ISSN 1870-4069

Análisis del comportamiento de estudiantes de programación durante su proceso de aprendizaje utilizando árboles de decisión

Víctor-Gonzalo Rivero-Martínez, Maricela Quintana-López, Asdrúbal López-Chau

Universidad Autónoma del Estado de México,
Centro Universitario UAEM Valle de México,
México

vriverom001@alumno.uaemex.mx,
{mquintanal,alchau}@uaemex.mx

Resumen. El proceso de enseñanza-aprendizaje de la programación es fundamental para el desarrollo de la ciencia y la tecnología, desafortunadamente las estadísticas muestran altos índices de reprobación en materias de programación. Realizar un análisis sobre los factores que influyen en el comportamiento de los estudiantes de programación, permitiría detectar áreas de oportunidad con el fin de proponer mejoras en el proceso. El objetivo del presente trabajo es descubrir patrones, tendencias o comportamientos de estudiantes de programación mediante árboles de decisión. Para obtener los datos, se aplicó una encuesta a 200 estudiantes de programación, posteriormente se generó el árbol de decisión mediante el algoritmo J48 y el método de validación cruzada para las clases correspondientes a los niveles de programación principiante e intermedio, alcanzando en el mejor experimento un 75.5% de acierto y un 24.5% de error. De acuerdo con el árbol de decisión generado se obtuvieron reglas de comportamiento para los alumnos principiantes e intermedios. Los atributos que proporcionan más información son el número de semestres de programación cursados, la herramienta de apoyo que más utilizan, la resolución de dudas por parte del profesor y la ayuda proporcionada por una herramienta computacional.

Palabras clave: Programación, clasificación, arboles de decisión.

Analysis of the Behavior of Programming Students during their Learning Process Using Decision Trees

Abstract. The teaching-learning process of programming is essential for the development of science and technology, unfortunately statistics show high failure rates in programming subjects. Carry on an analysis of the factors that have influence on the behavior of programming students will allow to detect opportunity areas in order to improve the process. The objective of this work is to discover patterns, trends, or behavior of programming students through decision trees. To obtain the data a survey was applied to 200 programming

students, later the decision trees were generated using the J48 algorithm and the cross-validation method to the classes corresponding to the beginner and intermediate programming levels, reaching in the best experiment 75.5% of success and 24.5% of error. According to the generated decision tree, behavioral rules were obtained for beginner and intermediate students. The attributes that provide more information are the number of semesters of programming coursed, the support tool they use the most, the resolution of doubts by the teacher and the help that a computational tool could provide.

Keywords: Programming, classification, decision trees.

1. Introducción

El progreso tecnológico influye en diversos sectores como el empleo, el hogar, la educación, el entretenimiento y la industria, entre otros. La humanidad se encuentra en un momento en el que los avances en la inteligencia artificial, el aprendizaje automático, ciencia de datos e internet de las cosas, entre otras, permiten innovar productos y servicios que utilizamos en nuestras actividades diarias. La tecnología forma gran parte de nuestras vidas cotidianas, por lo tanto, se requiere de personas con un conocimiento especializado que les permita modificarla ya sea con el fin de actualizarla o de crear una nueva tecnología.

De acuerdo con [1], la industria 4.0 requiere de la formación y actualización de profesionales que tengan la capacidad de resolver problemas a través de habilidades digitales, por ello es importante que desde sus primeros años en la universidad aprendan a programar.

La enseñanza de la programación usualmente consiste en impartición de conocimientos y realización de prácticas en una sala de cómputo. De esta forma, se les hace hincapié en que, para aprender a programar hay que programar de manera constante y aumentando la complejidad de manera progresiva tomando como base conceptos anteriores. Programar es crear la solución a un problema utilizando un lenguaje de programación, lo cual requiere un esfuerzo importante de razonamiento, capacidad de análisis y abstracción para así diseñar y proponer la solución más adecuada, por lo que se considera a la tarea de programar una tarea compleja [2].

Aunado a esto se encuentra el entorno de programación donde la función del compilador es detectar los errores léxicos, sintácticos y algunos errores semánticos, como los que pueden ocurrir al realizar la comprobación de tipos. Que un estudiante no pueda corregir este tipo de errores, implica un desconocimiento del lenguaje, así como la falta de interpretación para establecer la relación entre el error y el tema que lo causa.

Además, se ha identificado que un porcentaje importante del grupo se apoya en el docente para corregir los errores de compilación, pero debido al número de estudiantes, no se puede dar una atención personalizada en el tiempo que se tiene causando en ocasiones frustración en los estudiantes.

Por lo anterior, sería conveniente tener información acerca del comportamiento y opinión de los estudiantes en su proceso de aprendizaje, que se pueda analizar con el fin de descubrir patrones o tendencias para posteriormente detectar áreas de oportunidad y proponer líneas de mejora a dicho proceso. Para lograr lo anterior, en este trabajo se utilizará el algoritmo J48 utilizando ocho atributos y las clases

principiante e intermedio del nivel de programación. A partir del árbol generado se observarán los atributos que aportan mayor información al análisis y las reglas generadas, lo cual corresponderá a las tendencias y comportamientos de los estudiantes.

La presente investigación se organiza de la siguiente manera. En la sección 2 se describen los trabajos relacionados, en la sección 3, el desarrollo del clasificador. En la sección 4, se describen los experimentos realizados y sus resultados. Por último, en la sección 5 se presentan las conclusiones y trabajo futuro.

2. Trabajos relacionados

La problemática de la enseñanza de la programación ha sido abordada en investigaciones realizadas principalmente en el nivel superior motivadas por los altos índices de reprobación en estas asignaturas.

Por ejemplo, en un estudio realizado en el año 2021 en una institución de educación superior en Veracruz, se utilizaron árboles de decisión para determinar el rendimiento académico de los estudiantes de carreras relacionadas con la programación. En el estudio se aplicó un cuestionario a 341 de los estudiantes, obteniendo como resultado que el 48.1% de ellos necesitan apoyo académico y que las variables de aprendizaje en el aula y tutorías tienen relación con la variable rendimiento académico [3].

En otro estudio realizado en el año 2019 para la Licenciatura en Ingeniería de Software de la Unidad Multidisciplinaria Tizimín en Yucatán, se tomaron muestras de los indicadores, de los años 2017 y 2018, de la asignatura de Algoritmia, donde en 2017 el 46.15% tuvo un desempeño entre Satisfactorio y Sobresaliente y el 53.84% entre Suficiente y No aprobado, concluyendo que el rendimiento en general de los grupos no era el óptimo y esto influía en los semestres posteriores [4].

Por otro lado, Fuentes y Moo determinan y clasifican las dificultades que tienen los estudiantes de las ingenierías en sistemas y electromecánica al momento de solucionar problemas mediante la programación básica. La clasificación de las dificultades y su propuesta de solución se presenta en la ¡Error! No se encuentra el origen de la referencia. [5].

3. Metodología

Para desarrollar el clasificador se utilizó la metodología para el descubrimiento de conocimiento (KDD) propuesta por Fayyad [9], la cual consta de las etapas de selección, preprocesamiento, transformación, minería de datos y, por último, interpretación y evaluación (ver Fig. 1). La explicación de cada etapa se presenta a continuación:

Selección de datos. Se reúnen los datos a utilizar en la investigación y se seleccionan a los más relevantes.

Preprocesamiento y transformación de los datos. Se seleccionan los atributos a utilizar y son convertidos en un formato computacionalmente apropiado para el análisis de minería de datos.

Tabla 1. Clasificación de las dificultades al programar.

Dificultad	Descripción	Propuesta de solución
Fobia a los problemas complejos	Los estudiantes tienden a no solucionar problemas que los impacten	Enseñar al estudiante a dividir el problema en pequeños subproblemas
Lógica incompleta	No consiguen establecer los pasos necesarios para llegar a una solución completa.	Fomentar la solución de acertijos en los que se apliquen los pasos para solucionar un problema.
Desconocen el lenguaje	Desconocen las palabras reservadas o bibliotecas del lenguaje	Elaborar material didáctico que sirva como guía
Desconocer las herramientas del entorno de desarrollo (IDE)	Los estudiantes escribían sus códigos, pero no lograban corregir los errores de lógica.	Enseñar a los estudiantes como correr el programa paso a paso empleando el entorno de programación

Minería de datos. Se aplican las técnicas y algoritmos que se encargan de extraer y representar el conocimiento de forma adecuada el propósito de identificar conocimientos, patrones o tendencias de los datos.

Interpretación y evaluación del conocimiento descubierto. Se analiza la consistencia de los resultados alcanzados.

4. Desarrollo del clasificador

A continuación, se presenta la aplicación de las etapas de la metodología con el fin de clasificar a los estudiantes en principiantes e intermedios, así como obtener las reglas de comportamiento durante su aprendizaje.

4.1. Selección de datos

Para este trabajo, la población a considerar fueron los estudiantes de los primeros semestres de programación de las carreras de Ingeniería en Sistemas y Comunicaciones e Ingeniería en computación del Centro Universitario UAEM Valle de México. Esta población actualmente está constituida por aproximadamente 500 estudiantes, por lo que con un nivel de confianza del 93% y un margen de error máximo de 5% y la ecuación para el muestreo aleatorio simple (Ecuación 1), se tuvieron que encuestar a 200 estudiantes mediante un cuestionario en línea consistente en 23 preguntas, 18 de opción múltiple y cinco preguntas abiertas:

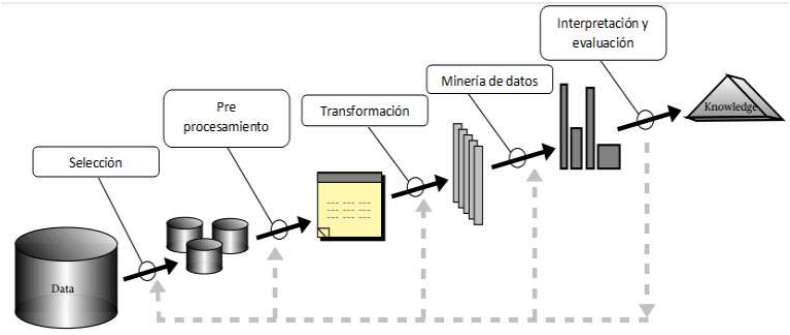
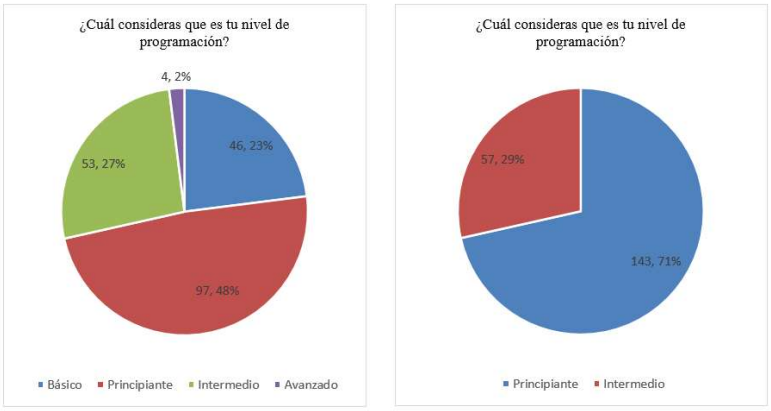


Fig. 1. Etapas del proceso de extracción del conocimiento.



a) b)

Fig. 2. Preprocesamiento de los resultados de las preguntas a) Pregunta original consistente en 4 opciones b) Reducción a dos opciones.

$$M = \frac{\frac{z^2 * p(1-p)}{e^2}}{1 + \frac{z^2 * p(1-p)}{e^2 N}}, \quad (1)$$

donde M es el tamaño de la muestra, N es el tamaño de la población, e es el margen de error y z es la cantidad de desviaciones estándar que una proporción determinada se aleja de la media.

4.2. Preprocesamiento y transformación de datos

Debido a que algunas de las respuestas de las preguntas tenían un porcentaje muy bajo se optó por agrupar las opciones en dos. En la Fig. 2 se muestra un ejemplo de este proceso.

Además, en la encuesta aplicada se eliminaron aquellas preguntas de carácter personal ya que no aportaban información para el estudio, además de las preguntas

Tabla 2. Preguntas para el análisis y resultados.

Pregunta	Resultados
Semestres cursados de programación en cualquier lenguaje:	1: 9% 2: 32% 3: 25% 4: 13% 5 o más: 21%
Nivel de programación:	Principiante: 71.5% Intermedio: 28.5%
Tipo de material didáctico utilizado durante el curso:	Video: 60.5% Libro: 4.5% Otra: 1.5%
Si el profesor me dedicara más tiempo aprendería más:	Sí: 57.5% No: 42.5%
Cuando mis compañeros tienen dudas el profesor las resuelve oportunamente:	Sí: 87.5% No: 12.5%
Las clases son generales y no están adaptadas a los objetivos particulares de cada estudiante.	Sí: 57.5% No: 42.5%
Considero que una herramienta computacional ayudaría a mejorar mi aprendizaje.	Si: 89.5% No: 10.5%
De los errores del compilador, sé a qué se refieren y de que tema de estudio se trata:	Frecuentemente: 47% Algunas veces: 53%
Me cuesta trabajo entender los mensajes de error del compilador porque está en otro idioma y eso no me ayuda:	Si: 30.5% No: 69.5%
Considero que cuando no puedo realizar el ejercicio de programación que me dejaron es por qué:	No pude comprender el problema: 16.5% Comprendo el problema, pero no sé cómo resolverlo: 17% Comprendo el problema, ya sé cómo resolverlo, pero no sé cómo programarlo: 39% Comprendo, sé cómo resolverlo, lo programo, pero el compilador me marca errores: 23% Comprendo, sé cómo resolverlo, lo programo pero no hace lo que me pidieron: 4.5%

abiertas. Considerando finalmente un total de ocho preguntas como atributos para la clasificación del nivel de programación en intermedio o avanzado. En la Tabla 2 se muestra como quedaron las preguntas para el análisis y sus resultados.

Tabla 3. Experimentos en el Algoritmo J48 para el nivel de programación con 10 pliegues.

No.	Confidencia	Acierto	Error	Matriz de confusión	
				Principiantes	**Intermedios**
1	0.05	71.5%	28.5%	143	0
				57	0
2	0.1	70.5%	29.5%	141	2
				57	0
3	0.15	72%	28%	138	5
				51	6
4	0.20	75%	25%	131	12
				38	19
5	0.25	75.5%	24.5%	129	14
				35	22

4.3. Minería de datos

En esta investigación se utilizó el algoritmo J48 del software Weka [7], el cual genera un árbol de decisión que partición a el conjunto de entrenamiento con base en la selección de atributos que aportan más información para la determinación de la clase. Este proceso es recursivo para cada división que se realiza [8].

El factor de confianza es el parámetro que influye en el tamaño y capacidad de predicción del árbol construido y define la probabilidad de error que se permite en cada operación.

A menor probabilidad, se exige que la diferencia en los errores de predicción antes y después de podar sea más significativa para no podar. El valor por defecto es del 25%, si este valor baja se permiten más operaciones de poda [9].

4.4 Evaluación

El modelo que se utilizó en la investigación fue el de validación cruzada con n pliegues, en el cual los datos se dividen en n partes, utilizándose n-1 partes para entrenamiento y uno para prueba, en este caso, como se utilizaron 10 pliegues, 90% de los datos se utilizaron para entrenamiento y 10% para prueba. Este proceso se hace 10 veces y los errores calculados serán el promedio de todas las ejecuciones [9]**.**

4.5 Interpretación y uso

Mediante el modelo generado por el algoritmo J48 se analizó el comportamiento de los estudiantes durante su proceso de aprendizaje de programación y para observar la influencia y relación que tienen en la determinación de su nivel de programación.

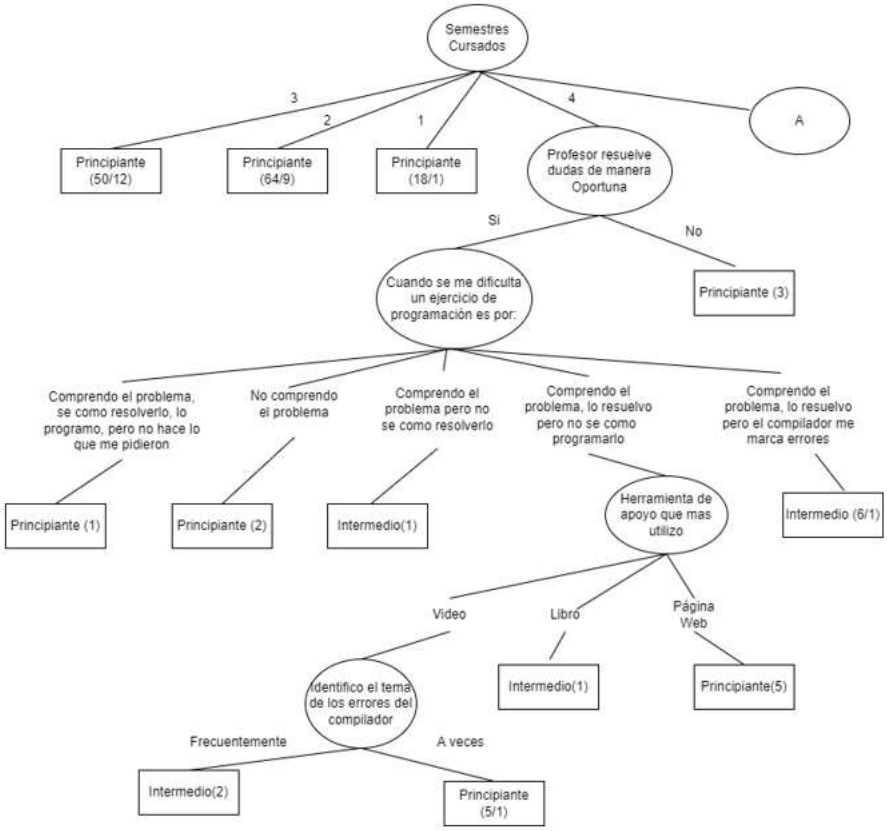


Fig. 3. Rama izquierda del árbol de decisión del nivel de programación.

5. Experimentos y resultados

Se generó el clasificador para las 200 instancias (143 principiantes y 57 intermedios) utilizando el algoritmo J48 y se realizó la prueba del modelo utilizando el método de validación cruzada variando el nivel de confidencia. Los resultados obtenidos se presentan en la Tabla 3, en la cual se observa que observa que el mejor resultado es alcanzado en el experimento 5 donde con una confidencia de 0.25 se alcanza un porcentaje de instancias clasificadas correctamente de 75.5% y un porcentaje de instancias clasificadas incorrectamente de 24.5%.

Para el mismo experimento, la matriz de confusión, indica que, de 143 alumnos que se consideran principiantes, 129 son clasificados correctamente, y 14 incorrectamente como intermedios. Por otro lado, de los 57 intermedios, solo se logra clasificar correctamente a 22 e incorrectamente a 35.

El árbol de decisión generado para el experimento 5 fue dividido a fin de poder presentarlo y analizarlo. La parte izquierda del árbol se presenta en la Fig. 3 y la parte derecha en la Fig. 4. A continuación se realiza un análisis de cada parte.

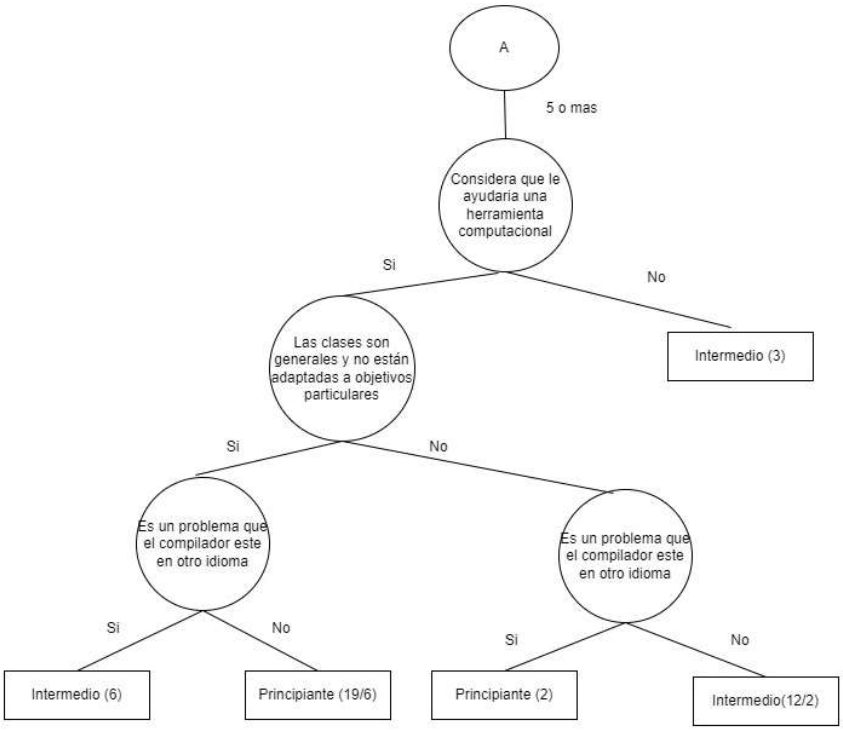


Fig. 4. Rama derecha del árbol de decisión del nivel de programación.

En la Fig. 3. se observa que, si el número de semestres está entre uno y tres, el estudiante es clasificado como principiante, específicamente, 132 instancias se clasifican como principiantes (18 para uno, 64 para dos y 50 para tres), con 22 instancias mal clasificadas.

También que un estudiante se considera principiante cuando su programa no hace lo que le pidieron o no comprenden el problema que les solicitan resolver. Por otro lado, los intermedios comprenden el problema, saben cómo resolverlo, lo programan, pero el compilador les marca errores.

En el mismo árbol se pueden observar que los estudiantes cuando comprenden el problema y no saben cómo resolverlo, recurren a una herramienta de apoyo, los principiantes recurren a páginas web y video tutoriales, mientras que los intermedios a libros y video tutoriales.

En la Fig. 4 se observa a los estudiantes han llevado 5 o más cursos de programación, de los cuales se observa que son intermedios si consideran que no necesitan una herramienta computacional de apoyo para su aprendizaje, aunque se siguen considerando intermedios si consideran varios factores como la ayuda de la herramienta computacional, que las clases son generales y no están adaptadas a sus objetivos particulares y que es un problema que el compilador este en otro idioma, lo cual habla que consideran importante el entorno de programación.

Se puede destacar de los alumnos intermedio que, si no consideran sus clases generales, entonces no es un problema que el compilador este en otro idioma, lo que puede hablar de cierta independencia.

Por otro lado, se consideran principiantes si para ellos es importante la herramienta computacional, que las clases con generales, pero no consideran que sea un problema que el compilador este en otro idioma, esto puede ser debido a que no toman en cuenta los mensajes del compilador.

Las reglas de comportamiento que se generaron en el árbol de decisión son las siguientes:

Reglas de comportamiento de los estudiantes.

Los alumnos son principiantes si:

1. Han cursado de uno a tres semestres de programación.
2. No consideran que su profesor resuelva dudas de manera oportuna.
3. Si al momento de realizar un ejercicio de programación han cursado cuatro semestres, consideran que su profesor revuelve las dudas oportunamente y cuando les dejan un programa para resolver un problema ellos:
   - Comprenden el problema, lo resuelven, pero al programarlo no hace lo que les pidieron.
   - No comprenden el problema a resolver.
4. Cuando les dejan un ejercicio de programación, comprenden el problema, saben cómo resolverlo, pero no programarlo, además la herramienta de apoyo que utilizan es una página web y a veces identifican el tema de estudio por medio de los errores del compilador.
5. Si han cursado 5 semestres o más de programación, les gustaría utilizar una herramienta computacional de apoyo y:
   - Consideran que los cursos son muy generales y no están adaptados a sus objetivos particulares.
   - No consideran un problema que el compilador este en otro idioma.

Los alumnos son intermedios si:

1. Si han cursado al menos 5 semestres de programación y no necesitan utilizar una herramienta computacional de apoyo.
2. Si al momento de realizar un ejercicio de programación han cursado 4 semestres, consideran que su profesor es oportuno en la resolución de dudas y cuando les dejan un programa para resolver un problema ellos:
   - Comprenden el problema, pero no saben cómo resolverlo.
   - Lo programan y el compilador les marca errores,
   - Cuando les dejan un ejercicio de programación, comprenden el problema, saben cómo resolverlo, pero no programarlo, la herramienta de apoyo que utilizan es un video tutorial o libro y frecuentemente identifican el tema de estudio por medio de los errores del compilador.

3. Si han cursado 5 semestres o más de programación, les gustaría utilizar una herramienta computacional de apoyo y:
   - No es un problema que los mensajes de error del compilador estén en otro idioma.
   - No consideran relevante que sus clases son generales y no estén alineadas a sus objetivos particulares.

6. Conclusiones y trabajo a futuro

En el presente trabajo se realizó una encuesta a 200 estudiantes de los cuales 143 se consideran principiantes y 57 intermedios. Se aplicó el modelo J48 para obtener el modelo de árbol de decisión obteniéndose un 75.5% instancias clasificadas correctamente y 24.5% de instancias clasificadas incorrectamente. De árbol de decisión obtenido se observa que el atributo que proporciona más información para la clasificación de las clases es la cantidad de semestres que los alumnos han cursado programación, obteniendo que de las 200 instancias 132 alumnos que han cursado de uno a tres semestres, se consideran principiantes. Otro atributo relevante es la herramienta de apoyo que utilizan en su aprendizaje, donde se encontró que el video tutorial es la herramienta que más se utiliza y que si utilizan el libro de texto se consideran intermedios.

En el caso del entorno de programación se puede concluir que, si no comprenden el problema o no lo pueden resolver, entonces se consideran principiantes y si llegan a programarlo y el compilador les marca errores se consideran intermedios. Cabe resaltar que en la encuesta se les pregunto a los alumnos si les gustaría utilizar una herramienta computacional que sirva como apoyo a su aprendizaje, este atributo fue relevante para los estudiantes que ya estudiaron cinco o más semestres de programación, quienes seguramente ya tienen necesidades más específicas en su aprendizaje.

Se considera que el objetivo del presente trabajo se cumplió ya que se descubrieron patrones de comportamiento relevantes de los estudiantes durante su proceso de aprendizaje.

Las áreas de oportunidad que se pueden destacar en este estudio son, que, aunque los estudiantes ya hayan cursado cinco semestres o más, se consideran principiantes, por lo que se puede proponer que elaboren o participen en proyectos con aplicaciones reales o más avanzadas, por otro lado, las herramientas computacionales pueden jugar un rol importante en el soporte a estudiantes durante la codificación ya que manejo del entorno también resulto relevante en este estudio.

Como trabajo futuro queda analizar mediante minería de textos las respuestas de los estudiantes a las preguntas abiertas y verificar si hay alguna relación con el presente análisis.

Referencias

1. Muñoz-La Rivera, F., Hermosilla, P., Delgadillo, J., Echeverría, D.: Propuesta de construcción de competencias de innovación en la formación de ingenieros en el contexto de la industria 4.0 y los objetivos de desarrollo sostenible (ODS). Formación Universitaria, vol. 14, no. 2 (2021) doi: 10.4067/S0718-50062021000200075

2. Compañ-Rosique, P., Satorre-Cuerda, R., Llorens-Largo, F., Molina-Carmona, R.: Enseñando a programar: un camino directo para desarrollar el pensamiento computacional. Revista de Educación a Distancia., no. 46 (2015)
3. Díaz-Martínez, M. A., Ahumada-Cervantes M. A., Melo-Morín, J. P.: Arboles de decisión como metodología para determinar el rendimiento académico en educación superior. Revista Lallista de Investigación, vol. 18, no 2 (2022) doi: 10.22507/rli.v18n2a8
4. Narváez Díaz, L. E., López Martínez, R. E.: Propedéutico como estrategia de apoyo en una asignatura de la Licenciatura en Ingeniería de Software de la Unidad Multidisciplinaria Tizimín. Revista Electrónica sobre Tecnología, Educación y Sociedad, vol. 7, no. 14, pp. 1–18 (2020)
5. Fuentes-Rosado, J., Moo-Medina, M.: Dificultades de aprender a programar. Educación en Ingeniería, vol. 12, no. 24, pp. 76–82 (2017) doi: 10.26507/rei.v12n24.728
6. Osman, A. S.: Data Mining Techniques: Review. International Journal of Data Science Research, vol. 2, no. 1 (2019)
7. Weka 3: Machine learning software in java [En línea]. Available: https://www.cs.waikato.ac.nz/ml/weka/ [Último acceso: 23 04 2022]
8. Cardona-Taborda, C. H., Gelvez-García, N., Palacios-Rozo, J.: Análisis de datos mediante el algoritmo de clasificación J48, sobre un cluster en la nube de AWS. Redes de Ingeniería, pp. 3–25 (2017). doi: 10.14483/udistrital.jour.redes.2016.3.a01
9. García-Jiménez, M., Álvarez-Sierra, A.: Análisis de Datos en WEKA – Pruebas de selectividad (2010) [En línea]. Available: http://www. it. uc3m. es/jvillena/irc/practicas/06-07/28. pdf [Último acceso: 23 04 2022]
10. Fayyad, U., Piatetky-Shapiro, G., Smyth, P.: The kdd process for extracting useful knowledge from volumes of data. Communications of the ACM, vol. 39, no. 11, pp. 27–34 (1996)