Actas de las Jenui, vol. 6. 2021.
Páginas: 43-50

Modelos de inteligencia artificial para asesorar el proceso evaluador de trabajos informáticos complejos

Jose Divasón, F. Javier Martínez de Pisón, Ana Romero, Eduardo Sáenz de Cabezón
[1]Departamento de Matemáticas y Computación; [2]Escuela Técnica Superior de Ingeniería Industrial
Universidad de La Rioja
{jose.divason,fjmartin,ana.romero,eduardo.saenz-de-cabezon}@unirioja.es

Resumen

Planteamos el uso de modelos de inteligencia artificial (IA) para asesorar el proceso evaluador de trabajos informáticos complejos, con componentes técnica y creativa. El objetivo es generar un procedimiento para analizar las variables principales que describen la evaluación realizada, descubrir posibles sesgos y discrepancias y generar rúbricas adecuadas que los eviten. La metodología propuesta se ha aplicado a una asignatura de introducción a la informática (grados de informática y matemáticas, primer curso) en que una tarea consiste en la elaboración de una página web por parte de los estudiantes. Dicho trabajo debe cumplir unos requisitos técnicos (compatibilidad con estándares, número de documentos HTML y CSS, etc.) y tiene una componente creativa (maquetación, aspecto, etc.). Se han desarrollado modelos de IA optimizados mediante algoritmos evolutivos para identificar las variables que intervinieron en la calificación de dichos trabajos durante cinco cursos. Los resultados obtenidos permiten extraer conclusiones sobre la práctica evaluadora, posibles mejoras para la objetividad de la evaluación y la posibilidad de generar rúbricas adecuadas para la evaluación de este tipo de trabajos. La metodología es aplicable a otras materias al ser esta tipología técnica-creativa frecuente en trabajos universitarios.

Abstract

We propose the use of artificial intelligence (AI) models to help evaluate complex projects in computer science courses that involve technical and creative components. The goal is to provide a methodology to analyze the main variables that describe the evaluation, to discover possible biases and discrepancies, and to generate appropriate rubrics that avoid them. The proposed methodology has been applied to a first-year introductory course on computer science, which is taught in both computer science and mathematics degrees. In that course, the students must develop a web page according to some technical requirements (compatibility with standards, number of HTML and CSS files, etc.), but it also possesses a creative component (layout, appearance, etc.). Optimized AI models have been developed using evolutionary algorithms to identify the most important variables that took part in the evaluation of these projects during five courses. The results obtained allow us to draw conclusions about the evaluation process, possible improvements in the impartiality of the evaluation, and the possibility of generating suitable rubrics for grading assignments. The methodology is applicable to projects in other subjects since this technical-creative typology is frequent in university tasks.

Palabras clave

Evaluación, inteligencia artificial, algoritmos evolutivos, sesgos de evaluación.

1. Introducción

En muchas asignaturas informáticas y técnicas de los diferentes estudios de grado y máster una parte de la evaluación se basa en la elaboración por parte de los estudiantes de proyectos y trabajos. El auge de esta metodología de enseñanza y evaluación, en particular del aprendizaje basado en proyectos (PBL), es debido sobre todo a su eficiencia en el desarrollo en los estudiantes de habilidades profesionales y competencias transferibles [6]. Sin embargo, el empleo de este tipo de trabajos y proyectos plantea diversas dificultades que tienen ciertas particularidades en los cursos fundamentales de informática [12]. Entre las dificultades y retos encontrados en relación a los métodos basados en trabajos y proyectos están las relacionadas con su evaluación [5, 13], que resulta en ocasiones difícil por tener los trabajos un doble carácter técnico y creativo y por resultar compleja la evaluación de competencias transferibles y habilidades adquiridas en el desarrollo del

proyecto [14]. Otro factor complicado de acometer es la posible presencia de sesgos, discrepancias e inconsistencias en la evaluación, que resultan muy difíciles de detectar para los propios docentes implicados [24]. La detección de discrepancias, inconsistencias y sesgos en la evaluación se lleva a cabo en la literatura de dos formas: exógena (basada en medidas externas del desempeño de los estudiantes) y endógena (basada en las propias calificaciones) habiéndose mostrado más adecuada la segunda [1]. Por otro lado, se han propuesto diferentes metodologías para detectar discrepancias entre profesores en la evaluación de proyectos en enseñanzas de ingeniería e informática [17, 20].

Nuestra contribución consiste en una metodología basada en aprendizaje automático para identificar discrepancias, inconsistencias y sesgos en la calificación de un proyecto informático. Estudiamos su aplicación en la evaluación de un trabajo en el primer semestre del primer curso de los grados en ingeniería informática y matemáticas en la Universidad de La Rioja. El proyecto consiste en la realización de un sitio web formado por varias páginas, se realiza por grupos pequeños (de dos o tres miembros) y se califica por el equipo de profesores de la asignatura de forma que cada trabajo es calificado por solo uno de los profesores implicados. Los estudiantes reciben una lista de requisitos que debe cumplir el sitio web relacionados con la calidad del código HTML y CSS generado, adecuación con los estándares y cantidad de páginas e información requerida. Aquellos trabajos que cumplen los requisitos son evaluados atendiendo a la calidad de la información en el sitio web, legibilidad y calidad del código, diseño y navegabilidad. Utilizamos algoritmos genéticos para identificar las variables que influyen en la calificación final de los trabajos. En particular podemos detectar discrepancias entre profesores observando la variable `profesor`, de forma que identificamos trabajos que son equivalentes respecto a los criterios de corrección pero difieren en la nota final dependiendo del profesor que los califica. Observando variables como el género, la titulación que cursa cada estudiante u otras podemos detectar sesgos en la corrección. La virtud de los algoritmos genéticos es que nos permiten identificar aquellas variables que *sobreviven* gracias a su capacidad interpretativa en los modelos evaluados, lo que indica una prevalencia significativa.

Tras una primera sección de introducción, en la segunda sección de este trabajo exponemos el contexto académico y describimos el proyecto y los criterios de evaluación. En la tercera sección mostramos la metodología seguida. En la cuarta sección presentamos los algoritmos genéticos y los resultados obtenidos en el ejemplo de aplicación. Finalmente, exponemos las conclusiones extraídas y el trabajo futuro sugerido por ellas así como otros campos de aplicación.

2. Contexto académico y descripción de la tarea

Sistemas Informáticos es una asignatura común al grado en ingeniería informática y al grado en matemáticas en la Universidad de La Rioja. Se imparte en el primer semestre del primer curso como asignatura común a ambos grados y consta de seis créditos ECTS. Las horas lectivas se dividen en una hora semanal de teoría y dos sesiones semanales de laboratorio informático (de 90 minutos cada una). El trabajo autónomo de los estudiantes se estima en 90 horas a lo largo del semestre. Los estudiantes de ambos grados están mezclados tanto en las sesiones de teoría como en las sesiones prácticas. Cada año cursan la asignatura aproximadamente 75 estudiantes (50 del grado en ingeniería informática y 25 del grado en matemáticas).

La primera parte de la asignatura es una introducción básica a la arquitectura de ordenadores, protocolos de internet, codificación de caracteres, sistemas de ficheros y administración básica de sistemas operativos mediante línea de comandos e interfaz gráfica. La segunda parte se centra en los lenguajes HTML5 y CSS3 y consiste en una introducción al diseño de páginas web. El curso no sigue un libro de texto en particular, pero los contenidos están cubiertos en [2, 21].

La evaluación de la asignatura se divide en tres partes. Un 60 % de la nota final depende de una prueba escrita, un 20 % de los informes de prácticas del laboratorio informático y un 20 % de un proyecto realizado en grupos consistente en la realización de un sitio web relacionado con los contenidos de la asignatura.

Para realizar el proyecto los estudiantes son divididos en grupos de dos o tres personas sin tener en cuenta el grado que están estudiando. Se asigna un tema a cada grupo y los estudiantes deben colaborar para crear un sitio web que tenga exactamente el mismo contenido para todos los miembros del grupo (los ficheros HTML han de ser iguales) pero cada estudiante debe aplicar a su sitio web un diseño diferente (diferentes ficheros CSS). Así, cada grupo presenta un proyecto por cada estudiante, que recibirá una calificación individual. Los estudiantes disponen de treinta días para finalizar sus proyectos. El proyecto entregado debe satisfacer una serie de requisitos, de los que los más importantes son los siguientes:

1. Se deberá respetar la diferenciación entre contenido y aspecto, es decir, los ficheros HTML únicamente deberán centrarse en el contenido y su estructura y no deberán incluir ningún elemento sobre el aspecto o presentación de la información.
2. Debe haber al menos ocho ficheros HTML.
3. Debe haber al menos dos ficheros CSS.
4. Todos los ficheros HTML deberán verse afectados por alguna hoja de estilo y, al menos dos, deberán

verse afectados por al menos dos hojas de estilo.
5. Todos los ficheros HTML deberán pasar la validación de HTML5 según las especificaciones del World Wide Web Consortium (W3C) para HTML versión 5.
6. Los ficheros CSS deben ser validados por el validador CSS versión 3 del W3C.
7. Todas las páginas HTML deben incluir el atributo `charset`.
8. Las páginas web deben ser subidas al servidor que la universidad proporciona para los estudiantes de ingeniería informática y matemáticas.
9. Toda la web debe funcionar correctamente si es migrada a otro servidor. Para ello todos los enlaces internos deberán ser enlaces relativos (incluidas las imágenes).
10. Todas las páginas HTML deben incluir los iconos de validación HTML y CSS, que deben ser enlazados por rutas relativas y sin usar atributos de estilo en las etiquetas HTML correspondientes.
11. El `charset` declarado en cada fichero HTML debe coincidir con el usado al codificar y almacenar el fichero.
12. El sitio web debe tener un sistema de navegación que permita moverse de forma razonable por las diferentes páginas del sitio. Cada fichero HTML debe tener un menú que permita volver al primer nivel de páginas de la web.
13. No pueden usarse elementos `table` o `frame` para maquetar las páginas del sitio.
14. El trabajo debe ser implementado sin usar herramientas automáticas de desarrollo web.
15. Las páginas deben verse correctamente en los navegadores más habituales (Firefox, Chrome, Microsoft Edge).

Estos requisitos están diseñados para que los estudiantes no solo desarrollen código HTML y CSS correcto, sino también de acuerdo a los estándares [8, 10] y siguiendo los principios de diseño de HTML5: compatibilidad, utilidad, interoperabilidad y acceso universal [22]. Se añaden también algunas tareas opcionales que pueden suponer un aumento de la nota. Por ejemplo, se valora positivamente que se faciliten versiones del sitio web en distintos idiomas, que se haga un diseño adaptativo (*responsive design*) o que se cumplan las directrices de accesibilidad WCAG en alguno de sus niveles [3].

A lo largo de la asignatura se pone especial énfasis es la importancia de la validación de los documentos HTML y CSS y de la adhesión a los estándares. Por un lado, esto facilita desarrollos futuros y actualizaciones del trabajo, evita posibles problemas y contribuye a la adquisición de buenos hábitos, lo que resulta crucial en los primeros cursos de informática [7, 11]; siendo estas algunas de las habilidades transferibles y competencias de desarrollo profesional propias de la asignatura que se pretenden mejorar con el proyecto. Por otro lado, la validación es un método eficaz para comprobar errores en HTML y CSS, facilitando el aprendizaje de estos lenguajes [18, 19]. Pese a estas ventajas muchos estudiantes no validan su código en los cursos de introducción al desarrollo web [19], por lo que se decidió imponerlo como un requisito.

3. Metodología

La evaluación del trabajo tiene aspectos técnicos que son fácilmente cuantificables, como comprobar si se cumplen los requisitos mínimos exigidos, el número de etiquetas HTML y propiedades CSS diferentes que se utilizan, el número de idiomas o si la página web es o no *responsive*. Por el contrario, la parte creativa no es tan fácil de evaluar por ser más subjetiva, como por ejemplo el aspecto general, las funcionalidades, el contenido, etc. Debido a que la asignatura de Sistemas Informáticos pertenece al primer curso de dos titulaciones y que además tiene muchos grupos de prácticas, a lo largo de los años ha habido un número variable de profesores que han impartido docencia y han tenido que calificar trabajos. De hecho, es habitual que cada curso haya nuevos docentes que no han impartido antes la asignatura. Esto hace que sospechemos de que no haya una uniformidad en la manera de corregir, sobre todo al evaluar la parte creativa. Por tanto, el objetivo de este estudio es intentar detectar posibles discrepancias e inconsistencias entre la corrección de los distintos profesores. Un segundo objetivo es detectar qué variables tienen más influencia en la calificación, y lo haremos mediante algoritmos genéticos: las variables que más persisten en la carrera evolutiva de los modelos son aquellas que más contribuyen a explicar la nota. Esto nos permitiría generar una rúbrica adecuada que se ajuste mejor a los objetivos de la evaluación de los trabajos. Observando variables como el género, la titulación que cursa cada estudiante u otras también podemos detectar sesgos en la corrección.

Lo primero que hicimos fue un análisis para intentar determinar qué variables podían influir en la calificación final del trabajo. Consideramos un total de 33, que fueron separadas en varios grupos:

- 23 variables acerca de la parte técnica y los requisitos mínimos, como el número de ficheros HTML, de ficheros CSS, su tamaño, el número de etiquetas distintas HTML y CSS utilizadas, etc.
- 6 variables categóricas acerca de la parte creativa: aspecto general, funcionalidad, contenido, posicionamiento, contrastes y legibilidad del código.
- 4 variables acerca del contexto del estudiante: titulación, género del estudiante, profesor que co-

rrigió y número de miembros de su grupo.

Esta separación en grupos nos permite descubrir sesgos y discrepancias, siendo una metodología más sencilla que otros métodos que se han propuesto [17].

Una vez determinadas las variables, procedimos a extraer datos de las entregas de trabajos realizadas en los cursos 15-16, 16-17, 17-18, 18-19 y 19-20. En total, hicimos el estudio con 325 estudiantes. Para extraer la información de las 23 variables de la parte técnica, obtuvimos los datos automáticamente mediante una herramienta de desarrollo propio implementada para analizar todos los trabajos [9]. De manera similar, las variables del contexto pudieron obtenerse también automáticamente a partir de nuestros listados de estudiantes y notas. Sin embargo, la parte creativa no permite una automatización y sus variables son más difíciles de medir. A la hora de recabar dichos datos, intentamos evitar introducir discrepancias de diferentes profesores, de forma que un único profesor de la asignatura analizó manualmente todos los trabajos. De esta manera, en esta fase de recogida de datos para alimentar los modelos, todos los trabajos se analizaron en igualdad de condiciones. Para poder abordar de forma homogénea la utilización de diversos métodos estadísticos y de aprendizaje automático, se realizaron diversas transformaciones básicas como la normalización de las variables numéricas con *Z-Score* y la binarización de las variables categóricas. Previamente, fue necesario eliminar casos anómalos y reemplazar valores ausentes. La base de datos quedó finalmente formada por 322 instancias (se eliminaron 3 entregas que estaban incompletas) y 38 características de entrada más la variable de salida a estimar, `Calificación`.

La siguiente fase del análisis se centró en la creación de modelos de regresión que pudieran condensar el conocimiento intrínseco existente de las variables cuantitativas, que caracterizaban los aspectos técnicos del trabajo, así como de aquellas variables cualitativas que pudieran ayudar a detectar ausencia o presencia de discrepancias debidas a criterios de evaluación de cada profesor o, incluso, posibles sesgos que pudieran existir relacionados con las características propias de cada estudiante o grupo de estudiantes evaluados (género, titulación que cursan, etc.). El problema se abordó mediante dos métodos: uno clásico basado en modelos lineales Ridge con selección de variables hacia atrás y otro basado en optimización evolutiva que, mediante algoritmos genéticos (AG), realiza conjuntamente el ajuste de los modelos de aprendizaje automático (ML por sus siglas en ingles) y la selección de las variables más importantes. Todos los experimentos se desarrollaron en los lenguajes `R` y `Python`.

4. Modelos, análisis y resultados

4.1. Selección inicial de modelos

Inicialmente se seleccionaron los algoritmos de ML que pudieran ser más adecuados para un problema de regresión de este tipo. En este caso, las técnicas usadas fueron: regresión Ridge (RIDGE), máquinas vectores soporte para regresión (SVR), redes neuronales artificiales (ANN), métodos basados en vecinos más próximos (KNN), árboles de decisión (DT) y *Extreme Gradient Boosting Machines* (XGB). Los modelos fueron entrenados y ajustados mediante búsqueda en rejilla usando la librería `scikit-learn` de `Python`. Para homogeneizar el proceso de ajuste y validación de los diversos modelos de regresión, se seleccionó como métrica la raíz cuadrada del error cuadrático medio (RMSE por sus siglas en inglés). El motivo de elegir esta métrica es intentar evitar casos extremos donde la predicción falle en demasía, es decir, usando RMSE se penalizan, en mayor medida, las estimaciones más alejadas del valor real que si usásemos otras métricas como el error medio absoluto (MAE). La relación entre RSME y MAE también es importante, pues si se observa que la proporción del RMSE es mucho mayor que el MAE, nos estará indicando que el modelo de regresión tiene un elevado número de errores mucho mayores que la media de los mismos. Debido al reducido tamaño de la base de datos, se diseñó una validación cruzada de 16 pliegues repetida 25 veces que aseguraba la robustez en las estimaciones. En el Cuadro 1 se muestran los parámetros de los mejores modelos obtenidos así como el error medio absoluto (MAE), el RMSE, la desviación estándar de los errores (SD), el coeficiente de determinación (R2) y el intervalo de confianza del error de las estimaciones al 95 %. De entre estas técnicas, se seleccionaron los modelos RIDGE, por su simplicidad e interpretabilidad, y las SVR. Aunque la ANN demostró ser el método que obtuvo los menores MAE y RMSE, se eligió SVR por su facilidad a la hora de entrenar y su capacidad para alcanzar un mínimo global en el proceso de ajuste.

4.2. Análisis con *GA-PARSIMONY*

La búsqueda de modelos parsimoniosos (modelos de baja complejidad), es uno de los retos actuales en el campo del aprendizaje automático. Siguiendo el principio de parsimonia, también denominado “la regla de la navaja de Ockham”; en igualdad de condiciones, la explicación más sencilla suele ser la más probable. Aplicado al mundo del aprendizaje automático, entre modelos con similar precisión se recomienda elegir aquellos que tengan menor complejidad pues suelen ser mejores generalizadores del problema; además de que son más fáciles de comprender y más robustos

Modelo	MAE	RMSE	SD	R2	CI95 %
RIDGE (alpha=1.0e+02)	0,106	0,138	0,138	0,542	[-0,244, 0,328]
KNN (n_neighbors=2.0)	0,103	0,149	0,149	0,464	[-0,370, 0,280]
ANN (num_neuronas=6, alpha=0.000010)	0,082	0,115	0,115	0,683	[-0,248, 0,204]
DTreeRegressor (mae,random,max_depth=22)	0.094	0,132	0,132	0,582	[-0,306, 0.270]
SVR (C=1.0, epsilon=0.003, gamma=0.025119)	0,088	0,123	0,122	0,638	[-0,292, 0,252]
XGB (mdep=7,nroun=150,subsam=0.95,colsam=0.70)	0,091	0,124	0,124	0,629	[-0,261, 0,266]

Cuadro 1: Resultados obtenidos por los distintos modelos (junto con sus hiperparámetros optimizados mediante algoritmos genéticos) en Python.

frente a perturbaciones en sus entradas. Los mecanismos utilizados dentro de los algoritmos de ML, como la regularización o la selección de características, están enfocados en este sentido.

En este estudio, el entrenamiento y selección de los mejores modelos de ML se realizó con la metodología *GA-PARSIMONY* [25]. Esta metodología realiza una búsqueda de modelos de baja complejidad mediante algoritmos genéticos. El objetivo final es obtener modelos de alta precisión y baja complejidad a través del uso de selección de características, ajuste de los parámetros de entrenamiento del algoritmo y selección basada en parsimonia. Esto último es lo que diferencia a *GA-PARSIMONY* de otros métodos parecidos, pues la selección de los mejores individuos o soluciones de cada generación se realiza siguiendo un principio de búsqueda de parsimonia que consiste de dos pasos consecutivos: una preselección de los modelos más precisos y, de entre aquellos con similar coste, una promoción a puestos superiores de los que tengan menor complejidad. Todos los experimentos fueron implementados con el paquete `GAparsimony` [15] desarrollado en lenguaje `R`. Para realizar la optimización de AG con `GAparsimony` es necesario definir los cromosomas de cada individuo que va a ser entrenado con el algoritmo de ML correspondiente. El cromosoma viene definido por una combinación de los valores que se asignarán a los parámetros de entrenamiento del algoritmo y un vector que indicará cuáles son los atributos de entrada seleccionados para ese individuo. En particular, cada individuo i de cada generación g se define con un cromosoma λ_g^i formado por la concatenación de dos vectores P y Q, donde P corresponde con los parámetros de entrenamiento del algoritmo, y Q representa un vector de probabilidades usado en la selección de las características de modo que la variable j será incluida en el modelo si $q_j \geq 0{,}5$. Como función de ajuste J, se utilizó el RMSE de validación cruzada, $RMSE_{val}$. Por último, la complejidad del modelo quedó definida como el número de atributos seleccionados N_{FS}. Esta medida de complejidad ha mostrado ser muy eficaz en experiencias pasadas [16]. El proceso de optimización con AG se definió con una población de 40 individuos evaluados en 40 iteraciones pero con un criterio de parada si el error $RMSE_{val}$ no mejoraba en 20 generaciones seguidas. El proceso de selección utilizó el $20\,\%$ de los mejores individuos (elitistas). Con respecto a la elección de dichos parámetros, tanto el criterio de parada como el número de iteraciones se obtuvieron tras un proceso de ajuste mediante prueba y error. El resto de parámetros se eligieron de otras experiencias previas con bases de datos de similar tamaño y número de variables [16]. Posteriormente a la selección de los mejores individuos de cada generación, `GAparsimony` realizó los procesos clásicos de cruce de los cromosomas de los mejores individuos para crear la siguiente generación de individuos; así como la mutación de los cromosomas para poder crear más diversidad de soluciones en generaciones posteriores.

4.3. Resultados

Mediante el proceso de optimización con `GAParsimony`, se obtuvieron los mejores modelos RIDGE y SVR, con 15 y 18 variables respectivamente. Concretamente, el mejor modelo RIDGE obtuvo un MAE de 0,11 y RMSE 0,14; mientras que el mejor modelo SVR consiguió un MAE de 0,103 y RSME 0,137. A la vista de los datos mostrados en el Cuadro 2, podemos observar los siguientes resultados:

- Las variables `Genero` y `Titulacion` no son seleccionadas para ninguno de los dos modelos, lo que nos lleva a concluir que no se producen sesgos en las calificaciones de los trabajos en base a estas dos características.
- Las variables correspondientes a tres de los profesores (`Profesor4`, `Profesor5`, `Profesor6`) no han sido seleccionadas, luego entre estos profesores no se producen discrepancias. Sin embargo, dos de los profesores (`Profesor1` y `Profesor3`) sí que aparecen como características seleccionadas para ambos modelos, un hecho que nos ha permitido detectar posibles discrepancias en las correcciones. Dichas discrepancias en la corrección han sido confirmadas por medio de test estadísticos que muestran diferencias significativas ($p < 0,05$).
- Las variables que más influyen en la calificación

RIDGE	SVR
Ul (100)	**AspectoGeneral (99,69)**
EtiquetasHTML (100)	**Profesor1 (99,06)**
Profesor1 (100)	**FicherosImagenes (98,75)**
AspectoGeneral (100)	**Responsive (98,13)**
Contrastes (99,69)	**Profesor3 (97,81)**
Profesor3 (99,06)	**noCumplenAccesibilidad (96,88)**
Posicionamiento (99,06)	**Contrastes (96,88)**
EtiquetaIMG (98,75)	**TotalBytesCSS (95,31)**
LegibilidadCodigo (98,75)	**NumJavaScript (94,69)**
HTMLAnalizados (98,44)	**Contenido (94,69)**
PosiblesProblemasAcc (96,25)	**Idiomas (94,06)**
NumJavaScript (96,25)	**Ul (90,94)**
EtiquetasCSS (92,5)	**Funcionalidad (87,19)**
Contenido (90,63)	**ModificadoresCSS (85,63)**
Tablas (80,94)	ErroresPotencialesAcc (76,56)
MiembrosGrupo (46,56)	**EtiquetasHTML (71,56)**
Form (38,75)	Profesor6 (56,25)
Genero (27,82)	**Profesor2 (55,31)**
Ol (25,63)	ErroresAccesibilidad (47,19)
Profesor2 (23,8)	CSSAnalizados (40,31)
Videos (23,44)	Tablas (40,31)
Profesor6 (22,5)	**LegibilidadCodigo (34,38)**
ErroresPotencialesAcc (20,31)	UsosValidador (28,75)
HTMLcon2CSS (15)	**TotalBytesHTML (22,19)**
TotalBytesCSS (14,38)	Profesor5 (19,06)
Funcionalidad (13,12)	HTMLcon2CSS (17,5)
Titulacion (12,81)	EtiquetasCSS (16,56)
ErroresAccesibilidad (10)	PosiblesProblemasAcc (14,69)
TotalBytesHTML (7,81)	HTMLAnalizados (14,38)
Responsive (7,5)	Videos (14,38)
UsosValidador (7,5)	MiembrosGrupo (13,44)
Idiomas (7,19)	Genero (13,12)
noCumplenAccesibilidad (6,88)	Form (12,81)
Profesor4 (6,88)	Ol (9,69)
Profesor5 (6,88)	Profesor4 (9,69)
CSSAnalizados (6,25)	Titulacion (8,75)
ModificadoresCSS (3,75)	EtiquetaIMG (8,44)
FicherosImagenes (2,81)	Posicionamiento (6,56)

Cuadro 2: Variables seleccionadas (en negrita) para los mejores modelos RIDGE y SVR. Entre paréntesis se muestra el porcentaje de aparición de cada variable dentro de los elitistas de las últimas generaciones.

(son elegidas para ambos modelos), además de los profesores comentados anteriormente, son las siguientes: `AspectoGeneral`, `Contenido`, `LegibilidadCodigo`, `Contrastes`, `NumJavaScript`, `EtiquetasHTML` y `Ul`. Hacemos notar que cuatro de ellas son variables cualitativas acerca de la parte creativa del trabajo y, por lo tanto, subjetivas; lo que puede explicar el hecho de que las calificaciones sufran diferencias significativas entre algunos de los profesores y posibles inconsistencias en cada profesor. El hecho de que la variable `EtiquetasHTML` aparezca entre las seleccionadas parece razonable ya que uno de los aspectos que se tiene en cuenta para la evaluación por parte de los profesores es la utilización del mayor número posible de elementos HTML explicados en clase.

- Las características que menos influyen en la calificación (no son seleccionadas para ninguno de los dos modelos), además del género, la titulación y los profesores anteriormente comentados, son: `CSSAnalizados`, `HTMLcon2CSS`, `Form`,

`Ol`, `Videos`, `ErroresAccesibilidad`, `ErroresPotencialesAcc`, `MiembrosGrupo` y `UsosValidador`. Algunas de estas variables tienen un valor similar en casi todos los trabajos o bien responden a procesos auxiliares en la realización del tabajo cuyo efecto en la nota final no es significativo.

- Algunas variables son elegidas para un modelo y no para el otro, aunque las consideramos relevantes porque en ambos modelos existen variables relacionadas que sí que son elegidas. Por ejemplo, podemos observar que para el modelo RIDGE no se elige la variable correspondiente al número de propiedades CSS (`ModificadoresCSS`); sin embargo, sí que se selecciona la variable `EtiquetasCSS`, por lo que podemos deducir que en la corrección sí que se tiene en cuenta el número de propiedades CSS utilizadas. Además, en SVR hay variables que apenas aparecen en los elitistas de las últimas generaciones (por ejemplo, `TotalBytesHTML`), sin embargo sí que han sido seleccionadas en el mejor modelo. Esto nos indica que su importancia podría ser en realidad menor, a pesar de haber sido elegidas.

5. Conclusiones y trabajo futuro

Los resultados obtenidos aplicando algoritmos genéticos a los resultados de evaluación de un trabajo informático con doble componente técnica y creativa nos permiten sacar varias conclusiones. En primer lugar, estos modelos, al basar su funcionamiento en la persistencia de las variables al ir evolucionando los modelos explicativos del fenómeno, nos permiten hacer una valoración ajustada de la influencia de las variables en el resultado final. En el caso de la evaluación de un trabajo como el que estamos analizando, nos permite valorar el papel que cada una de las variables tiene en la explicación de variaciones en las calificaciones, lo que supone una aproximación al propio proceso de evaluación de los trabajos. La evaluación de los métodos de enseñanza y corrección se basa usualmente en la propia subjetividad del profesor [4], algo que la metodología que presentamos puede contribuir a reducir. Por otro lado, uno de los retos fundamentales de la aplicación de metodologías basadas en proyectos o trabajos es el diseño de sistemas de evaluación adecuados [6, 13]. Una rúbrica correctamente establecida proporciona una estandarización de la evaluación al especificar una guía para la calificación de acuerdo a criterios claramente establecidos de antemano. Esto tiene el efecto de reducir discrepancias, inconsistencias y sesgos. El diseño de rúbricas útiles es una componente importante para una mejora de la evaluación en particular en lo que se refiere a aspectos que mezclan subjetividad y objetividad, como la calidad del código en cursos de programación [23]. La metodología que presentamos proporciona una forma de iniciar el trabajo de desarrollo de rúbricas basada en la experiencia docente previa y con la intención explícita de reducir sesgos y discrepancias. La discriminación de las variables que tienen más influencia en la variabilidad de la nota son señaladas por esta metodología y serán por tanto tenidas en cuenta para establecer de forma precisa los criterios de evaluación en torno a ellas en el diseño de rúbricas. Finalmente, la evaluación de trabajos con componente técnica y componente creativa es muy común en muchos estudios de grado y máster, en particular en estudios técnicos. La detección de discrepancias y sesgos en la evaluación puede ser complicada y costosa. La metodología propuesta automatiza la mayor parte del proceso de selección de las variables que nos permiten descubrir sesgos y discrepancias, de modo que ambos puedan ser evitados en el diseño de la evaluación. Partiendo de un conjunto de variables que definan las características que intervienen en el trabajo, los algoritmos genéticos nos permiten una evaluación adecuada de estas variables y al ser el proceso automático, podemos refinar o ampliar el conjunto de variables en sucesivas interacciones. Como trabajo futuro nos planteamos facilitar una interfaz y guía que permita el uso de la metodología propuesta en otras situaciones similares, además de una herramienta que genere automáticamente una posible rúbrica.

Referencias

[1] Daniel W. Allen. Toward a Common Currency: Using Item Response Theory to Adjust for Grading Inconsistency. *Mathematics, Statistics, and Computer Science Honors Projects*, Macallester College, 2007. Disponible en `https://digitalcommons.macalester.edu/mathcs_honors/7`. Fecha de acceso: 16 de abril de 2021.

[2] Glenn Brookshear y Dennis Brylow. *Computer Science: An Overview*. Pearson Education, 2018.

[3] Alastair Campbell, Andrew Kirkpatrick, Michael Cooper y Joshue O Connor. Web Content Accessibility Guidelines (WCAG) 2.1 *W3C Reccomendation*, 2018. Disponible en `https://www.w3.org/TR/2018/REC-WCAG21-20180605/`. Fecha de acceso: 16 de abril de 2021.

[4] Angela Carbone y Jens J. Kaasbøll. A survey of methods used to evaluate computer science teaching. *SIGCSE Bulletin ACM*, 30(3):41–45, 1998.

[5] Julián Chaparro-Peláez, Santiago Iglesias-Pradas, Félix J. Pascual-Miguel y Ángel

Hernández-García. Factors Affecting Perceived Learning of Engineering Students in Problem-Based Learning Supported by Business Simulation. *Interactive Learning Environments*, 21(3):244–262, 2013.

[6] Juebei Chen, Anette Kolmos y Xiangyun Du. Forms of implementation and challenges of PBL in engineering education: a review of literature. *European Journal of Engineering Education*, 46(1):90–115, 2021.

[7] Al Cuoco, E. Paul Goldenberg y June Mark. Habits of mind: An organizing principle for mathematics curricula. *The Journal of Mathematical Behavior*, 15(4):375–402, 1996.

[8] Alex Danilo, Sangwhan Moon, Steve Faulkner, Arron Eicholz y Travis Leithead. HTML 5.2 *W3C Recommendation*, 2017. Disponible en https://www.w3.org/TR/2017/REC-html52-20171214/. Fecha de acceso: 16 de abril de 2021.

[9] Jose Divasón, Ana Romero y Eduardo Sáenz-de-Cabezón. Experiencia con una herramienta para automatizar la comprobación de requisitos de los trabajos HTML–CSS de Sistemas Informáticos. En *Actas de las XXIV Jornadas de Enseñanza Universitaria de Informática, Jenui 2018*, vol. 3, páginas 173 – 180, Universitat Oberta de Catalunya, 2018.

[10] Elika J. Etemad, Tab Atkins Jr. y Florian Rivoal. CSS Snapshot 2020, Disponible en https://www.w3.org/TR/2020/NOTE-css-2020-20201222/. Fecha de acceso: 16 de abril de 2021.

[11] Saj-Nicole A. Joni y Elliot Soloway. But my program runs! Discourse rules for novice programmers. *Journal of Educational Computing Research*, 2(1):95–125, 1986.

[12] Judy Kay, Michael Barg, Alan Fekete, Tony Greening, Owen Hollands y Jeffrey H. Kingston. Problem-Based Learning for Foundation Computer Science Courses. *Computer Science Education*, 10(2):109–128, 2000.

[13] Tanya Kumberger. Revising a Design Course from a Lecture Approach to a Project-Based Learning Approach. *European Journal of Engineering Education*, 38(3):254–267, 2013.

[14] José Luis Martín Núñez, Edmundo Tovar Caro y José Ramón Hilera González. From Higher Education to Open Education: Challenges in the Transformation of an Online Traditional Course. *IEEE Transactions on Education*, 60(2):134–142, 2016.

[15] Francisco Javier Martínez-De-Pisón. GAparsimony: GA-based optimization R package for searching accurate parsimonious models. R package version 0.9-4, 2020. Disponible en https://https://github.com/jpison/GAparsimony.

[16] F.J. Martinez-de-Pison, J. Ferreiro, E. Fraile y A. Pernia-Espinoza. A comparative study of six model complexity metrics to search for parsimonious models with GAparsimony R Package. *Neurocomputing*, In Press., 2020. DOI: 10.1016/j.neucom.2020.02.135

[17] Juan M. Montero, Rubén San-Segundo, Javier Macías-Guarasa, Ricardo de Córdoba y Javier Ferreiros. Methodology for the Analysis of Instructors' Grading Discrepancies in a Laboratory Course. *International Journal of Engineering Education*, 22(5):1053–1062, 2006.

[18] Thomas H. Park y Susan Wiedenbeck. Learning Web Development: Challenges at an Earlier Stage of Computing Education. In *Proceedings of the Seventh International Workshop on Computing Education Research*, 2011. DOI: 10.1145/2016911.2016937

[19] Thomas H. Park, Brian Dorn y Andrea Forte. An Analysis of HTML and CSS Syntax Errors in a Web Development Course. *ACM Transactions on Computing Education*, 15(1):1–21, 2015.

[20] Stefan Pero y Tomas Horváth. Detection of Inconsistencies in Student Evaluations. In *Proceedings of the 5th International Conference on Computer Supported Education (CSEDU-2013)*, Mayo 2013. DOI:10.5220/0004385602460249

[21] G. Michael Schneider y Judith Gersting. *An Invitation to Computer Science*. Wadsworth Publishing Co. Inc., sexta edición, 2012.

[22] Maciej Stachowiak y Anne van Kesteren. HTML Design Principles *W3C Working Draft*, 2007. Disponible en https://www.w3.org/TR/2007/WD-html-design-principles-20071126/. Fecha de acceso: 16 de abril de 2021.

[23] Martijn Stegeman, Erik Barendsen y Sjaak Smetsers. Designing a Rubric for Feedback on Code Quality in Programming Courses. En *Proceedings of the 16th Koli Calling International Conference on Computing Education Research*, 2016. DOI:10.1145/2999541.2999555

[24] Pamela Steinke y Peggy Fitch. Minimizing Bias When Assessing Student Work. *Research & Practice in Assessment*, 12:87–95, 2017.

[25] R. Urraca, E. Sodupe-Ortega, J. Antonanzas, F. Antonanzas-Torres y F.J. Martinez-de-Pison. Evaluation of a novel GA-based methodology for model structure selection: The GA-PARSIMONY. *Neurocomputing*, 271:9–17, 2018.