	Agregar, modificar, consultar o eliminar problemas dentro del sistema.
CU05	Admin	Gestionar categorías	Crear, consultar, modificar y eliminar según sea necesario.
CU06	Admin	Gestionar usuarios	Consultar, agregar, eliminar y modificar usuarios del sistema.

ITESM MTY [16]; la Escuela Superior de Cómputo del Instituto Politécnico Nacional, con su Club de Algoritmia [9] y la Universidad Nacional Autónoma de México, con el Club de Programación Competitiva UNAM [29].

Actualmente existen sitios populares como Codeforces [4] u omegaUp [21], donde los interesados en programación competitiva pueden entrenar resolviendo problemas del gran repertorio con el que estas plataformas cuentan. Ambas separan sus problemas en categorías dependiendo de las estrategias que se utilizan para resolverlos. Tienen jueces automáticos que dan respuesta a las soluciones enviadas por los participantes, y de igual manera permite el acceder a concursos en tiempo real; al igual que revisar concursos anteriores para conocer los problemas expuestos en cada uno.

La falta de apoyo y entrenamiento personalizado para los equipos de programación competitiva puede limitar su desempeño y potencial, debido a que actualmente el resultado de los equipos en las competencias mundiales y regionales depende en gran parte a cómo entrenan y el tratar de resolver muchísimos problemas diferentes [18]. El mejorar la manera en que se entrena para este tipo de competencias no sólo ofrece beneficios directos para la programación competitiva, sino que también provee a sus participantes con la mejora en sus capacidades para resolver problemas, trabajo en equipo y entrega de resultados bajo presión [1].

En este sentido, el aprender a resolver problemas con distintos métodos aumentan las posibilidades de obtener entrevistas de trabajo en empresas como Google, Meta y Amazon, mejorando en las habilidades sobre teoría de números, combinatorias y algoritmos geométricos [28]. El objetivo de este trabajo es proponer una arquitectura y herramienta de software de apoyo a la falta de mecanismos personalizados de entrenamiento para los equipos de programación competitiva. Los resultados de este proyecto pueden tener un gran impacto dentro de las competencias de programación al brindar apoyo para mejorar el desempeño de los equipos y, por tanto, aumentar

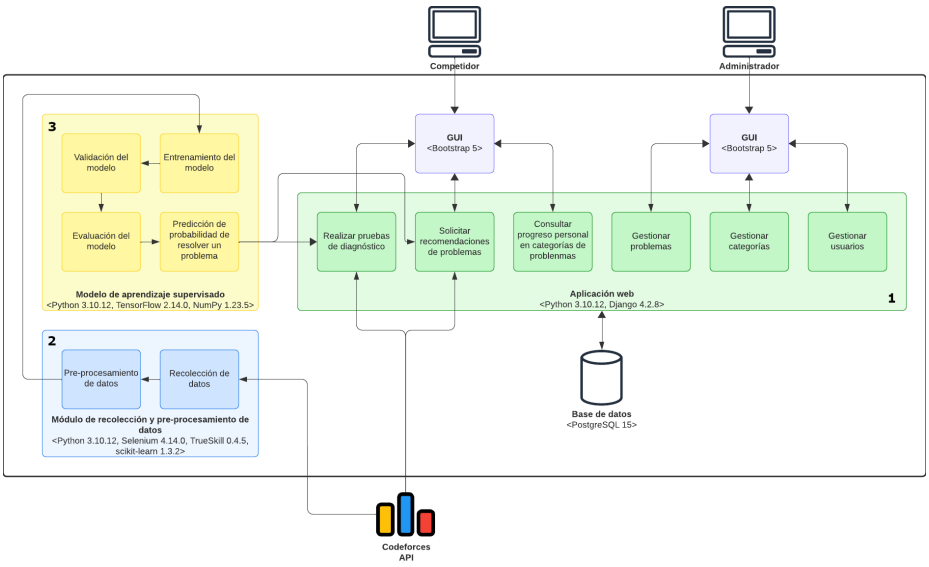


Fig. 1. Arquitectura del sistema acorde a los 3 módulos desarrollados.

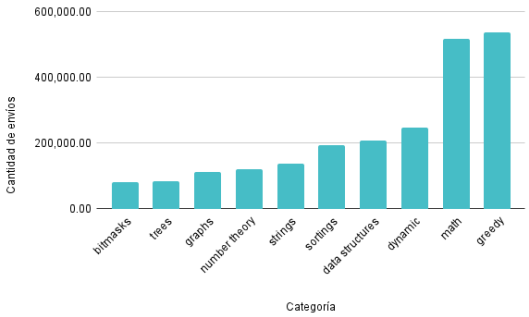
sus posibilidades de éxito en las competencias. El resto del documento se organiza de la siguiente forma, la sección 2 incluye una descripción de trabajos relacionados, mientras que la sección 3 incluye la propuesta de la aplicación desarrollada. En tanto que la sección 4 muestra los resultados obtenidos. Finalmente, la sección 5 comenta las consideraciones finales de conclusiones y trabajo futuro.

2. Trabajos relacionados

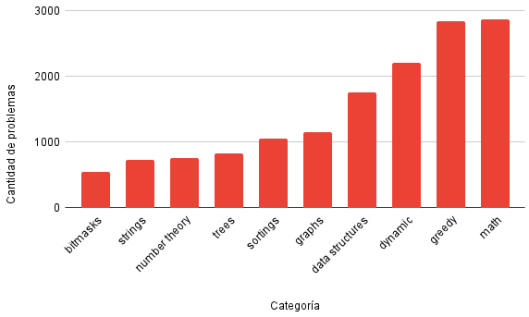
La programación competitiva es una disciplina que ha ganado cada vez más popularidad en los últimos años [14], debido a su capacidad para mejorar el pensamiento crítico y solución de problemas de los equipos de programadores [24]. En este contexto, la inteligencia artificial se puede presentar como una herramienta valiosa para el desarrollo de sistemas de entrenamiento personalizado, permitiendo mejorar el rendimiento de los equipos de programación en competencias.

Recientemente se desarrolló un sistema para generar entrenamientos en programación competitiva que permitiera crear rutas de aprendizaje para estudiantes y utilizar insignias para rastrear su progreso [6], empleando gamificación. Dentro de otro de los sistemas propuestos se generó un módulo capaz de clasificar problemas de programación competitiva a partir de su descripción y categorías, así como recomendar problemas para practicar con base en el historial de soluciones previas del usuario [3]. Este sistema utilizó técnicas de machine learning para realizar la clasificación y recomendación de problemas.

Otro ejemplo de este tipo de proyectos [22] nos presenta un sistema implementado en una universidad que regulariza a los estudiantes de ingeniería en computación utilizando problemas de programación de un juez en línea.



(a) Envíos por categoría.



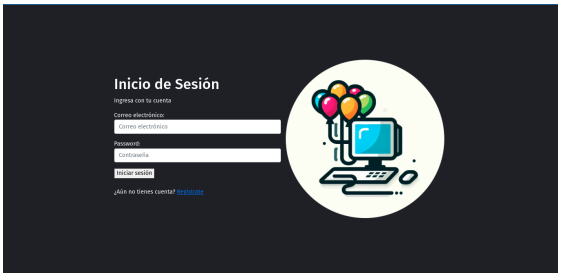
(b) Problemas por categoría.

Fig. 2. Información general del dataset extraído: envíos y problemas.

El sistema recomienda problemas a los estudiantes en función de su puntuación y la de los problemas, utilizando el sistema de clasificación ELO [20]. En este otro caso de estudio [27], el objetivo de la investigación es utilizar una red neuronal convolucional (CNN) [11] para identificar las técnicas empleadas en las soluciones, y recomendar problemas en función de características similares a los problemas previamente resueltos por los usuarios.

Mientras que en otras investigaciones se proponen sistemas de recomendación para programadores novatos en competencias de programación en línea. En ello se utiliza filtrado colaborativo para encontrar problemas similares y recomendarlos a los usuarios [31]. De manera similar, también existe otro sistema de recomendación de problemas, utilizando el algoritmo de vecinos más cercanos (KNN) [12], para encontrar usuarios similares y realiza recomendaciones basadas nuevamente en el filtrado colaborativo y la similitud de contenido [23].

Estos trabajos han abordado la problemática de la programación competitiva y la falta de entrenamiento personalizado, ofreciendo sistemas basados en inteligencia artificial y otras técnicas que permiten recomendar problemas adecuados al nivel y temas de interés de los programadores.



(a) Inicio de sesión con credenciales.

(b) Creación de cuenta.

Fig. 3. Pantallas iniciales del sistema.

3. Esquema propuesto

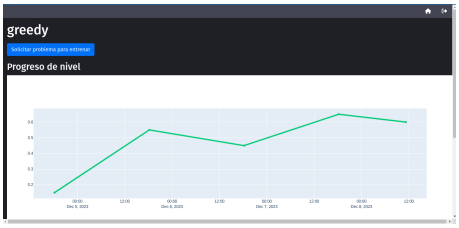
Para el desarrollo de la propuesta, se empleó la metodología de prototipo, que es un método de desarrollo de sistemas en el que se construye un prototipo, se prueba y después se retrabaja según sea necesario hasta que se obtenga un resultado aceptable del cual se puede desarrollar el sistema completo [26, 30]. La fases de esta metodología incluyen: i) definición de objetivos, ii) definición de funcionalidad, iii) diseño y desarrollo del prototipo, y iv) evaluación del prototipo.

3.1. Definición de los objetivos del prototipo

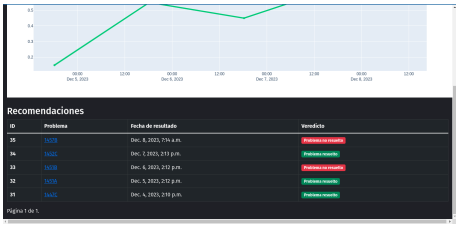
Los objetivos representan las funcionalidades clave que los competidores (principales usuarios del sistema) podrán acceder y utilizar en el sistema, además del experimento necesario para medir la eficacia del prototipo. Estos incluyen: i) gestionar problemas, categorías y usuarios en el sistema; ii) recomendar problemas a los competidores que se adapten a su habilidad actual en la categoría en la que estén entrenando; iii) realizar diagnósticos iniciales a los competidores para evaluar su nivel de habilidad en las categorías de problemas en las que quieran entrenar; iv) proporcionar un seguimiento visual del progreso del competidor en sus categorías de interés a lo largo del tiempo; y v) experimento para evaluar la eficacia del sistema.

3.2. Definición de la funcionalidad

El sistema se centra en ofrecer una experiencia de entrenamiento para equipos de programación competitiva.



(a) Gráfica de progreso por categoría.



(c) Recomendaciones por categoría.

Fig. 4. Progresos y recomendaciones de usuario acorde a su evolución.

Los miembros de estos equipos recibirán recomendaciones personalizadas de problemas para su práctica, así como realizar un seguimiento de su progreso a través de estadísticas detalladas centradas en los temas de algoritmia de su interés. La recomendación de problemas se llevará a cabo mediante un enfoque de aprendizaje supervisado, seleccionando aquellos para los cuales el modelo haya predicho una probabilidad intermedia de resolución. Esta estrategia busca proporcionar retos adecuados a los usuarios, fomentando así su mejora continua. La Tabla 1 presenta los casos de uso (CU) que forman parte de la funcionalidad del sistema.

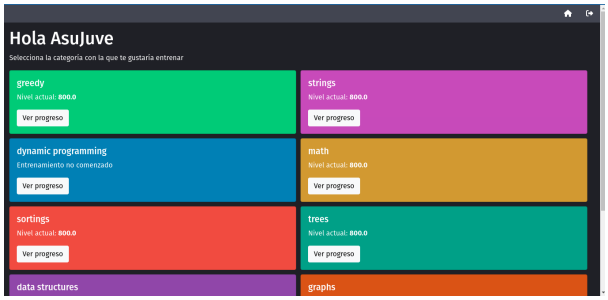
3.3. Desarrollo

Diseño del prototipo. El sistema se estructura en tres módulos esenciales (ver Fig. 1): i) una aplicación web para competidores y administradores; ii) un módulo de recolección y pre-procesamiento de datos; iii) y un modelo de aprendizaje supervisado.

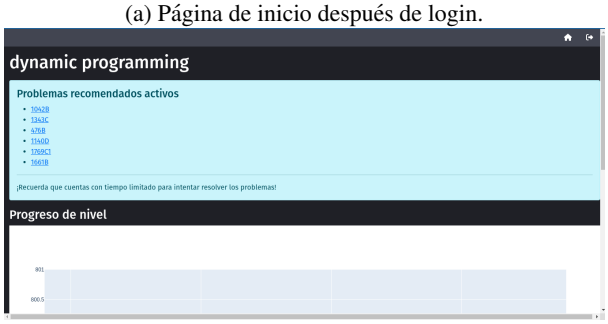
Módulo de aplicación web. En el desarrollo de la aplicación web se ha empleado el framework Django [7], una herramienta que facilita la creación de aplicaciones web eficientes en Python. Este framework sigue el patrón de diseño Modelo-Vista-Controlador (MVC), donde la base de datos PostgreSQL y su interacción con el controlador nos define el modelo, la GUI (Front-end con Bootstrap) nos define la Vista, y los módulos de gestión de manipulación de datos y consultas (módulos 1, 2 y 3) nos define el controlador.

En cuanto a la gestión de datos, se eligió PostgreSQL como sistema de gestión de bases de datos relacional. Para comprender la manera en la que se representa la habilidad de un competidor dentro de la tabla ”Nivel”, es necesario hablar de TrueSkill. TrueSkill es un sistema de ranking basado en habilidades desarrollado por Microsoft Research, diseñado originalmente para evaluar el rendimiento de jugadores en Xbox Live.

Este sistema modela la habilidad de un jugador con dos parámetros clave: μ (mu), que representa la habilidad promedio, y σ (sigma), que mide la incertidumbre asociada a esa habilidad. La combinación de μ y σ genera una curva de creencia sobre la habilidad del jugador. Cuando σ es alto, el sistema tiene menos certeza sobre la verdadera habilidad, reflejando un rango más amplio de posibles habilidades. En contraste, un σ bajo indica una mayor certeza de que la habilidad del jugador está cerca del promedio. TrueSkill utiliza este enfoque en partidas de juego para ajustar el nivel de habilidad de los jugadores en función de los resultados.



(a) Página de inicio después de login.



(b) Recomendaciones activas de usuario.

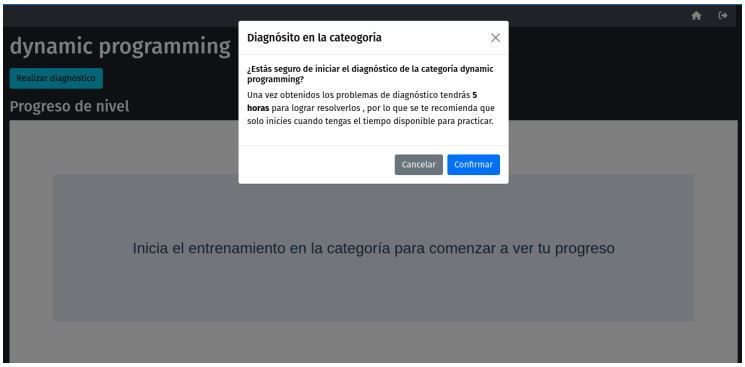
Fig. 5. Pantallas funcionales del sistema una vez ingresado el usuario.

Durante la implementación y adaptación de TrueSkill dentro del sistema de entrenamiento, se decidió que los problemas actúen como ”jugadores”, y la resolución de problemas se equipara a participar en una ”partida”, permitiendo que los competidores mejoren su nivel a medida que resuelven desafíos más complejos. Estos niveles son registrados, incorporando μ y σ para mantener una representación dinámica y precisa de la habilidad del competidor en categorías específicas. El valor μ aumenta en caso de que se haya resuelto el problema y disminuye en el caso contrario. Este proceso dinámico asegura una adaptación continua del sistema TrueSkill a medida que los competidores participan en más ”partidas”de resolución de problemas.

Módulo de recolección y procesamiento de datos

Recolección de datos. En este módulo se llevaron a cabo varias etapas para obtener un conjunto de datos significativo y limpio que sirviera como base para el entrenamiento del modelo. Primeramente se utilizaron técnicas de web scraping con Selenium para extraer 6000 nombres de usuarios de Codeforces. Estos usuarios abarcaban diferentes valores de ranking dentro de la plataforma.

Una vez con los nombres de usuarios, se empleó la API de Codeforces para obtener todos sus envíos realizados. Cada envío representa un intento de resolver un problema y contiene información valiosa; como la dificultad del problema, sus categorías y el veredicto sobre si el problema fue resuelto o no.



(a) Confirmación de inicio de diagnóstico.

(b) Confirmación de recomendación.

Fig. 6. Modales de confirmación dependiendo de la acción realizada.

Pre-procesamiento de datos. Se le dio un formato correcto a los campos de los envíos para facilitar el entrenamiento del modelo. Esto implicó transformar la información de manera que fuera adecuada para su análisis y uso en el sistema, como normalizar y categorizar campos. Además del uso de los campos obtenidos mediante la API de Codeforces, se llevó acabo la creación y actualización del campo user_rating.

Este campo representa el nivel del usuario y se calculó utilizando el sistema de rankeo TrueSkill, del cuál se habló en el módulo anterior. El procedimiento se llevó a cabo tomando en cuenta que los envíos obtenidos desde la API de Codeforces están ordenados cronológicamente. Después de cada envío, se actualizó el nivel del usuario. Este enfoque permitió reflejar los cambios en la habilidad del usuario a lo largo del tiempo, considerando sus interacciones con problemas de distintas categorías y dificultades.

Filtrado de envíos. Para optimizar la calidad del conjunto de datos se realizaron algunas condiciones de filtrado. Por un lado, solo se consideró el último envío por cada usuario para un problema específico, evaluando si finalmente lograron resolver el problema. También se eliminaron envíos de problemas en los que el usuario no realizó, al menos, tantos intentos como el promedio de intentos para resolver ese problema. Esto se hizo para centrarse en problemas que los usuarios realmente intentaron resolver.

Tabla 2. Métricas obtenidas en los modelos de predicción.

Modelo	Precisión	Exactitud	Sensibilidad	F1	Matriz de confusión
Red neuronal	0.9004	0.9004	1	.95	[257, 24312]
					[152, 221078]
Regresión logística	0.9002	0.9002	1	.95	[152, 24417]
					[93, 221137]

Resultados del proceso. Después del proceso, se obtuvo un conjunto de datos limpio y significativo que consta de 1 228 994 envíos realizados por 6 000 usuarios en relación con 8 299 problemas. En la Fig. 2a se muestra la cantidad de envíos por categoría, y en la Fig. 2b la cantidad de problemas por categoría.

Modelo de aprendizaje supervisado

Objetivo de predicción. El objetivo central del modelado de datos en este proyecto es crear un sistema de recomendación de problemas eficaz y personalizado para los competidores de programación competitiva. El sistema debe ser capaz de proporcionar recomendaciones de problemas basadas en la probabilidad de que un competidor resuelva un problema en función de su nivel de habilidad en las categorías del problema, la cual a su vez se basa en el historial de envíos y su desempeño en las mismas categorías.

Procesamiento de datos. Para lograr los objetivos de predicción, se utilizó el conjunto de envíos de soluciones generado en el módulo del sistema anterior. Tomando el campo de *veredicto* como la variable objetivo, siendo este la probabilidad de que un usuario, con cierto nivel de habilidad, solucione un problema con un cierto nivel de dificultad y perteneciente a categorías de algoritmia específicas. Para poder utilizar el conjunto de datos pre-procesado dentro de los modelos a entrenar, se utilizaron tecnologías como Pandas para la manipulación de datos, y scikit-learn para la división del conjunto en datos de entrenamiento y datos de prueba para evaluar el rendimiento del modelo en datos no vistos.

Métodos de modelado. Para lograr el objetivo de un sistema de recomendación preciso y personalizado, se consideraron dos enfoques de modelado principales: redes neuronales y regresión logística. La idea de utilizar redes neuronales en el sistema radica en su capacidad para modelar relaciones complejas y no lineales en los datos, lo que es esencial para capturar los patrones que determinan si un competidor es probable que resuelva un problema específico [8].

Para construir y entrenar la red neuronal se utilizó el modelo Sequential de la librería TensowFlow, donde se trabajo con hiperparámetros tales como el optimizador (Adam, en nuestro caso), función de pérdida (entropía cruzada, en nuestro caso), funciones de activación y tasa de aprendizaje (0.01 en este ejemplo); esto para intentar mejorar el rendimiento del modelo y obtener mayor precisión. La estructura de la red neuronal (completamente conectada secuencial) está definida por 13 neuronas de entrada en la capa 1, dos capas ocultas con 200 y 100 neuronas (relu) respectivamente, mientras que la capa de salida consta de una neurona sigmoidea.

Todo este proceso de entrenamiento realizado en 10 épocas con un tamaño de batch de 32. El total de elementos del dataset fue 2 538 392 muestras, divididas en 80 % para entrenamiento y 20 % para pruebas. Por otro lado, la regresión logística es un método eficiente y ampliamente utilizado para modelar la probabilidad de un resultado discreto, como sí o no, verdadero o falso [17]. En este caso, se está interesado en clasificar si un competidor logrará resolver un problema específico o no y en qué medida, lo que se ajusta perfectamente a la naturaleza binaria de la regresión logística. Para la implementación de la regresión logística se utilizó el modelo LogisticRegression (con solucionador *liblinear*), proporcionado por la librería scikit-learn.

Experimentación. Para evaluar la eficacia del prototipo se propone llevar a cabo un experimento que involucre dos grupos de equipos de programación competitiva. Por un lado, habría un grupo que se entrene utilizando el sistema desarrollado, mientras que el otro grupo seguiría con su entrenamiento habitual. El escenario de prueba se desarrollaría a lo largo de diversas competencias de programación competitiva tipo RPC, similares a las competiciones ICPC.

Durante una competencia reciente, se recopiló información sobre el rendimiento de todos los equipos que podrían participar en el experimento. Se registraron datos como la cantidad de problemas resueltos, el puntaje obtenido y la posición en la tabla de clasificación. Entre las diversas competencias subsecuentes, los equipos continuarían su entrenamiento de acuerdo con los grupos a los que pertenecen. La subsecuentes competencias proporcionarían datos adicionales sobre el desempeño de los equipos. La evaluación de la eficacia del sistema se llevaría a cabo comparando los resultados continuamente.

Esta comparación permitirá determinar si los equipos que utilizaron el sistema mostraron una mejora significativa en comparación con el grupo de control. Es importante destacar que la comparación se realizaría a nivel individual, considerando que los equipos no tienen un nivel de habilidad inicial homogéneo. Además de los datos recopilados de las competencias, se tomará en cuenta el tiempo y la dedicación al entrenamiento por parte de cada equipo. Esta metodología proporcionará una visión integral de la eficacia del sistema en el contexto específico de la programación competitiva, permitiendo identificar el impacto del sistema en el rendimiento individual de los equipos a lo largo del tiempo.

3.4. Evaluación

Durante la primera iteración del desarrollo del prototipo se enfocó en implementar la gestión de problemas, categorías y usuarios en el sistema. Esto le permite a los usuarios administradores el registro y modificación de instancias de estas entidades en el sistema. Para la segunda iteración se trabajó en la generación de predicciones, para lo que fue necesario el pre-procesamiento de los datos. Posterior al procesamiento de los datos limpios, se implementaron las funcionalidades de recomendación de problemas y realización de diagnósticos a los competidores, debido a que ambas utilizan la predicción de probabilidad de un competidor de resolver un problema. Se usaron modelos tanto de red neuronal y regresión logística, y el modelo final fue el que obtuvo el mejor rendimiento.

Durante la tercera fase se agregaron al sistema los elementos visuales para el seguimiento del progreso en categorías de interés. Esto incluyó el graficar el nivel de los competidores a través del tiempo con base a los resultados de los problemas recomendados, además de una tabla con información de cada recomendación.

Además del experimento, para obtener una comprensión completa de la experiencia de los competidores, se ha diseñado una encuesta para obtener retroalimentación de los usuarios. Esta encuesta se enfocará en la facilidad de uso de la aplicación web, la presencia de posibles problemas técnicos en el sistema y la adecuación de los problemas recomendados. Los participantes, además de contestar a las preguntas, podrán proporcionar opiniones sobre su experiencia y sugerencias para posibles mejoras.

4. Discusión de resultados

4.1. Aplicación web del sistema

Para la autenticación de usuarios se tiene la página de inicio de sesión, Fig. 3a, y la página de creación de cuenta, Fig. 3b. En esta última se incluye un campo para el nombre de usuario de Codeforces, el cual se confirma mediante el uso de la API de la misma plataforma. Esto es requerido para que el sistema pueda verificar las soluciones de los problemas recomendados hechas por el usuario, permitiendo actualizar el nivel de habilidad en las categorías a entrenar. Dentro de la página inicial del sitio, Fig. 3a, se encuentran listadas las categorías disponibles para practicar, ahí se muestra el nivel actual en cada una en caso de haber comenzado el entrenamiento en esa categoría.

Dentro de la página de cada categoría se encuentra una gráfica que muestra el nivel del usuario en esa categoría a través del tiempo, Fig. 4a. También se observa una tabla con la información de cada una de los problemas que se le han recomendado al usuario, Fig. 4b. En caso de que el usuario aún no haya comenzado el entrenamiento de una categoría, tendrá la opción de realizar un diagnóstico inicial para conocer su nivel en la categoría. Al seleccionar esta opción se abre un modal, Fig. 4a, para confirmar la elección. Esto último debido a que se tiene que tener en cuenta que el usuario tendrá solo 5 horas para resolver los problemas que se le recomienden en el diagnóstico. La aplicación verificará si se lograron resolver los problemas antes del tiempo límite utilizando la API de Codeforces.

Una vez que se ha completado el diagnóstico en una categoría, se pueden solicitar más problemas para entrenar. Similar al modal de confirmación de diagnóstico, se abre también un modal para confirmar esta acción, Fig. 5b, solo que ahora para una sola recomendación de problema; en este caso tendremos 2 horas para solucionar el problema. El sistema de igual manera utilizará la API de Codeforces para verificar que se haya resuelto el problema. Cuando existen recomendaciones de problemas para las que aún queda tiempo de resolverlas, estas se mostrarán en la parte superior de la página, a manera de enlaces para que el competidor pueda redirigirse a Codeforces a intentar resolverlas, Fig. 5b.

4.2. Modelos de aprendizaje supervisado

Se utilizó el conjunto de datos pre-procesados para el entrenamiento de dos modelos de aprendizaje supervisado, una red neuronal y un modelo de regresión logística.

Las métricas obtenidas de cada modelo se pueden ver en la Tabla 2. Tras evaluar exhaustivamente el rendimiento de dos modelos, la red neuronal y la regresión logística, con base a diversas métricas clave, ambos demostraron un desempeño sobresaliente. Ambos modelos exhibieron una alta precisión, exactitud y sensibilidad, con un F1-score que refleja un equilibrio notable entre precisión y exactitud.

Las matrices de confusión revelan un bajo número de falsos positivos y falsos negativos para ambas arquitecturas. Sin embargo, al analizar minuciosamente los resultados, se observa que la red neuronal presenta una ligera ventaja en términos de precisión, con un valor ligeramente superior. En consecuencia, la elección provisional recae en la red neuronal, respaldada por su leve mejora en precisión, una métrica crítica para este contexto particular.

5. Conclusiones y trabajo futuro

Con el fin de apoyar en el desafío de desconocer la forma efectiva de entrenarse para perfeccionar las habilidades de los concursantes de programación competitiva en los diversos temas abordados en este ámbito, se desarrolló un sistema de recomendación de problemas de programación. Esta propuesta utiliza aprendizaje supervisado para predecir la probabilidad que tiene un usuario de resolver un ejercicio, y a partir de ahí recomendar actividades con una probabilidad intermedia de resolución, proporcionando retos que ayuden a mejorar a los usuarios.

Se ofrece así la oportunidad de obtener sugerencias de problemas para entrenar a partir de las temáticas de algoritmos en las que se quiera mejorar. Para el entrenamiento del modelo de aprendizaje supervisado, se utilizó la información de envíos de soluciones de problemas realizadas por miles de usuarios en la plataforma de Codeforces. En este sentido, para la ponderación del nivel de habilidad de los usuarios se trabajó con el sistema de clasificación TrueSkill.

Como trabajo futuro, se requiere ir aplicando el modelo sugerido en diversas competencias generadas a lo largo del año, lo cual irá proporcionando elementos que brinden argumentos de su nivel de utilidad. A la vez se pueden arrojar factores que muestren si los algoritmos empleados pueden mejorarse o substituirse.

Referencias

1. ACM MPSTME: Benefits of Competitive Programming | LinkedIn (2022) https://www.linkedin.com/pulse/benefits-competitive-programming-association-for-computing-machiner/?trk=organization-update-content_share-article
2. Alnahhas, A., Mourtada, N.: Predicting the performance of contestants in competitive programming using machine learning techniques. Olympiads in Informatics, pp. 3–20 (2020) doi: 10.15388/ioi.2020.01
3. Avkirkar, P.: FYProject - competitive coding problem classifier and problem recommendation (2023) https://github.com/ParasAvkirkar/-Competitive-Coding-Problem-Classifier-and-Recommender/blob/2a93e19905f3922b85a0f0674ef56334ab4f57d1/final_report.pdf
4. Codeforces: Codeforces (2023) https://codeforces.com/
5. CUCEI: Club de Algoritmia CUCEI (2023) http://algoritmia.club/#/home

6. Di-Mascio, T., Laura, L., Temperini, M.: A framework for personalized competitive programming training. In: 17th International Conference on Information Technology Based Higher Education and Training, IEEE, pp. 1–8 (2018) doi: 10.1109/ITHET.2018.8424620
7. Django: Django project (2023) https://www.djangoproject.com/
8. Edgar, T. W., Manz, D. O.: Chapter 4 - Exploratory Study. In: Research Methods for Cyber Security, Syngress, pp. 95–130 (2017), doi: 10.1016/B978-0-12-805349-2.00004-2
9. ESCOM: Clubs Académicos Estudiantiles - ESCOM (2023) https://www.escom.ipn.mx/htmls/escomunidad/clubs.php
10. Google's Coding Competitions: Google's Coding Competitions - Code Jam, Hash Code and Kick Start (2023) https://codingcompetitions.withgoogle.com/
11. IBM: What are convolutional neural networks? | IBM (2023) https://www.ibm.com/topics/convolutional-neural-networks
12. IBM: What is the k-nearest neighbors algorithm? | IBM (2023) https://www.ibm.com/topics/knn
13. ICPC: ICPC (2018) https://icpc.global/
14. ICPC: ICPC FACT SHEET. Technical report (2021) https://icpc.global/community/history/Factsheet2020.pdf
15. IEEE: IEEE standard glossary of software engineering terminology (1990) doi: 10.1109/IEEESTD.1990.101064
16. ITESM MTY: Club de algoritmia ITESM MTY | Facebook (2023) https://www.facebook.com/algoritmiaITESM/
17. Kiang, M. Y.: Neural networks. Encyclopedia of Information Systems, pp. 303–315 (2003) doi: 10.1016/B0-12-227240-4/00121-0
18. Liu, R.: Training ICPC teams: A technical guide (2014)
19. Meta: Meta hacker cup (2022) https://www.facebook.com/codingcompetitions/hacker-cup/
20. Mittal, R.: What is an ELO rating? (2022) https://medium.com/purple-theory/what-is-elo-rating-c4eb7a9061e0
21. omegaUp: Welcome – omegaUp (2023) https://omegaup.com/
22. Prisco, A., dos-Santos, R., Nolibos, A., Botelho, S., Tonin, N., Bez, J.: Evaluating a programming problem recommendation model - a classroom personalization experiment. In: 2020 IEEE Frontiers in Education Conference (FIE), pp. 1–6 (2020) doi: 10.1109/FIE44824.2020.9274028
23. Ramesh, P., Subramanian, K.: Context-aware practice problem recommendation using learners' skill level navigation patterns. Intelligent Automation & Soft Computing, vol. 35, pp. 3845–3860 (2023) doi: 10.32604/iasc.2023.031329
24. Sharma, R.: Unlocking the world of competitive programming: A comprehensive guide (2023) https://emeritus.org/blog/coding-competitive-programming/
25. Sinza, Y., Oliva, J., Guerrero, S.: Análisis de los componentes relacionados en programación competitiva: Un mapeo siste-mático deliteratura. Revista politécnica, vol. 19, no. 38 (2023)
26. Sommerville, I.: Software engineering. Addison-Wesley (2010)
27. Sudha, S., Arun Kumar, A., Muthu Nagappan, M., Suresh, R.: Classification and recommendation of competitive programming problems using CNN. In: International Conference on Intelligent Information Technologies, Springer, pp. 262–272 (2018) doi: 10.1007/978-981-10-7635-0_20
28. Tangri, A.: What are the benefits of learning competitive programming? (2019) https://www.quora.com/What-are-the-benefits-of-learning-competitive-programming
29. UNAM: Club de Programación Competitiva UNAM | Mexico City | Facebook (2023) https://www.facebook.com/clubpumasmas
30. Weitzenfeld, A., Guardati, S.: Capítulo 12: Ingeniería de software: el proceso para el desarrollo de software, Introducción a la Computación. CENGAGE Learning (2007)

31. Yera Toledo, R., Caballero Mota, Y., Martínez, L.: A recommender system for programming online judges using fuzzy information modeling. Informatics, vol. 5, no. 2, pp. 17 (2018) doi: 10.3390/informatics5020017