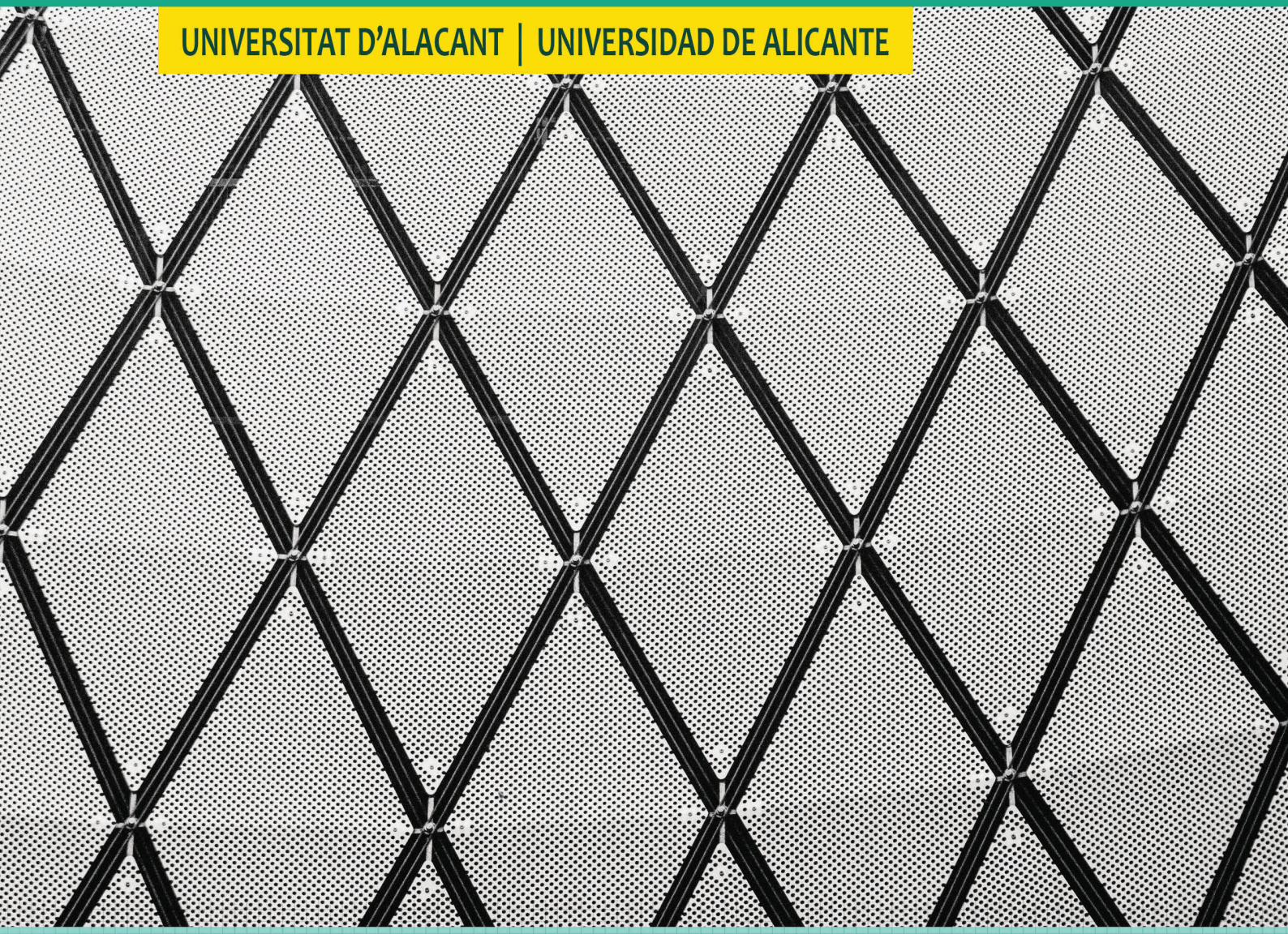
# Redes de Investigación e Innovación en Docencia Universitaria

Volumen 2022

# *Xarxes d'investigació i Innovació en Docència Universitària*

Volum 2022

UNIVERSITAT D'ALACANT | UNIVERSIDAD DE ALICANTE



UA | ICE
UNIVERSITAT D'ALACANT
UNIVERSIDAD DE ALICANTE
Institut de Ciències de l'Educació
Instituto de Ciencias de la Educación

Satorre Cuerda, Rosana (Coordinación)
Menargues Marcilla, María Asunción
Díez Ros, Rocío
Pellín Buades, Neus (Eds.)

# Redes de Investigación e Innovación en Docencia Universitaria. Volumen 2022

Rosana Satorre Cuerda (Coord.),

Asunción Menargues Marcilla, Rocío Díez Ros & Neus Pellín Buades(Eds.)

2022

Redes de Investigación e Innovación en Docencia Universitaria. Volumen 2022

Organització: Institut de Ciències de l'Educació de la Universitat d'Alacant/ *Organización: Instituto de Ciencias de la Educación de la Universidad de Alicante*

Edició / *Edición: Rosana Satorre Cuerda (Coord.), Asunción Menargues Marcilla, Rocío Díez Ros & Neus Pellín Buades(Eds.)*

Comité tècnic / *Comité técnico*:

Cristina Mansilla Martínez

Sergio Andrés Mijangos Sánchez

Neus Pellín Buades

Revisió i maquetació: ICE de la Universitat d'Alacant/ Revisión y maquetación: ICE de la Universidad de Alicante

Primera edició: / *Primera edición*:



ice@ua.es

ISBN: 978-84-09-39082-3



Producció: Institut de Ciències de l'Educació (ICE) de la Universitat d'Alacant / *Producción: Instituto de Ciencias de la Educación (ICE) de la Universidad de Alicante*

# 15. Extracción de perfiles de alumnos de programación en estudios superiores mediante aprendizaje automático en evaluaciones tipo juez en línea

Rico-Juan, Juan Ramón; Sánchez-Cartagena, Víctor M.; Valero-Mas, Jose J.; Gallego, Antonio-Javier; Rizo, David

*Universidad de Alicante*

**RESUMEN**

Los sistemas de juez en línea (JL) se utilizan en el ámbito de la programación, ya que permiten realizar evaluaciones rápidas y objetivas del código desarrollado por los estudiantes. Este tipo de evaluación suele proporcionar una única decisión sobre si el envío ha cumplido con éxito la tarea. Sin embargo, dado que en un contexto educativo dicha información puede considerarse insuficiente, sería beneficioso tanto para el estudiante como para el instructor recibir información adicional sobre el desarrollo general de la tarea. Este trabajo pretende abordar esta limitación considerando la explotación adicional de la información recopilada por el JL e infiriendo automáticamente la retroalimentación, tanto para el estudiante como para el instructor. Consideramos el uso de esquemas basados en Aprendizaje Automático para modelar el comportamiento del estudiante y de Inteligencia Artificial Explicable para proporcionar una retroalimentación comprensible para el ser humano. La propuesta ha sido evaluada en un caso de estudio considerando más de 2.600 envíos y 90 estudiantes de la asignatura *Desafíos de Programación* (Ingeniería en Informática, Universidad de Alicante). Los resultados obtenidos validan la propuesta: el modelo es capaz de predecir de forma significativa el resultado del usuario (aprobar o suspender la tarea) basándose únicamente en el patrón de comportamiento inferido por los envíos realizados al JL, además retroalimentar al estudiante y al instructor.

**PALABRAS CLAVE:** Identificación del perfil del estudiante, Sistemas tipo juez en línea, Inteligencia artificial explicable, Aprendizaje automático.

## 1. INTRODUCCIÓN

Acuñado originalmente por Kurnia et al. (2001), el término juez en línea (JL) designa a los sistemas concebidos para la evaluación y calificación automatizada de tareas de programación. Más concretamente, estos métodos representan los servicios de evaluación en línea capaces de recoger códigos fuente, compilarlos, evaluar sus resultados y calcular las puntuaciones en base a diferente criterios (Wasik et al., 2018). Estas herramientas automatizadas han sido especialmente consideradas en dos escenarios (Yera & Martínez, 2017): (i) concursos y competiciones de programación, y (ii) contextos educativos en titulaciones académicas. Este trabajo se centra en este último escenario, en concreto, en los cursos de programación de los estudios de Informática en los centros de enseñanza superior.

A pesar de sus claras ventajas, los sistemas de JL no proporcionan al estudiante ni al instructor ninguna información sobre el envío real, aparte de si el código proporcionado cumplió con éxito la tarea (Mani et al., 2014). Sin embargo, la información recopilada por el sistema JL puede aprovecharse para enriquecer el proceso educativo mediante extrayendo automáticamente información adicional, como los hábitos de los estudiantes o los patrones de comportamiento relacionados con el éxito (o el fracaso) de la tarea (Asif et al., 2017).

Formalmente, la Minería de Datos Educativos (en adelante EDM, del inglés *Educational Data Mining*) representa la disciplina destinada a inferir patrones descriptivos y predicciones a partir de entornos educativos, tales como la caracterización de los comportamientos y logros de los alumnos, el análisis del contenido del conocimiento del dominio, la autonomía evaluaciones o funcionalidades educativas, entre otras (Peña-Ayala, 2014). Dentro de esta disciplina, el aprendizaje automático (en adelante ML, del inglés *Machine Learning*) se presenta como uno de los principales marcos para llevar a cabo este proceso de inferencia de conocimiento debido a su potencia y flexibilidad.

Cuando se utiliza un JL para calificar una tarea de programación, suele haber un periodo de tiempo en el que los alumnos pueden realizar tantos envíos como consideren. La calificación de un estudiante en la actividad se calcula en base al mejor envío. Durante la evaluación es posible que no se disponga de otros datos adicionales como las calificaciones en actividades anteriores (Gray & Perkins, 2019), el éxito académico en otros cursos (Pal & Pal, 2013; Alturki et al., 2021) o el origen socioeconómico que desde un punto de vista ético introduciría los posibles sesgos. A pesar de la falta de estos datos seguiría siendo deseable poder detectar a los estudiantes en riesgo antes de la fecha límite de la actividad.

En nuestro caso, pretendemos aplicar enfoques de ML a los sistemas de JL para las tareas de programación. Basándonos en esto, y ayudados por el uso de meta-información recopilada de los procedimiento, pretendemos inducir dos tipos de resultados: (i) la probabilidad de éxito de un nuevo estudiante, y (ii) la identificación de diferentes perfiles de alumnos para proporcionar retroalimentación tanto al instructor como al propio estudiante. Obsérvese que este tipo de información puede

utilizarse no sólo para prevenir actitudes inadecuadas de los estudiantes, proporcionando las observaciones adecuadas sobre el desarrollo de la tarea, sino también para ajustar adecuadamente la dificultad de las distintas tareas, entre otras posibles acciones correctivas para el éxito del curso.

Sin embargo, el hecho de que la estrategia de ML generalmente funcione en modo de caja negra dificulta su aplicación en este contexto orientado a la retroalimentación (Komárek et al., 2021). En este sentido, el campo de la Inteligencia Artificial eXplicable (en adelante XAI, del inglés *eXplainable Artificial Intelligence*) está ganando gradualmente la atención para hacer frente a esta limitación mediante el diseño de metodologías que permiten a los seres humanos entender e interpretar las decisiones tomadas por un modelo computacional (Burkart & Huber, 2021).

Teniendo en cuenta todo lo anterior, este trabajo presenta un método para identificar los perfiles de los estudiantes en los sistemas educativos de JL con el objetivo de proporcionar retroalimentación tanto a los estudiantes como a los instructores sobre el desarrollo de la tarea. Más concretamente, la propuesta se basa exclusivamente en la meta-información extraída de estos sistemas JL y considera un marco ML para inferir automáticamente estos perfiles junto con métodos XAI para proporcionar interpretabilidad sobre los comportamientos estimados. La metodología propuesta se ha evaluado en un caso de estudio que comprende tres cursos académicos relacionados con la programación y más de 2.600 envíos correspondientes a dos actividades distintas.

## 2. MÉTODO

Predecir el rendimiento de los estudiantes al abordar tareas de programación y proporcionarles retroalimentación en consecuencia representa un planteamiento considerablemente amplio que requiere ser particularizado. En este sentido, este trabajo plantea un conjunto de preguntas de investigación a las que nuestra propuesta metodológica pretende dar respuesta:

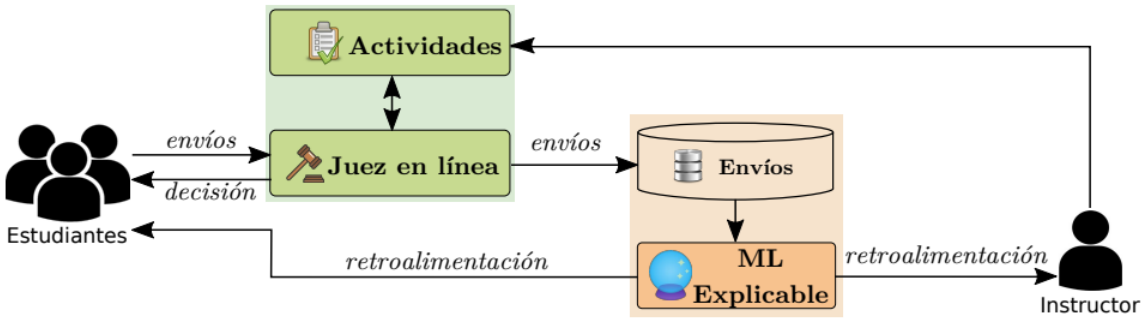
**Pregunta de investigación 1 (RQ1)**: ¿Cuándo deben los estudiantes empezar a presentar sus solicitudes al sistema de JL?
**Pregunta de investigación 2 (RQ2)**: ¿Cuántos envíos son razonables para el éxito de la tarea?
**Pregunta de investigación 3 (RQ3)**: ¿Hay algún consejo para que los estudiantes aborden adecuadamente las tareas?

La Figura 1 muestra gráficamente el esquema propuesto para abordar cuantitativamente estas cuestiones, que comprende los siguientes pasos:

1. El profesor define las diferentes actividades que deben resolver los alumnos y configura el sistema JL en consecuencia.
2. Los alumnos abordan la tarea planteada y presentan sus implementaciones (envíos).
3. El JL evalúa estos envíos y proporciona a los alumnos una nota de corrección basada exclusivamente en la evaluación del código de programación enviado.

4. Al mismo tiempo, estos envíos son procesados por un módulo adicional que proporciona información tanto al profesor –que puede asistir al estudiante– como al propio estudiante –que puede ajustar su compromiso con la tarea–. Nótese que este elemento representa el núcleo del trabajo, ya que pretende modelar el comportamiento del usuario considerando un marco de aprendizaje supervisado

Figura 1. Esquema propuesto.




El EDM comprende una amplia gama de esquemas capaces de modelar y predecir el rendimiento de los estudiantes. En nuestro caso, abordamos esta tarea considerando una adaptación de algoritmos ML estándar que permite el uso de técnicas XAI para utilizar una retroalimentación comprensible para el ser humano.

Las técnicas de XAI suelen dividirse en dos familias (Kenny et al., 2021): (i) métodos intrínsecamente explicables que representan los que transmiten directamente el funcionamiento del modelo; y (ii) explicaciones post-hoc, que intentan proporcionar justificaciones sobre la razón por la que el modelo llegó a sus predicciones. Este trabajo se enmarca en este último caso ya que permite utilizar una amplia gama de algoritmos. A su vez dentro de este marco post-hoc, los dos enfoques más comunes se basan en realizar permutaciones sobre los valores de cada predictor (variable independiente) para evaluar su influencia en las predicciones (variable dependiente) o en crear un modelo lineal alternativo y sencillo de explicar que imite el comportamiento del que nos ocupa (Arrieta et al., 2020). Dado que esta última familia de enfoques suele obtener resultados más precisos es la que usaremos en este estudio. Más concretamente, recurrimos a las denominadas *Shapley Additive Explanations* (SHAP) introducidas por Lundberg & Lee (2017), ya que representa una de las técnicas para XAI más aceptadas en la literatura (Fryer et al., 2021). SHAP se basa en el concepto de los valores de Shapley, una técnica muy conocida en el campo de la teoría de los juegos cooperativos ideada para medir la contribución individual de cada jugador al juego (Roth, 1988). Obsérvese que este proceso se corresponde con el bloque *ML Explicable* representado en la Figura 1.

## 3. CASO DE ESTUDIO

El caso de estudio se refiere a la asignatura *Desafíos de Programación* del último curso del Grado en Informática de la Universidad de Alicante. Una parte de la evaluación en esta asignatura se

basa en dos actividades de tres semanas de duración cada una (A1 y A2). Se trata de la aplicación de técnicas de optimización y se desarrollan de forma autónoma por los alumnos con una supervisión semanal. A lo largo del proceso, se envía el código con las posibles soluciones al JL que informa sobre si la entrega es correcta o no.

Los datos utilizados en este caso de estudio se recogieron durante tres cursos académicos (2019-2020, 2020-2021 y 2021-2022). La Tabla 1 presenta una descripción de esta colección en términos de número de alumnos, cantidad de envíos y tasa de éxito.

Tabla 1. Descripción de los datos recogidos para este estudio. Para cada año académico y tarea, se proporciona el número de estudiantes matriculados, la cantidad de envíos y las estadísticas sobre el éxito/fracaso de la tarea.

| Año | Actividad | Estudiantes | | | Envíos | | | Promedio de envíos por estudiante | | Envíos hasta el éxito |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Total | Éxitos | Fallos | Total | Éxitos | Fallos | Éxitos | Fallos | |
| 2019-20 | A1 | 21 | 13 | 8 | 317 | 7.9% | 92.1% | 13 | 18 | 12.3 |
| | A2 | 18 | 11 | 7 | 316 | 7.6% | 92.4% | 24 | 8 | 22.2 |
| 2020-21 | A1 | 37 | 31 | 6 | 626 | 43.3% | 56.7% | 17 | 15 | 8.7 |
| | A2 | 33 | 27 | 6 | 452 | 25.4% | 74.6% | 14 | 14 | 9.2 |
| 2021-22 | A1 | 28 | 25 | 3 | 411 | 13.9% | 86.1% | 15 | 11 | 13.6 |
| | A2 | 24 | 21 | 3 | 431 | 16.7% | 83.3% | 17 | 27 | 13.9 |

El sistema JL utilizado es Javaluador (Carrasco et al., 2010), el cual comprende una colección de más de 55 problemas de optimización especialmente diseñados para ser abordados con técnicas de programación *dinámica* o *ramificación y poda*. Como sugiere su nombre, este JL evalúa envíos implementados en el lenguaje de programación Java.

Javaluador evalúa la eficacia y corrección de los programas enviados según con límite de tiempo (10 segundos) y uso de memoria 100 MB. Obsérvese que, si bien este JL proporciona cierta información al alumno en cada uno de los envíos, no se proporciona ninguna información sobre el rendimiento general del alumno dentro de la propia actividad o del curso. En este sentido, consideramos que, si se procesan adecuadamente mediante sistemas basados en el aprendizaje y técnicas de XAI, estos envíos individuales pueden proporcionar dicha información tanto al propio estudiante como al instructor del curso.

A continuación, definimos los distintos parámetros ideados para esta tarea que se extraen de los envíos individuales:

- **Días hasta la fecha límite**: Número de días que quedan desde la presentación actual hasta la fecha límite de la tarea.
- **Días hasta la fecha límite de la primera entrega**: Este valor se establece para cada es-

tudiante y actividad, y permanece constante para el resto de los envíos.

- **Envíos hasta la fecha**: Número de envíos realizados anteriormente incluyendo el actual.
- **Días de entrega hasta la fecha**: Número de días distintos en los que un alumno ha intentado resolver la actividad.
- **Actividad**: A1 (programación dinámica) o A2 (ramificación y poda).
- **Éxito:** Representa si el resultado del envío actual pertenece a un historial de intentos exitosos o fallidos.

Hay que tener en cuenta que, debido a los diferentes rangos que pueden abarcar estas variables, se aplica un proceso de normalización estándar para evitar posibles sesgos en los esquemas basados en el aprendizaje.

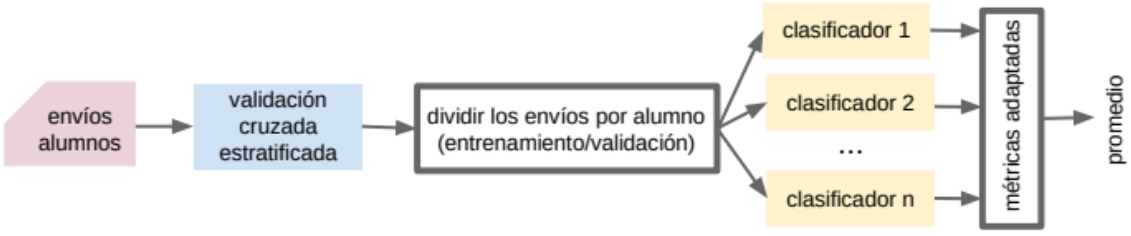
### 3.1. EXPERIMENTACIÓN

Para evaluar la bondad de nuestra propuesta, consideramos como métrica el Área Bajo la Curva de Características Operativas del Receptor (AUC), ya que se basa en la confianza de los modelos en sus predicciones, se utiliza cuando tenemos clases ligeramente desequilibradas y penaliza las diferencias más grandes entre el valor real y el predicho (Duda et al., 2001).

Un procedimiento comúnmente considerado para evitar resultados sesgados en la evaluación de modelos de ML es la validación cruzada estratificada (Mitchell, 1997). Esta estrategia divide los datos disponibles en un número fijo de particiones –normalmente 10– manteniendo la proporción de muestras de cada clase en cada partición. Se utiliza una de ellas para verificar los resultados y el resto para entrenar el modelo. Este proceso se repite tantas veces como el número de particiones establecido.

En el contexto particular de este trabajo, hay que tener en cuenta que, al realizar el proceso de partición, todos los envíos de un alumno tienen que formar parte de la misma partición y cuando se calculen las predicciones de los envíos hay que agruparlos por alumno para conocer finalmente su estimación –en nuestro caso hemos elegido el máximo de las predicciones del historial del alumno– . La figura 2 muestra gráficamente estas ideas.

Figura 2. Esquema de validación cruzada estratificada para problemas de ML. En este caso, aunque ya no existe el concepto de bolsa, todas las instancias que originalmente pertenecían a la misma bolsa deben estar en la misma partición para evitar cualquier posible sesgo en los resultados.



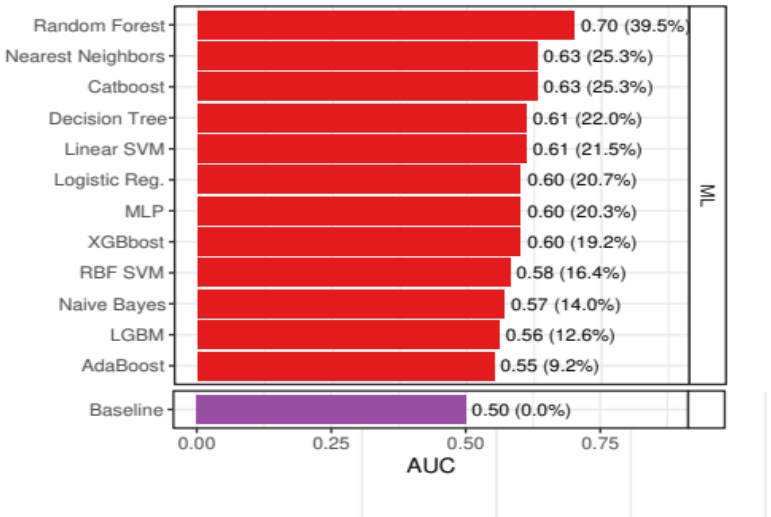
Los algoritmos ML utilizados en esta experimentación están diversificados y pertenecen a diferentes estilos de aprendizaje con la intención de abordar este problema desde distintos puntos de vista. A continuación los vamos a enumerarlos con las referencias correspondientes para conocer más detalles sobre ellos. Como método bayesiano, Naive Bayes (NB) (Webb, 2010); como regresión, Logistic regression (LogReg) (Menard, 2002); método de vecindad k-Nearest Neighbours (kNN) (Cover & Hart, 1967); árboles de decisión Decision Tree (DT) (Breiman, 2017) y Random Forest (RF) (Breiman, 2001), métodos avanzados de Boosting, Adaptive Boosting (AdaBoost) (Freund & Schapire, 1997), eXtreme Gradient Boosting (XGBoost) (Chen & Guestrin, 2016), Categorical Boosting (CatBoost) (Dorogush et al., 2018), and Light Gradient Boosting Machine (LGBM) (Ke et al., 2017); máquinas de vectores soporte, Support Vector Machine (SVM) (Cortes & Vapnik, 1995); y redes neuronales, Multilayer Perceptron (MLP) (Hinton, 1990). Como algoritmo base (Baseline) estableceremos aquel que siempre realiza la misma predicción atendiendo a la clase mayoritaria (en nuestro que todos los alumnos superan la actividad).

Las herramientas de software, hemos realizado este análisis utilizando una variedad de herramientas de código abierto: Se ha utilizado Python como lenguaje de programación base, así como las librerías scikit-learn (v0.24.0), xgboost (v0.90), y catboost (0.24.0) para algoritmos ML, y SHAP (v0.37) para analizar aspectos de XAI-ML.

### 3.2. RESULTADOS

La Figura 3 muestra los resultados obtenidos para la predicción del rendimiento de los estudiantes –ya sea éxito o fracaso– en términos de la métrica AUC. En todos los casos, estos valores representan la media de las 10 particiones del esquema de validación cruzada.

Figura 3. Resultados medios de los diferentes algoritmos de clasificación en términos de la métrica AUC. La mejora relativa de con respecto al caso de referencia base se indica entre paréntesis. A mayor valor AUC mejor es el resultado.



Podemos observar como todos los algoritmos ML seleccionados mejoran el caso base (Baseline); el vecino más cercano (Nearest Neighbors) y Catboost obtienen muy buenos resultados (AUC 0.63) pero Random Forest es que mejora al resto con un AUC de 0.70, por lo que lo seleccionamos para aplicar las técnicas de explicabilidad.

Figura 4. Análisis SHAP de las variables de entrada con respecto a la predicción del método Random Forest. Las características están ordenadas de mayor a menor.

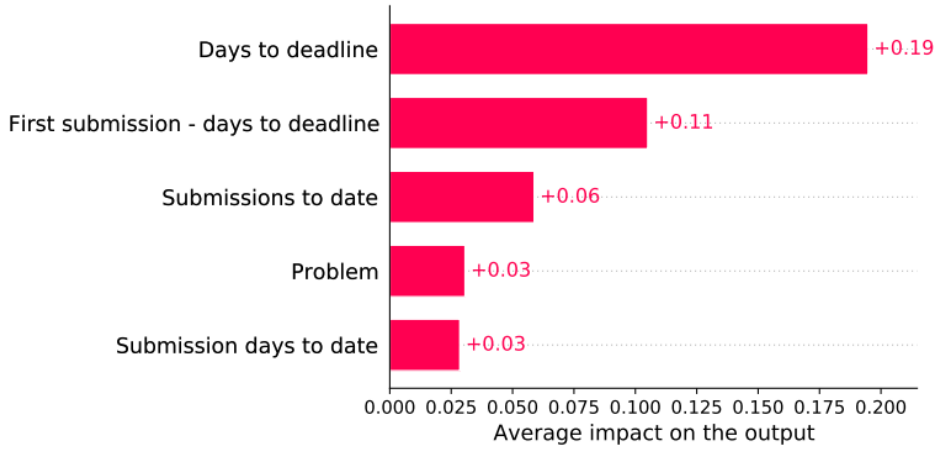
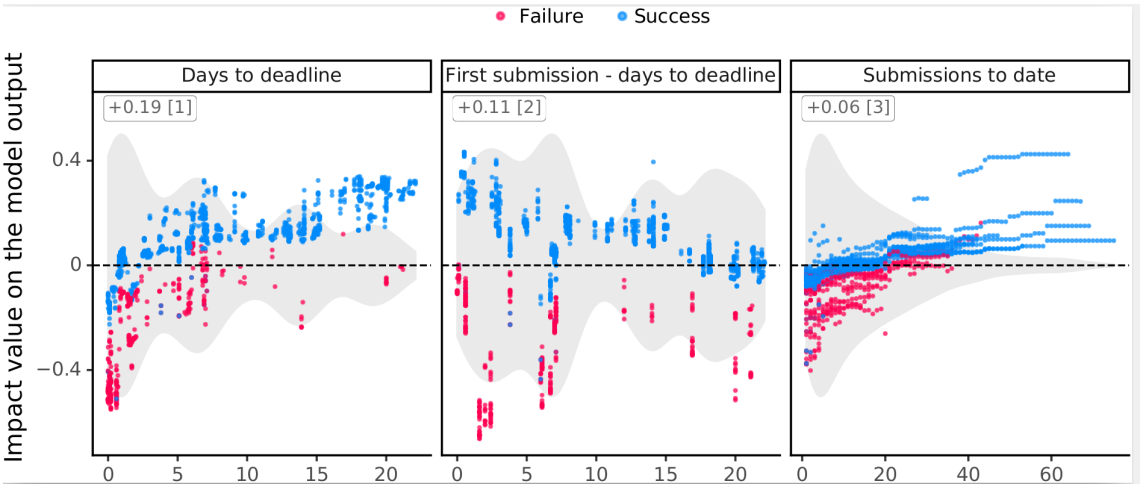



Figura 5. Impacto individual de las tres principales variables de entrada sobre la predicción según el análisis SHAP. La densidad de muestras en cada área se muestra como un gráfico de violín en el fondo. El impacto medio junto con con la posición de la clasificación se indican en la parte superior izquierda de cada caso particular. Lo ocurrido en la realidad se identifica con colores de los puntos (envíos) rojo para casos de alumnos que no han superado la actividad y azul para los que sí.




Realizamos ahora el análisis XAI de nuestra propuesta para constatar la influencia en el resultado del método de predicción de cada una de las variables planteadas. Para ello, consideramos el método SHAP que estima el impacto de cada variable independiente en el éxito del problema en términos de los valores de Shapley. Las figuras 4 y 5 muestran los resultados de este estudio según el impacto general y detallado, respectivamente.

## 4. DISCUSIÓN Y CONCLUSIONES

Atendiendo a los resultados de la sección anterior vamos a responder a las preguntas de investigación.

**RQ1:** ¿Cuándo deben los estudiantes empezar a presentar sus solicitudes al sistema de JL? Se observa que cuando los envíos se realizan hasta 7 días antes del plazo los resultados son mayoritariamente positivos.

**RQ2:** ¿Cuántos envíos son razonables para el éxito de la tarea? Para asegurar el éxito el número de envíos tiene que ser alto, concretamente mayor de 40.

**RQ3:** ¿Hay algún consejo para que los estudiantes aborden adecuadamente las tareas? Que el alumno empiece a desarrollar la actividad lo antes posible y que no desista aunque sea elevado en número de envíos incorrectos.

El uso de sistemas de juez en línea se han considerado como complemento positivo en los cursos relacionados con la programación. Este trabajo aborda la limitación sobre la información de riesgos tanto al estudiante como al instructor. Proponemos el uso de la Inteligencia Artificial eXplicable para obtener dicha retroalimentación. Hemos probado la metodología en Desafíos de Programación con unos 90 alumnos junto a sus 2.600 envíos.

Como trabajos futuros consideramos aumentar el número de alumnos a considerar e incorporar características sobre el factor humano como la personalidad o la motivación.

## 5. REFERENCIAS


Alturki, S., Alturki, N., & Stuckenschmidt, H. (2021). Using educational data mining to predict students' academic performance for applying early interventions.*Journal of Information Technology Education: Innovations in Practice*,*20*, 121–137.

Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R. et al. (2020). Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*,*58*, 82–115.

Asif, R., Merceron, A., Ali, S. A., & Haider, N. G. (2017). Analyzing undergraduate students' performance using educational data mining.*Computers & Education*,*113*, 177–194.

Breiman, L. (2001). Random forests.*Machine learning*, *45*, 5–32.

Breiman, L. (2017).*Classification and regression trees*. Routledge.

Burkart, N., & Huber, M. F. (2021). A survey on the explainability of supervised machine learning. *Journal of Artificial Intelligence Research*,*70*, 245–317.

Carrasco, R. C., Rico-Juan, J. R., & Varó, M. Á. (2010). Aprendizaje de algoritmia mediante desafíos de programación. In *XVI Jornadas de Enseŭanza Universitaria de la Informática*(pp. 519–522). Universidade de Santiago de Compostela. Escola Técnica Superior d'Enxeñaría.

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *CoRR*,

*abs/1603.02754*.

Cortes, C., & Vapnik, V. (1995). Support-vector networks.*Machine Learnning*,*20*, 273–297.

Cover, T. M., & Hart, P. E. (1967). Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*,*13*, 21–27.

Dorogush, A. V., Ershov, V., & Gulin, A. (2018). Catboost: gradient boosting with categorical features support. *arXiv preprint arXiv:1810.11363*, .

Duda, R. O., Hart, P. E., & Stork, D. G. (2001).*Pattern Classification*. (2nd ed.). Wiley.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting.*Journal of computer and system sciences*,*55*, 119–139.

Fryer, D., Strümke, I., & Nguyen, H. (2021). Shapley values for feature selection: The good, the bad, and the axioms.*IEEE Access*,*9*, 144352–144360.

Gray, C. C., & Perkins, D. (2019). Utilizing early engagement and machine learning to predict student outcomes.*Computers & Education*,*131*, 22–32.

Hinton, G. E. (1990). Connectionist learning procedures. In*Machine Learning, Volume III* (pp. 555–610). Elsevier.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree.*Advances in neural infor- mation processing systems*,*30*, 3146–3154.

Kenny, E. M., Ford, C., Quinn, M., & Keane, M. T. (2021). Explaining black-box classifiers using post-hoc explanations-by-example: The effect of explanations and error-rates in xai user studies.*Artificial Intelligence*,*294*, 103459.

Komárek, T., Brabec, J., & Somol, P. (2021). Explainable multiple instance learning with instance selection randomized trees. In*Joint European Conference on Machine Learning and Knowledge Discovery in Databases*(pp. 715–730). Springer.

Kurnia, A., Lim, A., & Cheang, B. (2001). Online judge.*Computers & Education*,*36*, 299–315.

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In*Advances in Neural Information Processing Systems*(pp. 4765–4774).

Mani, A., Venkataramani, D., Petit Silvestre, J., & Roura Ferret, S. (2014). Better feedback for educational online judges. In*Proceedings of the 6th International Conference on Computer Supported Education, Volume 2: Barcelona, Spain, 1-3 April, 2014*(pp. 176– 183). SciTePress.

Menard, S. (2002). *Applied logistic regression analysis,* volume 106. Sage.

Mitchell, T. M. (1997). *Machine Learning*. New York: McGraw-Hill.

Pal, A. K., & Pal, S. (2013). Analysis and mining of educational data for predicting the per- formance of students.*International Journal of Electronics Communication and Computer Engineering*,*4*, 1560–1565.

Peña-Ayala, A. (2014). Educational data mining: A survey and a data mining-based analysis of recent works.*Expert Systems with Applications*,*41*, 1432–1462.

Roth, A. E. (1988).*The Shapley value: essays in honor of Lloyd S. Shapley*. Cambridge University Press.

Wasik, S., Antczak, M., Badura, J., Laskowski, A., & Sternal, T. (2018). A survey on online judge systems and their applications.*ACM Computing Surveys (CSUR)*,*51*, 1–34.

Yera, R., & Martínez, L. (2017). A recommendation approach for programming online judges supported by data preprocessing techniques.*Applied Intelligence*,*47*, 277–290.

**AGRADECIMIENTOS**

Este trabajo ha sido parcialmente financiado por el ‘’Programa Redes-I3CE de investigación en docencia universitaria del Instituto de Ciencias de la Educación (REDES-I3CE-2020-5069)” de la Universidad de Alicante.