

UNIVERSIDAD DE CÓRDOBA

**Instituto de Estudios de Posgrado**

**Máster en Inteligencia Computacional e Internet de las Cosas**

Trabajo Fin de Máster

# Aplicación de inteligencia artificial explicable para la resolución de problemas predictivos en analítica software

**Alumno: Francisco Javier Alcaide Mengual**

**Tutores: Dra. Aurora Ramírez Quesada – Dr. José Raúl Romero Salguero**

---



22/09/2023

## Tabla de contenidos

Resumen .....	2
1. Introducción .....	3
2. Antecedentes .....	4
2.1. Aplicación del aprendizaje automático.....	4
2.2. Técnicas de explicabilidad .....	6
2.3. Problemas predictivos en ingeniería del software.....	8
3. Objetivo.....	9
4. Definición del problema .....	10
4.1. Estimación de esfuerzo en ingeniería de software .....	10
4.2. Predicción de conflictos en fusiones de código .....	11
5. Metodología .....	11
5.1. Metodología común .....	12
5.2. Metodología en estimación de esfuerzo .....	14
5.3. Metodología en Predicción de conflictos .....	21
6. Análisis de resultados .....	27
6.1. Estimación de esfuerzo.....	27
6.2. Clasificación de conflictos.....	58
6.3. Conclusiones del análisis de resultados .....	86
7. Conclusiones .....	89
Anexos.....	93
Anexo 1: Preprocesado estimación de esfuerzo.....	93
Anexo 2: Preprocesado clasificación de conflictos .....	96
Anexo 3: Explicabilidad estimación de esfuerzo.....	96
Anexo 4: Explicabilidad predicción de conflictos .....	101

## Resumen

En el ámbito de la analítica software, la estimación precisa del esfuerzo requerido para el desarrollo y la detección temprana de conflictos en el proceso colaborativo son cruciales para el éxito de los proyectos al facilitar el desarrollo y ahorrar costes para la ingeniería de software. En este contexto, este trabajo aborda estos dos aspectos esenciales aplicando algoritmos de aprendizaje automático para identificar patrones y características que permitan por un lado estimar el esfuerzo necesario durante el ciclo de desarrollo software mediante modelos basados en regresión, y por otro lado modelos de clasificación para predecir los escenarios de fusión de código que son propensos a convertirse en casos conflictivos en el futuro para que puedan ser abordados por los desarrolladores antes de que se vuelvan complicados y costosos. En ambas tareas se exploran diversos algoritmos con el objetivo de mejorar la precisión y fiabilidad de los resultados, además de permitir una comparación de rendimiento entre los distintos tipos de modelos empleados. Un aspecto distintivo de esta investigación es la incorporación de técnicas de explicabilidad, que proporcionan transparencia a los modelos de estimación y clasificación empleados. La interpretación de cómo los modelos de aprendizaje automático llegan a sus predicciones y toman decisiones es esencial para generar confianza y permitir una colaboración más efectiva entre los usuarios y los algoritmos al permitir una toma de decisiones más informada. Esta investigación busca mejorar la precisión en la estimación de esfuerzo y a agilizar el proceso de desarrollo al anticipar y mitigar los conflictos en el ciclo de vida del software.

**Palabras clave:** Ingeniería de software, analítica software, aprendizaje automático, estimación de esfuerzo, predicción de conflictos, técnicas de explicabilidad, modelos predictivos, evaluación de rendimiento.

## 1. Introducción

El campo de la ingeniería de software, encargado del diseño, desarrollo, prueba y mantenimiento de aplicaciones y sistemas informáticos, se ha convertido en un pilar esencial en la actualidad. Impulsado por la continua expansión de la tecnología que utilizamos en distintas actividades de nuestra vida diaria para satisfacer nuestras necesidades, ayudarnos a solucionar problemas, como medio para comunicarnos o incluso en sistemas de control y gestión en la industria. Por lo tanto, la ingeniería software se ha convertido en un motor que impulsa el progreso e innovación tecnológica.

En este contexto expuesto, el desarrollo de software ha evolucionado hacia un enfoque más colaborativo donde en lugar de realizar proyectos aislados formados por un grupo de programadores, los proyectos de desarrollo de software en la actualidad involucran a equipos interdisciplinarios compuestos por profesionales en distintas áreas relacionadas, que además pueden estar dispersos geográficamente mientras trabajan en diferentes partes de un software. Este enfoque colaborativo permite aprovechar la experiencia y los conocimientos especializados de profesionales en diversas áreas. Sin embargo, también introduce desafíos significativos en términos de coordinación y comunicación efectiva, ya que una gestión deficiente en este área puede originar importantes problemas que afecten a la calidad del producto desarrollado o retrasos en los plazos de desarrollo.

De esta forma, el ámbito de la analítica de software, es decir, el proceso de recopilación, medición y análisis de datos relacionados con el uso y el rendimiento del software o las aplicaciones informáticas se enfrenta a algunas cuestiones importantes: ¿Cómo se puede estimar correctamente el esfuerzo que se debe invertir en el desarrollo de un proyecto con precisión? ¿Cómo se puede evitar que los problemas asociados al trabajo conjunto de varios grupos, como los conflictos en la fusión de código, afecten la calidad del software final y retrasen su desarrollo?

La analítica software (SA, *Software Analytics*) se ha beneficiado de los avances en minería de datos y aprendizaje automático (ML, *Machine Learning*) en los últimos años. Los modelos de ML han demostrado ser efectivos en la resolución de problemas de analítica software, como la clasificación, regresión y detección de anomalías [1]. Junto al creciente uso de técnicas de ML en la analítica software para analizar los datos generados por los procesos de planificación, desarrollo y prueba del software, otro hecho importante es el aumento de la complejidad y tamaño de los problemas que se resuelven mediante este tipo de algoritmos en este campo. Esto ha dado lugar a modelos cada vez más complejos y opacos complicando la interpretación de los resultados y el proceso interno que ha llevado a obtener dichos resultados. Por este motivo los modelos de ML a menudo se consideran "cajas negras" (*black-box*) en el sentido de que el procesamiento interno, o la forma en la que el algoritmo está construido, es opaca. La dificultad para comprender este procesamiento provoca que se deba de confiar ciegamente en los resultados que proporciona. Esto hace más difícil para los equipos de desarrollo y gestión de proyectos tomar decisiones precisas, que se basan en información útil y fiable.

Como se ha comentado, la analítica software cumple un papel importante como ayuda para detectar errores de manera temprana, mejorar la calidad del software y reducir los costes y tiempos de desarrollo. Por ello, se aplica en una amplia variedad de áreas, como la estimación de esfuerzo, la predicción de la calidad del software, la detección de errores y la mejora del proceso de desarrollo. Para conseguir estas mejoras hace uso del gran volumen de datos generados por los sistemas de control de versiones, los sistemas de seguimiento de problemas y los sistemas de gestión de proyectos para extraer patrones y conocimientos útiles. La falta de transparencia y comprensión de los modelos de ML utilizados en la analítica software puede generar desconfianza en su uso y problemas por tomar

decisiones erróneas por la falta de compresión del modelo empleado. Así, surge la necesidad de mejorar la transparencia de los modelos de ML utilizados en este ámbito, además de la compresión de los resultados obtenidos. La inteligencia artificial explicable (XAI, *Explainable Artificial Intelligence*) [2,3] es un campo emergente que busca abordar estos problemas mencionados proporcionando métodos y técnicas para mejorar la transparencia y comprensión de los modelos de ML, permitiendo a los usuarios comprender cómo y/o por qué se llega al resultado obtenido. De este modo, se mejora la confianza en los resultados y la capacidad para tomar las decisiones basadas en dichos resultados, consiguiendo decisiones más precisas y fiables al tener un mejor conocimiento de la información que proporcionan los resultados [4,5].

En este trabajo se lleva a cabo una adecuada revisión bibliográfica sobre trabajos previos en el ámbito de estimación de esfuerzo y predicción de conflictos mediante ML en ingeniería de software, al igual que sobre métodos XAI, para obtener los fundamentos teóricos necesarios. Se define y establece una metodología acorde a los objetivos específicos del trabajo, la cual marca los pasos que se siguen en el estudio experimental realizado. Los resultados de este estudio experimental pasan por una fase de evaluación y análisis, para finalmente destilar unas conclusiones. Por lo tanto, para reflejar el alcance del trabajo se ha organizado este documento con una estructura que se compone de las siguientes secciones: La Sección 2 presenta los principales conceptos relevantes junto a los trabajos sirven como base y principal fuente de información; En la Sección 3, se establecen los objetivos de este TFM; En la Sección 4 se definen claramente los dos problemas que se busca abordar; En la Sección 5 se recoge la metodología que se seguirá para abordar los problemas expuestos; La Sección 6 contiene los resultados obtenidos a lo largo del trabajo siguiendo la metodología explicada; La Sección 7 se recogen las conclusiones finales del trabajo. Por último, debido a la magnitud de la experimentación realizada, se ha visto conveniente incluir anexos con la información detallada.

## 2. Antecedentes

Como se ha visto anteriormente, en esta sección se recogen y desarrollan los conceptos importantes para conseguir una mejor comprensión del contexto y las herramientas empleadas en este trabajo.

### 2.1. Aplicación del aprendizaje automático

*Machine learning* es una rama de la inteligencia artificial capaz de crear sistemas capaces de aprender y mejorar automáticamente a partir de la experiencia previa, es decir, su punto fuerte reside su capacidad para nutrirse y analizar grandes volúmenes de datos previos con el objetivo de que aprendan patrones y puedan tomar decisiones usando la información extraída de esos patrones. Debido a su utilidad, versatilidad y adaptabilidad, el ML tiene aplicaciones en una amplia variedad de sectores, en los que actualmente tiene un gran impacto, y de cuya capacidad se beneficia también la ingeniería de software.

En relación con el auge del ML, Y. Yang et al. [5] realizan una revisión sistemática de artículos relacionados con modelos predictivos en ingeniería de software, publicados entre 2009 y 2020, proporcionando una visión completa de la aplicación de modelos predictivos en la ingeniería de software. El estudio buscaba describir los modelos y enfoques clave más utilizados, resumir las áreas de aplicación comunes y analizar los resultados de las investigaciones. Con este estudio se observó un crecimiento en el uso de modelos predictivos, que se deduce del aumento en el número de estudios relacionados con modelos predictivos en ingeniería de software entre 2009 y 2020 con la propuesta de

enfoques novedosos. Además, descubren que las tareas comunes de modelos predictivos en ingeniería de software son predicción de defectos en el software y al mantenimiento de software.

Ante la creciente necesidad de adoptar y desarrollar metodologías avanzadas para solventar los inconvenientes que aparecen al estimar el esfuerzo para el desarrollo software y prever posibles conflictos en la fusión de código durante un desarrollo colaborativo de distintos grupo de trabajo, el uso de técnicas de ML ha emergido como una herramienta poderosa para ayudar a la ingeniería de software en la toma de decisiones basada en datos previos y predicción precisa de eventos futuros. A lo largo del tiempo se ha desarrollado métodos para abordar los problemas mencionados, pero el uso de métodos basados en el ML ha ganado terreno por su capacidad para analizar los datos y características específicas de un proyecto, aprender de datos anteriores y descubrir relaciones y patrones complejos, lo que se traduce en que pueden ofrecer ventajas significativas en eficiencia y precisión frente a los métodos tradicionales de estimación empleados hasta ahora y permitir la construcción de modelos predictivos que alertan a los desarrolladores sobre posibles conflictos antes de que ocurran.

En cuanto a la mejora que proporcionan los modelos de ML, J. Wen et al [1] realizó una revisión sistemática de la literatura sobre modelos de estimación de esfuerzo en desarrollo de software basados en ML desde distintas perspectivas, entre la que se encontraba la comparación entre modelos de ML frente a modelos sin uso de ML o tradicionales. Este análisis sistemático reveló que las técnicas de ML son prometedoras en este campo, y en su mayoría, los modelos de ML superan a los modelos que no son de este tipo en términos de precisión.

La comparación de rendimiento entre técnicas de ML es un tema importante y ha sido objeto de estudio por parte de muchos investigadores. El estudio previamente mencionado de J. Wen et al. [1] también realizan una comparación entre tipos de modelos de ML en el ámbito de la estimación de la esfuerzo y recogen que los tipos de técnicas de ML más comunes son el razonamiento basado en casos (CBR, *Case Based Reasoning*), redes neuronales artificiales (ANN, *Artificial Neural Network*) y árboles de decisión (DT, *Decision Tree*). En general, la precisión de la estimación de la mayoría de los modelos de ML se acerca a niveles aceptables. De la misma forma, Y. Yang et al. [5] exploran la diversidad de modelos de ML que son empleados para tareas de ingeniería de software y observan un aumento en la aplicación de técnicas de aprendizaje profundo, como modelos basados en redes neuronales convolucionales (CNN, *Convolutional Neural Network*) y redes neuronales recurrentes (RNN, *Recurrent Neural Network*).

Por otro lado, Y. Mahmood et al [4] han evaluado el rendimiento de las técnicas de ML en la precisión de la estimación de esfuerzo en el desarrollo de software, tanto en enfoques individuales como de tipo *ensemble*. Los resultados de este estudio respaldan la adopción de técnicas de *ensemble* en la estimación de esfuerzo al concluir que las técnicas de *ensemble* en ML superan a las técnicas individuales en la precisión de la estimación de esfuerzo en el desarrollo de software en la mayoría de los casos. Esto es debido a la ventaja que proporciona la combinación de métodos individuales en el *ensemble*, permitiendo abordar las debilidades de los métodos individuales para obtener estimaciones más precisas. En la misma línea, los resultados de Cabral J. et al [6] destacan que las técnicas de ML adecuadas para construir modelos de estimación de esfuerzo y que las técnicas de *ensemble* superan a los modelos individuales, pero que no existe una clara superioridad de un modelo de *ensemble* sobre otros.

Sin embargo, el ML tiene algunos problemas. Entre los que destaca la dificultad de interpretación de los resultados y la falta de explicación de cómo se llega a ellos en muchos casos. Es por ello por lo que

suele conocerse como la "caja negra" del ML. Estas dificultades dan lugar a la necesidad de la explicabilidad. Mediante las técnicas de explicabilidad se busca "abrir la caja negra", permitiéndonos comprender y explicar por qué un modelo de ML toma una decisión en particular.

## 2.2. Técnicas de explicabilidad

Como se ha comentado en la sección anterior, a pesar de que los algoritmos de ML nos permiten tener una poderosa herramienta, un aspecto crítico cuando se trabaja en el campo del ML es la interpretación y explicación de los modelos. Debido a que muchos sectores toman decisiones importantes en base a los resultados de los algoritmos de ML, comprender el porqué se convierte en una cuestión fundamental. En relación con las debilidades de las técnicas de ML, R. Guidotti et al. [3] exponen los principales problemas que se enfrentan en el uso de ML y proponen métodos de explicabilidad útiles para abordar los desafíos en ML.

Los beneficios de las técnicas de explicabilidad residen en aumentar la confianza en los resultados, al comprender el cómo y el porqué, y también proporcionar información sobre las características y los patrones valiosos para los modelos. Esto significa que las técnicas de explicabilidad no solo aumentan nuestra comprensión de los modelos, sino que también nos ayudan a tomar decisiones más sólidas y respaldadas por la información. Otro beneficio de las técnicas de explicabilidad es que nos proporcionan la capacidad de identificar la causa de los errores al comprender mejor el modelo, ya que nos permiten estudiar que ha llevado a un modelo a cometer un error para posteriormente mejorarlo.

En la ingeniería de software, especialmente en la estimación de esfuerzo y la predicción de conflictos en fusiones de código, la toma de decisiones tiene un impacto importante en la calidad del software y el éxito del proyecto. Por lo tanto, no es suficiente conseguir modelos de ML que obtengan buenos resultados, también es necesario comprender el porqué para aliviar las preocupaciones y dudas generadas en los desarrolladores al no comprender los modelos. Debido a la tendencia creciente en la complejidad de los proyectos de desarrollo de software que ha provocado una expansión en el uso de técnicas de ML en esta área, ha traído también una creciente demanda de transparencia y comprensión en las decisiones derivadas de los modelos de ML.

Como explican en su libro de P. Biecek y T. Burzykowski [2] existen distintas técnicas de explicabilidad. La técnicas de explicabilidad global se centran en obtener una comprensión general de cómo funciona el modelo en conjunto y de las características o factores que más influyen en él. Por otro lado, existen técnicas que permiten entender el razonamiento detrás de una decisión individual, es decir, se centra en decisiones concretas para obtener información en profundidad de qué características influyen principalmente en la decisión del modelo para ese caso, conocidas como técnicas de explicabilidad local. Por ello, se van a explorar ambos enfoques para proporcionar una visión completa, y a continuación se explican brevemente cada una de las técnicas empleadas en este trabajo usando la información procedente del libro de P. Biecek y T. Burzykowski [2]:

- **Importancia por permutación:** Esta técnica proporciona una comprensión global del modelo y evalúa la contribución de cada característica en el proceso de toma de decisiones. Funciona comparando el modelo original con ese modelo sin el efecto de la característica de la que se está evaluando la importancia. Para eliminar el efecto de una característica en el modelo se toma dicha característica y se realiza una permutación de sus valores en el conjunto de datos, es decir, se realizar una reorganización aleatoria de sus valores. Luego se observa cómo afecta esto al rendimiento del

modelo. Si el rendimiento disminuye significativamente, podemos inferir que esa característica es importante para el modelo.

- **Break-down:** El método de *Break-down* es una técnica de explicabilidad local que proporciona una descomposición detallada de la contribución de cada característica en la predicción del modelo, es decir, nos proporciona información de que forma cada característica contribuye al resultado de la predicción, esto incluye un valor de contribución y su signo. Para llegar a dichos valores esta técnica registra los valores de todas las características utilizadas por el modelo para una instancia concreta. El modelo de ML proporciona una predicción promedio que sirve como punto de partida del método sin tener en cuenta los valores una característica concreta. Posteriormente el algoritmo de *Break-down* mide cuánto cambia la predicción cuando incluimos una característica específica. Esto implica ajustar el valor de una característica y observar cómo afecta la predicción permitiendo obtener su contribución. Este proceso se repite sucesivamente con todas las características.
- **Shapley Values (Valores de Shapley):** Es una técnica de explicabilidad local que se basa en la teoría de juegos cooperativos al asignar una contribución justa y equitativa en función del impacto de cada característica en un modelo de ML a la predicción final. Las características del modelo son los "jugadores" en el juego mientras que las combinaciones de características se denominan "coaliciones". Debido a su forma equitativa de asignar las contribuciones destaca en casos donde se busca equidad entre las características, como la toma de decisiones basadas en algoritmos en la atención médica o la evaluación financiera.

Para calcular los valores de Shapley de una característica consideramos todas las posibles coaliciones en las que esa característica podría participar. Para este método es importante tener en cuenta la predicción promedio del modelo. Para cada coalición se hace una predicción del modelo contando con esa coalición y se calcula la diferencia respecto a la predicción promedio. Luego, se realiza una predicción del modelo sin esa coalición y de nuevo, se calcula la diferencia respecto de la predicción promedio. Finalmente, para cada coalición se calcula cuánto cambió la predicción del modelo por esa característica, es decir, la diferencia entre las diferencias previamente calculadas, esto es conocido como valor marginal. Por lo tanto, el valor de Shapley de una característica es la media ponderada del valor marginal que esa característica proporciona al modelo en todas las coaliciones posibles. La ponderación refleja cuántas veces aparece cada característica en diferentes coaliciones. Un valor de Shapley alto en una característica indica que esa característica tiende a contribuir significativamente a las predicciones del modelo. Un valor bajo sugiere que la característica no tiene un impacto importante en las predicciones.

- **LIME (*Local Interpretable Model-Agnostic Explanations*):** La técnica de explicabilidad local LIME comienza generando muestras perturbadas, es decir, hace pequeñas modificaciones aleatorias en los valores originales de las características para posteriormente usar el modelo de ML para obtener predicciones de las muestras perturbadas. Esto crea un nuevo conjunto de etiquetas de clase que reflejan cómo el modelo reacciona a las pequeñas modificaciones de las características. Con cada paso LIME ajusta un modelo explicativo simple, como una regresión lineal, a las muestras perturbadas y las predicciones del modelo original. Este modelo simple actúa como una aproximación local del modelo original. Al evaluar la contribución de cada característica al modelo explicativo simple puede determinar qué características influyen más en la predicción de entrada y cómo.

Los resultados de las técnicas que se han explicado, en general, se presentan en forma de gráfico de barras con el valor de la contribución relativa de una característica, así como el signo de la contribución. De esta forma los usuarios pueden interpretar los resultados de forma intuitiva y sencilla, y entender qué características concretas están impulsando la predicción en cuestión.

## 2.3. Problemas predictivos en ingeniería del software

La estimación de esfuerzo se refiere a la predicción de la cantidad de tiempo, recursos y personal necesarios para llevar a cabo un proyecto de manera exitosa. Este proceso es crucial ya que tiene impacto directo en la distribución de recursos, los plazos de entrega establecidos y, en consecuencia, la satisfacción del cliente y el ahorro de costes para las empresas. En el campo de la ingeniería de software es fundamental para la planificación de los proyectos de desarrollo, pues si la estimación de esfuerzo no es precisa, los proyectos pueden enfrentar problemas significativos como retrasos en la entrega, presupuestos desbordados y resultados insatisfactorios. Por otro lado, las metodologías tradicionales poco a poco se quedan atrás en términos de precisión por los continuos avances y cambios que tienen lugar en este campo. Estos hechos han sido expuestos por los autores y estudios previamente mencionados [1,4,5,6]. Por lo tanto, una vez se comprende la importancia de la estimación de esfuerzo, veamos que dificultades presenta el proceso de estimación de esfuerzo en ingeniería de software:

- **Complejidad de los proyectos:** Los proyectos pueden ir desde pequeñas aplicaciones hasta sistemas enormes y complejos. Además, debido a los avances que tienen lugar siempre hay una tendencia creciente en la complejidad de los sistemas de software. Una estimación precisa se vuelve cada vez más difícil a medida que aumenta la complejidad por la aparición de un mayor número de factores desconocidos con influencia en el esfuerzo necesario.
- **Cambios constantes:** La ingeniería de software es un campo en constante evolución. Los requisitos del proyecto pueden no estar claramente definidos al comienzo o pueden cambiar durante el ciclo de desarrollo, los recursos pueden variar y nuevas tecnologías pueden surgir durante el desarrollo de un proyecto, lo que dificulta la estimación precisa del esfuerzo desde el principio. Además, a medida que avanza un proyecto es necesario actualizar la estimación de esfuerzo realizada y ajustarla a la nueva información y reducción de las incógnitas iniciales.
- **Diversidad de equipos:** Como se ha comentado previamente los equipos de desarrollo de software a menudo están compuestos por profesionales con diversas habilidades y experiencias. La estimación debe tener en cuenta estas diferencias para evitar subestimar o sobreestimar el esfuerzo necesario.
- **Falta de datos históricos:** En proyectos innovadores, puede faltar un conjunto de datos históricos sólido y relevante para basar la estimación. Sin datos históricos que sirvan como referencia, los equipos de desarrollo se enfrentan a un desafío adicional para prever el esfuerzo requerido de manera precisa.

La fusión de código, es decir, combinar las contribuciones de diferentes desarrolladores en las mismas líneas de código o áreas cercanas en una única base de código, es una práctica habitual en proyectos de desarrollo de software. Pero es una tarea compleja por la posibilidad de generar conflictos de código que ralentizan el proceso y dificultan la entrega puntual de un software de calidad. Predecir los casos de conflicto de manera eficiente y precisa es esencial para mantener la integridad del software. Debido a la importancia de encontrar mejores soluciones a este problema, M. Owhawi-

kareshk et al. [7] evalúan la viabilidad de predecir conflictos de fusión en el desarrollo colaborativo de software mediante el uso de características extraídas de Git en proyectos en distintos lenguajes de programación. Se encontró que los clasificadores logran una alta precisión en la predicción de fusiones seguras. Sin embargo, al predecir escenarios de fusión conflictivos los resultados son peores. Los autores comprobaron que, si bien la predicción precisa de conflictos es un desafío, los clasificadores son altamente efectivos para identificar escenarios de fusión seguros. A pesar de la falta de correlación significativa entre las características y los conflictos de fusión, demostraron que es posible construir clasificadores efectivos para la predicción de conflictos utilizando características extraídas de Git. Sin embargo, la predicción de conflictos presenta varios desafíos:

- **Gran volumen de contribuciones y evolución continua:** En proyectos colaborativos se incorporan contribuciones por parte de cada desarrollador, además dado que los proyectos de software evolucionan constantemente, esto se traduce en un alto volumen de cambios, en los cuales hay que evaluar su potencial para causar conflictos. Por otro lado, las nuevas correcciones y mejoras se integran regularmente, lo que significa que la predicción de conflictos debe ser continua y adaptarse a medida que el proyecto avanza.
- **Detección de cambios:** Los conflictos pueden no ser obvios a simple vista. Algunos cambios sutiles en el código pueden interactuar de maneras inesperadas y causar conflictos difíciles de identificar sin un análisis profundo que requiere un gran esfuerzo adicional por parte de los equipos de desarrollo que afecta al ciclo de desarrollo.
- **Integración en el flujo de trabajo:** La predicción de conflictos debe integrarse de manera efectiva en el flujo de trabajo de desarrollo ya que requiere obtener información relevante por parte de los desarrolladores en el momento adecuado y en el contexto de su trabajo. La colaboración eficiente entre desarrolladores y sistemas de predicción de conflictos es esencial, lo que significa una correcta la interpretación de los resultados de predicción y permitir tomar decisiones para resolver el conflicto con la seguridad de comprender qué ha llevado a obtener esa predicción por parte del sistema de predicción.

### 3. Objetivo

El objetivo de este TFM es estudiar y aplicar un enfoque de ML, junto a distintos tipos de métodos de explicabilidad (como las explicaciones globales y locales), a fin de analizar el rendimiento de los algoritmos de ML y comprobar la utilidad de los métodos XAI en el campo de la analítica software. Concretamente, se busca ayudar al campo de la analítica software a hacer frente a los desafíos que surgen al realizar la estimación precisa del esfuerzo requerido para un proyecto y la detección proactiva de conflictos en el proceso de fusión de código, expuestos en la sección anterior.

Este enfoque nos permite aprovechar los beneficios que proporciona el ML y análisis de datos para su resolución, comprobados por los investigadores y los estudios mencionados también en la sección previa. Por otro lado, al mismo tiempo que se intenta afrontar los dos problemas mencionados, un aspecto remarcable en este trabajo es que se emplearán algoritmos de ML diversos (algoritmos SVM, algoritmos basados en árboles de decisión, algoritmos de *ensemble*, entre otros) tanto para regresión como clasificación. Este hecho contribuye a enriquecer el contenido del trabajo al ofrecer una mejor comparativa de cómo rinden cada algoritmo en el contexto del trabajo, de forma similar a como se han hecho en los trabajos usados como referencia.

Para complementar la contribución que aporta de este trabajo y conseguir una visión más completa del ámbito del ML, no solo se estudia y aplican diversos algoritmos de ML. También profundizaremos en la explicabilidad de estos modelos ante la necesidad fundamental de comprender el cómo y porqué los modelos de ML hacen sus predicciones, y saber qué ha influido más en cada caso y revelar interacciones complejas entre las características. De esta forma, se podrá valorar la importancia de los métodos XAI en la mejora de la transparencia y compresión de modelos de ML empleados para resolver estos problemas en analítica software.

En resumen, para abordar los problemas de precisión en la estimación del esfuerzo como la eficiencia en la detección de conflictos se tiene como objetivo aprovechar tanto las virtudes del ML como de las técnicas de explicabilidad. Al mismo tiempo que se obtiene una transparencia e interpretación adecuada de los modelos empleados, se contribuye a expandir y confirmar los resultados obtenidos en el uso de ML en esta área por estudios previos, y añadir diversidad a esos estudios mediante el uso de otros algoritmos. Además de profundizar en el análisis de resultados mediante las técnicas de explicabilidad y asentar el uso de este tipo de técnicas en la industria. Por último, presentamos ambos problemas de forma individual y con más detalle.

## 4. Definición del problema

Como se ha comentado previamente en el ámbito de la analítica software, la estimación precisa del esfuerzo necesario para el desarrollo de proyectos y la predicción de conflictos en casos de fusión de código son desafíos cruciales que afectan directamente a la eficiencia y la calidad del proceso de desarrollo.

### 4.1. Estimación de esfuerzo en ingeniería de software

El objetivo en esta parte es obtener como salida de los diversos modelos de ML una predicción en forma de valor numérico, correspondiente al esfuerzo necesario para completar un proyecto. La interpretación de los resultados de la estimación de esfuerzo implica valores numéricos, entonces tener una salida continua es esencial. Así, se buscará predecir el valor del esfuerzo necesario en función de las características de entrada recogidas en conjuntos de datos representativos en el ámbito del estudio de la estimación de esfuerzo en el desarrollo de software, tomando como referencia el repositorio [8]. Estos conjuntos cuentan con características que se pueden extraer de un proyecto de desarrollo software y que son útiles para estimar el esfuerzo, como por ejemplo características relacionadas con el grupo de trabajo (tamaño del equipo, experiencia, etc.) y con el tamaño o complejidad del proyecto (duración, recursos, número de consultas, etc.). Por lo tanto, estamos ante un problema que debe de ser abordado mediante algoritmos de ML basados en regresión, por su capacidad para generar una salida continua al predecir valores numéricos y trabajar con características de entrada de tipo numérico.

Por otro lado, mediante la comparación de los distintos modelos de ML podemos obtener una visión más completa de los puntos fuertes de cada uno y mejorar la precisión de la estimación de esfuerzo en proyectos de desarrollo de software. Se investigará cómo estos modelos pueden identificar patrones y características que influyen significativamente en el esfuerzo requerido para completar un proyecto. Esto no solo permitirá una estimación más precisa, sino que también proporcionará una comprensión más profunda de los factores que impactan en el proceso de desarrollo.

## 4.2. Predicción de conflictos en fusiones de código

La predicción de conflictos tiene como objetivo construir modelos de ML capaces de determinar si ocurrirá un conflicto o no durante una fusión de código, es decir, la salida es de tipo binario. Por lo tanto, al contrario del problema de estimación de esfuerzo, en el caso de predicción de conflictos en fusiones de código la naturaleza del problema sugiere que la salida debe ser discreta. Además, en este contexto una salida discreta es más interpretable y útil en la práctica.

Por lo tanto, se emplearán técnicas de ML de clasificación en combinación con conjuntos de datos que contienen características relevantes de proyectos de desarrollo colaborativo relacionados con el trabajo de M. Owhadi-Kareshk et al. [7] y contenidos en el repositorio [9]. Las características de estos conjuntos de datos están relacionadas con las métricas de código (número de líneas y ficheros modificados, cantidad de comentarios, etc.) y con la metainformación del repositorio (frecuencia de las contribuciones, número de desarrolladores, duración, número de *commits*, etc.). Además, al tratarse de un problema de clasificación las instancias han sido etiquetadas previamente como “conflictos” (1) o “no conflictos” (0).

## 5. Metodología

En esta sección, se presenta una descripción detallada de los pasos que se van a seguir en el desarrollo del presente trabajo. Como se ha comentado previamente, este trabajo de fin de máster se enfoca en dos problemas concretos en el campo de la analítica software: el problema de estimación de esfuerzo abordándolo desde un enfoque de regresión y la predicción de conflictos de fusión desde un enfoque de clasificación. Ya que en ambas partes se utilizan técnicas de ML y análisis de datos para abordar estos problemas específicos es necesario detallar los conjuntos de datos utilizados para cada parte, así como el preprocesamiento y la configuración de parámetros de los algoritmos de ML que van a ser empleados. Además, se presentarán las métricas utilizadas para evaluar el rendimiento de los modelos y se describirán los procedimientos seguidos para obtener resultados confiables y los resultados de las técnicas de explicabilidad empleadas. Como ambos problemas que serán abordados tienen similitudes en algunas de las etapas del proceso se van a presentar en primer lugar aquellos procedimientos comunes en ambos problemas y posteriormente los procedimientos que son específicos de cada problema. En la Fig. 1 se muestra una sencilla representación de la metodología seguida de forma general para alcanzar los objetivos de este trabajo.

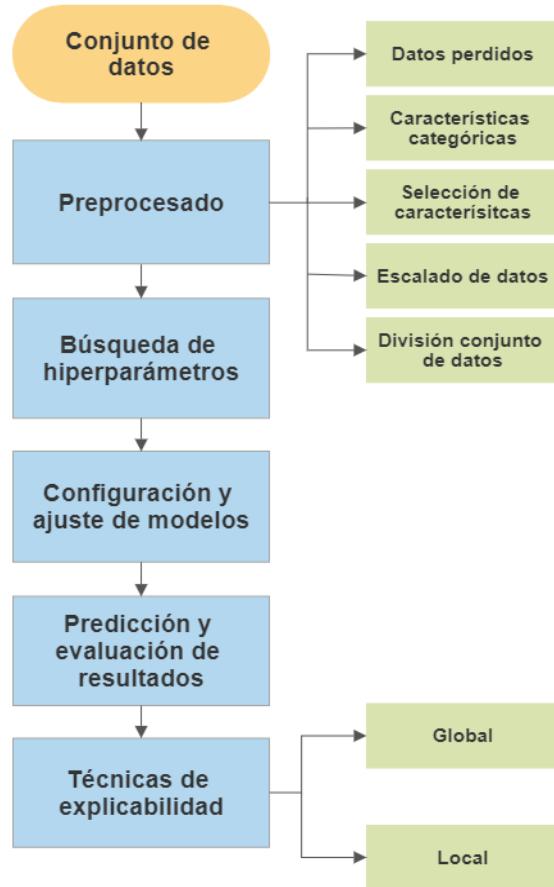


Fig. 1. Diagrama de los pasos seguidos en el presente trabajo.

## 5.1. Metodología común

### 5.1.1. Tratamiento de valores perdidos

Cada conjunto de datos seleccionado será analizado en busca de valores perdidos, así como las instancias que los contienen. Para solventar el problema que encontramos en los casos que presenten valores perdidos se explorarán dos enfoques principales: eliminación de las instancias con valores perdidos y la sustitución de los valores por la moda en atributos nominales o la media en atributos numéricos. De esta forma, se estudiará el impacto de cada enfoque en el rendimiento de los modelos de regresión mediante una evaluación comparativa usando validación cruzada para determinar cuál de los enfoques produce un mejor resultado en términos de rendimiento predictivo.

### 5.1.2. Transformación de características categóricas

Debido a que los modelos de ML que vamos a emplear suelen tener dificultades con la existencia de características de tipo categórico se va a analizar las características de los conjuntos de datos en busca de características de tipo categórico, para posteriormente en los conjuntos de datos que presenten características de este tipo se aplicarán técnicas de codificación para convertirlas en variables binarias como el método "*OneHotEncoder*" de la librería *Scikit-learn*. Esta transformación permite representar las categorías de las características categóricas como variables binarias independientes, lo cual facilita su utilización en modelos de ML.

### 5.1.3. Selección de características

Tendrá lugar un proceso de selección de características para identificar las más relevantes para el problema que se está estudiando. En primer lugar, se realizará un estudio de la correlación entre las características de cada conjunto de datos y se identificarán los pares de características con una correlación alta, estableciendo un umbral de 0.8. Para cada característica altamente correlacionadas con otra, se evalúa su impacto en el rendimiento predictivo mediante validación cruzada, comparando el rendimiento incluyendo y eliminando la característica con alta correlación en los distintos modelos. Aquellas características que muestren una correlación alta y no aportaban un beneficio significativo al modelo serán eliminadas del conjunto de datos. Además, se eliminará cualquier característica relacionada directamente con la salida de nuestros modelos, como por ejemplo en el caso de la estimación de esfuerzo, aquellas características que en sí mismas representen la estimación de esfuerzo. Ya que dicha salida es nuestro objetivo por lo que su presencia resultaba redundante y evidentemente mostrarán una correlación excesivamente alta con la variable de salida.

### 5.1.4. División del conjunto de datos

Se ha dividido el conjunto de datos en conjuntos de entrenamiento y prueba utilizando la función “*train\_test\_split*” proporcionada por Scikit-learn, con una proporción de 70% para entrenamiento y 30% para prueba. Esta división permite evaluar el rendimiento de los modelos en datos no vistos y evitar el sobreajuste. Para los casos de clasificación con desbalanceo de clases, se empleará el parámetro “*stratify*” con el objetivo de mantener la misma distribución de clases en los conjuntos de entrenamiento y prueba.

### 5.1.5. Entrenamiento del modelo

Cada modelo será entrenado en cada conjunto de datos utilizando los mejores hiperparámetros encontrados, para posteriormente hacer predicciones sobre el conjunto de prueba y evaluar el rendimiento utilizando diversas métricas. Para obtener una visión más fiable del rendimiento se empleará validación cruzada, usando un número de divisiones adecuada a cada conjunto de datos e igual para todos los modelos de un mismo conjunto de datos.

### 5.1.6. Técnicas de explicabilidad del modelo

Un análisis de la explicabilidad global de los modelos se llevará a cabo mediante la función “*permutation\_importance*” de *Scikit-Learn*, que proporciona la importancia relativa de cada característica en el rendimiento del modelo. Para evitar un exceso de características en el resultado de este método se establece un umbral de 0,01.

Además, se aplicarán técnicas de explicabilidad local mediante técnicas como *Break-Down*, *Shapley Values* y LIME proporcionadas por la librería *Dalex*. Estas técnicas permiten comprender cómo las características individuales influyen en las predicciones del modelo, proporcionando una mayor transparencia y comprensión del proceso de estimación de esfuerzo. En el caso de las técnicas de explicabilidad local el procedimiento difiere ligeramente entre el problema de estimación de esfuerzo y la predicción de conflictos debido a la propia naturaleza de los problemas:

- Para el problema de estimación de esfuerzo estas técnicas serán aplicadas sobre las estimaciones de esfuerzo con valor máximo, mínimo y mediana permitiendo analizar cómo cada modelo asigna importancia a cada característica en el contexto de diferentes niveles de esfuerzo estimado. De esta

forma, se han obtenido información específica sobre el impacto de cada característica en las predicciones realizadas por el modelo y ha brindado una comprensión más detallada de cómo el modelo interpreta y utiliza la información para estimar el valor de esfuerzo en el desarrollo de software.

- Para el problema de predicción de conflictos durante la fusión de código las técnicas de explicabilidad local se aplicarán sobre las instancias de cada tipo de resultado posible obtenido de la predicción, es decir, aquellas instancias que son verdaderos positivo, verdadero negativo, falso positivo o falso negativo. Pero para realizar un estudio más profundo de como los modelos predicen el resultado para cada uno de los cuatro resultados posibles se estudiarán aquellas instancias que obtengan la probabilidad máxima, mínima y en la mediana para ser clasificadas en dicho resultado.

Es importante mencionar que debido al gran volumen de resultados que se van a obtener de las técnicas de explicabilidad, se emplearán tablas resumen y gráficas para permitir una mejor comparación de los resultados a nivel de instancia, es decir, entre las instancias estudiadas con distintos nivel de estimación (en la parte de estimación de esfuerzo) e instancias con distintos niveles de probabilidad (en la parte de clasificación de conflictos) y comparación a nivel de modelos empleados. Para elaborar las tablas resumen se tendrá en cuenta tanto el *ranking* en valor de la contribución de la característica a la predicción como el número de veces que una característica aparece entre las cinco primeras características en contribución. De esta forma, buscaremos evitar características cuyos contribuciones sean ínfimas y por lo tanto poco relevantes para el estudio. También sirve para evitar los casos de características que aparezcan con un valor alto únicamente en una ocasión se establezcan entre las características más importantes, introduciendo ruido y dificultando la comparación y extracción de información relevante de los resultados.

## 5.2. Metodología en estimación de esfuerzo

### 5.2.1. Aplicación de técnicas de *bootstrapping*

Como se verá a continuación en la descripción de los conjuntos de datos seleccionados para el problema de estimación de esfuerzo, tenemos conjuntos de datos con un número reducido de instancias. Este hecho provoca un rendimiento muy bajo y por lo tanto se van a aplicar técnicas de *bootstrapping* para aumentar el número de instancias por un factor adecuado para evitar sobreajuste posteriormente. Posteriormente, se comparará los resultados obtenidos mediante el uso de estas técnicas con respecto a los resultados iniciales.

### 5.2.2. Escalado de características

Tras el preprocesado y limpieza de los conjuntos de datos se aplicará un escalado en las características utilizando el método "*StandardScaler*" de la biblioteca *Scikit-learn*. Este escalado se ha realizado en las características numéricas del conjunto de datos, excluyendo aquellas características binarias obtenidas mediante "*OneHotEncoder*". El escalado de las características numéricas permite que todas las variables tengan una escala comparable, lo cual es beneficioso para algunos algoritmos de regresión que vamos a emplear, como SVR.

### 5.2.3. Descripción de los conjuntos de datos

Con el objetivo de realizar un estudio y análisis en profundidad se estudiarán varios conjuntos de datos relevantes en este campo [8], que se exponen a continuación: *Miyazaki* [10], *Kitchenham* [11], *ISBSG* [12], *Albrecht* [13], *CHINA* [14], *Desharnais* [15], *Atkinson* [9] y *Maxwell* [16].

Se realizará una descripción estadística de los conjuntos de datos utilizados mediante el uso de la biblioteca *Pandas*, que nos permitirá explorar las características de cada conjunto analizando la distribución de los datos. Una descripción detallada de los pasos seguidos en el preprocesado de los conjuntos de datos de esta parte y cómo se han afrontado las dificultades encontradas se recoge en el Anexo 1.

- **Desharnais**

Inicialmente, este conjunto de datos contaba con 81 instancias y 13 características, no contiene ningún valor perdido. La variable “*Effort*” es la variable dependiente que será estimada por cada uno de los modelos.

**Tabla 1. Datos estadísticos de las características que contiene el conjunto de datos Desharnais**

Característica	mean	std	min	25%	50%	75%	max
<i>TeamExp</i>	2.2	1.4	-1.0	1.0	2.0	4.0	4.0
<i>ManagerExp</i>	2.5	1.6	-1.0	1.0	3.0	4.0	7.0
<i>YearEnd</i>	85.7	1.2	82.0	85.0	86.0	87.0	88.0
<i>Length</i>	11.7	7.4	1.0	6.0	10.0	14.0	39.0
<i>Effort</i>	5046.3	4418.8	546.0	2352.0	3647.0	5922.0	23940.0
<i>Transactions</i>	182.1	144.0	9.0	88.0	140.0	224.0	886.0
<i>Entities</i>	122.3	84.9	7.0	57.0	99.0	169.0	387.0
<i>Adjustment</i>	27.6	10.6	5.0	20.0	28.0	35.0	52.0
<i>PointsAjust</i>	289.2	185.8	62.0	152.0	255.0	351.0	1116.0
<i>Language</i>	1.6	0.7	1.0	1.0	1.0	2.0	3.0

- **CHINA**

El conjunto de datos final consta de 499 instancias y 12 características numéricas, siendo la característica ‘*Effort*’ la variable dependiente de nuestro problema. Además, no presenta valores perdidos en ninguna instancia.

**Tabla 2. Datos estadísticos de las características que contiene el conjunto de datos China.**

Característica	mean	std	min	25%	50%	75%	max
<i>AFP</i>	486.8	1059.2	9.0	100.5	215.0	437.5	17518.0
<i>Input</i>	167.1	486.3	0.0	27.0	63.0	152.5	9404.0
<i>Output</i>	113.6	221.3	0.0	13.0	42.0	112.0	2455.0
<i>Enquiry</i>	61.6	105.4	0.0	6.0	24.0	68.5	952.0

<b>File</b>	91.2	210.3	0.0	14.0	36.0	84.0	2955.0
<b>Interface</b>	24.2	85.0	0.0	0.0	0.0	20.0	1572.0
<b>Added</b>	360.4	829.8	0.0	38.0	135.0	325.5	13580.0
<b>Changed</b>	85.1	290.9	0.0	0.0	4.0	87.0	5193.0
<b>Deleted</b>	12.4	124.2	0.0	0.0	0.0	0.0	2657.0
<b>PDR_UFP</b>	12.1	12.8	0.3	4.2	8.0	15.75	96.6
<b>Resource</b>	1.5	0.8	1.0	1.0	1.0	2.0	4.0
<b>Duration</b>	8.7	7.3	1.0	4.0	7.0	11.0	84.0
<b>Effort</b>	3921.0	6480.9	26.0	703.5	1829.0	3826.5	54620.0

- **Maxwell**

Conjunto de datos con 62 instancias y 27 características numéricas sin valores perdidos. La variable dependiente que contiene el valor del esfuerzo es '*Effort*'.

**Tabla 3. Datos estadísticos de las características que contiene el conjunto de datos Maxwell**

Característica	mean	std	min	25%	50%	75%	max
<b>App</b>	2.4	0.9	1.0	2.0	2.0	3.0	5.0
<b>Har</b>	2.6	0.9	1.0	2.0	2.0	3.0	5.0
<b>Dba</b>	1.0	0.4	0.0	1.0	1.0	1.0	4.0
<b>Ifc</b>	1.9	0.2	1.0	2.0	2.0	2.0	2.0
<b>Source</b>	1.9	0.3	1.0	2.0	2.0	2.0	2.0
<b>Telonuse</b>	0.2	0.4	0.0	0.0	0.0	0.0	1.0
<b>Nlan</b>	2.5	1.0	1.0	2.0	3.0	3.0	4.0
<b>T01</b>	3.0	0.9	1.0	2.0	3.0	4.0	5.0
<b>T02</b>	3.0	0.7	1.0	3.0	3.0	3.0	5.0
<b>T03</b>	3.0	0.9	2.0	2.0	3.0	4.0	5.0
<b>T04</b>	3.2	0.7	2.0	3.0	3.0	4.0	5.0
<b>T05</b>	3.0	0.7	1.0	3.0	3.0	3.0	5.0
<b>T06</b>	2.9	0.7	1.0	3.0	3.0	3.0	4.0
<b>T07</b>	3.2	0.9	1.0	3.0	3.0	4.0	5.0
<b>T08</b>	3.8	0.9	2.0	3.0	4.0	5.0	5.0
<b>T09</b>	4.1	0.7	2.0	4.0	4.0	5.0	5.0
<b>T10</b>	3.6	0.9	2.0	3.0	4.0	4.0	5.0
<b>T11</b>	3.4	0.9	2.0	3.0	3.0	4.0	5.0
<b>T12</b>	3.8	0.7	2.0	4.0	4.0	4.0	5.0
<b>T13</b>	3.1	0.9	1.0	2.0	3.0	4.0	5.0

- **Miyazaki94**

Es un conjunto de datos pequeño con 48 instancias y solo 6 características de tipo numérico. 'MM' representa el valor del esfuerzo. Este conjunto de datos no tiene valores perdidos.

**Tabla 4. Datos estadísticos de las características que contiene el conjunto de datos Miyazaki**

Característica	mean	std	min	25%	50%	75%	max
<b>KLOC</b>	70.8	87.6	6.9	24.8	45.4	65.3	417.6
<b>SCRN</b>	33.7	47.2	0.0	9.0	23.5	36.25	281.0
<b>FORM</b>	22.4	20.5	0.0	9.0	16.0	31.0	91.0
<b>ESCRN</b>	525.6	626.1	0.0	127.25	322.5	604.25	3000.0
<b>EFORM</b>	460.7	396.8	0.0	180.5	342.5	636.75	1566.0
<b>EFILE</b>	1854.6	6398.6	57.0	480.75	872.5	1318.0	45000.0
<b>MM</b>	87.5	228.8	5.6	26.375	38.1	58.95	1586.0

- **Albrecht**

No presenta valores perdidos en ninguna de sus 8 características y 24 instancias. ‘Effort’ es el valor del esfuerzo que buscamos estimar.

**Tabla 5. Datos estadísticos de las características que contiene el conjunto de datos Albrecht**

Característica	mean	std	min	25%	50%	75%	max
<b>Input</b>	40.3	36.9	7.0	23.0	33.5	43.5	193.0
<b>Output</b>	47.3	35.2	12.0	18.5	39.0	64.5	150.0
<b>Inquiry</b>	16.9	19.3	0.0	3.3	13.5	20.3	75.0
<b>File</b>	17.4	15.5	3.0	5.8	11.5	22.3	60.0
<b>FPAadj</b>	0.9	0.1	0.8	0.9	1.0	1.1	1.2
<b>AdjFP</b>	647.6	487.9	199.0	287.5	506.0	710.3	1902.0
<b>Effort</b>	21.9	28.4	0.5	7.15	11.5	19.5	105.2

- **Atkinson**

Este conjunto de datos cuenta con 48 instancias y 9 características numéricas sin ningún valor perdido y siendo ‘Act Effort’ la variable dependiente.

**Tabla 6. Datos estadísticos de las características que contiene el conjunto de datos Atkinson**

Característica	mean	std	min	25%	50%	75%	max
<b>Act Effort</b>	456.1	241.1	109.0	257.5	427.0	600.25	912.0
<b>Est Dur</b>	8.9	7.3	2.0	5.0	6.0	9.875	29.0
<b>EA-RTFP</b>	17.6	16.9	2.0	6.0	10.0	27.0	59.0
<b>IMT</b>	1.1	0.9	0.0	0.0	1.0	2.0	3.0
<b>IAT</b>	2.9	3.3	0.0	0.0	2.0	4.25	10.0
<b>OAT</b>	3.3	6.5	0.0	0.0	0.0	2.25	21.0
<b>OT</b>	5.3	10.4	0.0	0.0	0.0	4.0	35.0
<b>ER</b>	3.3	3.5	0.0	0.0	2.5	6.25	9.0
<b>ERA</b>	8.4	8.8	0.0	0.0	6.5	17.25	24.0

- **Kitchenham**

Este conjunto de datos contiene 135 instancias y 9 características numéricas de las cuales 6 son binarias. “Actual.effort” es el esfuerzo que requiere el proyecto.

**Tabla 7. Datos estadísticos de las características que contiene el conjunto de datos Kitchenham**

Característica	mean	std	min	25%	50%	75%	max
<i>Project.type_A</i>	0.02	0.17	0.0	0.0	0.0	0.0	1.0
<i>Project.type_C</i>	0.01	0.12	0.0	0.0	0.0	0.0	1.0
<i>Project.type_D</i>	0.38	0.49	0.0	0.0	0.0	1.0	1.0
<i>Project.type_P</i>	0.56	0.50	0.0	0.0	1.0	1.0	1.0
<i>Project.type_Pr</i>	0.01	0.09	0.0	0.0	0.0	0.0	1.0
<i>Project.type_U</i>	0.01	0.09	0.0	0.0	0.0	0.0	1.0
<i>Actual.duration</i>	200.96	130.56	37.0	121.5	171.0	240.0	946.0
<i>Adjusted.function.points</i>	527.75	1572.90	18.9	123.9	258.2	554.6	18137.5
<i>Actual.effort</i>	3169.13	9933.63	219.0	880.5	1557.0	2877.5	113930.0

- **ISBSG**

El conjunto de datos empleado se ha obtenido realizando un filtrado del conjunto de datos original para obtener un subconjunto de datos formado por 197 instancias y 37 características, además de la variable dependiente, ‘Normalised Work Effort Level 1’.

**Tabla 8. Datos estadísticos de las características que contiene el conjunto de datos ISBSG.**

Característica	mean	std	min	25%	50%	75%	max
<i>Development Type_Enhancement</i>	0.41	0.49	0.0	0.0	0.0	1.0	1.0
<i>Development Type_New Development</i>	0.53	0.50	0.0	0.0	1.0	1.0	1.0
<i>Development Type_Re-Development</i>	0.01	0.07	0.0	0.0	0.0	0.0	1.0
<i>Development Platform_MF</i>	0.57	0.50	0.0	0.0	1.0	1.0	1.0
<i>Development Platform_MR</i>	0.13	0.33	0.0	0.0	0.0	0.0	1.0
<i>Development Platform_Multi</i>	0.17	0.38	0.0	0.0	0.0	0.0	1.0
<i>Development Platform_PC</i>	0.13	0.34	0.0	0.0	0.0	0.0	1.0
<i>Primary Programming Language_ABAP</i>	0.01	0.10	0.0	0.0	0.0	0.0	1.0
<i>Primary Programming Language_Access</i>	0.03	0.16	0.0	0.0	0.0	0.0	1.0
<i>Primary Programming Language_C</i>	0.023	0.16	0.0	0.0	0.0	0.0	1.0

<i>Primary Programming Language_C#</i>	0.06	0.23	0.0	0.0	0.0	0.0	1.0
<i>Primary Programming Language_C++</i>	0.02	0.14	0.0	0.0	0.0	0.0	1.0
<i>Primary Programming Language_CLIPPER</i>	0.02	0.14	0.0	0.0	0.0	0.0	1.0
<i>Primary Programming Language_COBOL</i>	0.29	0.46	0.0	0.0	0.0	1.0	1.0
<i>Primary Programming Language_COOL:GEN</i>	0.07	0.26	0.0	0.0	0.0	0.0	1.0
<i>Primary Programming Language_EASYTRIEVE</i>	0.04	0.20	0.0	0.0	0.0	0.0	1.0
<i>Primary Programming Language_HPS</i>	0.01	0.07	0.0	0.0	0.0	0.0	1.0
<i>Primary Programming Language_Java</i>	0.02	0.12	0.0	0.0	0.0	0.0	1.0
<i>Primary Programming Language_JavaScript</i>	0.01	0.07	0.0	0.0	0.0	0.0	1.0
<i>Primary Programming Language_Lotus Notes</i>	0.01	0.10	0.0	0.0	0.0	0.0	1.0
<i>Primary Programming Language_NATURAL</i>	0.05	0.22	0.0	0.0	0.0	0.0	1.0
<i>Primary Programming Language_ORACLE</i>	0.01	0.10	0.0	0.0	0.0	0.0	1.0
<i>Primary Programming Language_Other 4GL</i>	0.03	0.17	0.0	0.0	0.0	0.0	1.0
<i>Primary Programming Language_Other ApG</i>	0.01	0.10	0.0	0.0	0.0	0.0	1.0
<i>Primary Programming Language_PL/I</i>	0.09	0.28	0.0	0.0	0.0	0.0	1.0
<i>Primary Programming Language_PowerBuilder</i>	0.02	0.14	0.0	0.0	0.0	0.0	1.0
<i>Primary Programming Language_SQL</i>	0.07	0.25	0.0	0.0	0.0	0.0	1.0
<i>Primary Programming Language_Smalltalk</i>	0.01	0.07	0.0	0.0	0.0	0.0	1.0
<i>Primary Programming Language_TELON</i>	0.04	0.19	0.0	0.0	0.0	0.0	1.0
<i>Primary Programming Language_Visual Basic</i>	0.08	0.27	0.0	0.0	0.0	0.0	1.0
<i>Output count</i>	101.06	160.65	0.0	23.0	54.0	119.0	1337.0
<i>Enquiry count</i>	76.12	110.37	0.0	12.0	40.0	94.0	893.0
<i>File count</i>	104.21	239.53	0.0	20.0	52.0	110.0	2955.0
<i>Interface count</i>	17.44	34.38	0.0	0.0	5.0	17.0	284.0
<i>Adjusted Function Points</i>	526.53	1327.88	32.0	142.0	296.0	511.0	17518.0
<i>Normalised Work Effort Level 1</i>	5602.92	8208.05	252.0	1084.0	2761.0	5985.0	60826.0

#### 5.2.4. Modelado y ajuste de hiperparámetros

Se van a emplear varios algoritmos de regresión implementados en *Scikit-learn*, incluyendo "SVR" (*Support Vector Regression*), "RandomForest", "GradientBoostingRegressor" y "VotingRegressor". En primer lugar, se realizará una búsqueda exhaustiva de los mejores hiperparámetros para cada algoritmo utilizando la función "GridSearchCV". Debido al tamaño relativamente pequeño de los conjuntos de datos, se empleará la estrategia de validación cruzada repetida ("RepeatedKFold") para obtener una mayor confianza en los hiperparámetros seleccionados.

**Tabla 9. Rango de hiperparámetros evaluado en cada modelos empleado para estimación de esfuerzo.**

Modelo	Rango de parámetros
SVR	'kernel': linear, poly, rbf, sigmoid 'Degree': 1,2,3,4 'gamma': auto, scale, 0.1 'C': 0.5, 1.5, 2.5, 3.5, 4.5, 5.5, 6.5, 7.5, 8.5, 9.5, 10.5, 11.5, 12.5, 13.5, 14.5, 15.5, 16.5, 17.5, 18.5, 19.5, 20.5, 21.5
Random Forest Regressor	'n_estimators': 50,100,200,400 'criterion': squared_error, absolute_error 'min_samples_split': 2,3,4,5 'min_samples_leaf': 1,2,3,5 'bootstrap': True, False 'max_depth': 1,3,5,10
Gradient Boosting Regressor	'loss': squared_error, absolute_error 'learning_rate': 0.1, 0.3, 0.5, 1.0 'n_estimators': 50, 100, 200, 400 'criterion': friedman_mse, squared_error 'min_samples_split': 2, 3, 4, 5 'min_samples_leaf': 1, 2, 3, 5 'max_depth': 1, 3, 5, 10

#### 5.2.5. Evaluación de los modelos

Las métricas que se van a estudiar incluyen métricas habituales en el análisis de modelos de regresión MAE (*Mean Absolute Error*, Error Absoluto Medio ), RMSE (*Root Mean Squared Error*, Raíz del Error Cuadrático Medio), R<sup>2</sup> (Coeficiente de determinación) y también métricas habituales en la literatura sobre la estimación de esfuerzo en ingeniería de software [1,4,6,17,18] como MMRE (*Mean Magnitude of Relative Error*, Error relativo Medio) y PRED(0.25) (Porcentaje de predicciones dentro de un rango del 25%). Estas métricas proporcionan una visión completa del rendimiento y la precisión de los modelos.

- **Error Absoluto Medio:** es la media de las diferencias absolutas entre las predicciones y los valores reales. Representa la magnitud promedio de los errores de predicción, sin considerar su dirección (positiva o negativa). Matemáticamente se define como:

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \quad (1)$$

- **Raíz del Error Cuadrático Medio:** El error cuadrático medio calcula la media de los errores al cuadrado entre las predicciones y los valores reales. Los errores más grandes tienen un mayor

impacto debido al cuadrado, lo que lo hace sensible a valores atípicos. Es habitual usar la raíz cuadrada del error cuadrático medio pues proporciona una métrica en la misma escala que los valores originales, lo que facilita su interpretación. Matemáticamente:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2} \quad (2)$$

- **Coeficiente de Determinación:** El coeficiente de determinación, también conocido como  $R^2$ , indica la proporción de la varianza en la variable dependiente que es explicada por el modelo. Un coeficiente de determinación más alto sugiere que el modelo se ajusta mejor a los datos reales. Matemáticamente:

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2} \quad (3)$$

- **Error Relativo Medio:** calcula la media de los errores relativos (en valor absoluto) entre las predicciones y los valores reales. Mide la magnitud promedio de los errores en términos proporcionales. Matemáticamente:

$$MMRE = \frac{1}{n} \sum_{i=1}^n \left| \frac{y_i - \hat{y}_i}{y_i} \right| \quad (4)$$

- **Porcentaje de Predicciones dentro de un rango del 25%:** Esta métrica calcula el porcentaje de predicciones que caen dentro de un rango específico (en este caso, dentro del 25% del valor real). Es útil para evaluar la precisión del modelo en la predicción de valores cercanos al valor real. Matemáticamente:

$$Pred(0.25) = \left( \frac{\text{Número de predicciones dentro del 25\%}}{n} \right) \quad (5)$$

En las expresiones anteriores  $n$  es el número de instancias estimadas,  $y_i$  es el valor real de la instancia,  $\hat{y}_i$  es el valor predicho esa misma instancia e  $\bar{y}$  es la media de los valores reales. Los resultados obtenidos para cada conjunto de datos serán recopilados en tablas y gráficas. Esto permite una mejor comprensión y comparación de los resultados entre los distintos modelos y conjuntos de datos, así como para mostrar la variación del rendimiento en función de los diferentes modelos y métricas utilizadas.

## 5.3. Metodología en Predicción de conflictos

### 5.3.1. Descripción de los conjuntos de datos

Para estudiar con mayor precisión este problema se van a emplear los conjuntos de datos referentes a casos de fusión de código en cuatro lenguajes de programación, concretamente los lenguajes son Python, Java, PHP y Ruby, proporcionados por los autores del repositorio [9]:

Al igual que en la parte anterior se realizará una descripción estadística de los conjuntos de datos utilizados mediante el uso de la biblioteca *Pandas*. Todos los conjuntos de datos de esta parte tienen las mismas características. La descripción más detallada del preprocesado se recoge en el Anexo 2.

- **Java**

El conjunto de datos correspondiente al lenguaje Java presenta 27 características con 26699 instancias sin valores perdidos.

**Tabla 10. Datos estadísticos de las características que contiene el conjunto de datos del lenguaje Java.**

Característica	mean	std	min	25%	50%	75%	max
<i>parallel_changed_file_num</i>	5.8	180.1	0.0	0.0	0.0	0.0	11225.0
<i>commit_num</i>	73.3	329.1	1.0	1.0	4.0	19.0	8402.0
<i>file_added</i>	-158.5	1141.6	-33669.0	-3.0	0.0	0.0	20500.0
<i>file_removed</i>	-107.9	849.2	-24229.0	0.0	0.0	0.0	22253.0
<i>file_modified</i>	-583.3	4280.5	-139076.0	-19.0	1.0	3.0	69610.0
<i>line_added</i>	-78381.4	989103.5	-17558096.0	-489.0	1.0	62.0	17044123.0
<i>line_removed</i>	-43995.4	531357.2	-9044463.0	-183.5	1.0	20.0	23562126.0
<i>developer_num</i>	-2.1	10.9	-353.0	-1.0	0.0	1.0	143.0
<i>duration</i>	-9.4	1067.2	-38826.0	-32.0	-1.0	3.0	30704.0
<i>commit_density</i>	-44.4	794.9	-80264.0	-8.0	0.0	1.0	6695.0
<i>add_frequency</i>	1.3	1.8	0.0	0.0	1.0	2.0	25.0
<i>bug_frequency</i>	0.2	0.7	0.0	0.0	0.0	0.0	15.0
<i>change_frequency</i>	0.3	0.8	0.0	0.0	0.0	0.0	19.0
<i>delete_frequency</i>	0.0	0.3	0.0	0.0	0.0	0.0	10.0
<i>document_frequency</i>	0.1	0.5	0.0	0.0	0.0	0.0	12.0
<i>feature_frequency</i>	0.1	0.5	0.0	0.0	0.0	0.0	18.0
<i>fix_frequency</i>	1.6	2.1	0.0	0.0	1.0	2.0	19.0
<i>improve_frequency</i>	0.1	0.4	0.0	0.0	0.0	0.0	11.0
<i>refactor_frequency</i>	0.1	0.4	0.0	0.0	0.0	0.0	13.0
<i>remove_frequency</i>	0.4	0.9	0.0	0.0	0.0	1.0	19.0
<i>update_frequency</i>	0.6	1.4	0.0	0.0	0.0	1.0	33.0
<i>use_frequency</i>	0.6	1.2	0.0	0.0	0.0	1.0	23.0
<i>messages_min</i>	28.9	24.2	0.0	15.0	23.0	37.0	400.0
<i>messages_max</i>	81.1	55.4	0.0	49.0	69.0	94.0	408.0

<i>messages_mean</i>	49.3	25.0	0.0	37.0	45.9	55.7	400.0
----------------------	------	------	-----	------	------	------	-------

- **Python**

Para el lenguaje Python el conjunto de datos está formado por 49453 instancias y 27 características sin valores perdidos.

**Tabla 11. Datos estadísticos de las características para el lenguaje Python.**

Característica	mean	std	min	25%	50%	75%	max
<i>parallel_changed_file_num</i>	0.9	14.5	0.0	0.0	0.0	0.0	2328.0
<i>commit_num</i>	41.7	240.7	1.0	2.0	6.0	22.0	19787.0
<i>file_added</i>	-16.8	738.9	-21598.0	-2.0	0.0	0.0	72916.0
<i>file_removed</i>	-5.7	490.7	-15158.0	0.0	0.0	0.0	53222.0
<i>file_modified</i>	-105.7	1703.9	-144179.0	-23.0	0.0	3.0	54702.0
<i>line_added</i>	-6862.0	189516.1	-9691102.0	-427.0	1.0	35.0	13119916.0
<i>line_removed</i>	-4156.9	124735.6	-4979595.0	-173.0	0.0	13.0	9413449.0
<i>developer_num</i>	-1.8	14.3	-895.0	-2.0	0.0	1.0	313.0
<i>duration</i>	24.5	927.5	-33187.0	-24.0	-1.0	12.0	25171.0
<i>commit_density</i>	-48.5	701.3	-43637.0	-8.0	0.0	1.0	6363.0
<i>add_frequency</i>	1.3	1.9	0.0	0.0	1.0	2.0	19.0
<i>bug_frequency</i>	0.2	0.7	0.0	0.0	0.0	0.0	13.0
<i>change_frequency</i>	0.3	0.8	0.0	0.0	0.0	0.0	24.0
<i>delete_frequency</i>	0.0	0.3	0.0	0.0	0.0	0.0	9.0
<i>document_frequency</i>	0.1	0.4	0.0	0.0	0.0	0.0	12.0
<i>feature_frequency</i>	0.1	0.5	0.0	0.0	0.0	0.0	14.0
<i>fix_frequency</i>	1.6	2.1	0.0	0.0	1.0	2.0	19.0
<i>improve_frequency</i>	0.1	0.4	0.0	0.0	0.0	0.0	11.0
<i>refactor_frequency</i>	0.1	0.4	0.0	0.0	0.0	0.0	12.0
<i>remove_frequency</i>	0.4	0.9	0.0	0.0	0.0	1.0	20.0
<i>update_frequency</i>	0.6	1.4	0.0	0.0	0.0	1.0	38.0
<i>use_frequency</i>	0.6	1.1	0.0	0.0	0.0	1.0	23.0
<i>messages_min</i>	29.1	25.2	0.0	15.0	23.0	37.0	401.0
<i>messages_max</i>	81.3	56.3	0.0	49.0	69.0	94.0	418.0

<i>messages_median</i>	47.1	25.6	0.0	35.0	45.0	53.5	401.0
------------------------	------	------	-----	------	------	------	-------

- **Ruby**

El conjunto de datos empleado contiene 40129 instancias y 27 características. No presenta valores perdidos en ninguna característica.

**Tabla 12. Datos estadísticos de las características para el lenguaje Ruby.**

Característica	mean	std	min	25%	50%	75%	max
<i>parallel_changed_file_num</i>	1.2	23.0	0.0	0.0	0.0	0.0	2596.0
<i>commit_num</i>	72.7	1043.4	1.0	1.0	4.0	18.0	44155.0
<i>file_added</i>	-179.5	3417.4	-131063.0	-1.0	0.0	0.0	24248.0
<i>file_removed</i>	-156.6	3093.2	-104129.0	0.0	0.0	0.0	139785.0
<i>file_modified</i>	-404.5	8444.1	-422454.0	-17.0	1.0	2.0	257352.0
<i>line_added</i>	-19906.9	476631.8	-21369050.0	-209.0	1.0	21.0	35888503.0
<i>line_removed</i>	-19316.5	572538.2	-33084755.0	-84.0	1.0	7.0	53001109.0
<i>developer_num</i>	-2.2	18.7	-1364.0	-2.0	0.0	1.0	281.0
<i>duration</i>	79.6	1324.0	-41164.0	-40.0	-2.0	13.0	37540.0
<i>commit_density</i>	-40.2	523.3	-30730.0	-8.0	0.0	1.0	40865.0
<i>add_frequency</i>	1.3	1.9	0.0	0.0	1.0	2.0	26.0
<i>bug_frequency</i>	0.2	0.7	0.0	0.0	0.0	0.0	13.0
<i>change_frequency</i>	0.3	0.8	0.0	0.0	0.0	0.0	15.0
<i>delete_frequency</i>	0.0	0.3	0.0	0.0	0.0	0.0	12.0
<i>document_frequency</i>	0.1	0.4	0.0	0.0	0.0	0.0	13.0
<i>feature_frequency</i>	0.1	0.5	0.0	0.0	0.0	0.0	18.0
<i>fix_frequency</i>	1.6	2.1	0.0	0.0	1.0	2.0	48.0
<i>improve_frequency</i>	0.1	0.4	0.0	0.0	0.0	0.0	13.0
<i>refactor_frequency</i>	0.1	0.5	0.0	0.0	0.0	0.0	13.0
<i>remove_frequency</i>	0.4	0.9	0.0	0.0	0.0	1.0	21.0
<i>update_frequency</i>	0.6	1.3	0.0	0.0	0.0	1.0	45.0
<i>use_frequency</i>	0.6	1.1	0.0	0.0	0.0	1.0	20.0
<i>messages_min</i>	29.0	24.8	0.0	15.0	24.0	37.0	400.0
<i>messages_max</i>	81.4	55.7	0.0	49.0	69.0	94.0	410.0

<i>messages_mean</i>	49.4	25.7	0.0	37.0	45.9	55.75	400.0
----------------------	------	------	-----	------	------	-------	-------

- **PHP**

En el caso del conjunto de datos correspondiente al lenguaje PHP tenemos 50342 instancias y 27 características.

**Tabla 13. Datos estadísticos de las características para el lenguaje PHP.**

Característica	mean	std	min	25%	50%	75%	max
<i>parallel_changed_file_num</i>	2.0	68.5	0.0	0.0	0.0	0.0	9793.0
<i>commit_num</i>	74.1	559.3	1.0	2.0	6.0	27.0	31906.0
<i>file_added</i>	-83.3	1787.7	-87102.0	-3.0	0.0	0.0	85771.0
<i>file_removed</i>	-54.1	1103.7	-69175.0	0.0	0.0	0.0	50124.0
<i>file_modified</i>	-457.1	18816.5	-1662546.0	-40.0	0.0	2.0	1121206.0
<i>line_added</i>	-27681.0	413213.1	-16595349.0	-754.75	1.0	22.0	15842232.0
<i>line_removed</i>	-22710.5	332510.9	-11631939.0	-320.0	0.0	8.0	7880851.0
<i>developer_num</i>	-3.5	20.3	-727.0	-2.0	0.0	1.0	261.0
<i>duration</i>	31.3	1036.7	-62303.0	-29.0	0.0	10.0	31150.0
<i>commit_density</i>	-41.8	610.1	-43537.0	-8.0	0.0	1.0	34295.0
<i>add_frequency</i>	1.3	1.9	0.0	0.0	1.0	2.0	36.0
<i>bug_frequency</i>	0.2	0.7	0.0	0.0	0.0	0.0	13.0
<i>change_frequency</i>	0.3	0.8	0.0	0.0	0.0	0.0	13.0
<i>delete_frequency</i>	0.0	0.3	0.0	0.0	0.0	0.0	10.0
<i>document_frequency</i>	0.1	0.5	0.0	0.0	0.0	0.0	16.0
<i>feature_frequency</i>	0.1	0.5	0.0	0.0	0.0	0.0	18.0
<i>fix_frequency</i>	1.6	2.1	0.0	0.0	1.0	2.0	24.0
<i>improve_frequency</i>	0.1	0.4	0.0	0.0	0.0	0.0	11.0
<i>refactor_frequency</i>	0.1	0.4	0.0	0.0	0.0	0.0	13.0
<i>remove_frequency</i>	0.4	0.9	0.0	0.0	0.0	1.0	24.0
<i>update_frequency</i>	0.6	1.4	0.0	0.0	0.0	1.0	33.0
<i>use_frequency</i>	0.6	1.1	0.0	0.0	0.0	1.0	22.0
<i>messages_min</i>	28.8	24.4	0.0	15.0	23.0	37.0	400.0
<i>messages_max</i>	80.8	55.4	0.0	49.0	69.0	94.0	408.0
<i>messages_median</i>	46.9	25.0	0.0	35.0	45.0	53.0	400.0

### 5.3.2. Modelado y ajuste de hiperparámetros

Para este problema se van a emplear algoritmos de clasificación implementados en *Scikit-learn*, concretamente "*RandomForestClassifier*", "*GradientBoostingClassifier*" y "*ADABoost*". Además, debido al desbalanceo entre clases se implementarán algoritmos orientados a tratar con casos de desbalanceo de clases de la librería *Imbalanced-Learn* como "*BalancedRandomForest*", que usa "*Random Forest*" como algoritmo base, y "*RUSclassifier*" usando "*ADABoost*" como algoritmo base. Estos dos algoritmos están basados en los algoritmos "*RandomForestClassifier*" y "*ADABoost*" lo cual nos va a permitir comparar los resultados con los algoritmos que usan como base. En este caso también se buscará los mejores hiperparámetros para cada algoritmo utilizando la función "*GridSearchCV*".

**Tabla 14. Rango de hiperparámetros evaluado en cada modelo para la clasificación de conflictos**

Modelo	Rango de parámetros
<b>Random Forest Classifier</b>	'n_estimators': 10, 25, 50, 75, 100, 200 'min_samples_split': 2,3,5,10,20 'min_samples_leaf': 1,3,5,10,20 'class_weight': None, balanced 'max_depth': 1,3,5,10,20,30
<b>Balanced Random Forest</b>	'n_estimators': 10, 25, 50, 75, 100, 200 'min_samples_split': 2,3,5,10,20 'min_samples_leaf': 1,3,5,10,20 'class_weight': None, balanced 'max_depth': None,1,5,10,20,30
<b>Gradient Boosting Classifier</b>	'n_estimators': 10,25,50,75,100,200 'min_samples_split': 2,3,5,10,20 'min_samples_leaf': 1,3,5,10,20 'max_depth': 1,3,5,10,20,30
<b>ADABoost</b>	'n_estimators': 5, 10, 25, 50, 75, 100, 200 'learning_rate': 0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0, 3.0, 5.0
<b>RUSBoost</b>	'n_estimators': 5, 10, 25, 50, 75, 100, 200 'learning_rate': 0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0, 3.0, 5.0

### 5.3.3. Evaluación del modelo

Tras obtener las predicciones se calculan las métricas para evaluar el rendimiento del modelo, entre las que se encuentran aquellas métricas comunes en estudios de este tipo como *Accuracy*, *Precision*, *Recall* y *F1-score*, Y. Yang et al. [5] concluyen que *Precision*, *Recall* y *F1-score* son las métricas de rendimiento más utilizadas para evaluar la efectividad de los modelos predictivos en ingeniería de software. Además, son las métricas que los propios autores usan en su estudio [7]. Estas métricas se definen a continuación.

- **Accuracy:** mide la proporción de predicciones correctas en comparación con el total de predicciones. Es una métrica útil cuando las clases están balanceadas, es decir, cuando el número de muestras en cada clase es similar, por lo tanto para nuestro caso no es la mejor métrica para evaluar el rendimiento del modelo . Matemáticamente:

$$Accuracy = \frac{\text{Número de predicciones correctas}}{\text{Número total de predicciones}} = \frac{TP+TN}{TP+TN+FP+FN} \quad (6)$$

- **Recall:** mide la proporción de casos positivos que fueron correctamente identificados por el modelo. Esta métrica es útil cuando es importante detectar todos los casos positivos. Matemáticamente:

$$Recall = \frac{TP}{TP+FN} \quad (7)$$

- **Precision:** mide la proporción de casos positivos correctamente identificados entre todas las muestras clasificadas como positivas. Es útil cuando se desea evitar falsos positivos. Matemáticamente:

$$Precision = \frac{TP}{TP+FP} \quad (8)$$

- **F1-score:** es la media armónica de *precisión* y el *recall*, lo que significa que le da más peso a las métricas que tienen valores bajos, lo que lo hace útil para conjuntos de datos desbalanceados como en nuestro conjunto de datos. Es útil cuando se busca un equilibrio entre la detección de casos positivos y la minimización de falsos positivos y falsos negativos. Matemáticamente:

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (9)$$

Estas ecuaciones utilizan los acrónimos de verdadero positivo (TP, *True Positive*), verdadero negativo (TN, *True Negative*), falso positivo (FP, *False Positive*) y falso negativo (FN, *False Negative*) para expresar las métricas en función de los resultados de la clasificación. Además, de las métricas comentadas previamente el rendimiento de los modelos será ilustrado mediante la matriz de confusión permitiendo una mejor visualización de los resultados.

Los resultados obtenidos para cada conjunto de datos serán recopilados en tablas y gráficas. Esto permite una mejor comprensión y comparación de los resultados entre los distintos modelos y conjuntos de datos, así como para mostrar la variación del rendimiento en función de los diferentes modelos y métricas utilizadas.

## 6. Análisis de resultados

En esta sección se recopilan los resultados relevantes obtenidos siguiendo la metodología definida previamente para los problemas de estimación de esfuerzo y clasificación de conflictos en escenarios de fusión de código. La presentación de los resultados se acompaña con un análisis de los mismos para una mejor comprensión.

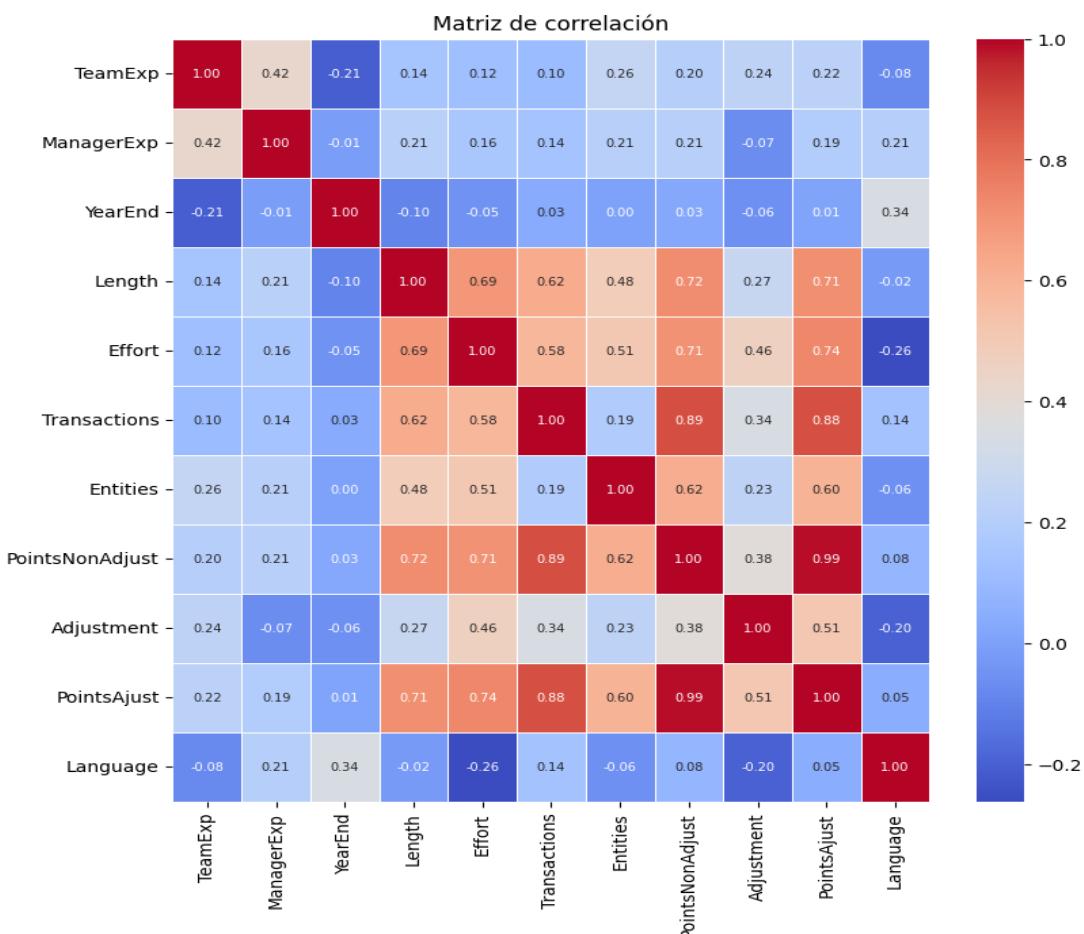
### 6.1. Estimación de esfuerzo

Recordemos que el problema de estimación de esfuerzo se ha abordado mediante algoritmos de ML de regresión. Para ofrecer una visión completa y profunda de todos los resultados obtenidos se usa la siguiente estructura: Comenzamos con un análisis de los conjuntos de datos empleados para comprender las relaciones entre características que componen cada conjunto de datos. Posteriormente

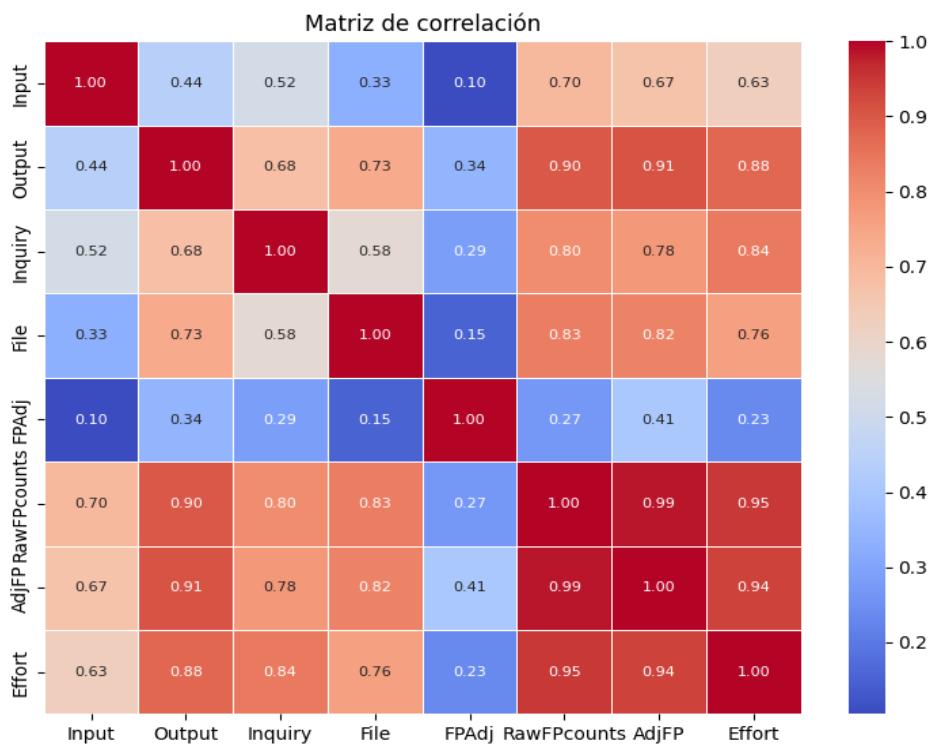
pasamos a la parte de evaluación del rendimiento obtenido por los modelos en cada conjunto de datos, incluyendo además los mejores hiperparámetros encontrados. Finalmente, nos centramos en los resultados de explicabilidad global y local en la estimación de esfuerzo.

### 6.1.1. Análisis de datos

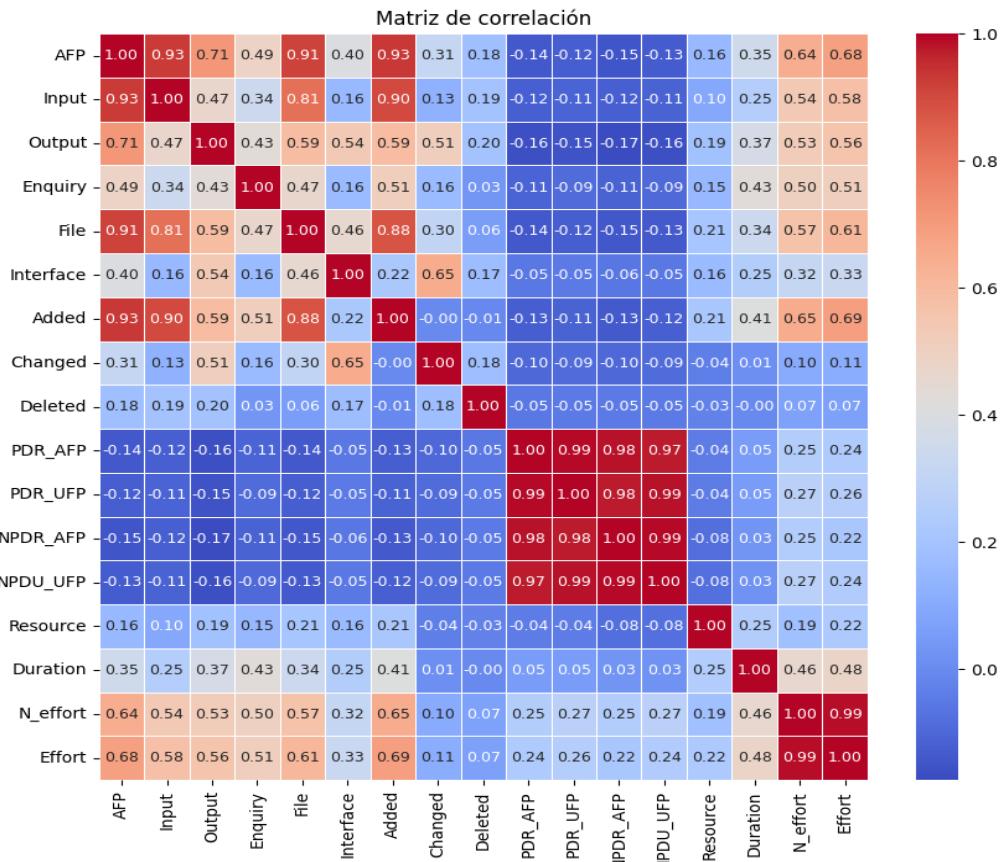
En primer lugar, se presentan los resultados de la correlación entre las características de los distintos conjuntos de datos empleados en la estimación de esfuerzo mediante representación de la matriz de correlación en las figuras desde la Fig. 2 a la Fig. 9 que se muestran a continuación. Como se ven en las figuras siguientes, los conjuntos de datos con mayor número de características correlacionadas son Miyazaki y Atkinson con 7 y 9 características correlacionadas, respectivamente. El resto de los conjuntos a pesar de que también presentan características correlacionadas son un mejor número. Por otro lado, el rango de correlación entre las características es muy amplio, llegando a valores de -0.88 como se ve en el conjunto Kitchenham hasta valores de 0.99 como por ejemplo en los conjuntos CHINA y Albretch.



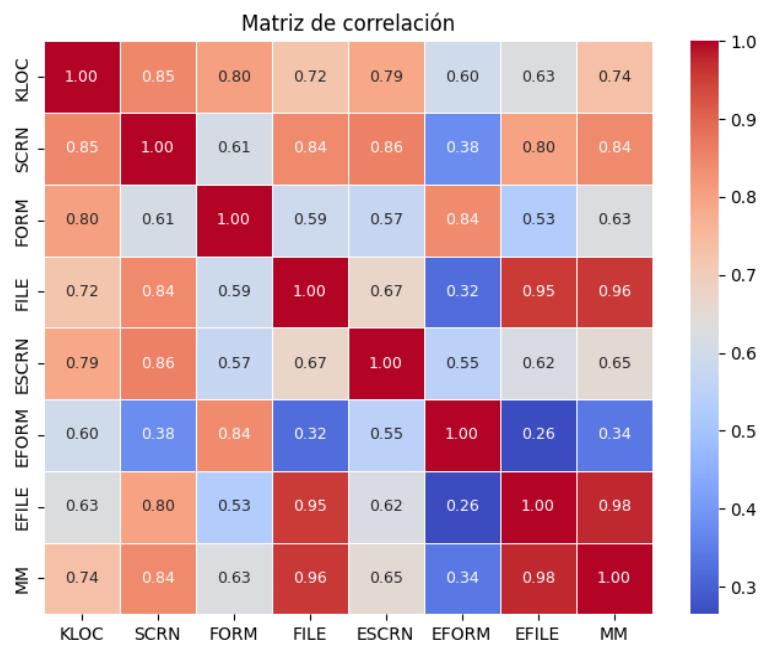
**Fig. 2. Matriz de correlación entre las características del conjunto de datos Desharnais.**



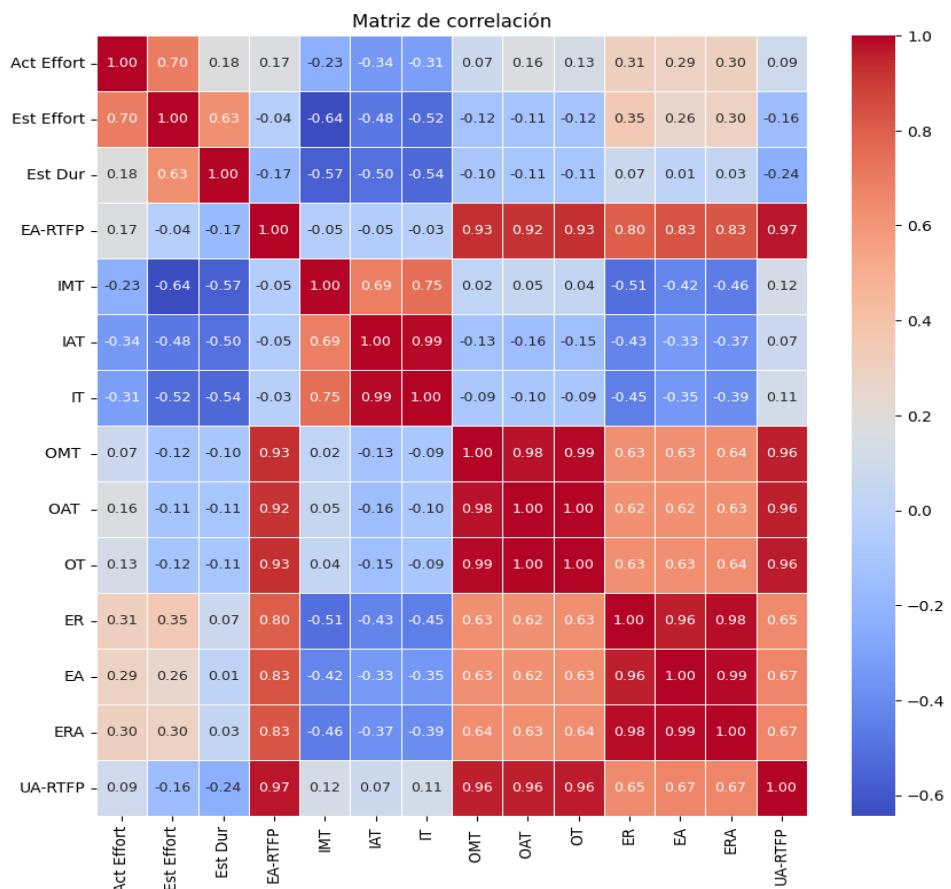
**Fig. 3. Matriz de correlación para las características del conjunto de datos Albretch.**



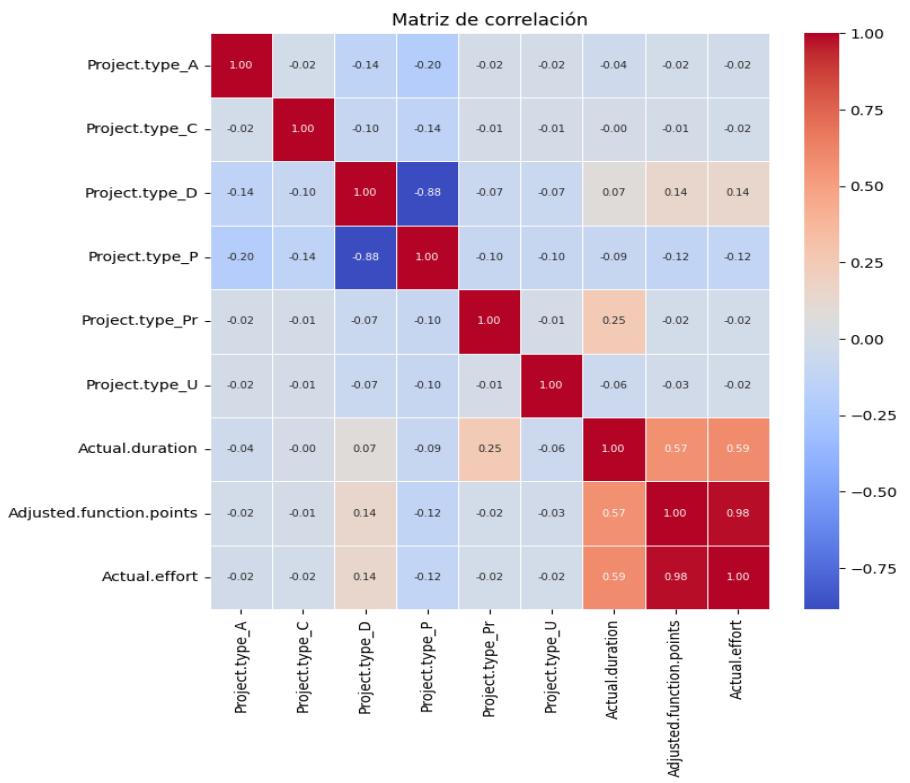
**Fig. 4. Matriz de correlación entre las características del conjunto de datos China.**



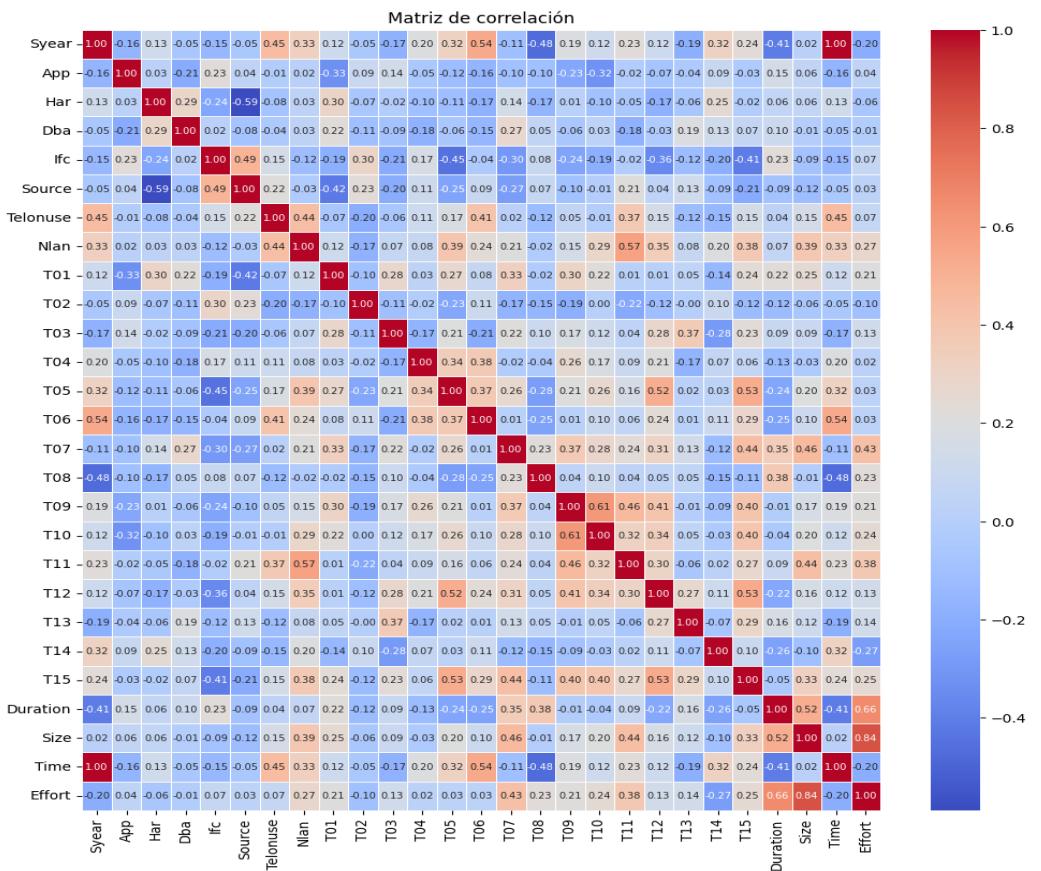
**Fig. 5. Matriz de correlación entre las características del conjunto de datos Miyazaki**



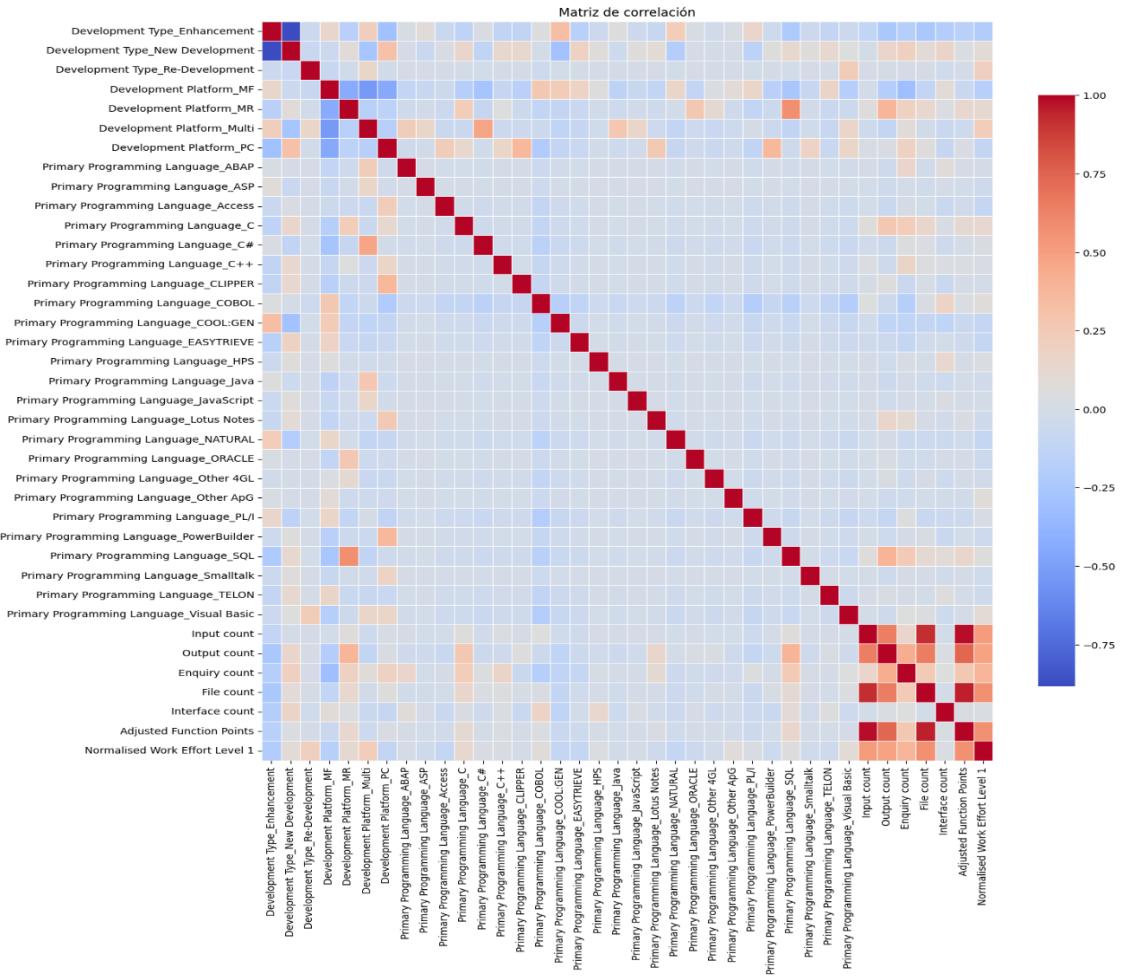
**Fig. 6. Matriz de correlación entre las características del conjunto de datos Atkinson.**



**Fig. 7. Matriz de correlación entre las características del conjunto de datos Kitchenham.**



**Fig. 8. Matriz de correlación entre las características del conjunto de datos Maxwell.**



**Fig. 9. Matriz de correlación entre las características del conjunto de datos ISBSG.**

### 6.1.2. Rendimiento predictivo

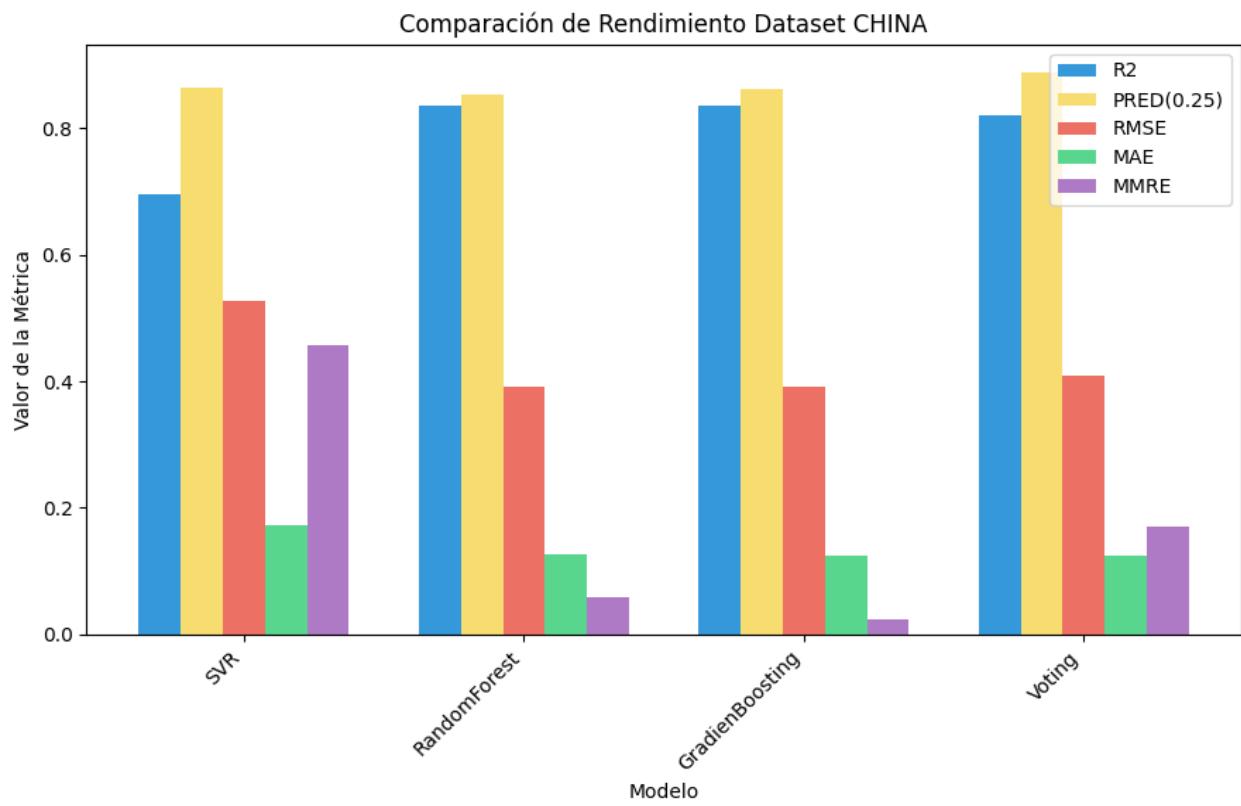
Antes de comenzar con los resultados y rendimiento de los modelos, es importante comentar los hiperparámetros empleados en cada modelo para obtener dicho resultados. En el caso de la estimación de esfuerzo los valores de hiperparámetros obtenidos por “*GridSearchCV*” se recogen en la Tabla 15. Recordemos que también se emplea el algoritmo de *ensemble “Voting”* sin embargo no se recoge en esta tabla ya que hace uso de los mismos algoritmos y configuraciones de hiperparámetros.

**Tabla 15. Valores de los hiperparámetros empleados para cada modelo y conjunto de datos, donde SVR, RFR y GBR hacen referencia a los algoritmos Support Vector Regression, Random Forest Regressor y Gradient Boosting Regressor, respectivamente.**

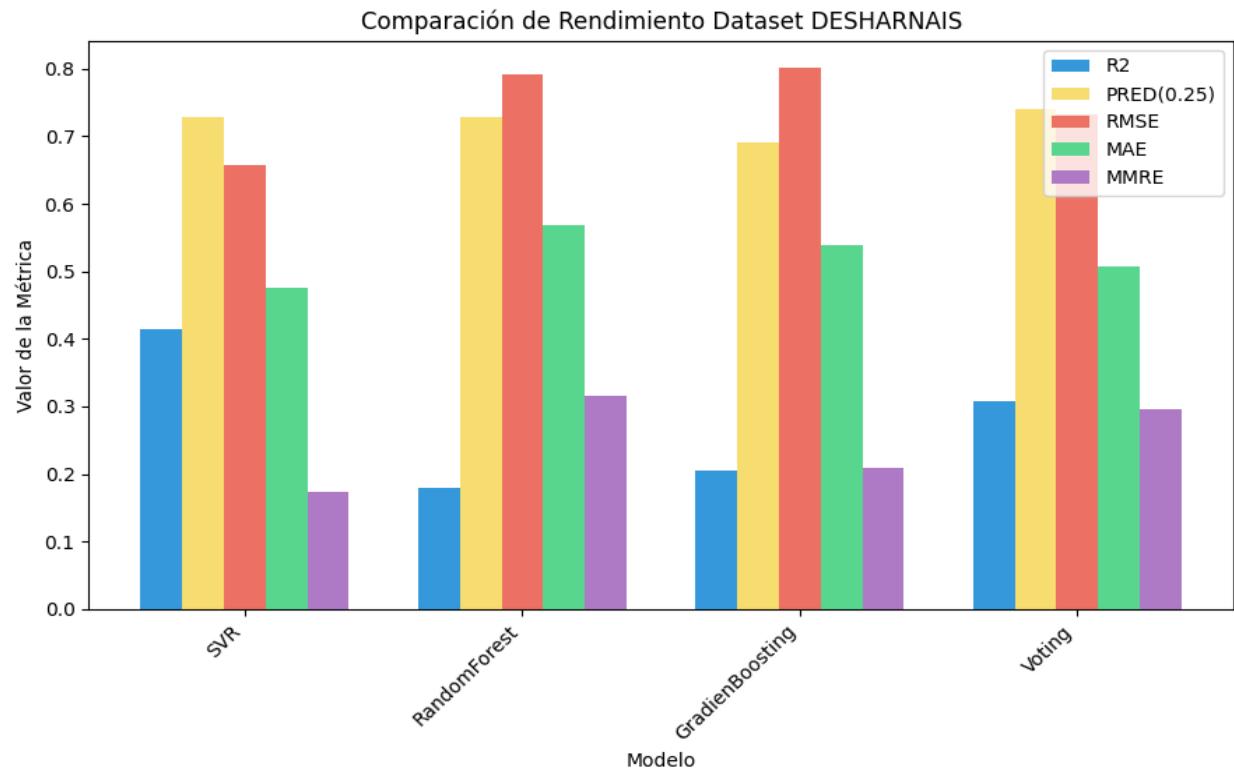
Conjunto	Modelo	Hiperparámetros
DESHARNAIS	<b>SVR</b>	kernel = “poly”, gamma = “scale”, degree = 1, C = 0.5
	<b>RFR</b>	Criterion = “absolute_error”, bootstrap = True, min_samples_leaf = 3, min_samples_split = 2, max_depth = 5, n_estimators = 200
	<b>GBR</b>	criterion = “friedman_mse”, loss = “absolute_error”, learning_rate = 0.3, min_samples_leaf = 1, min_samples_split = 4, max_depth = 3, n_estimators = 300
CHINA	<b>SVR</b>	kernel = “rbf”, gamma = “scale”, degree = 1, C = 7.5
	<b>RFR</b>	Criterion = “absolute_error”, bootstrap = True, min_samples_leaf = 1, min_samples_split = 2, max_depth = 10, n_estimators = 400

	<b>GBR</b>	criterion = "friedman_mse", loss = "absolute_error", learning_rate = 0.1, min_samples_leaf = 1, min_samples_split = 4, max_depth = None, n_estimators = 100
MAXWELL	<b>SVR</b>	kernel = "poly", gamma = "auto", degree = 1, C = 0.5
	<b>RFR</b>	Criterion = "squared_error", bootstrap = True, min_samples_leaf = 2, min_samples_split = 5, max_depth = None, n_estimators = 100
	<b>GBR</b>	criterion = "friedman_mse", loss = "absolute_error", learning_rate = 0.1, min_samples_leaf = 1, min_samples_split = 4, max_depth = 3, n_estimators = 300
MIYAZAKI	<b>SVR</b>	kernel = "rbf", gamma = "scale", degree = 1, C = 7.5
	<b>RFR</b>	Criterion = "squared_error", bootstrap = True, min_samples_leaf = 1, min_samples_split = 2, max_depth = 3, n_estimators = 100
	<b>GBR</b>	criterion = "friedman_mse", loss = "absolute_error", learning_rate = 0.1, min_samples_leaf = 1, min_samples_split = 2, max_depth = 3, n_estimators = 100
ALBRETTCH	<b>SVR</b>	kernel = "rbf", gamma = 0.1, degree = 1, C = 3.5
	<b>RFR</b>	Criterion = "absolute_error", bootstrap = True, min_samples_leaf = 1, min_samples_split = 2, max_depth = None, n_estimators = 400
	<b>GBR</b>	criterion = "squared_error", loss = "absolute_error", learning_rate = 0.5, min_samples_leaf = 1, min_samples_split = 2, max_depth = 1, n_estimators = 100
KITCHENHAM	<b>SVR</b>	kernel = "sigmoid", gamma = 0.1, degree = 1, C = 0.5
	<b>RFR</b>	Criterion = "absolute_error", bootstrap = True, min_samples_leaf = 1, min_samples_split = 2, max_depth = 10, n_estimators = 400
	<b>GBR</b>	criterion = "friedman_mse", loss = "absolute_error", learning_rate = 0.1, min_samples_leaf = 1, min_samples_split = 4, max_depth = None, n_estimators = 100
MAXWELL	<b>SVR</b>	kernel = "poly", gamma = "auto", degree = 1, C = 0.5
	<b>RFR</b>	Criterion = "absoluted_error", bootstrap = True, min_samples_leaf = 1, min_samples_split = 4, max_depth = 3, n_estimators = 400
	<b>GBR</b>	criterion = "squared_error", loss = "absolute_error", learning_rate = 0.3, min_samples_leaf = 5, min_samples_split = 2, max_depth = None, n_estimators = 100
ATKINSON	<b>SVR</b>	kernel = "rbf", gamma = "scale", degree = 1, C = 100.0
	<b>RFR</b>	Criterion = "absolute_error", bootstrap = False, min_samples_leaf = 3, min_samples_split = 2, max_depth = None, n_estimators = 200
	<b>GBR</b>	criterion = "squared_error", loss = "squared_error", learning_rate = 0.3, min_samples_leaf = 5, min_samples_split = 2, max_depth = 3, n_estimators = 500
ISBSG	<b>SVR</b>	kernel = "sigmoid", gamma = "auto", degree = 1, C = 9.5
	<b>RFR</b>	Criterion = "absolute_error", bootstrap = True, min_samples_leaf = 3, min_samples_split = 2, max_depth = None, n_estimators = 400
	<b>GBR</b>	criterion = "Friedman_mse", loss = "absolute_error", learning_rate = 0.1, min_samples_leaf = 1, min_samples_split = 2, max_depth = None, n_estimators = 300

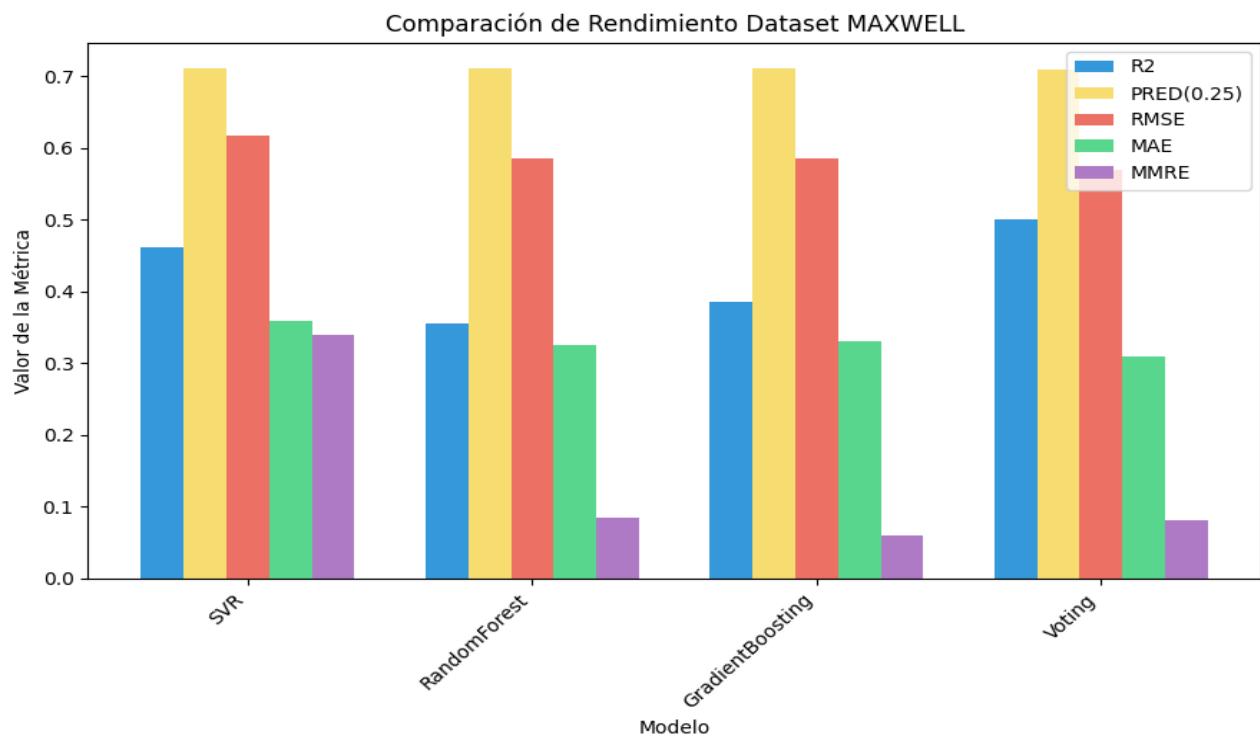
Teniendo en cuenta los hiperparámetros de la tabla anterior, desde la Fig. 10 a la Fig. 18 se muestran los resultados obtenidos en las métricas empleadas para evaluar el rendimiento en cada uno de los conjuntos de datos y modelos de ML para regresión usando esos hiperparámetros.



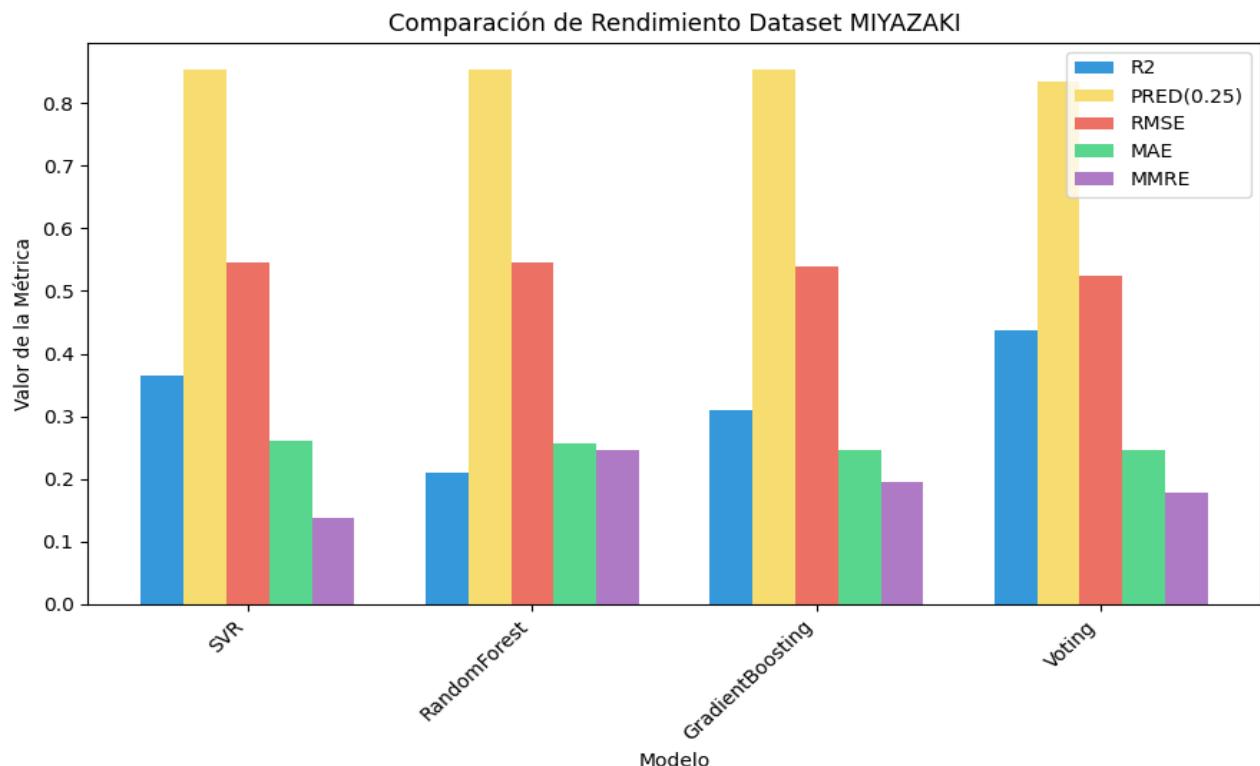
**Fig. 10.** Rendimiento en CHINA de los modelos empleados en regresión. Las métricas son; MAE (Error absoluto medio), RMSE (Raíz cuadrada del error cuadrático medio), R2 (Coeficiente de determinación), MMRE (Error relativo medio) y PRED(0.25) (Porcentaje de las predicciones dentro del 25%).



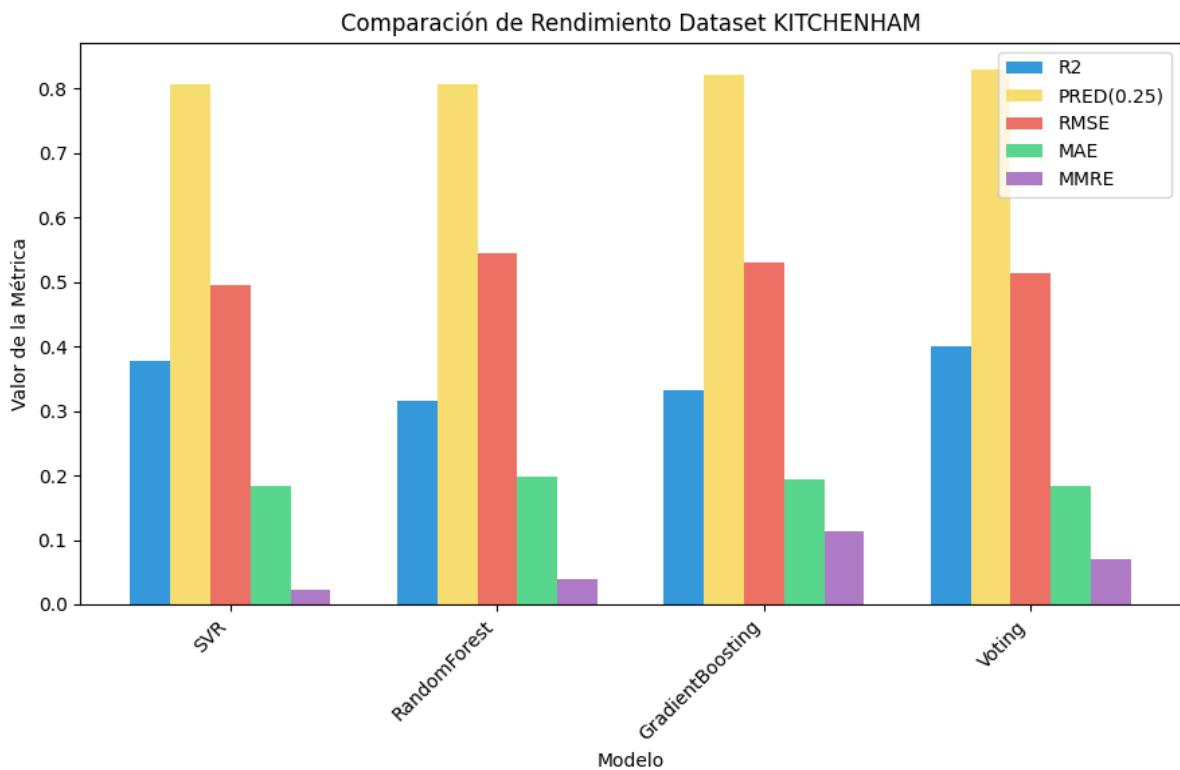
**Fig. 11.** Rendimiento en Desharnais de los modelos empleados en regresión.



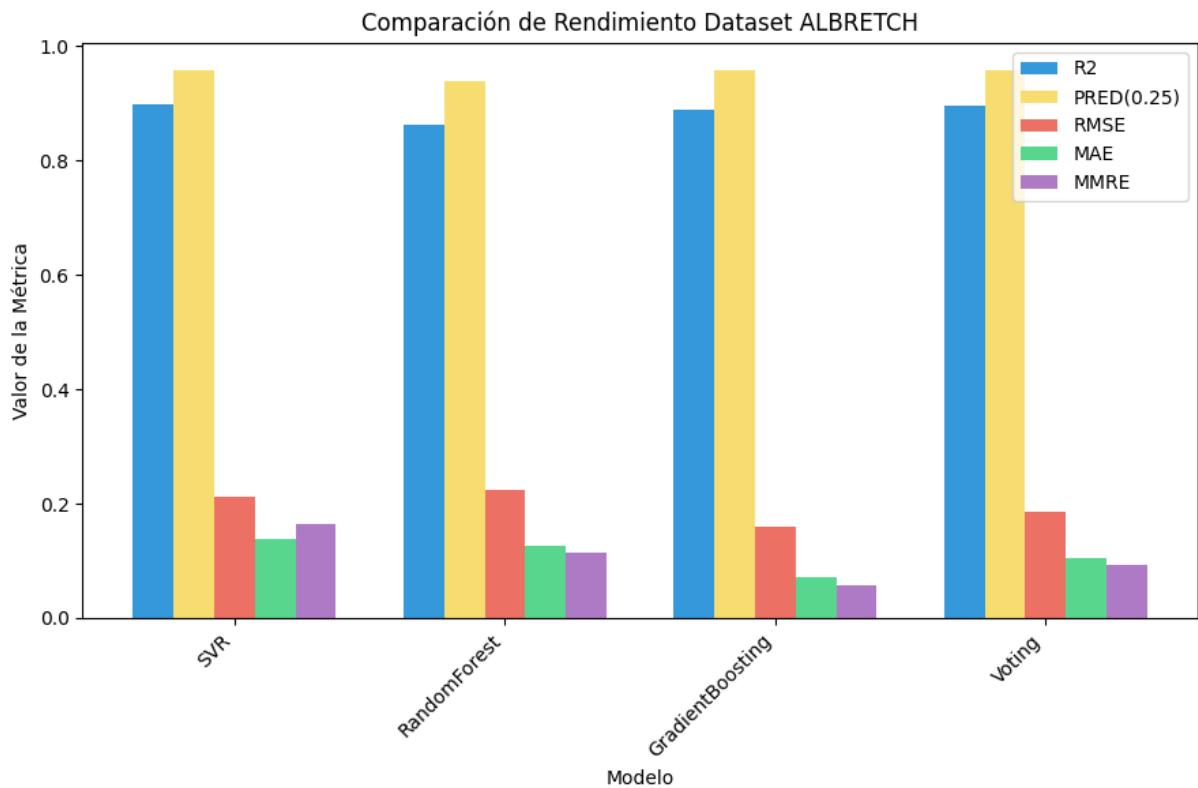
**Fig. 12.** Rendimiento en Maxwell de los modelos empleados en regresión.



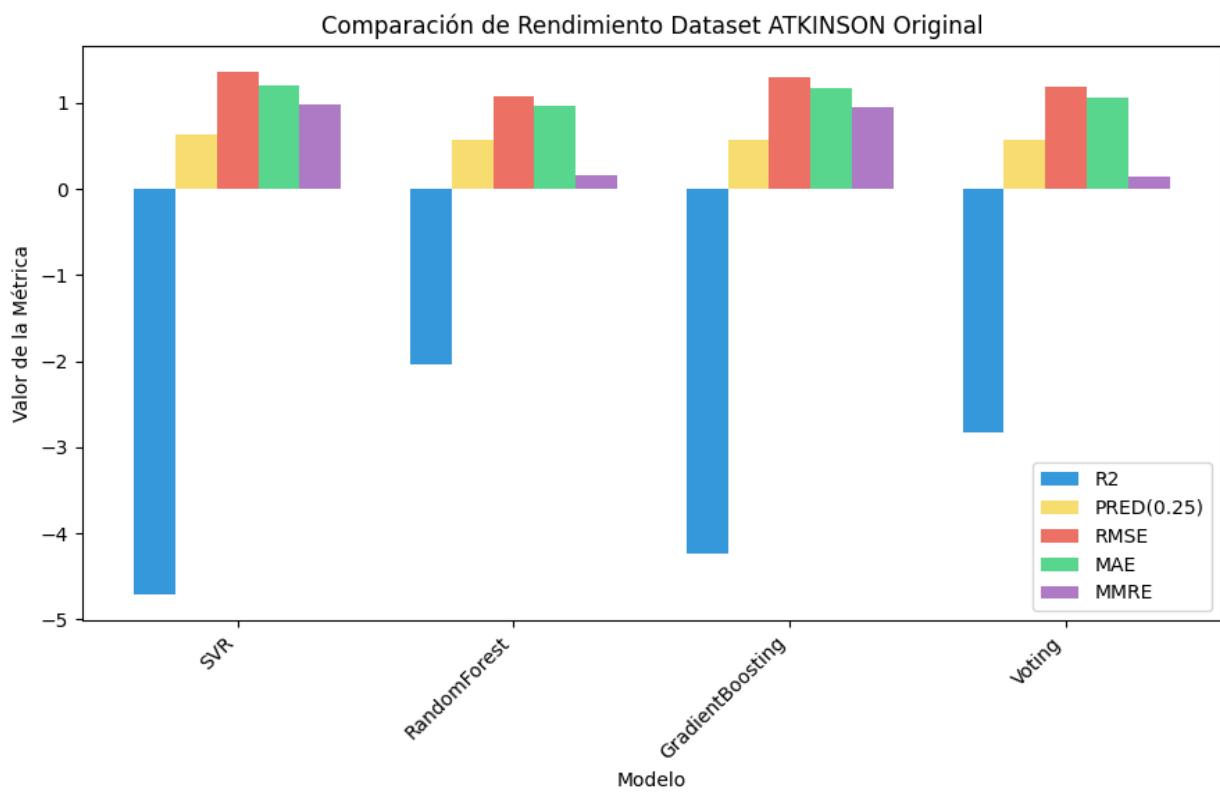
**Fig. 13.** Rendimiento en Miyazaki de los modelos empleados en regresión.



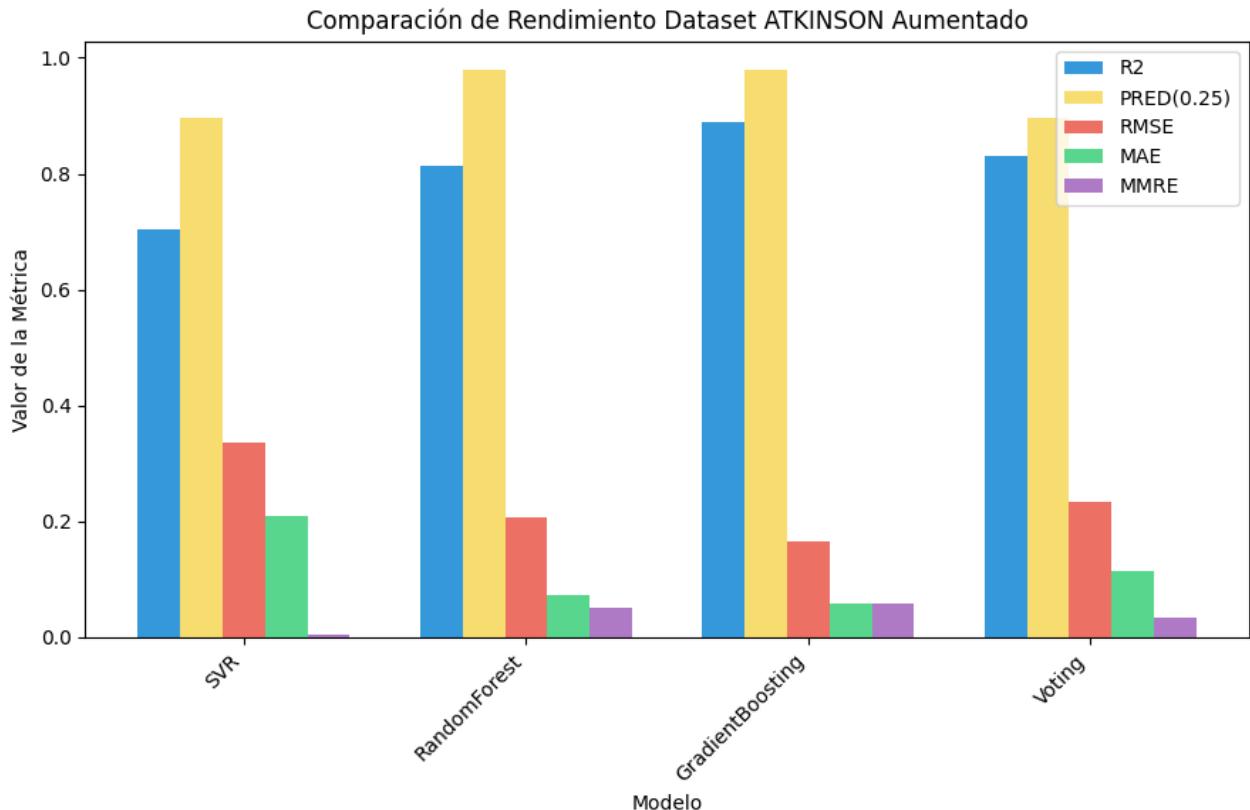
**Fig. 14.** Rendimiento en Kitchenham de los modelos empleados en regresión.



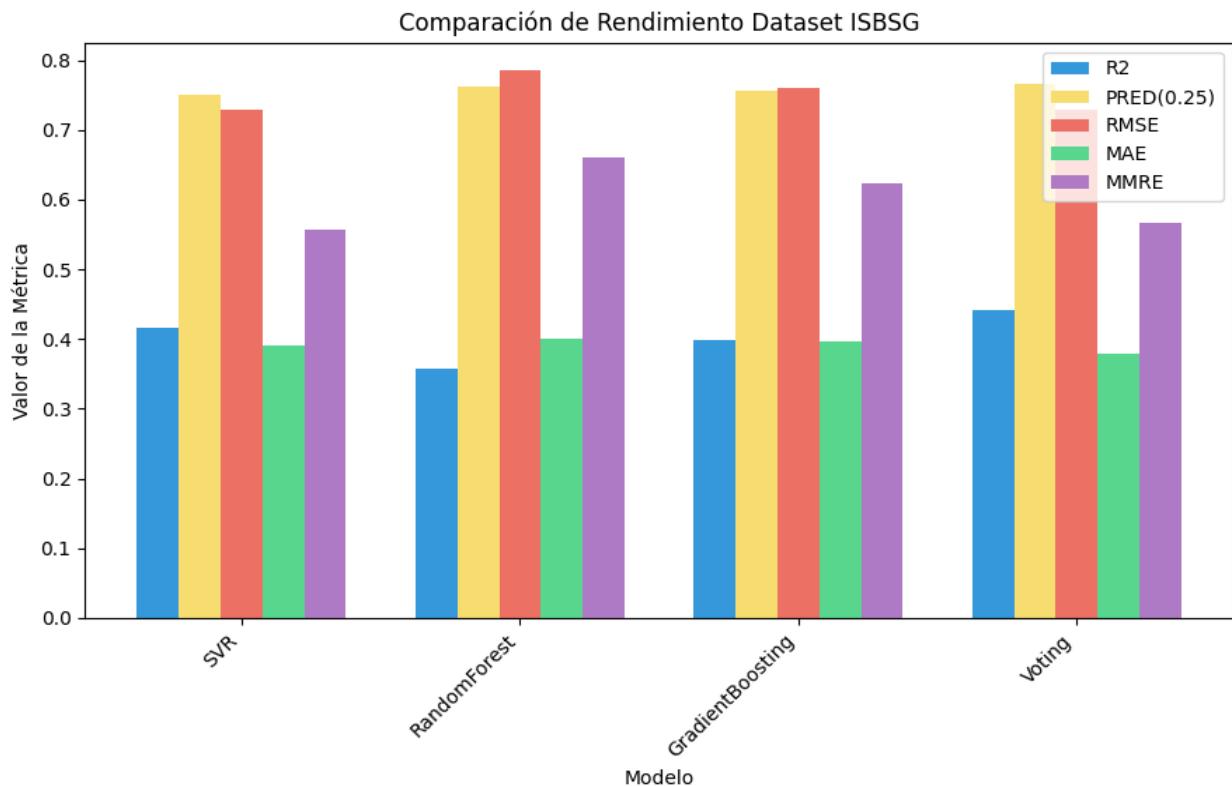
**Fig. 15.** Rendimiento en Albretch de los modelos empleados en regresión.



**Fig. 16.** Rendimiento en Atkinson original de los modelos empleados en regresión.



**Fig. 17.** Rendimiento en Atkinson aumentado de los modelos empleados en regresión.



**Fig. 18.** Rendimiento en ISBSG de los modelos empleados en regresión.

En general no se han obtenido buenos resultados en el rendimiento de los modelos, exceptuando los resultados del conjunto de datos CHINA (Fig. 10) y Albretch (Fig. 15) que han conseguido un coeficiente de determinación superior a 0.7 en todos los modelos del conjunto CHINA y superior a 0.8 para Albretch. Además, han obtenido valores bajos en MAE (error absoluto medio) y RMSE (error cuadrático medio). Por otro lado, el resto de los conjuntos tienen métricas bajas para el coeficiente de determinación siendo inferiores a 0.5 y valores altos en las métricas correspondientes a los errores. Un hecho remarcable es la mejora obtenida al aumentar el número de instancias del conjunto Atkinson como se puede ver en la Fig. 16 y Fig. 17.

El tamaño de los conjuntos de datos puede ser la causa de estos resultados. Por ejemplo, el conjunto CHINA tiene muchas más instancias que el resto de los conjuntos de datos (499 instancias), y ha conseguido un rendimiento más aceptable. Otra causa puede ser la baja correlación de algunas características en los conjuntos de datos con respecto a la variable objetivo, ya que Albretch ha obtenido mejores resultados a pesar de ser un conjunto de datos pequeño pero tiene características con mayor correlación que otros.

### 6.1.3. Explicabilidad del modelo

Para abordar el tema de la explicabilidad de los modelos para cada conjunto de datos se realiza desde el enfoque global y local, como ya se ha expuesto en la metodología. Es necesario comentar que para la evaluación de explicabilidad global a través de la importancia por permutación se ha llevado a cabo con respecto a tres de las métricas principales en el ámbito de la regresión: MAE, RMSE y  $R^2$ . Además, estas métricas han sido definidas en las ecuaciones (1), (2) y (3), y que están presentes en *Scikit-learn*. En cuanto a la explicabilidad local se han estudiado mediante técnicas *Break-down*, *Shapley*

*Values* y LIME en las instancias con estimación máxima, mínima y en la mediana. Debido a la cantidad de conjuntos de datos, modelos empleados e instancias estudiadas se ha obtenido una gran cantidad de resultados, los cuales pueden saturar en exceso esta sección. Por ello se han recopilado los resultados de cada conjunto de datos en tablas resumen que permitan una visualización adecuada para el análisis de los resultados, además se solo presentar aquellas gráficas que sean significativas para probar un hecho, evitando ser redundantes.

Para evitar saturar esta sección con demasiados elementos gráficos, en el Anexo 3 se puede consultar una colección parcial de los resultados originados en esta parte con el objetivo de complementar los que se presentan en esta sección y avalar los hallazgos. Los resultados de explicabilidad en esta sección serán presentados y analizados de forma individual para cada conjunto de datos, al igual que los resultados para cada modelo son independientes.

- **Desharnais**

Los resultados de explicabilidad global para este conjunto de datos se muestran en la Fig. 19, Fig. 20, Fig. 21, Fig. 22 las cuales corresponden a los modelos de SVR, *Random Forest*, *Gradiente Boosting* y *Voting* respectivamente.

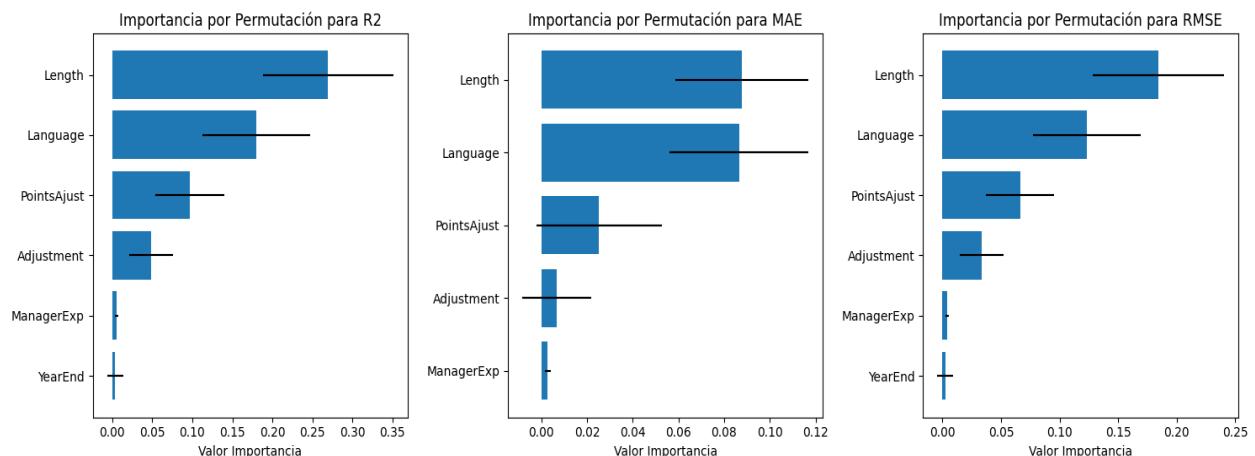


Fig. 19. Importancia por permutación Desharnais + Support Vector Regression.

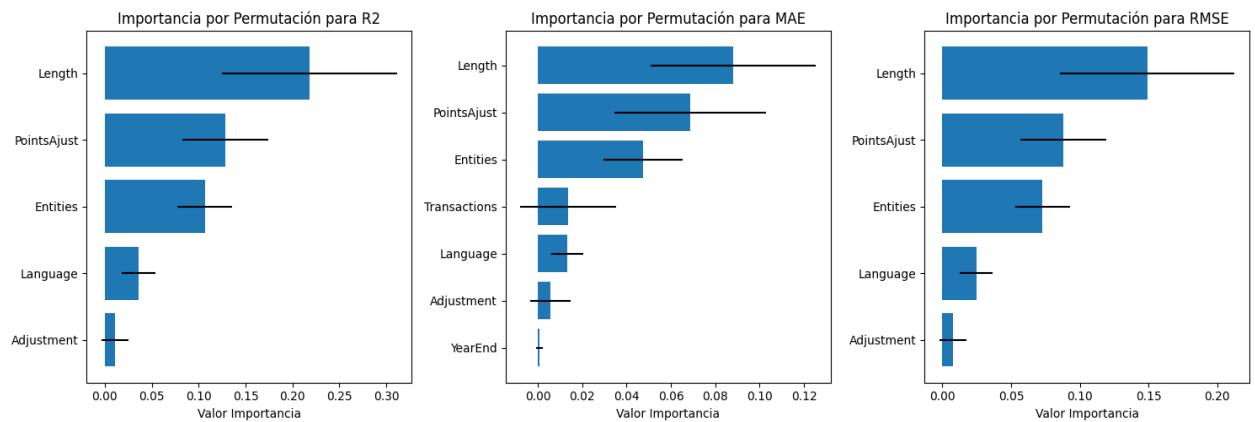
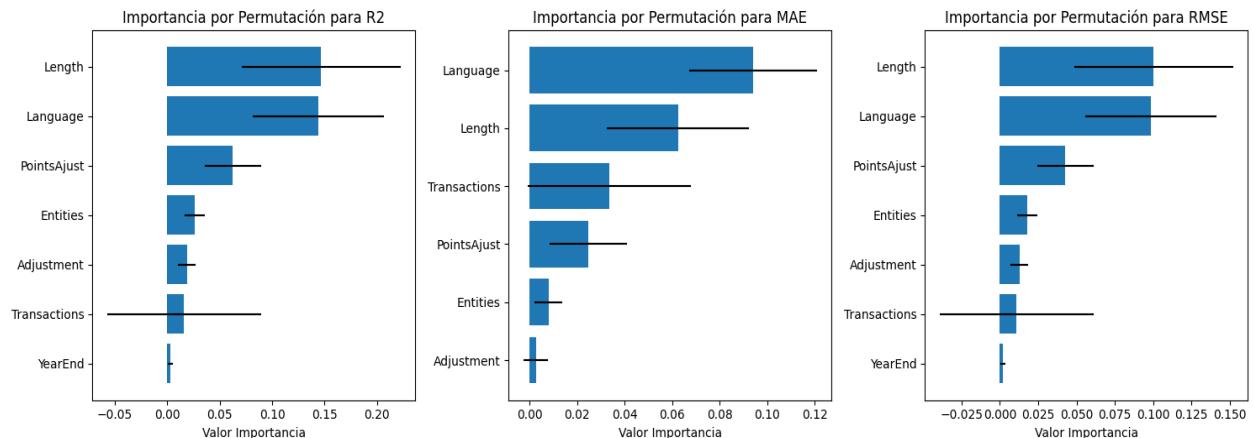
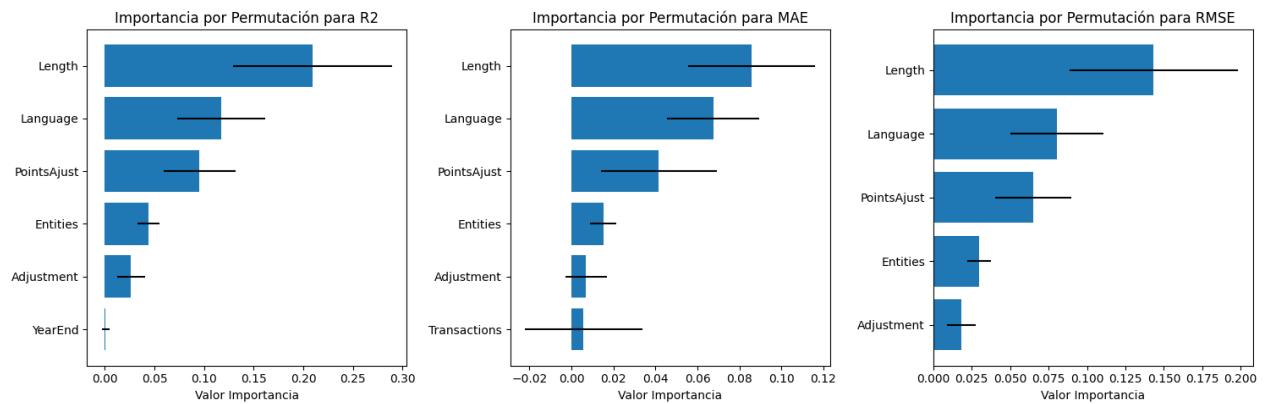


Fig. 20. Importancia por permutación Desharnais + Random Forest Regressor



**Fig. 21. Importancia por permutación Desharnais + Gradient Boosting Regressor.**



**Fig. 22. Importancia por permutación Desharnais + Voting**

Como se ha podido observar en estas figuras para un mismo modelo las características más importantes para cada métrica según el método de permutación suelen coincidir tanto en cuáles son las características más influyentes como en el orden de las mismas. Podemos ver en las cuatro figuras anteriores como las tres primeras características son las mismas en la mayoría de los casos, o solo han intercambiado un puesto entre ellas mismas.

De la misma forma, si realizamos una comparación a nivel de modelo también se aprecia una coincidencia en las características más importantes. Esto significa que el método funciona adecuadamente y con coherencia no solo dentro de un modelo sino también entre los modelos. Por lo tanto, podemos decir que en el conjunto de datos Desharnais las características “length” es la más importante, seguida por “language” y “PointsAjust”.

Tras los resultados de explicabilidad local, en la Tabla 16, se pueden comprobar que características tienen mayor influencia en cada modelo así como el número de veces que aparece esa característica en una de las técnicas de explicabilidad. Vemos como la característica “length”, “PointsAjust” y “Transactions” tienen un ranking alto en general y un número de apariciones alto. Esto concuerda con los resultados de explicabilidad global de este conjunto de datos, excepto “Transactions” que no aparecía con valores altos para la importancia global. Además, el modelo SVR da mayor importancia a

“language” mientras en los resultados de importancia por permutación era considerada menos relevante que “length”. Pero en general todos los resultados de explicabilidad local en este conjunto siguen el comportamiento visto en el caso global.

**Tabla 16. Resultados explicabilidad local Desharnais, donde SVR, RFR, GBR y VOT hacen referencia a los algoritmos Support Vector Regression, Random Forest Regressor, Gradient Boosting Regressor y Voting, respectivamente.**

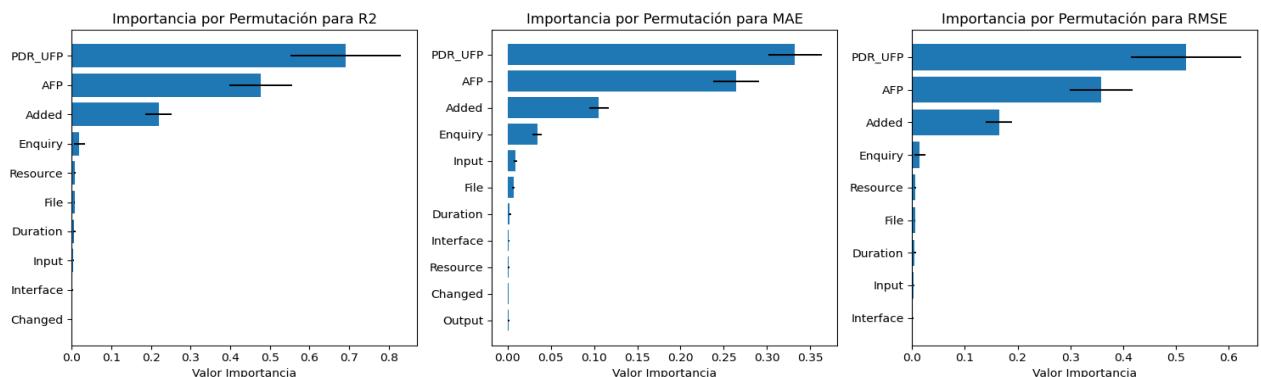
Característica	General		SVR		RF		GBR		VOT	
	Rank Final	Conteo Final	Rank	Conteo	Rank	Conteo	Rank	Conteo	Rank	Conteo
Length	1	35	2	8	1	9	2	9	1	9
PointsAjust	2	36	3	9	2	9	4	9	3	9
Transactions	3	32	4	6	4	8	1	9	2	9
Language	4	24	1	9	-	-	3	9	4	6
Entities	5	14	-	-	3	9	6	5	-	-
Adjustment	5	20	5	5	5	5	5	4	5	6
TeamExp	7	11	6	4	7	1	-	-	6	6
ManagerExp	8	4	-	-	6	4	-	-	-	-
YearEnd	9	4	7	4	-	-	-	-	-	-

- **CHINA**

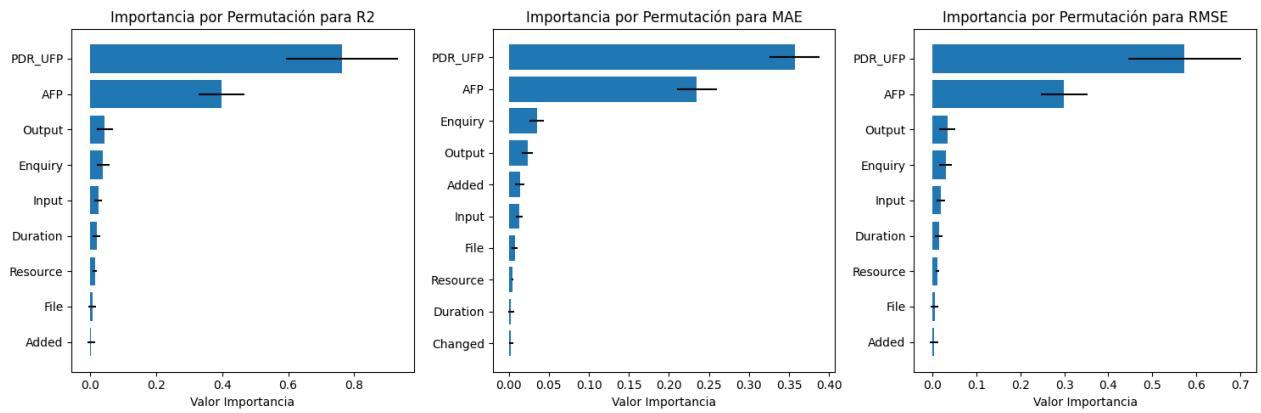
El siguiente conjunto para analizar su explicabilidad global es CHINA. Los resultados obtenidos se exponen en las Fig. 23, Fig. 24, Fig. 25 y Fig. 26.



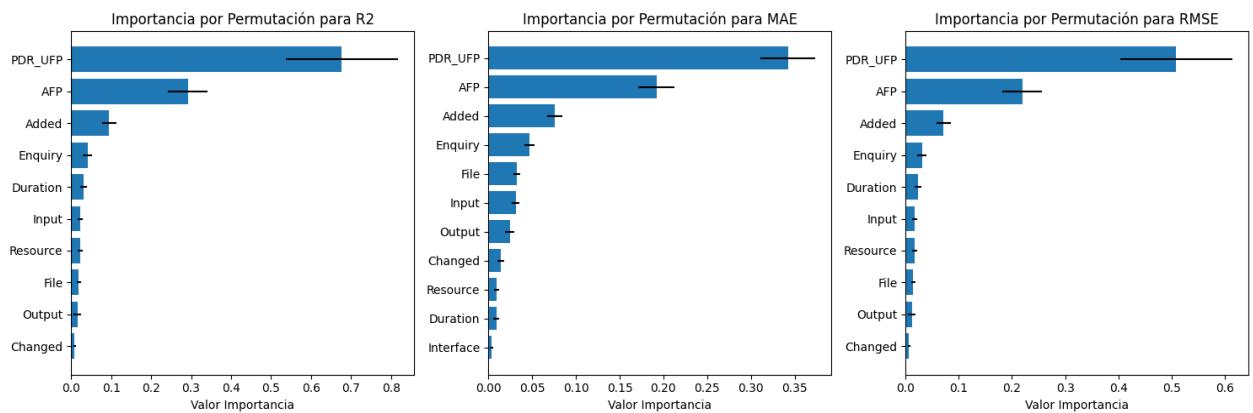
**Fig. 23. Importancia por permutación CHINA + SVR.**



**Fig. 24. Importancia por permutación CHINA + Random Forest Regressor**



**Fig. 25. Importancia por permutación CHINA + Gradient Boosting Regressor.**



**Fig. 26. Importancia por permutación CHINA + Voting**

Se aprecia una alta coincidencia en las características más importantes, tanto a nivel del mismo modelo como entre modelos. También se aprecia como los modelos basados en árboles, como *Random Forest* y *Gradient Boosting* en este caso, consiguen centrar la importancia en tres características y reducir la importancia del resto a valores muy bajos, hecho que no se observa en SVR. Sin embargo, es común en todo los modelos que las característica “PDR\_UFP” se desmarca claramente como la más importante.

De nuevo, los resultados de explicabilidad local de la Tabla 17 son coherente con los resultados en explicabilidad global, ya que las características “PDR\_UFP”, “AFP” y “Added” predominan en la parte alta del ranking y suelen aparecer en la mayoría de los resultados proporcionados por las técnicas. Además, como se aprecia en la Fig. 23, el modelo SVR da poca importancia a “AFP” mientras que el resto le daban un valor alto. Este hecho se refleja igualmente en explicabilidad local, donde “AFP” en SVR tiene rango bajo mientras que en el resto de los modelos es alto.

**Tabla 17. Resultados explicabilidad local CHINA.**

Característica	General		SVR		RF		GBR		VOT	
	Rank Final	Conteo Final	Rank	Conteo	Rank	Conteo	Rank	Conteo	Rank	Conteo
PDR_UFP	1	34	1	8	2	9	2	9	2	8
AFP	2	31	8	4	1	9	1	9	1	9
Added	3	31	2	8	3	9	5	5	3	9
Output	4	18	3	4	5	3	3	7	4	4
Input	5	18	9	4	4	5	4	4	5	5
Enquiry	6	11	5	4	7	2	7	3	5	2

<b>Duration</b>	7	9	7	2	6	3	7	1	7	3
<b>File</b>	8	10	10	4	9	2	6	3	9	1
<b>Resource</b>	9	1	6	1	-	-	-	-	-	-
<b>Changed</b>	10	6	11	1	7	2	-	-	7	3
<b>Deleted</b>	11	4	3	2	10	1	-	-	9	1
<b>Interface</b>	12	6	12	2	-	-	7	4	-	-

- **Maxwell**

La Fig. 27, Fig. 28, Fig. 29 y Fig. 30 contienen la importancia global del conjunto de datos Maxwell. Las características “Size” y “duration” son las más importantes para todos los modelos con amplia diferencia. De nuevo coinciden características y orden de las mismas en la mayoría de los casos, excepto en *Gradient Boosting* que atribuye más importancia a “duration” mientras que el resto de los modelos a “Size”. De nuevo, *Random Forest* es especialmente bueno para reducir la importancia de la mayoría de las características.

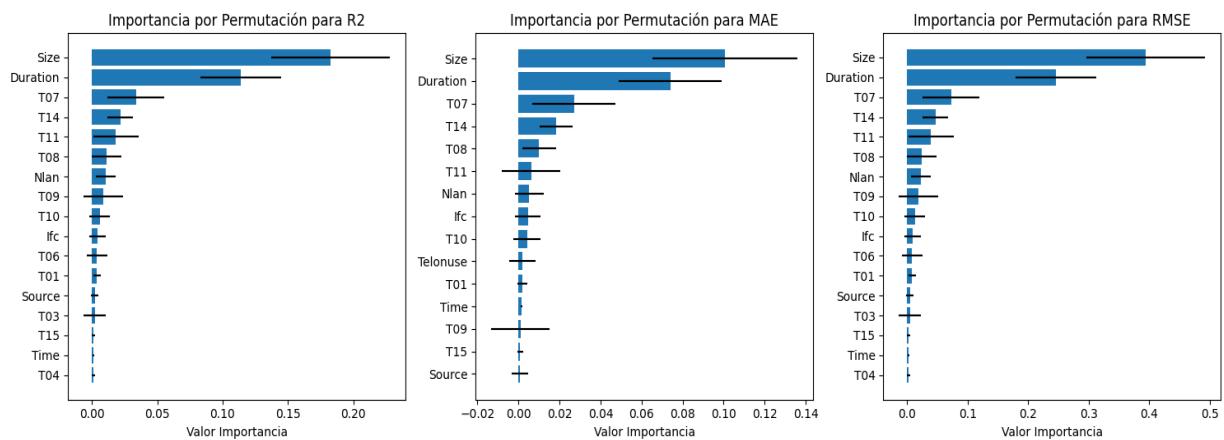


Fig. 27. Importancia por permutación Maxwell + SVR.

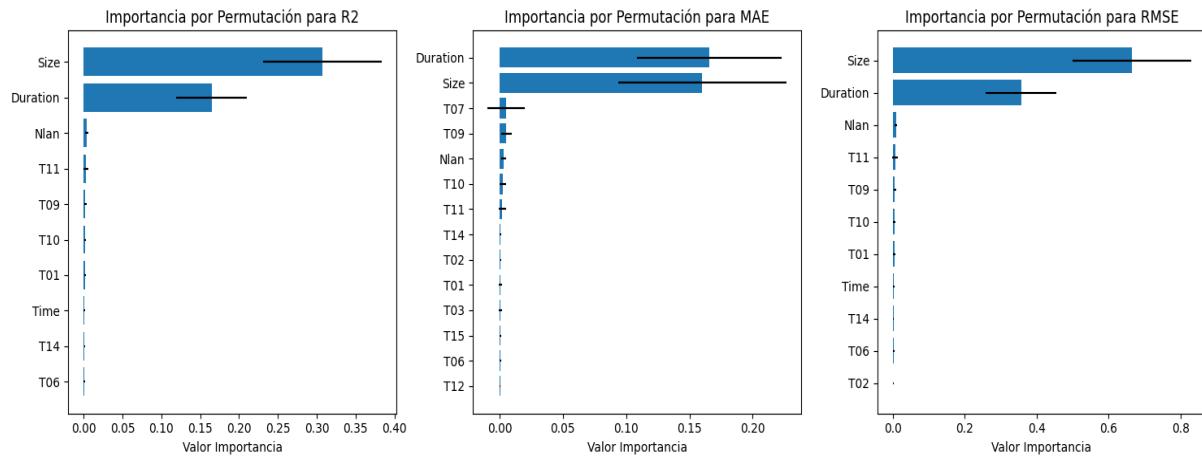
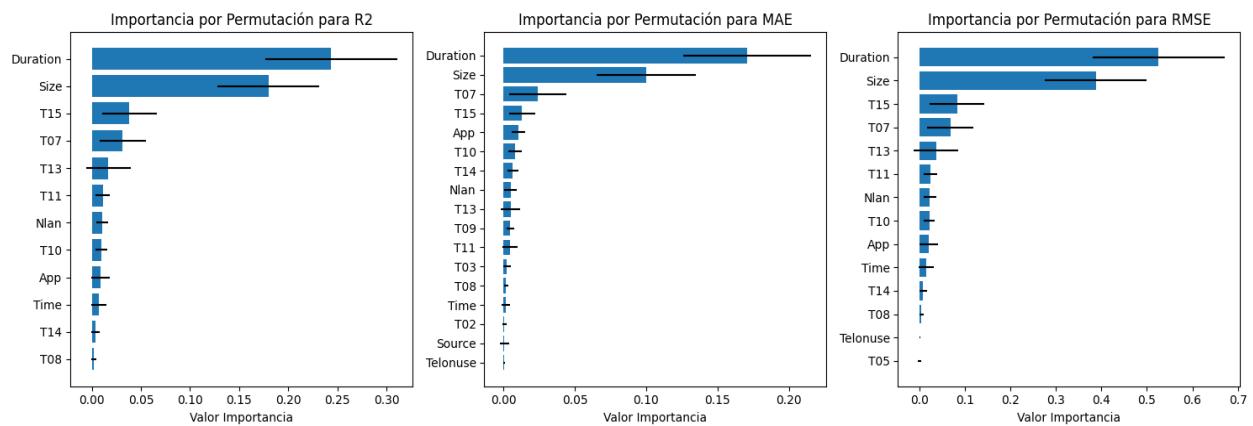
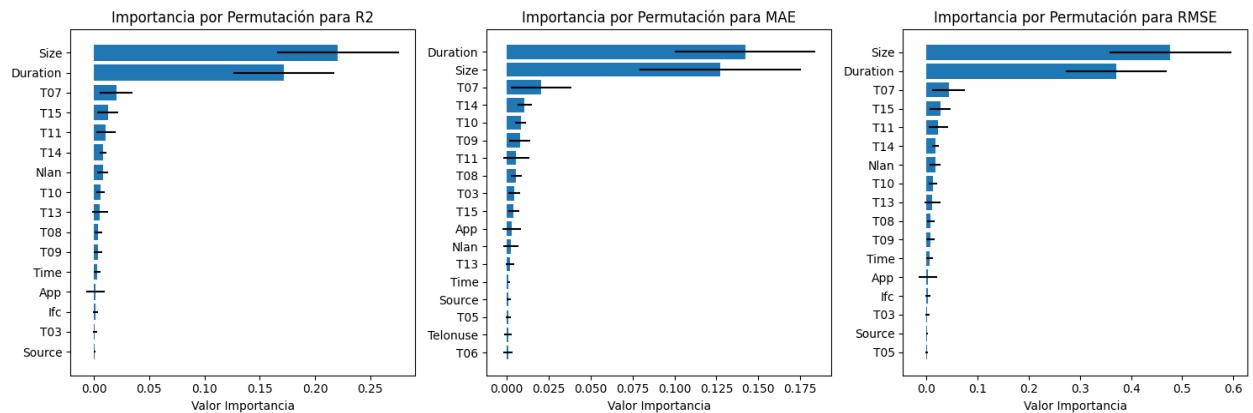


Fig. 28. Importancia por permutación Maxwell + Random Forest Regressor.



**Fig. 29. Importancia por permutación Maxwell + Gradient Boosting Regressor.**



**Fig. 30. Importancia por permutación Maxwell + Voting**

Para las técnicas de explicabilidad local del conjunto Maxwell (Tabla 18), la característica “Duration” es la que aporta una mayor contribución en general, seguida por “T07” y “Size”. Estos resultados concuerdan con los obtenidos en explicabilidad global, donde estas mismas características ocupaban los primeros puestos con diferencia respecto al resto. Además, al igual que los resultados de importancia por permutación se aprecia un gran número de características que tienen influencia.

**Tabla 18. Resultados explicabilidad local Maxwell.**

Característica	General		SVR		RF		GBR		VOT	
	Rank Final	Conteo Final	Rank	Conteo	Rank	Conteo	Rank	Conteo	Rank	Conteo
Duration	1	35	2	8	1	9	1	9	1	9
T07	2	36	1	9	3	9	3	9	2	9
Size	3	29	2	6	2	8	2	9	3	6
T09	4	14	4	5	4	3	-	-	5	6
T11	5	15	7	3	5	5	4	4	6	3
App	6	9	7	3	-	-	-	-	4	6
Nlan	7	8	6	2	6	4	8	2	-	-
T15	8	6	-	-	7	1	6	3	7	2
T05	9	4	-	-	7	1	5	3	-	-
T08	10	7	5	6	-	-	-	-	9	1
T10	11	3	-	-	7	1	7	2	-	-
T13	12	4	-	-	11	1	8	2	9	1
T03	13	1	-	-	7	1	-	-	-	-

<b>Har</b>	14	2	-	-	-	-	-	-	8	2
<b>T06</b>	14	2	-	-	-	-	-	8	2	-
<b>T14</b>	16	2	9	2	-	-	-	-	-	-
<b>Dba</b>	17	1	-	-	11	1	-	-	-	-

- **Miyazaki**

Las característica “KLOC” del conjunto de datos Miyazaki es la característica más importante según la permutación de todos los modelos. En este caso mientras que para *Random Forest* y *Gradient Boosting* “EFILE” y “ESCRN” son las dos siguientes características en el ranking de importancia, para SVR “EFILE” no tiene importancia, siendo una importante discrepancia a nivel de modelos como se puede ver en la Fig. 31, Fig. 32, Fig. 33 y Fig. 34.

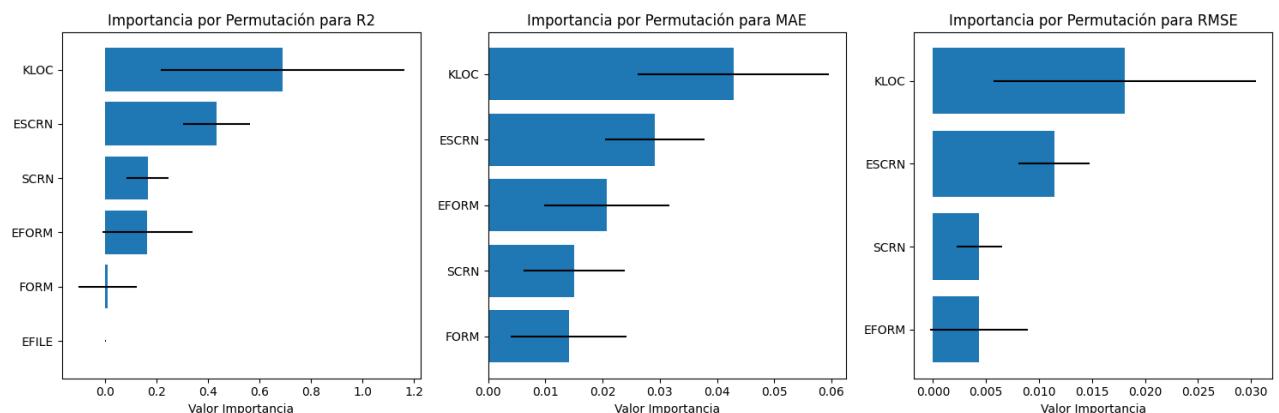


Fig. 31. Importancia por permutación Miyazaki + SVR.

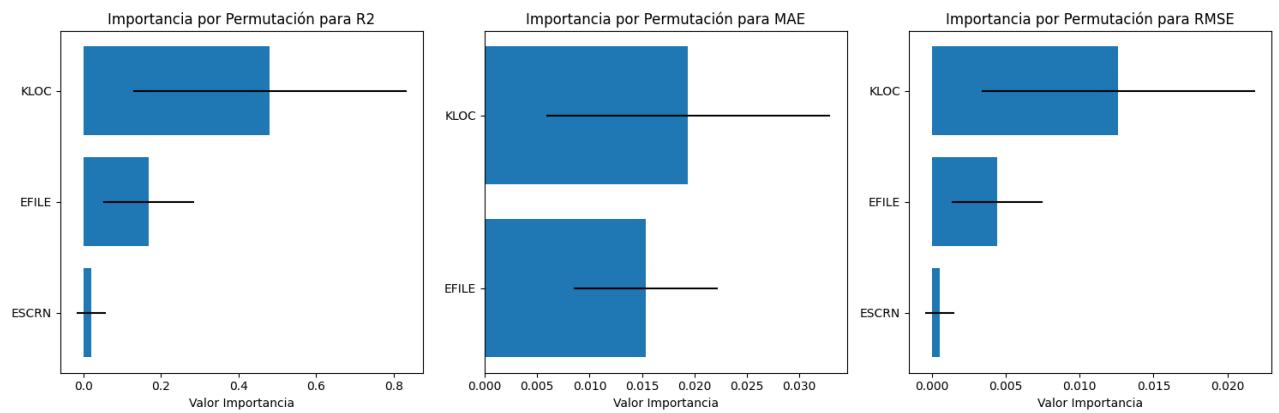
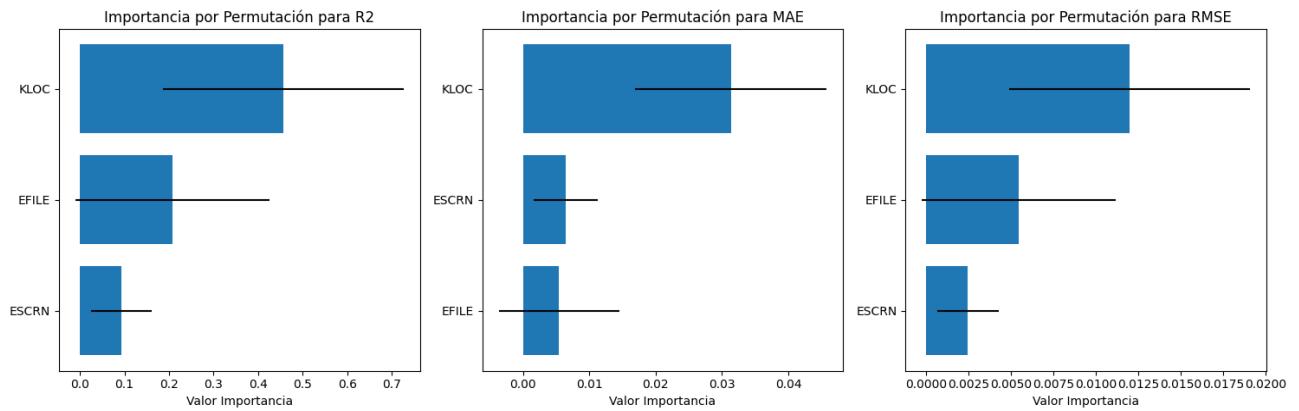
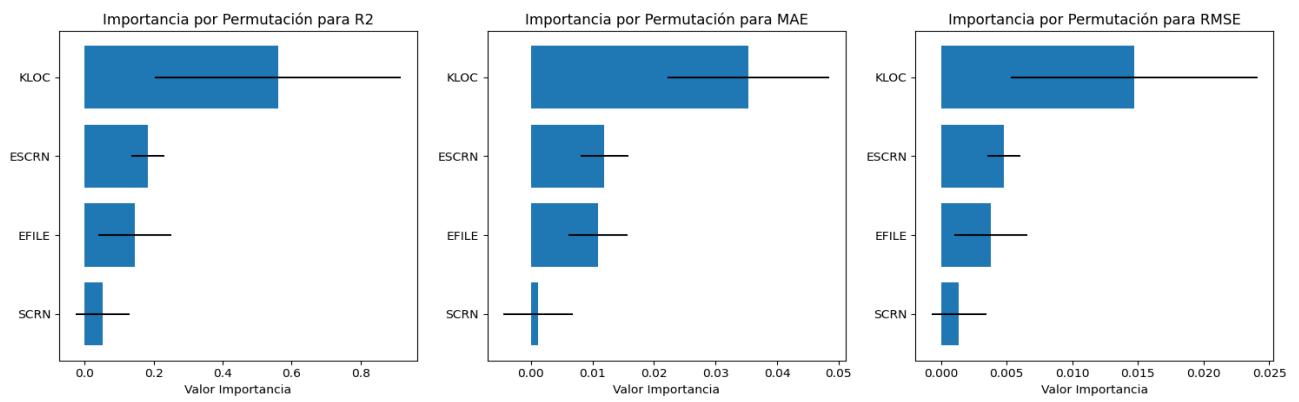


Fig. 32. Importancia por permutación Miyazaki + Random Forest Regressor.



**Fig. 33. Importancia por permutación Miyazaki + Gradient Boosting Regressor.**



**Fig. 34. Importancia por permutación Miyazaki + Voting**

Los resultados de explicabilidad local confirman que “KLOC” es la característica que contribuye de más de forma unánime para todos los modelos. Al igual que pasaba en explicabilidad global “EFILE” tiene rango algo en los modelos basados en árboles pero no es así para SVR. Por otro lado, “SCRN” y “FORM” son las siguientes en el orden de contribución y en número de apariciones en los resultados locales de forma general como se ve en la Tabla 19.

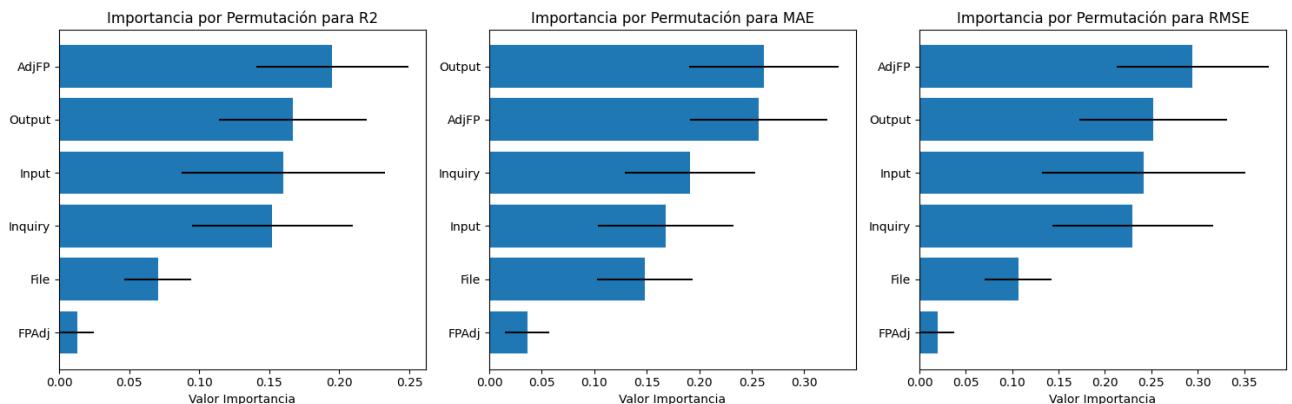
**Tabla 19. Resultados explicabilidad local Miyazaki.**

Característica	General		SVR		RF		GBR		VOT	
	Rank Final	Conteo Final	Rank	Conteo	Rank	Conteo	Rank	Conteo	Rank	Conteo
<b>KLOC</b>	1	36	1	9	1	9	1	9	1	9
<b>SCRN</b>	2	35	2	9	4	9	3	8	2	9
<b>FORM</b>	3	34	3	9	3	7	3	9	3	9
<b>EFILE</b>	4	29	5	4	2	9	2	9	4	7
<b>EFORM</b>	5	21	4	8	6	2	5	7	6	4
<b>ESCRN</b>	6	25	6	6	5	9	6	3	5	7

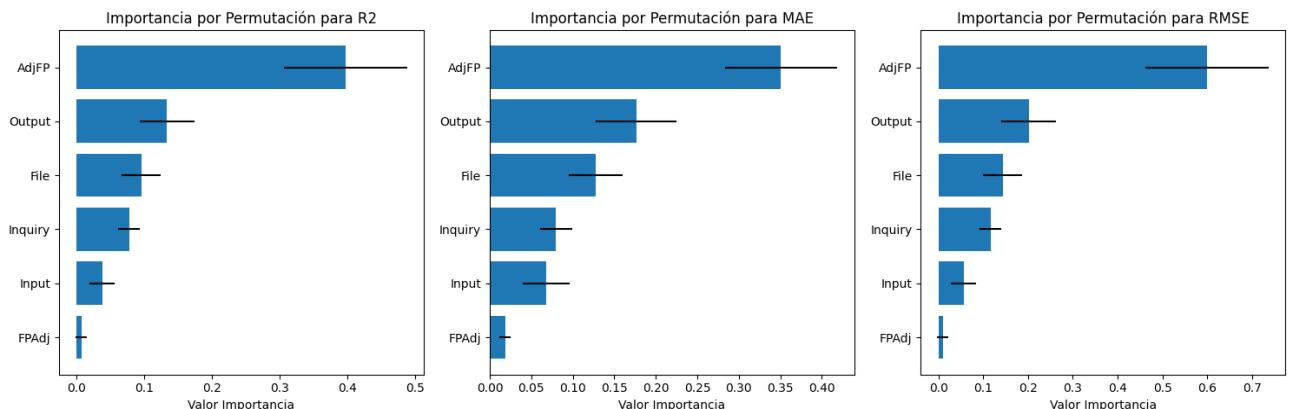
#### • Albretch

En el conjunto Albretch los resultados de explicabilidad global suelen coincidir entre modelos siendo “Output” y “AdjFP” las características más influyentes. El único hecho destacable es que SVR atribuye un valor de importancia similar a varias características mientras que en Random Forest destaca

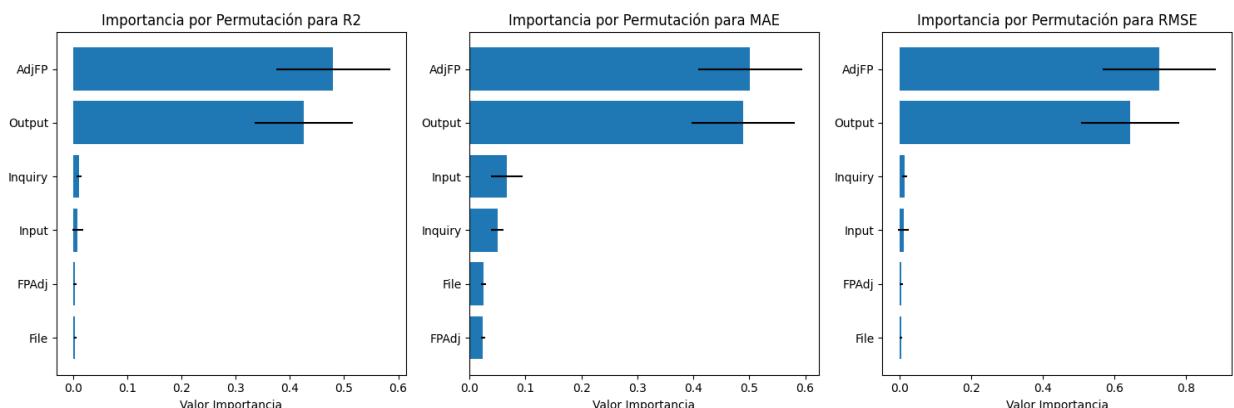
únicamente “AdjFP” y en *Gradient Boosting* destacan las dos características mencionadas según los datos recogidos en la Fig. 35, Fig. 36, Fig. 37 y Fig. 38.



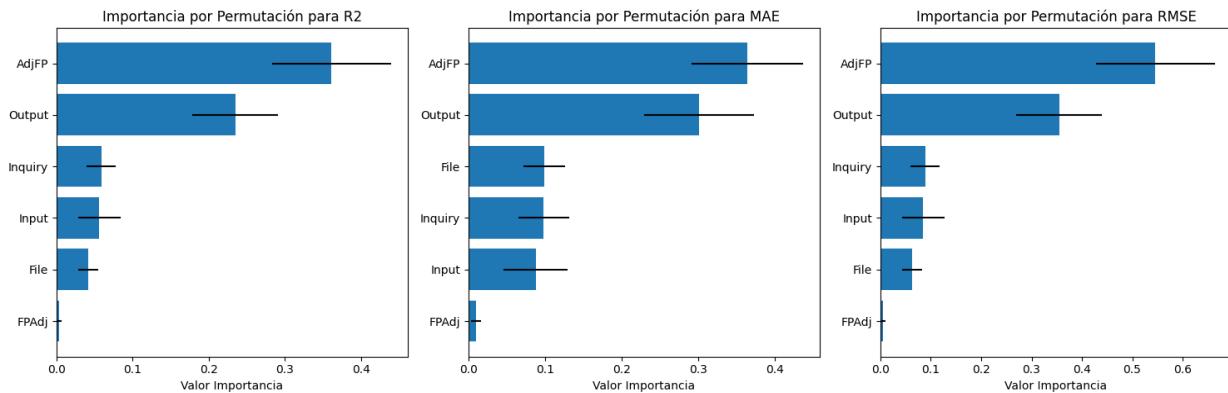
**Fig. 35. Importancia por permutación Albretch + SVR.**



**Fig. 36. Importancia por permutación Albretch + Random Forest Regressor**



**Fig. 37. Importancia por permutación Albretch + Gradient Boosting Regressor.**



**Fig. 38. Importancia por permutación Albretch + Voting.**

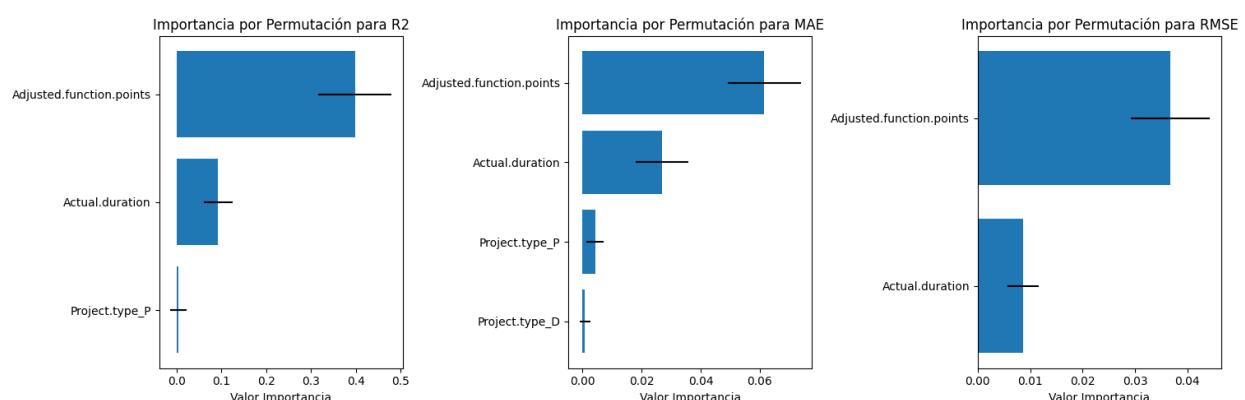
El orden de las características más influyentes para Albretch es “Output”, “AdjFP” y “Input” según el resumen de la Tabla 20. Estos resultados concuerdan con los obtenidos en explicabilidad global al igual que en los conjuntos anteriores.

**Tabla 20. Resultados explicabilidad local Albretch.**

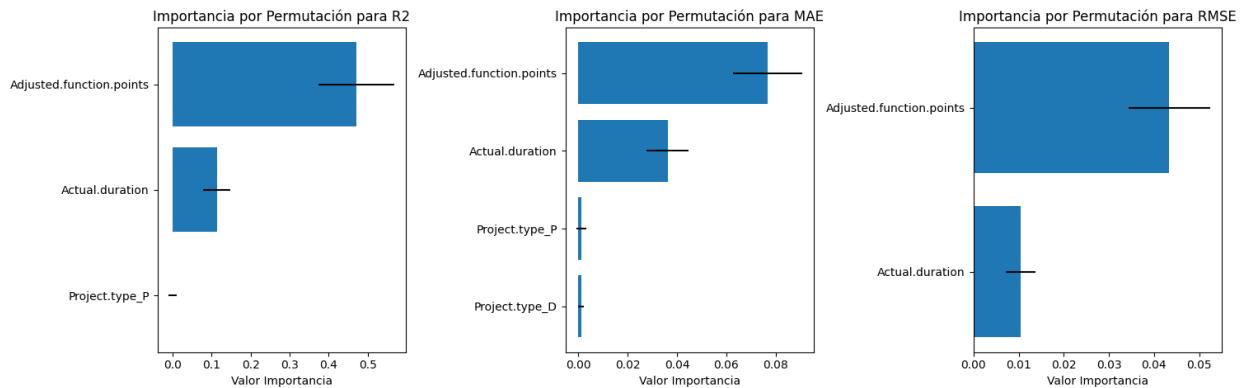
Característica	General		SVR		RF		GBR		VOT	
	Rank Final	Conteo Final	Rank	Conteo	Rank	Conteo	Rank	Conteo	Rank	Conteo
<b>Output</b>	1	36	1	9	2	9	1	9	1	9
<b>AdjFP</b>	2	36	3	9	1	9	2	9	2	9
<b>Inquiry</b>	3	35	2	9	5	8	4	9	3	9
<b>File</b>	4	35	5	9	4	9	5	8	5	9
<b>Input</b>	5	33	4	6	3	9	3	9	4	9
<b>FPAdj</b>	6	5	6	3	6	1	6	1	-	-

- **Kitchenham**

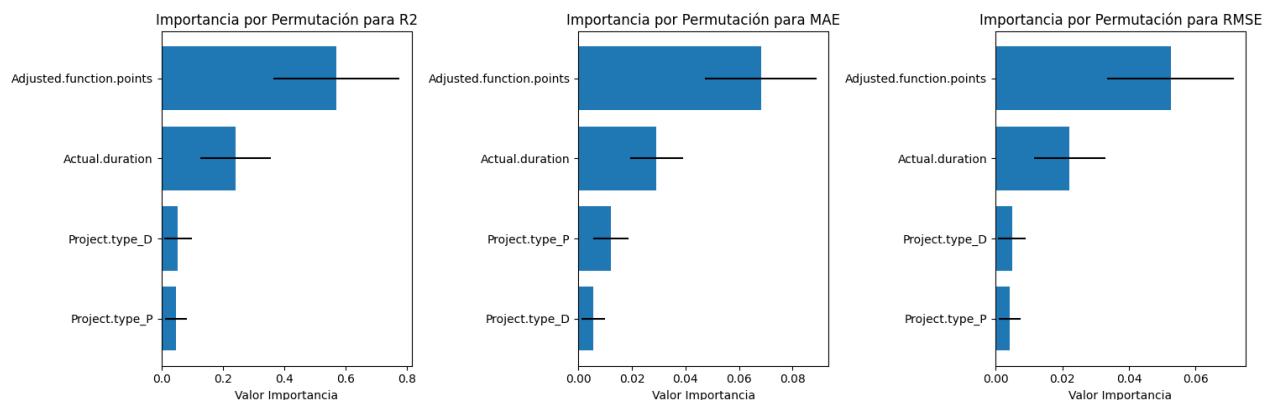
En las representaciones de la importancia por permutación del conjunto Kitchenham, Fig. 39, Fig. 40, Fig. 41 y Fig. 42, se observa el mismo comportamiento que en el resto de conjuntos de datos, prácticamente coincidencia total en las dos primeras características más importantes, que son “Adjusted.function.points” y “Actual.duration”, a nivel interno del modelo y entre modelos.



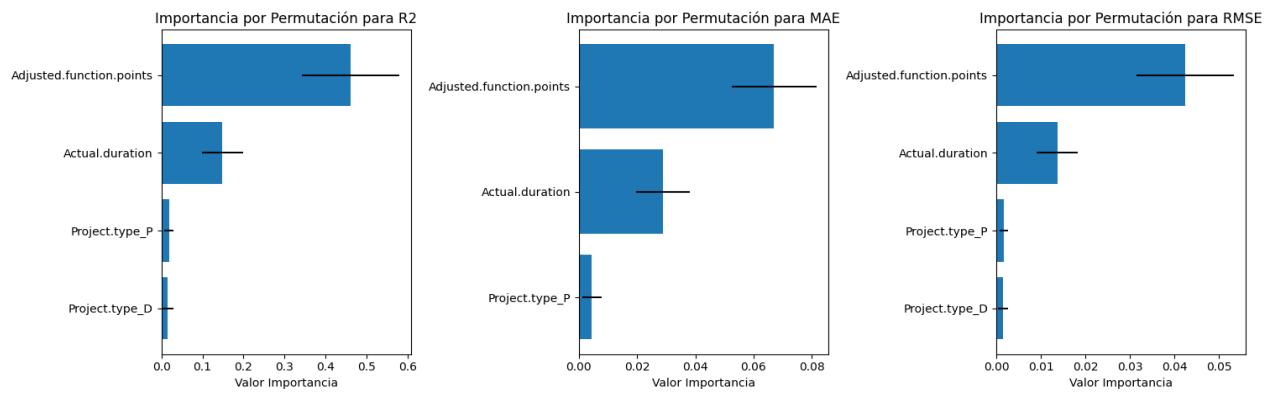
**Fig. 39. Importancia por permutación Kitchenham + SVR**



**Fig. 40. Importancia por permutación Kitchenham + Random Forest Regressor**



**Fig. 41. Importancia por permutación Kitchenham + Gradient Boosting Regressor.**



**Fig. 42. Importancia por permutación Kitchenham + Voting**

La Tabla 21 contiene los resultados de explicabilidad local para el conjunto Kitchenham. Se puede ver que muestran que “*Adjusted.function.points*” es la característica que más ha contribuido en todas las técnicas, seguida por “*Actual.duration*”. Este hecho coincide con lo esperado teniendo en cuenta la explicabilidad global.

Tabla 21. Resultados explicabilidad local Kitchenham.

Característica	General		SVR		RF		GBR		VOT	
	Rank Final	Conteo Final	Rank	Conteo	Rank	Conteo	Rank	Conteo	Rank	Conteo
Adjusted.function.points	1	36	1	9	1	9	1	9	1	9
Actual.duration	2	34	3	8	2	9	2	9	2	8
Project.type_P	3	34	2	9	3	9	3	9	4	7
Project.type_D	4	29	6	6	4	6	4	8	3	9
Project.type_A	5	19	7	1	6	6	6	6	6	6
Project.type_Pr	6	12	4	3	4	3	4	3	5	3
Project.type_C	7	14	5	8	7	3	-	-	7	3
Project.type_U	8	1	7	1	-	-	-	-	-	-

- **Atkinson**

Los valores de importancia por permutación para el Atkinson (Fig. 43, Fig. 44, Fig. 45 y Fig. 46) destacan las características “Est Dur” como la más importante de forma general para todos los modelos con bastante diferencia. En el segundo lugar hay diferencias entre los tres modelos, mientras en SVR se da más importancia a “IMT”, Random Forest tiene a “EA-RTFP” como segunda y para Gradient Boosting es “IAT”. Sin embargo, para los tres modelos tanto “IMT” como “IAT” están entre las tres primeras en prácticamente todos resultados.

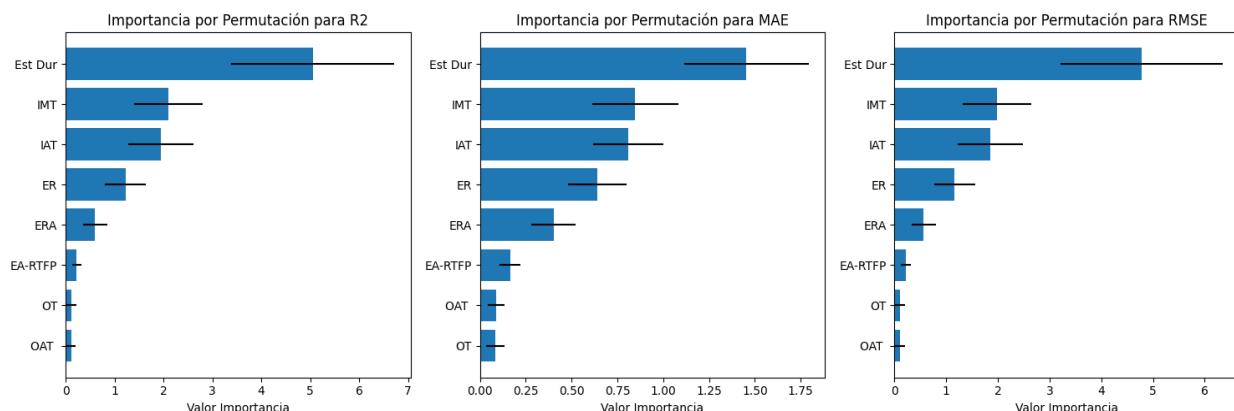


Fig. 43. Importancia por permutación Atkinson + SVR

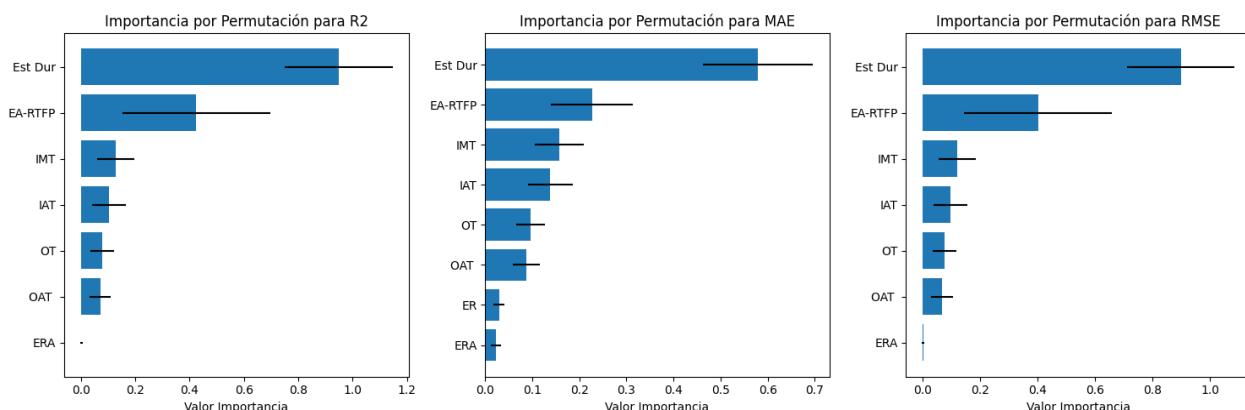
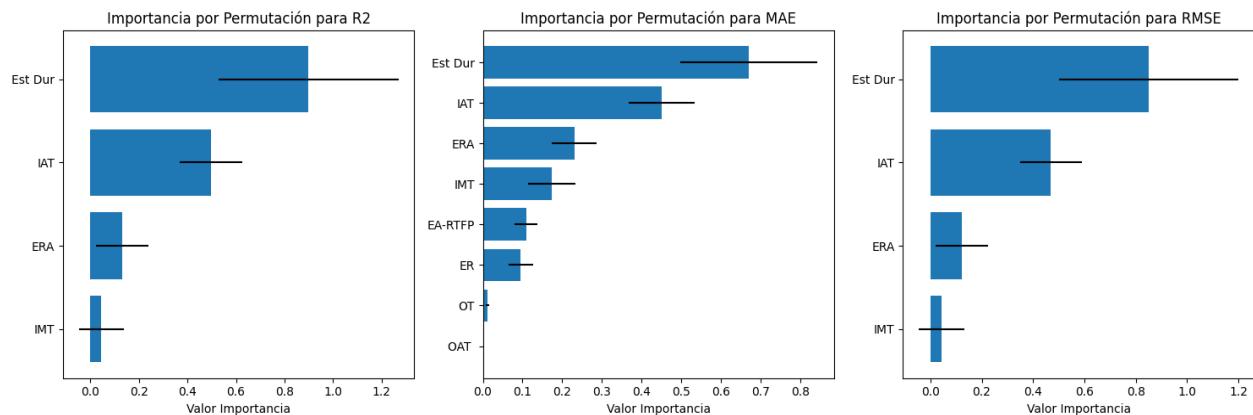
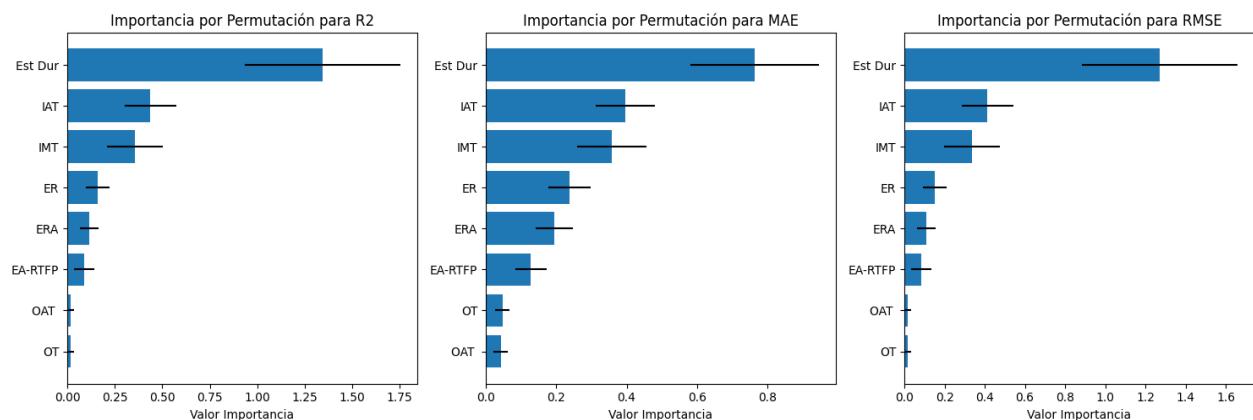


Fig. 44. Importancia por permutación Atkinson + Random Forest Regressor



**Fig. 45. Importancia por permutación Atkinson + Gradient Boosting Regressor.**



**Fig. 46. Importancia por permutación Atkinson + Voting**

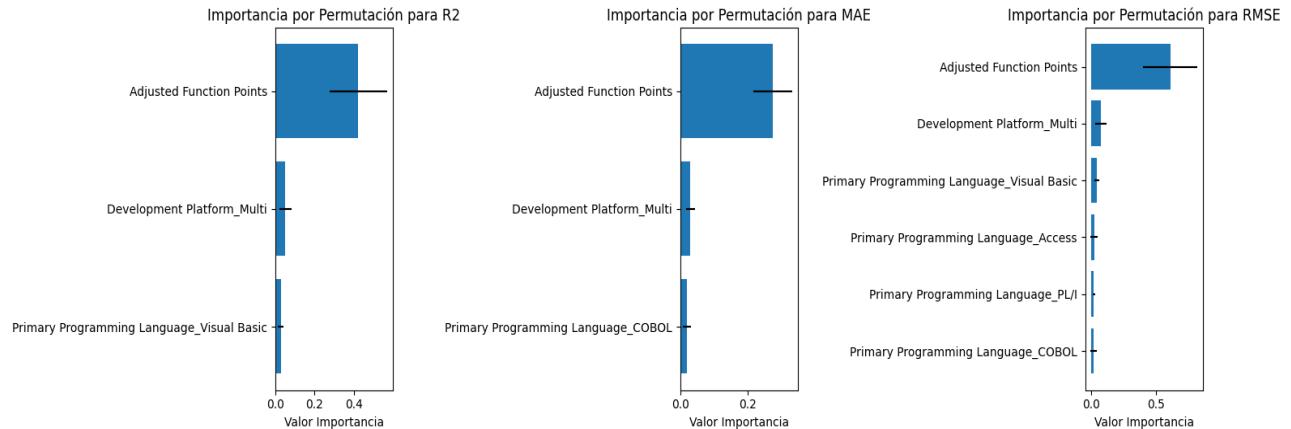
En este caso se aprecia diferencias entre los resultados obtenidos en explicabilidad global y en explicabilidad local, recogidos en la Tabla 22. Mientras que mediante los valores de importancia por permutación se observa que “*Est Dur*” es con diferencia la característica predominante con diferencia, sin embargo mediante las técnicas de explicabilidad “*IAT*” y “*IMT*” están por delante en la mayoría de los modelos.

**Tabla 22. Resultado explicabilidad local Atkinson.**

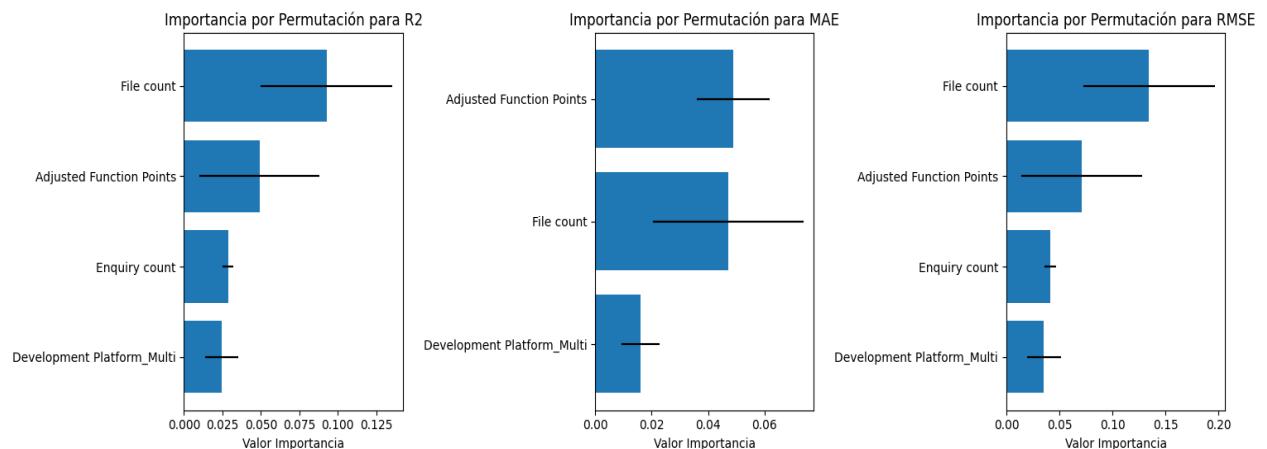
Característica	General		SVR		RF		GBR		VOT	
	Rank Final	Conteo Final	Rank	Conteo	Rank	Conteo	Rank	Conteo	Rank	Conteo
IAT	1	33	1	9	1	9	3	8	1	7
IMT	2	32	2	8	3	8	1	9	2	7
Est Dur	3	26	3	7	2	7	2	8	5	4
EA-RTFP	4	27	4	8	5	4	6	8	4	7
ER	5	19	7	7	-	-	7	3	3	9
Est	6	5	4	1	4	2	4	1	6	1
ERA	7	18	6	4	8	2	5	8	7	4
OT	8	13	8	1	6	8	-	-	8	4
OAT	9	7	-	-	7	5	-	-	9	2

- **ISBSG**

En el conjunto ISBSG las tres características más importantes son “*Adjusted function Points*”, “*Development Platform\_Multi*” y “*File count*”. Al contrario que en los conjuntos que hemos visto previamente donde la característica más influyente suele estar claramente por encima del resto en este caso la característica “*Adjusted Function Points*” y “*File count*” se intercambian ese puesto en los distintos modelos como se puede ver en Fig. 47, Fig. 48, Fig. 49 y Fig. 50.



**Fig. 47. Importancia por permutación ISBSG + SVR.**



**Fig. 48. Importancia por permutación ISBSG + Random Forest Regressor.**

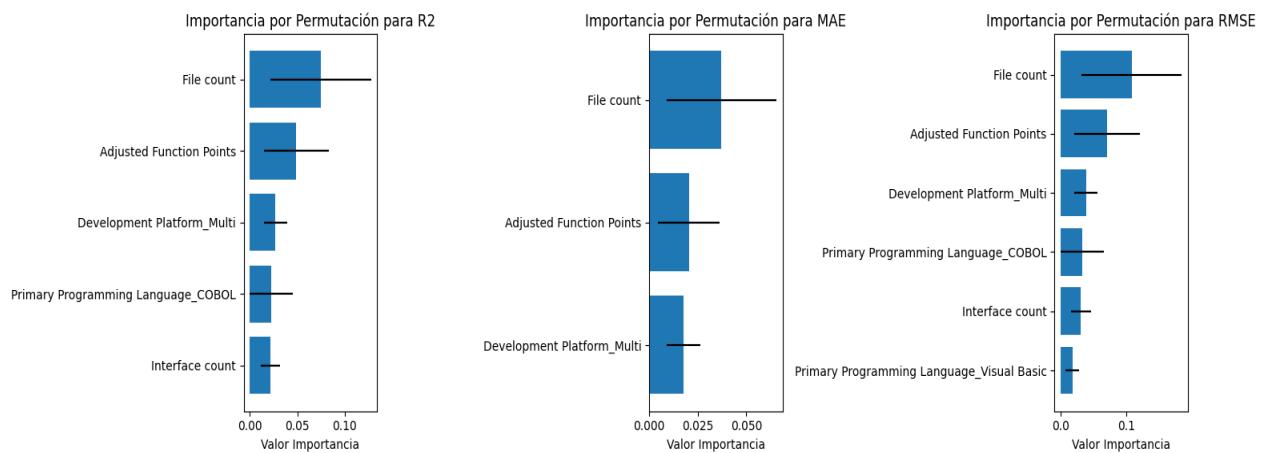


Fig. 49. Importancia por permutación ISBSG + Gradient Boosting Regressor.

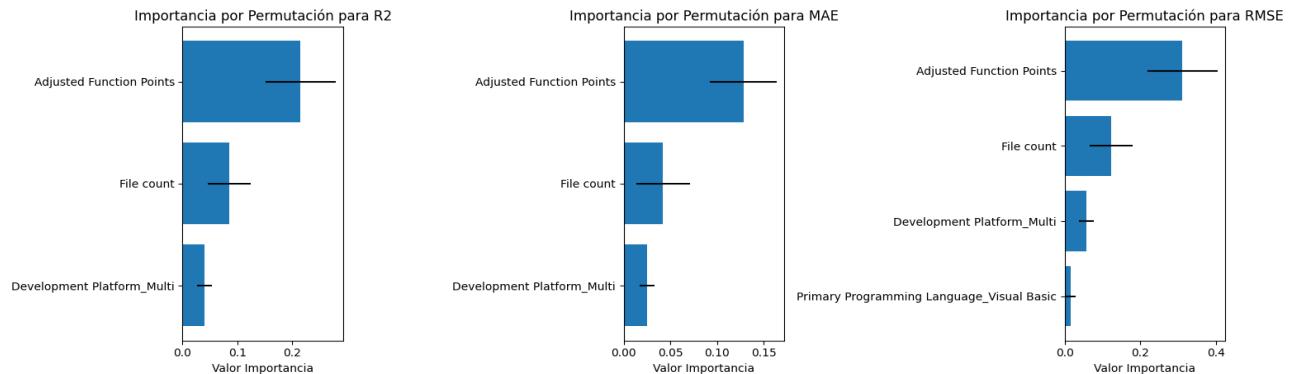


Fig. 50. Importancia por permutación ISBSG + Voting

Al igual que en las representaciones anteriores, las características “*File count*” y “*Adjusted Function Points*” se destacan como las características más influyentes en las predicciones de los modelos. Y al igual que en explicabilidad global estas características están igualadas en importancia intercambiando sus posiciones en los primeros puestos. Las siguientes características son “*Enquiry count*” y “*Output count*” también con un número alto de apariciones en los resultados locales. Sin embargo, hay una discrepancia entre la explicabilidad local y global en la característica “*Development Platform\_Multi*”, la cual queda en rango bajos a nivel local a pesar de tener bastante importancia a nivel global según las gráficas anteriores.

Tabla 23. Resultados explicabilidad local ISBSG.

Característica	General		SVR		RF		GBR		VOT	
	Rank Final	Conteo Final	Rank	Conteo	Rank	Conteo	Rank	Conteo	Rank	Conteo
<b>File count</b>	1	33	4	6	1	9	1	9	1	9
<b>Adjusted Function Points</b>	1	33	1	9	2	8	2	8	2	8
<b>Enquiry count</b>	3	25	3	9	3	5	3	6	3	5
<b>Output count</b>	4	19	2	7	6	2	10	3	3	7
<b>Primary Programming</b>	5	6	5	3	-	-	-	-	6	3

<b>Language_Smalltalk</b>											
<b>Development Platform_PC</b>	5	8	6	4	-	-	-	-	-	7	4
<b>Primary Programming Language_Other ApG</b>	7	3	-	-	-	-	-	4	3	-	-
<b>Development Platform_Multi</b>	8	6	-	-	4	4	8	2	-	-	-
<b>Primary Programming Language_COBOL</b>	9	7	-	-	-	-	5	4	9	3	
<b>Primary Programming Language_Lotus Notes</b>	10	4	-	-	6	1	6	3	-	-	
<b>Interface count</b>	11	7	-	-	5	5	11	2	-	-	
<b>Primary Programming Language_COOL:GEN</b>	11	2	-	-	-	-	-	-	5	2	
<b>Primary Programming Language_HPS</b>	13	2	-	-	6	2	-	-	-	-	
<b>Primary Programming Language_Visual Basic</b>	14	6	9	2	9	4	-	-	-	-	
<b>Development Type_ReDevelopment</b>	15	5	-	-	10	3	8	2	-	-	
<b>Primary Programming Language_Access</b>	16	3	9	2	-	-	-	-	8	1	
<b>Primary Programming Language_CLIPPER</b>	17	2	8	2	-	-	-	-	-	-	
<b>Primary Programming Language_SQL</b>	18	1	-	-	-	-	7	1	-	-	
<b>Primary Programming Language_NATURAL</b>	18	1	7	1	-	-	-	-	-	-	
<b>Primary Programming Language_Java</b>	20	2	-	-	11	1	-	-	11	1	
<b>Primary Programming Language_JavaScript</b>	21	2	-	-	12	1	12	1	-	-	
<b>Primary Programming Language_C</b>	22	1	-	-	-	-	-	-	10	1	
<b>Primary Programming Language_EASYTRIEVE</b>	23	1	-	-	-	-	-	-	11	1	

Una vez se han presentado todos los resultados, tanto a nivel global como local, que nos permiten comparar los resultados entre los dos tipos de técnicas y a nivel de modelo, es importante comparar los resultados a nivel de instancia, es decir, como contribuyen las características en las predicciones al estudiar la instancia con estimación máxima, mínima y mediana. Para realizar el estudio se han empleado las gráficas generadas por cada técnica de explicabilidad local aplicadas en esas instancias. Debido a que se ha observado un comportamiento consistente para esta comparación solo se van a presentar en esta sección los gráficos de unos ejemplos representativos. El resto de los gráficos se puede consultar en el Anexo 3, como se ha comentado previamente.

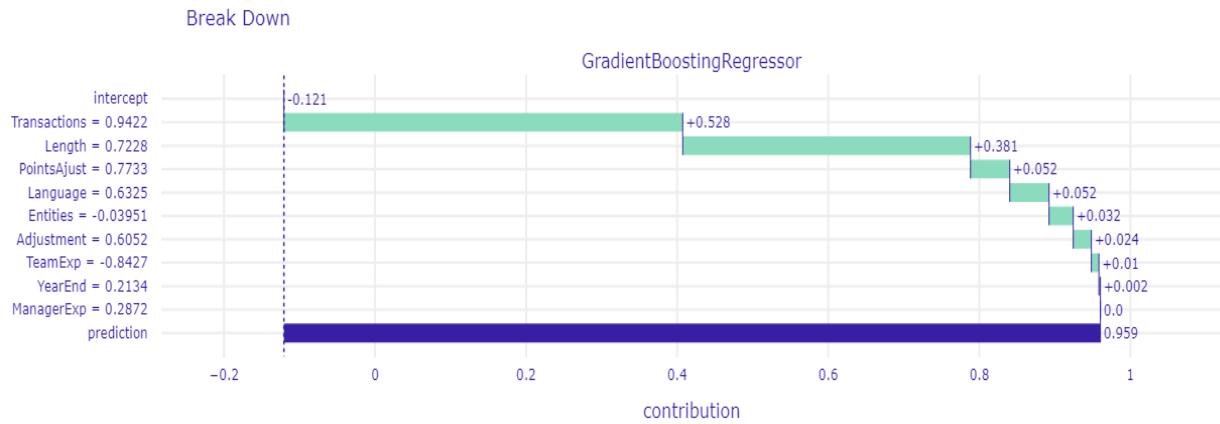


Fig. 51. Break-down en el conjunto Desharnais para el modelo Gradient Boosting Regressor y la instancia de estimación máxima.

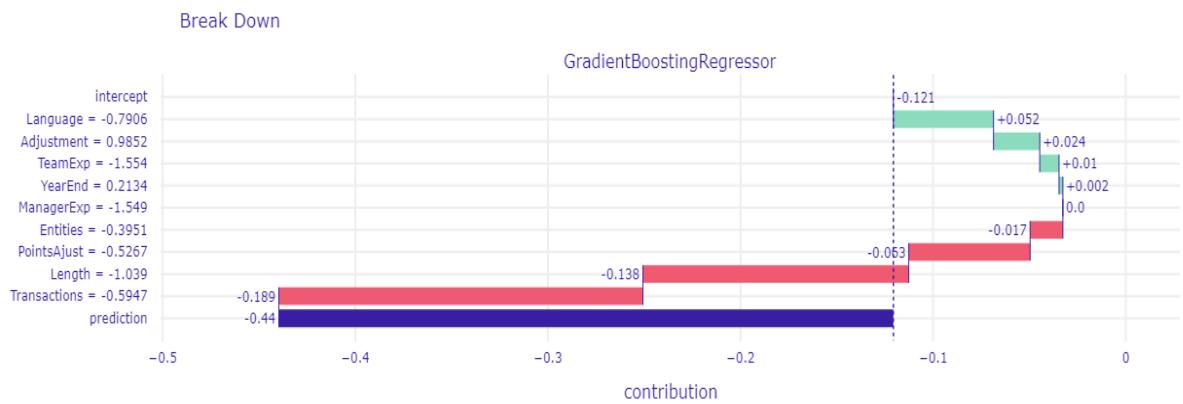


Fig. 52. Break-down en el conjunto Desharnais para el modelo Gradient Boosting Regressor y la instancia de estimación en la mediana.

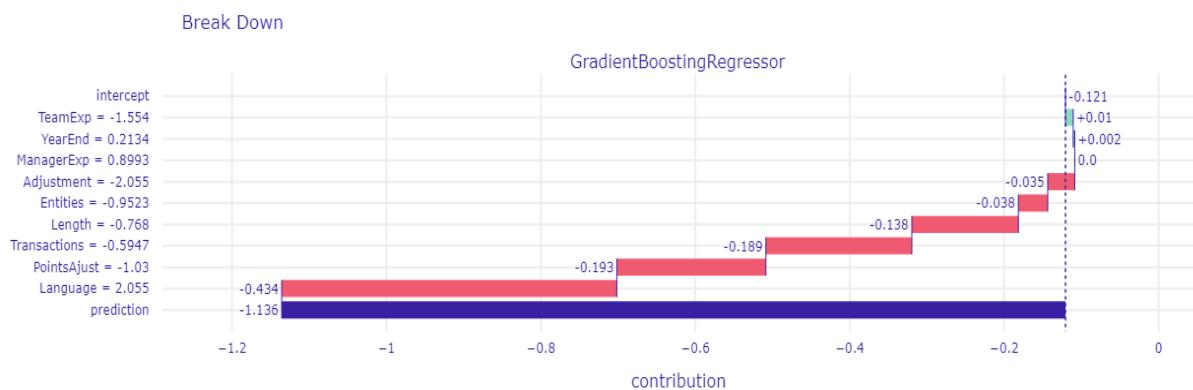


Fig. 53. Break-down en el conjunto Desharnais para el modelo Gradient Boosting Regressor y la instancia de estimación mínima.

Como podemos observar en la Fig. 51, Fig. 52 y Fig. 53, en el caso de la instancia con estimación máxima la mayoría de las instancias contribuyen de forma positiva a la predicción, lo cual es lógico debido a que estamos tratando con la instancia que ha conseguido la estimación máxima y por lo tanto la que más se distancia de la predicción media en sentido positivo. A medida que se reduce el nivel de estimación se reduce el número de características con contribución positiva y aparece características con grandes contribuciones en sentido negativo, hasta llegar al caso de la instancia de estimación mínima donde la mayoría de las contribuciones de valor alto son negativas.

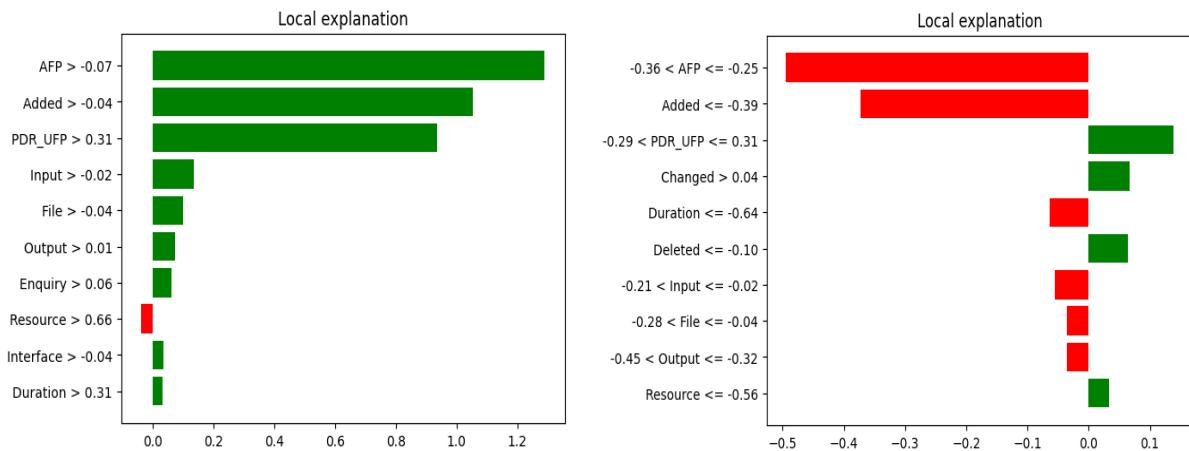


Fig. 54. En la izquierda se representa la técnica LIME en el conjunto CHINA para el modelo Random Forest y la instancia de estimación máxima. A la derecha esa misma técnica, modelo y conjunto de datos pero para la instancia en la mediana.

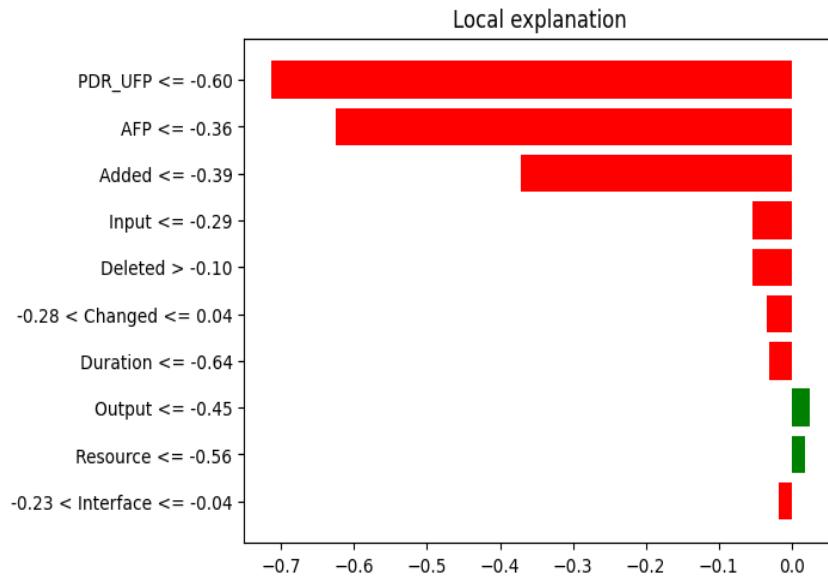


Fig. 55. LIME en el conjunto CHINA para Random Forest y la instancia de estimación mínima.

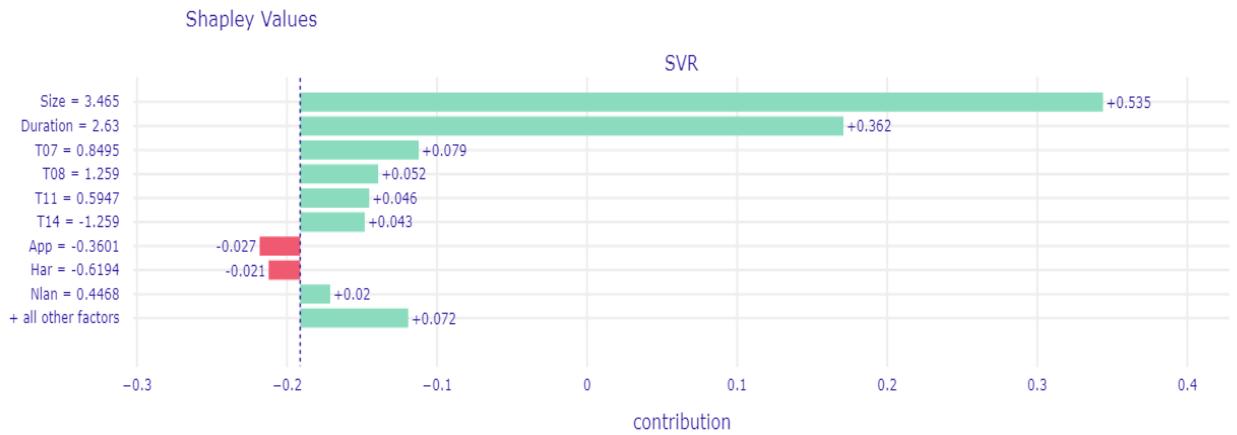


Fig. 56. Shapley Values en el conjunto Maxwell para SVR y la instancia de estimación máxima.

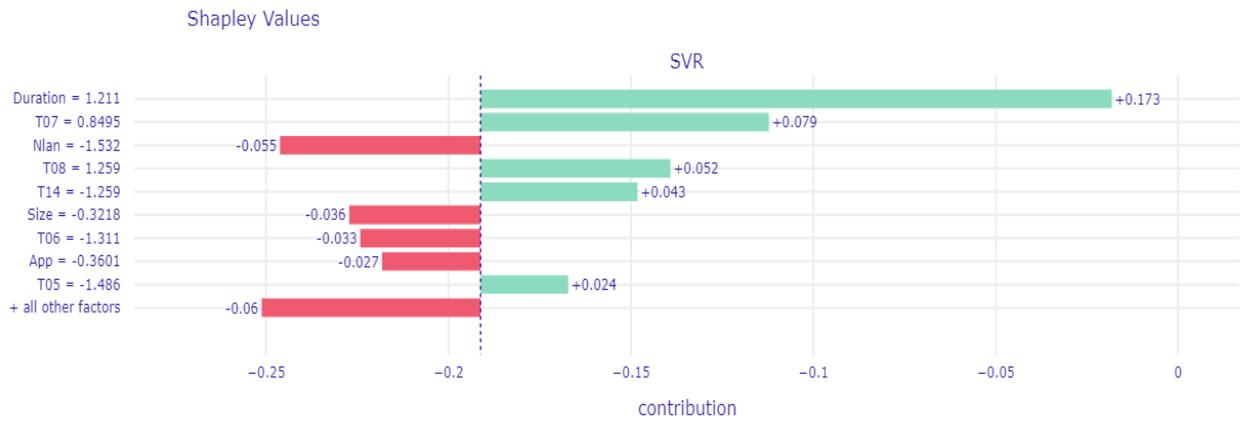


Fig. 57. Shapley Values en el conjunto Maxwell para SVR y la instancia de estimación en la mediana.

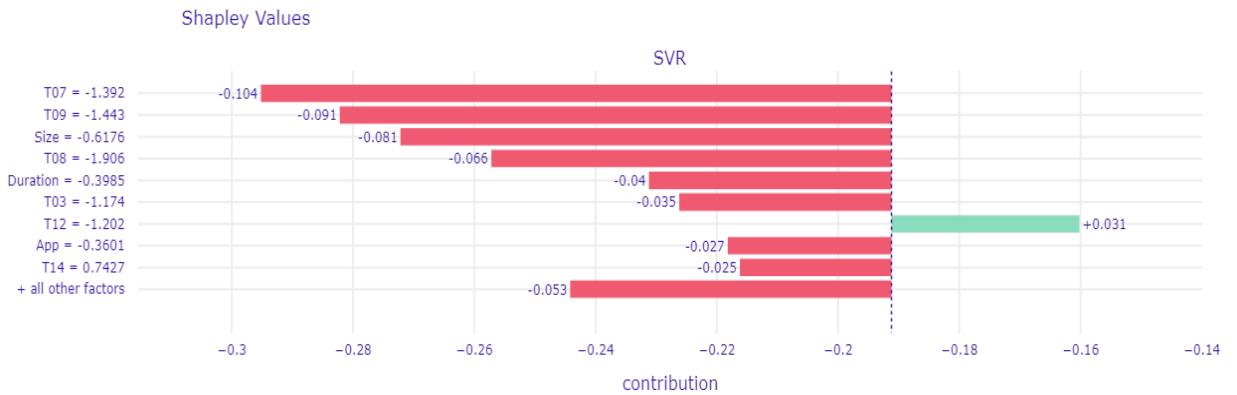


Fig. 58. Shapley Values en el conjunto Maxwell para SVR y la instancia de estimación mínima.

En comportamiento entre instancias con distinto nivel de estimación descrito previamente es confirmado por las otras dos técnicas de explicabilidad local como se muestra en la Fig. 54, Fig. 55, Fig. 56, Fig. 57 y Fig. 58. Por otro lado, si observamos las características que más contribuyen en las gráficas de todas las técnicas: “Length”, “Transactions” y “PointAdjust” en Desharnais; “AFP”, “PDR\_UFP” y “Added” en CHINA; “Size”, “Duration” y “T07” en Maxwell. Confirman los resultados de explicabilidad local comentado previamente. Con este análisis concluimos la parte de estimación de esfuerzo y pasamos a la parte de clasificación de conflictos.

## 6.2. Clasificación de conflictos

En esta sección se recopilan los resultados correspondientes a la clasificación binaria de conflictos. Se va a seguir la misma estructura que en la parte de la estimación de esfuerzo, es decir, análisis de datos, rendimiento predictivo de los modelos y explicabilidad global y local.

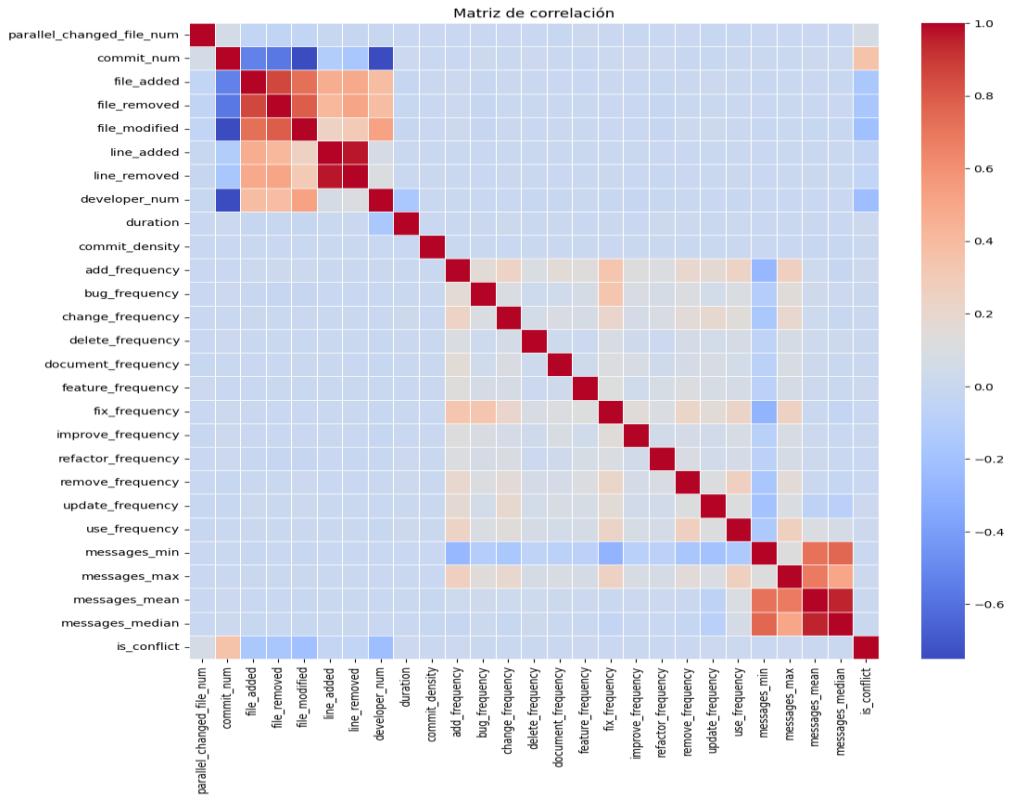
### 6.2.1. Análisis de datos

Para analizar los conjuntos de datos de la parte de clasificación, recordemos que se trata de conjuntos correspondientes a cuatro lenguajes de programación. Un hecho importante de estos conjuntos de datos es el desbalanceo de clases como se aprecia en la Tabla 24.

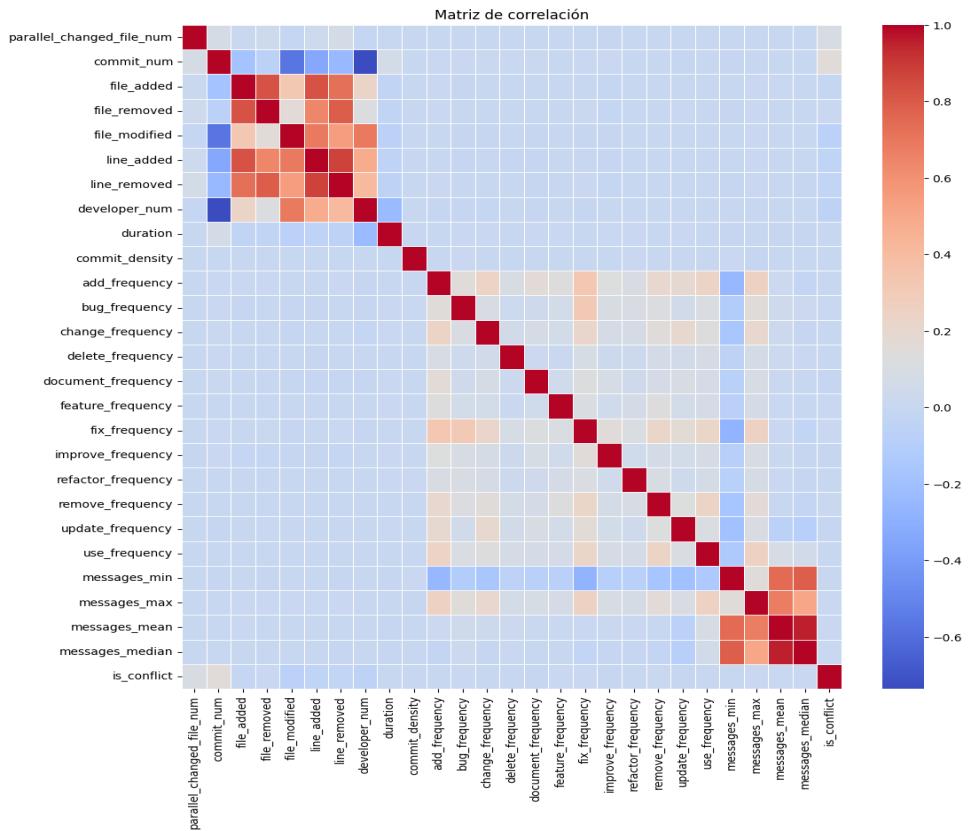
**Tabla 24. Cantidad y distribución de clases en cada conjunto de datos total, conjunto de entrenamiento y conjunto de prueba.**

Conjunto	Clase	Total	Conjunto de entrenamiento		Conjunto de prueba	
			Cantidad	Proporción	Cantidad	Proporción
JAVA	Clase 0	24519	17163	0.918	7356	0.918
	Clase 1	2180	1526	0.081	654	0.082
PYTHON	Clase 0	45925	32147	0.929	13778	0.929
	Clase 1	3528	2470	0.071	1058	0.071
RUBY	Clase 0	37098	25968	0.924	11130	0.924
	Clase 1	3031	2122	0.076	909	0.076
PHP	Clase 0	45605	31923	0.906	13682	0.906
	Clase 1	4737	3316	0.094	1421	0.094

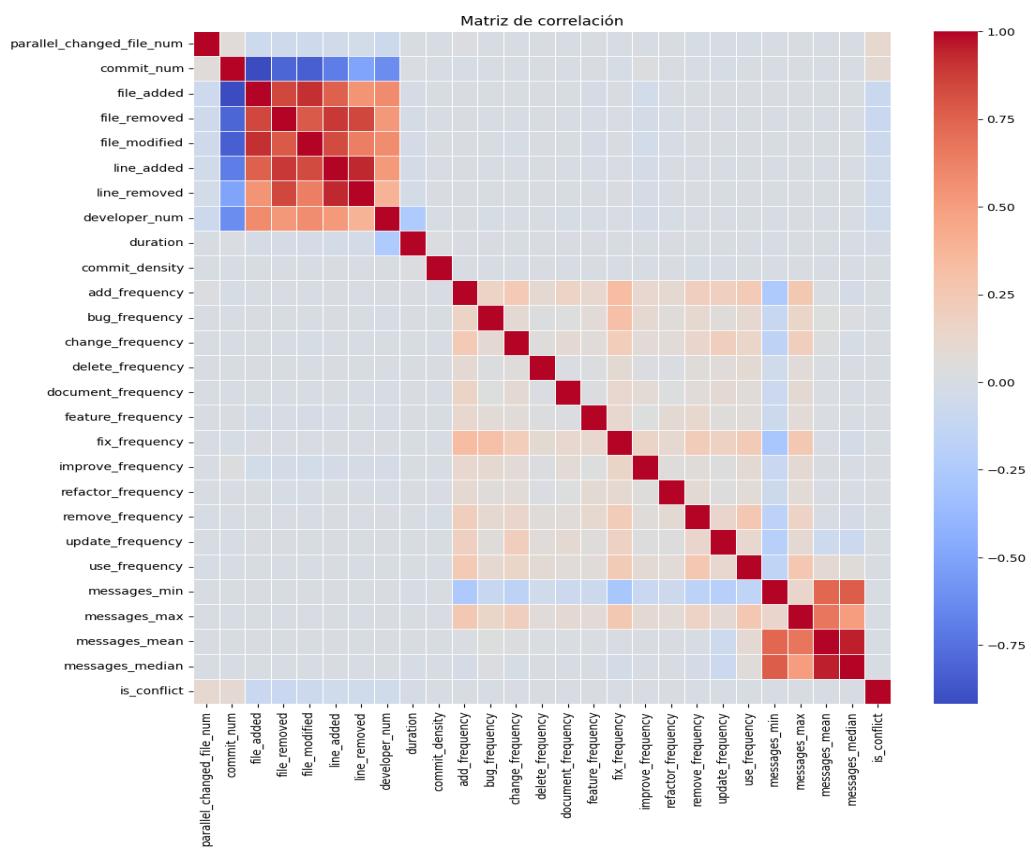
Finalmente, se muestran las matrices de correlación para los conjuntos de datos empleados en esta parte y que se recogen en la Fig. 59 - Fig. 62. En este caso, se observan dos grupos de características relacionados entre sí, como es el caso de las características correspondientes a los mensajes (“messages\_median”, “messages\_mean”, “messages\_max” y “messages\_min”). Por otro lado, encontramos mayor correlación entre las características referentes a los ficheros (“file\_added”, “file\_removed” y “file\_modified”) y a las líneas de código (“line\_added” y “line\_removed”)



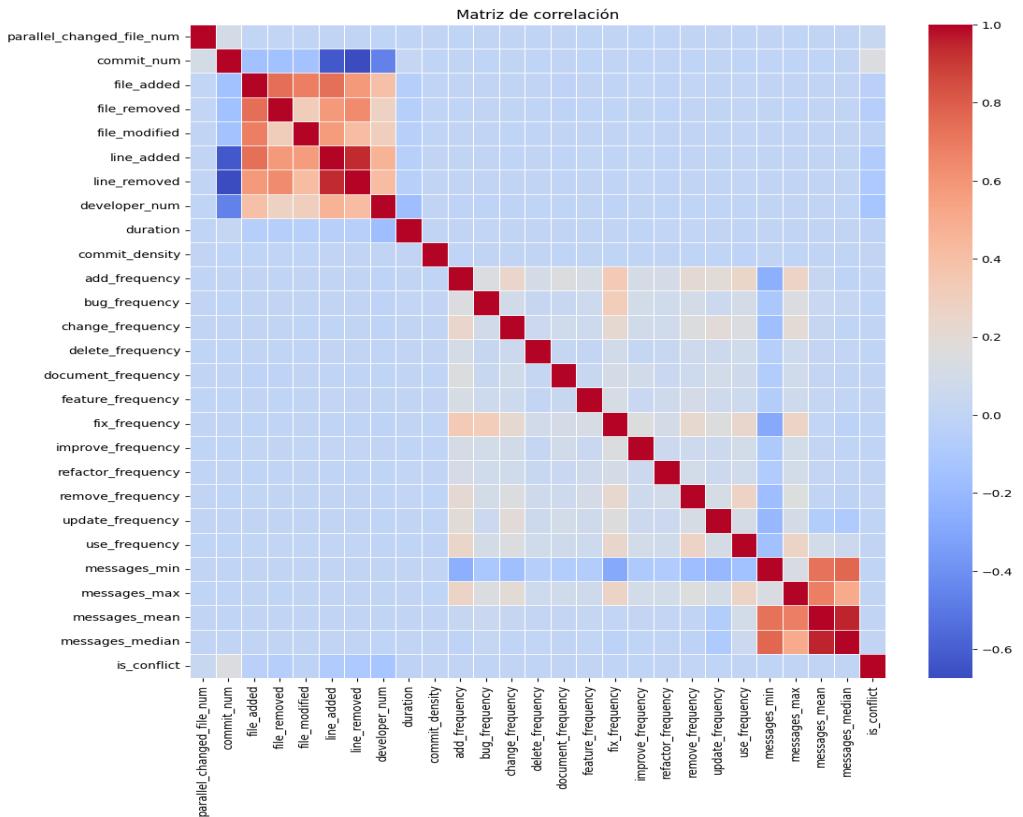
**Fig. 59. Matriz de correlación entre las características del conjunto Java**



**Fig. 60. Matriz de correlación entre las características del conjunto Python.**



**Fig. 61. Matriz de correlación entre las características del conjunto Ruby.**



**Fig. 62. Matriz de correlación entre las características del conjunto PHP.**

## 6.2.2. Rendimiento predictivo

Para los algoritmos de clasificación empleados, en la Tabla 25 se pueden encontrar la mejor configuración de hiperparámetros ha sido encontrada mediante “*GridSearchCV*”.

**Tabla 25. Valores de los hiperparámetros empleados para cada modelo y conjunto de datos.** , donde RFC, BRF, GBC, ADA y RUS hacen referencia a los algoritmos Random Forest Regressor, Balanced Random Forest, Gradient Boosting Regressor, ADABoost y RUSBoost, respectivamente.

Conjunto	Modelo	Hiperparámetros
Java	RFC	max_depth = 10, min_samples_leaf = 5, min_samples_split = 20, n_estimators = 50, class_weight = 'balanced'
	BRF	max_depth = 20, min_samples_leaf = 1, min_samples_split = 3, n_estimators = 25, class_weight = None
	GBC	max_depth = 5, min_samples_leaf = 25, min_samples_split = 2, n_estimators = 100
	ADA	learning_rate = 0.4, n_estimators = 25
	RUS	learning_rate = 1.6, n_estimators = 25
Python	RFC	max_depth = 30, min_samples_leaf = 5, min_samples_split = 2, n_estimators = 100, class_weight = 'balanced'
	BRF	class_weight = 'balanced', max_depth = None, min_samples_leaf = 1, min_samples_split = 2, n_estimators = 10
	GBC	max_depth = 10, min_samples_leaf = 10, min_samples_split = 2, n_estimators = 200
	ADA	learning_rate = 1.2, n_estimators = 200
	RUS	learning_rate = 1.6, n_estimators = 10
Ruby	RFC	class_weight = 'balanced', max_depth = 20, min_samples_leaf = 3, min_samples_split = 2, n_estimators = 300
	BRF	class_weight = None, max_depth = 20, min_samples_leaf = 1, min_samples_split = 2, n_estimators = 10
	GBC	max_depth = 5, min_samples_leaf = 25, min_samples_split = 2, n_estimators = 100
	ADA	learning_rate = 0.8, n_estimators = 5
	RUS	learning_rate = 1.8, n_estimators = 5
PHP	RFC	class_weight = 'balanced', max_depth = 10, min_samples_leaf = 5, min_samples_split = 20, n_estimators = 50
	BRF	class_weight = None, max_depth = 15, min_samples_leaf = 1, min_samples_split = 3, n_estimators = 75
	GBC	max_depth = 5, min_samples_leaf = 25, min_samples_split = 2, n_estimators = 100
	ADA	learning_rate = 1.0, n_estimators = 25
	RUS	learning_rate = 1.6, n_estimators = 25

Ahora que sabemos cómo están construidos nuestros modelos para abordar el problema de clasificación, vamos a ver qué rendimiento se ha obtenido con ellos. A la hora de evaluar el rendimiento de este problema es importante señalar que M. Owhadi-Kareshk et al. [7] reflejaron en su trabajo los resultados obtenidos, lo cual nos sirve para comparar con los resultados aquí obtenidos.

Antes de pasar directamente a la visualización de los resultados en las métricas elegidas para evaluar el rendimiento en clasificación y que han sido definidas en las ecuaciones (6),(7),(8) y (9). Para ello se muestra la matriz de confusión obtenida de cada modelo para un conjunto de datos y que sirven para darnos una idea de qué podemos esperar en las métricas. Para el conjunto de datos de Java, en la Fig. 63 se muestra la matriz de confusión obtenida en cada modelo, la Fig. 64 los resultados de las distintas métricas y la Fig. 65 esas mismas métricas pero evaluadas tras invertir las etiquetas del problema, igual que han hecho los autores.

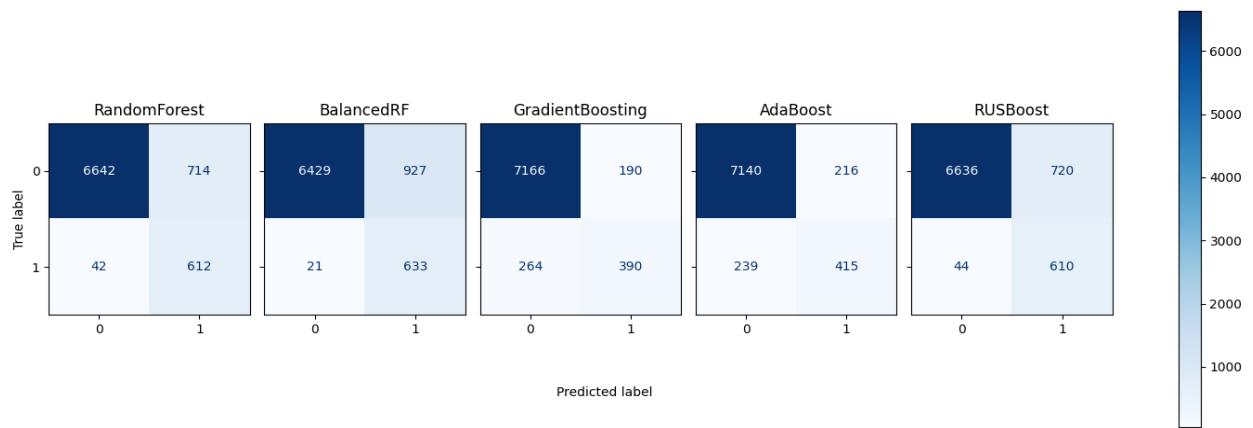


Fig. 63. Matriz de confusión obtenida en cada modelo de clasificación empleado en el conjunto Java.

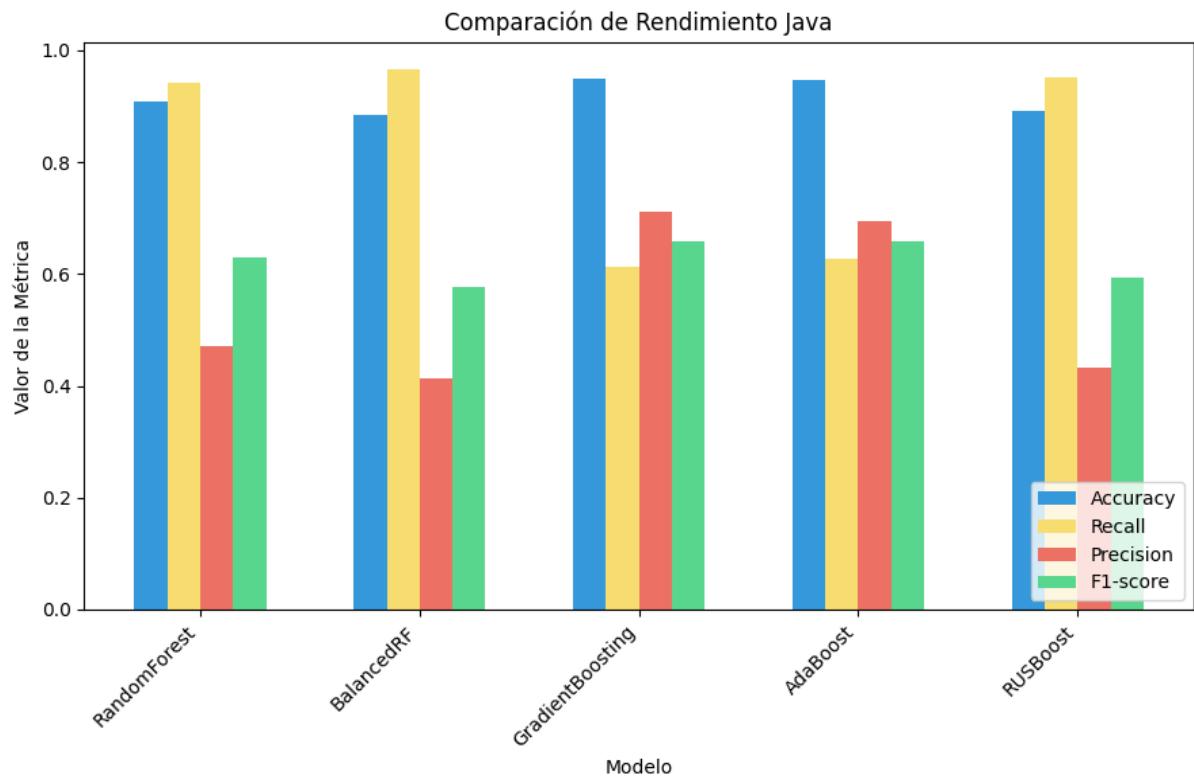
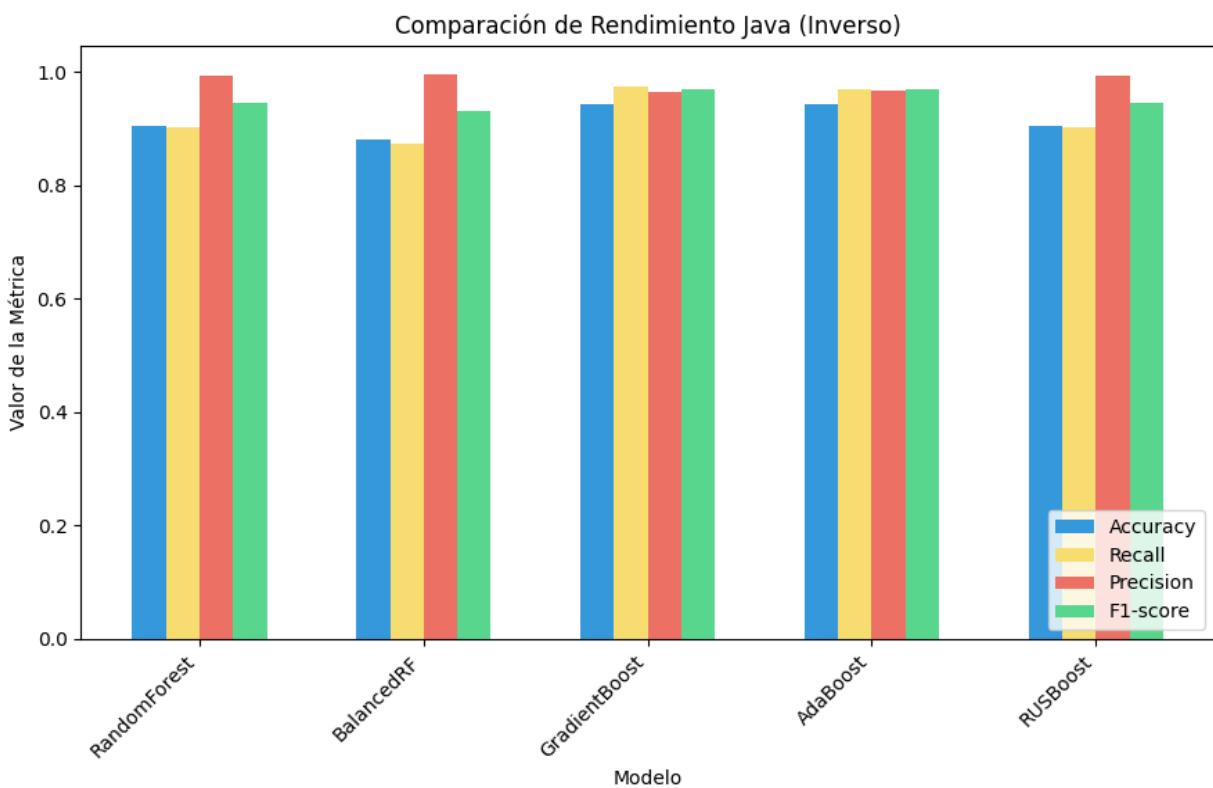


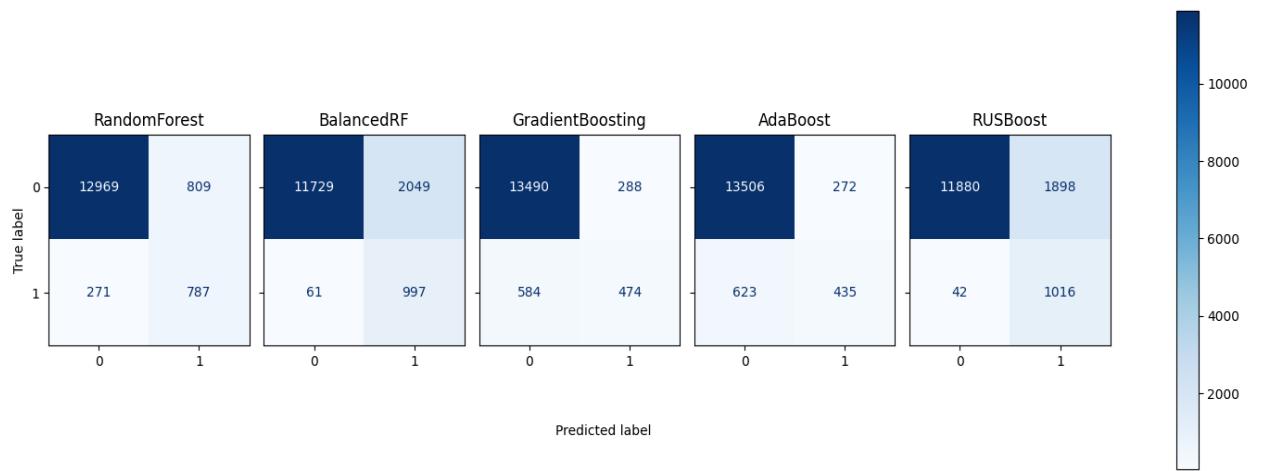
Fig. 64. Rendimiento en el conjunto Java, de los modelos de clasificación empleados.



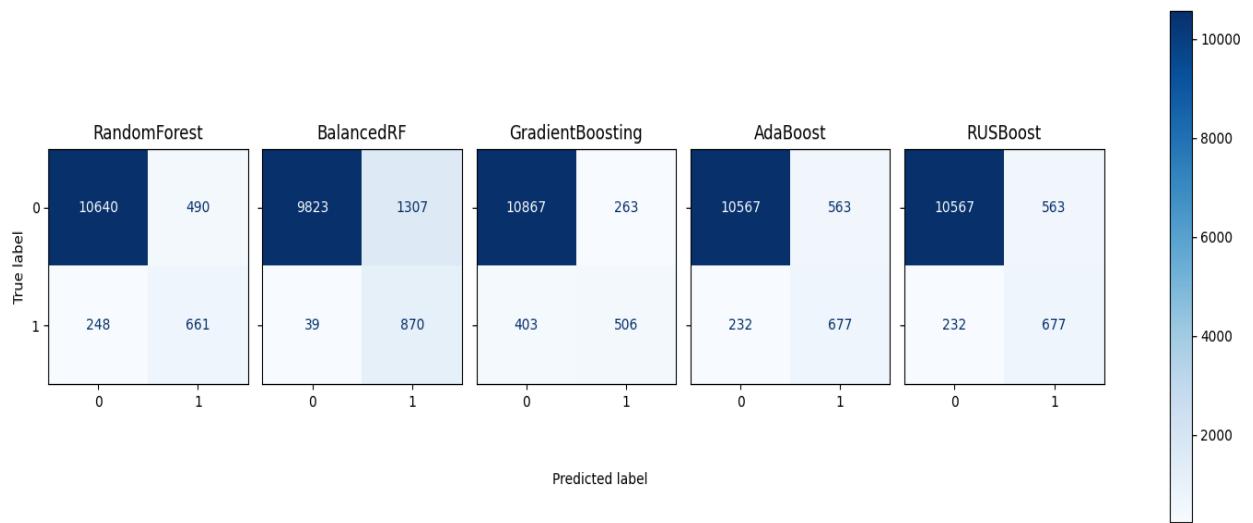
**Fig. 65. Rendimiento en modelos de clasificación empleados en el conjunto Java tras invertir las clases.**

De la gráfica con las métricas del conjunto Java (Fig. 64) se puede concluir que la métrica *Accuracy* no es adecuada en escenarios con desbalanceo de clases, ya que consigue valores altos gracias a la clase mayoritaria, en nuestro caso la clase 0, como también se puede apreciar en la matriz de confusión (Fig. 63). En relación con el desbalanceo de clases se observa que los algoritmos de *Imbalance-Learn* (“*Balanced Random Forest*” y “*RUSBoost*”) no consiguen mejor rendimiento que los algoritmos convencionales de *Scikit-learn*, a pesar de tratarse de un conjunto con desbalanceo de clases. Si nos fijamos en la métrica *F1-score*, la cual es más adecuada en escenarios con desbalanceo, vemos que “*ADABOOST*” y “*Gradient Boosting*” consiguen los mejores resultados, en parte al conseguir los mejores resultados en la métrica precisión. En cambio ambos “*Random Forest*” y “*RUSBoost*” consigue valores altos en *Recall*. En cuanto al estudio realizado con las clases invertidas, podemos ver que se obtienen muy buenos resultados en todas las métricas, lo cual era lo esperado ya que la clase mayoritaria es “No es un conflicto” y al construir los modelos para detectar los casos de no conflicto las métricas se disparan.

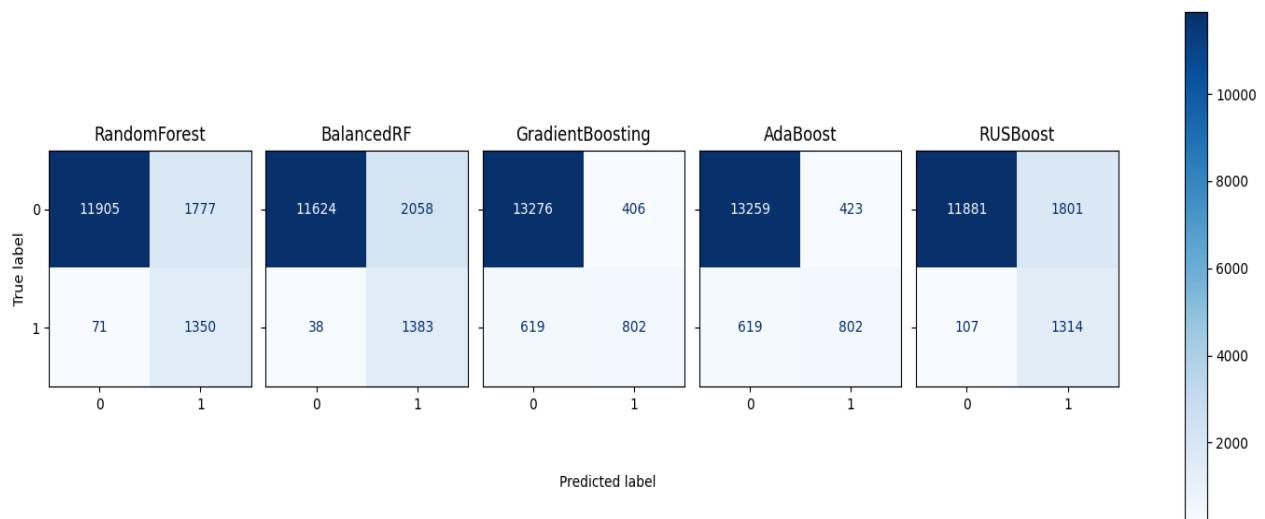
Debido a que los comportamientos que se han observado en el conjunto Java se aprecian de igual manera en el resto de los conjuntos de lenguajes de programación vamos a omitir una descripción individual de cada uno y solo se van a mostrar los resultados obtenidos para la matriz de confusión



**Fig. 66. Matriz de confusión obtenida en cada modelo de clasificación empleado en Python.**



**Fig. 67. Matriz de confusión obtenida para cada modelo de clasificación empleado en el conjunto Ruby.**



**Fig. 68. Matriz de confusión obtenida para cada modelo de clasificación empleado en el conjunto PHP.**

Tras las matrices de confusión de los conjuntos de datos de cada lenguaje de programación, se muestran los valores de las métricas obtenidos para esos mismos conjuntos de datos y modelos.

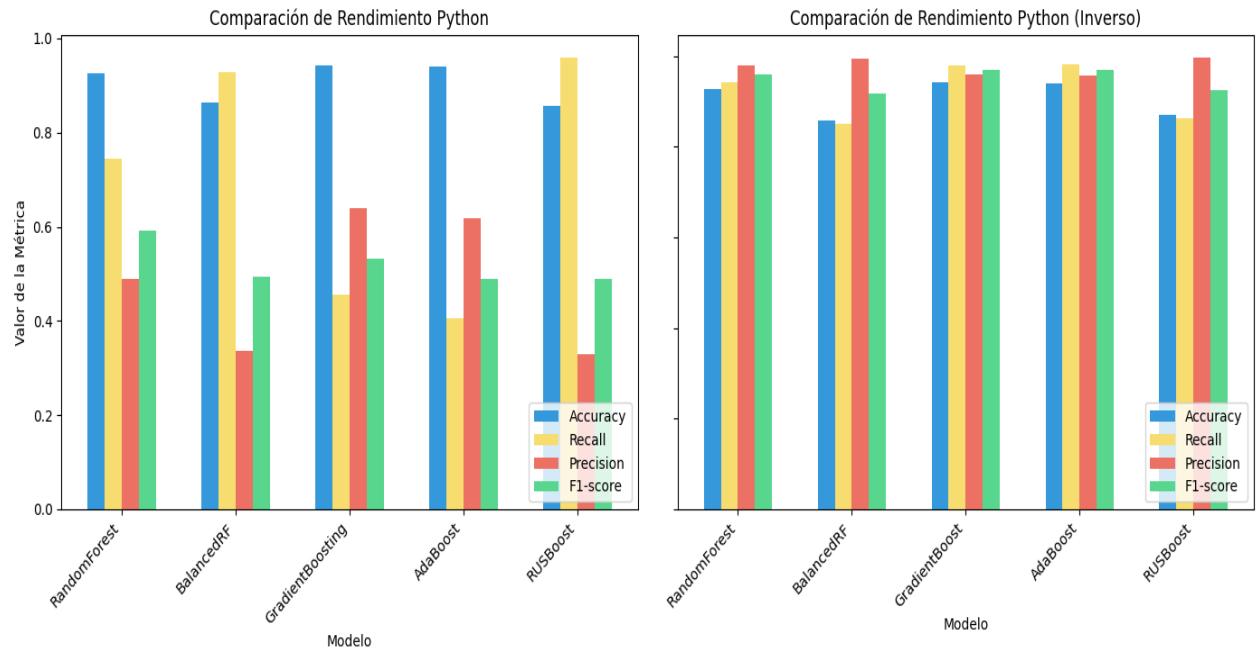


Fig. 69. En el gráfico de la izquierda se muestra el rendimiento en el conjunto Python para los modelos de clasificación empleados. A la izquierda el rendimiento cuando se invierten las clases.

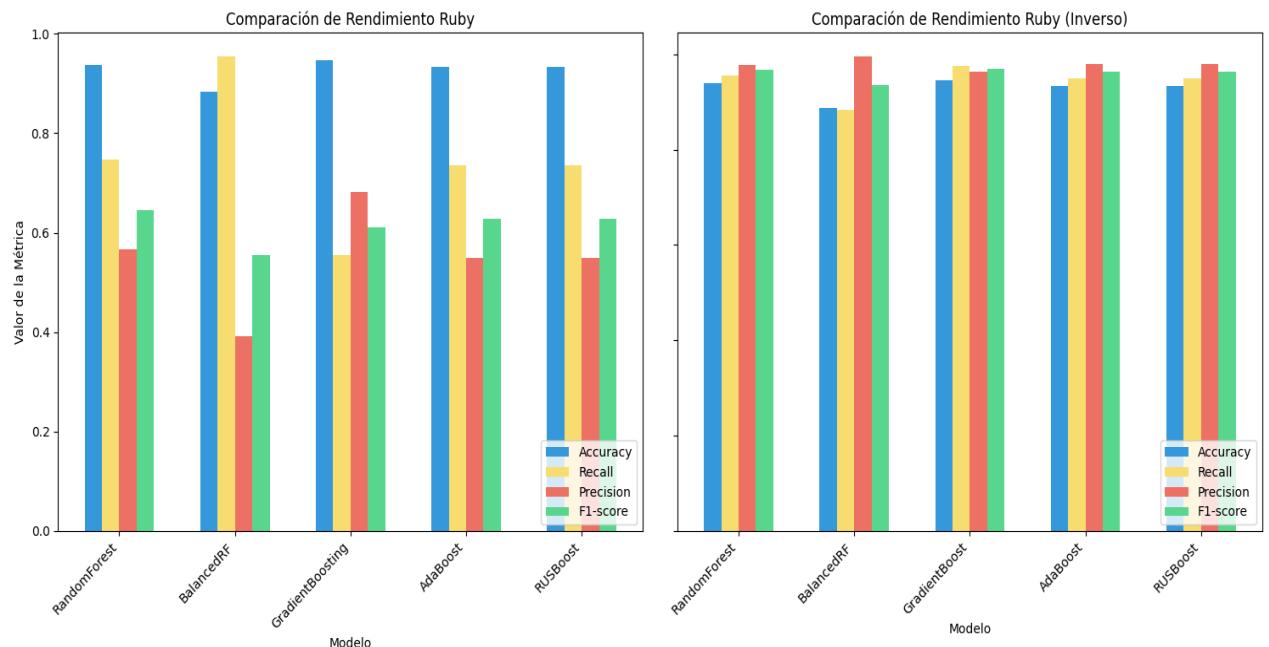


Fig. 70. En el gráfico de la izquierda se muestra el rendimiento en el conjunto Ruby para los modelos de clasificación empleados. A la izquierda el rendimiento cuando se invierten las clases.

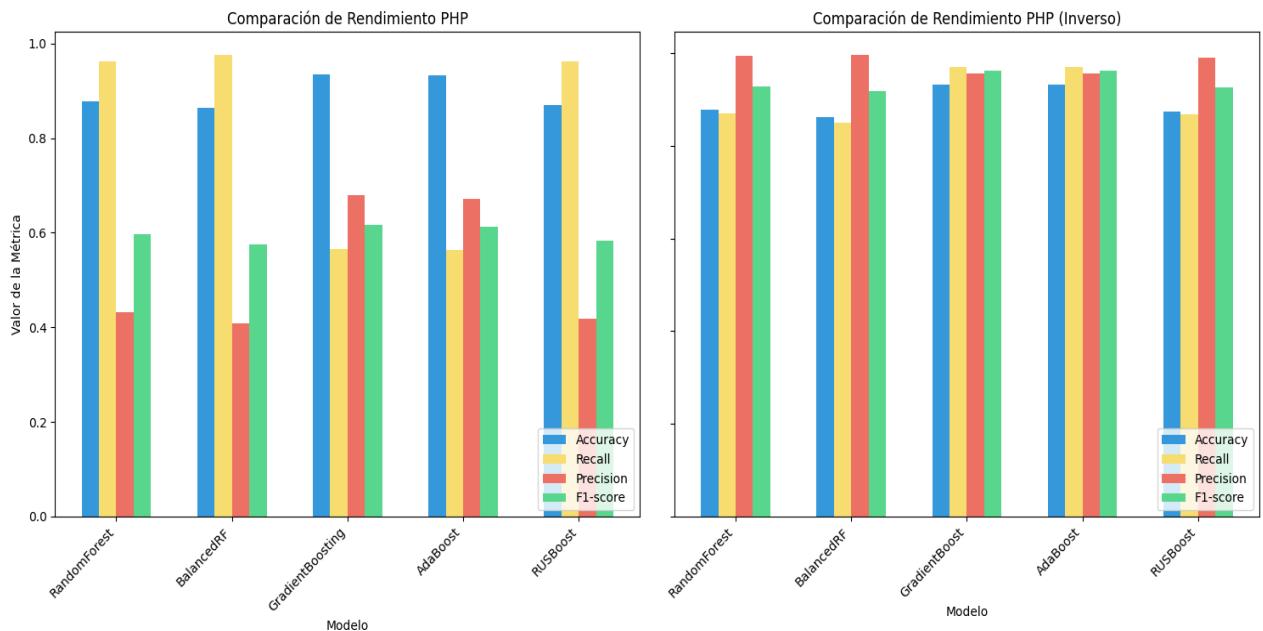


Fig. 71. En el gráfico de la izquierda se muestra el rendimiento en el conjunto PHP para los modelos de clasificación empleados. A la izquierda el rendimiento cuando se invierten las clases.

### 6.2.3. Explicabilidad del modelo

Siguiendo la forma de proceder en esta misma sección para la estimación de esfuerzo (Sección 6.1.3) se va a realizar una descripción individual de los resultados en explicabilidad global y local para cada conjunto de datos. En este caso las métricas empleadas en la importancia por permutación son *Precision* (ec.8), *Recall* (ec.7) y *F1-score* (ec.9).

- **Conjunto Java**

En primer lugar, se muestran los resultados obtenidos para la importancia por permutación, recordemos que nos da una visión de la explicabilidad global del modelo. Se pueden consultar los resultados en la Fig. 72 - Fig. 76.

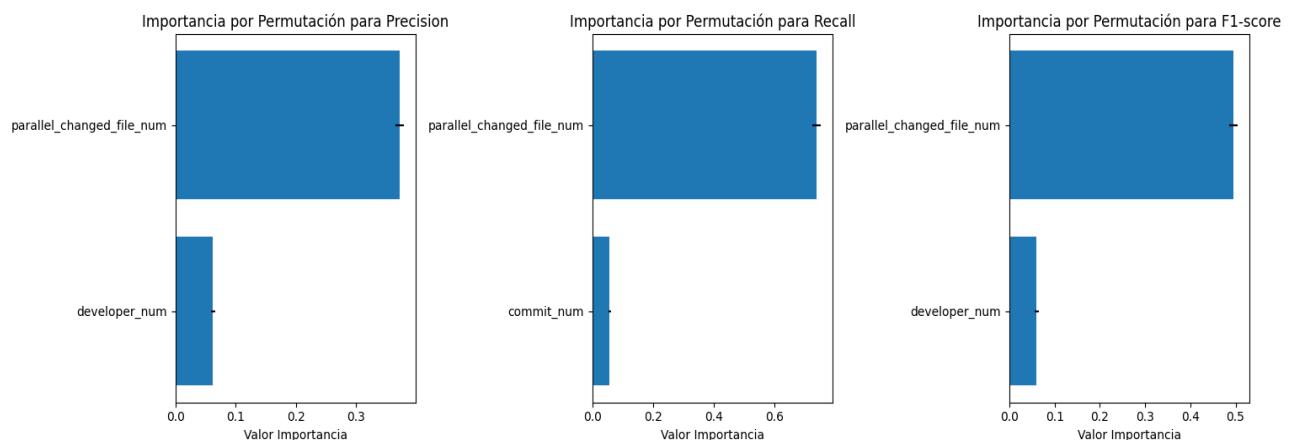
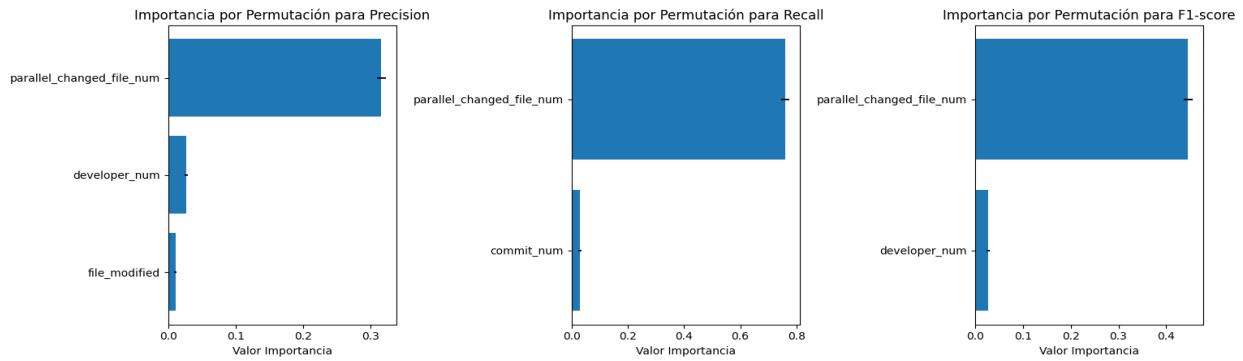
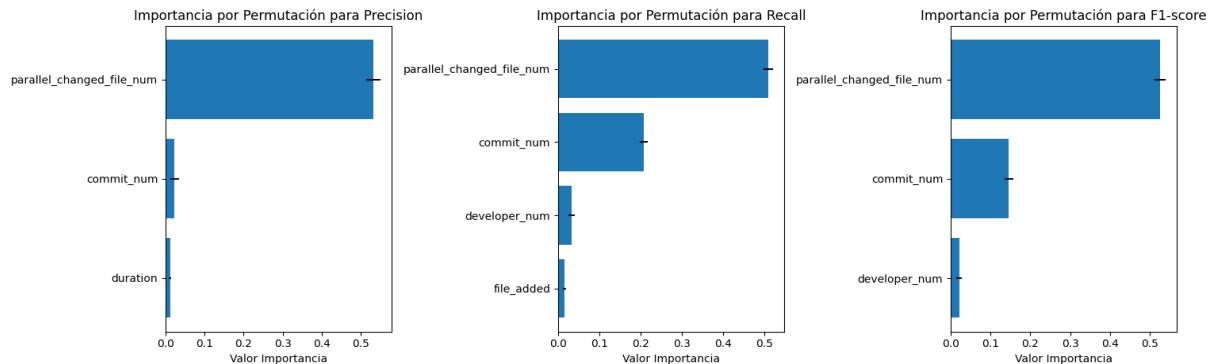


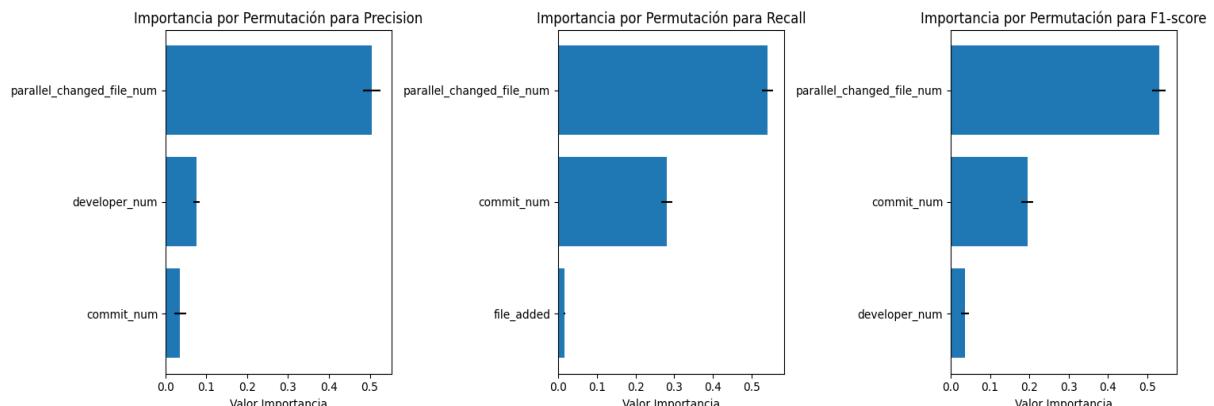
Fig. 72. Importancia por permutación Java + Random Forest



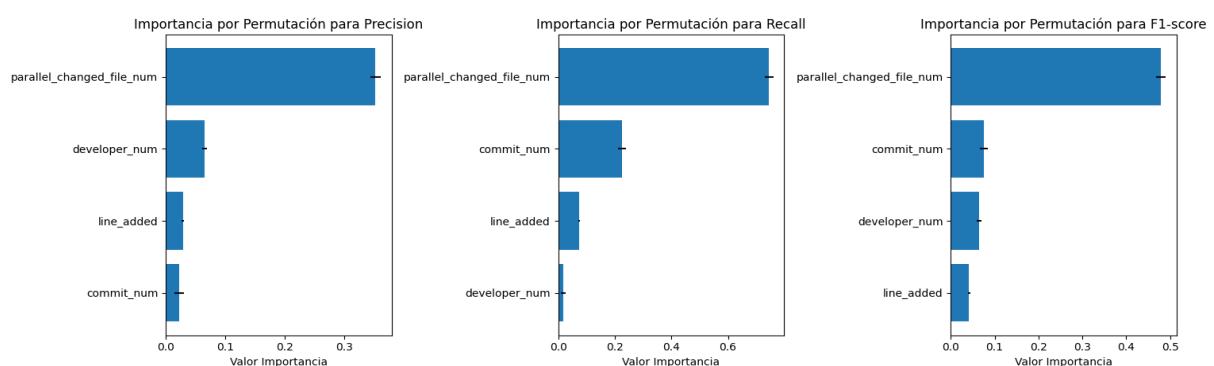
**Fig. 73. Importancia por permutación Java + Balanced Random Forest**



**Fig. 74. Importancia por permutación Java + Gradient Boosting**



**Fig. 75. Importancia por permutación Java + ADABoost**



**Fig. 76. Importancia por permutación Java + RUSBoost**

Podemos concluir que la característica “*parallel\_changed\_file\_num*” es en todas las métricas y modelos las más importante con diferencia con respecto al resto. Los siguientes puestos en importancia se encuentran las características “*commit\_num*” y “*developer\_num*”, que dependiendo del modelo y de la métrica intercambian sus puestos de segundo y tercer lugar. En algunos casos aparecen otras características como “*line\_added*” y “*duration*” con muy poca importancia. Tras visualizar los datos de explicabilidad global, podemos analizar los resultados de explicabilidad local contenidos en la Tabla 26, Tabla 27, Tabla 28 y Tabla 29, correspondientes a las predicciones de verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos respectivamente. Estos resultados obtenidos para explicabilidad local son coherentes con los resultados de explicabilidad global, pues “*parallel\_changed\_file\_num*”, “*commit\_num*” y “*developer\_num*” son las características con mayor ranking y con números altos de apariciones en todos los resultados de clasificación. Tras las características recién mencionadas, algunas características como “*file\_added*”, “*line\_added*” y “*file\_modified*” también tienen valores de rango y apariciones altos y por lo tanto se consideran importantes.

**Tabla 26. Resultados explicabilidad local en el conjunto Java para verdaderos positivos.** Rank representa la posición en valor de contribución y “Cont” el número de veces que una característica aparece en una técnica. RF, BRF, GB, ADA y RUS representan los modelos de Random Forest, Balanced Random Forest, Gradient Boosting, ADABoost y RUSBoost respectivamente.

Característica	General		RF		BRF		GB		ADA		RUS	
	Rank Final	Cont Final	Rank	Cont								
<i>parallel_changed_file_num</i>	1	45	1	9	1	9	1	9	1	9	1	9
<i>commit_num</i>	2	43	2	9	2	9	2	9	2	9	3	7
<i>developer_num</i>	3	34	3	8	7	3	4	6	3	9	4	8
<i>file_added</i>	4	26	5	2	3	6	5	4	4	9	5	5
<i>file_modified</i>	5	25	4	7	5	5	3	8	-	-	7	5
<i>line_added</i>	6	18	5	4	10	1	-	-	5	6	2	7
<i>line_removed</i>	7	9	7	2	4	4	8	1	-	-	5	2
<i>file_removed</i>	8	12	7	2	6	3	5	5	-	-	8	2
<i>delete_frequency</i>	9	3	-	-	-	-	-	-	6	3	-	-
<i>commit_density</i>	10	4	-	-	8	2	7	2	-	-	-	-
<i>duration</i>	11	2	9	1	-	-	8	1	-	-	-	-
<i>fix_frequency</i>	12	1	-	-	8	1	-	-	-	-	-	-
<i>messages_mean</i>	13	1	9	1	-	-	-	-	-	-	-	-
<i>document_frequency</i>	14	1	-	-	10	1	-	-	-	-	-	-
<i>Update_frequency</i>	14	1	-	-	10	1	-	-	-	-	-	-

**Tabla 27. Resultados explicabilidad local en el conjunto Java para verdaderos negativos.**

Característica	General		RF		BRF		GB		ADA		RUS	
	Rank Final	Cont Final	Rank	Cont								
<i>parallel_changed_file_num</i>	1	45	1	9	1	9	1	9	1	9	1	9
<i>commit_num</i>	2	40	2	9	2	8	2	8	2	9	4	6
<i>developer_num</i>	3	31	3	7	7	2	4	5	3	9	3	8
<i>line_added</i>	4	29	6	6	3	4	8	3	4	7	2	9

file_added	5	26	7	3	3	4	3	6	4	8	6	5
file_modified	6	18	4	4	8	4	5	2	-	-	5	8
line_removed	7	8	5	2	3	3	9	3	-	-	-	-
fix_frequency	8	6	9	1	6	2	5	2	7	1	-	-
file_removed	9	9	8	2	14	1	7	6	-	-	-	-
improve_frequency	10	1	-	-	-	-	-	-	6	1	-	-
duration	11	4	9	1	12	2	10	1	-	-	-	-
delete_frequency	12	1	-	-	-	-	-	-	7	1	-	-
document_frequency	13	2	-	-	10	2	-	-	-	-	-	-
messages_mean	14	1	-	-	8	-	-	-	-	-	-	-
commit_density	15	3	11	1	10	2	-	-	-	-	-	-
messages_min	16	1	-	-	13	1	-	-	-	-	-	-

Tabla 28. Resultados explicabilidad local en conjunto Java para falsos positivos.

Característica	General		RF		BRF		GB		ADA		RUS	
	Rank Final	Cont Final	Rank	Cont								
parallel_changed_file_num	1	45	1	9	1	9	1	9	1	9	1	9
commit_num	2	42	2	8	2	8	2	9	2	9	3	8
developer_num	3	36	3	8	3	4	3	6	3	9	3	9
file_added	4	23	4	4	5	4	4	4	4	8	6	3
line_added	5	24	7	4	8	2	9	2	4	7	2	9
file_modified	6	20	6	5	7	4	7	4	-	-	5	7
messages_min	7	2	-	-	4	2	-	-	-	-	-	-
file_removed	8	9	10	2	6	3	6	4	-	-	-	-
delete_frequency	9	2	-	-	-	-	-	-	6	2	-	-
messages_mean	10	4	5	3	9	1	-	-	-	-	-	-
commit_density	11	8	-	-	12	2	5	6	-	-	-	-
line_removed	12	5	8	1	10	3	8	1	-	-	-	-
improve_frequency	13	1	-	-	-	-	-	-	7	1	-	-
document_frequency	14	2	-	-	11	2	-	-	-	-	-	-
duration	14	1	8	1	-	-	-	-	-	-	-	-
use_frequency	16	1	-	-	13	1	-	-	-	-	-	-

Tabla 29. Resultados explicabilidad local en el conjunto Java para falso negativos.

Característica	General		RF		BRF		GB		ADA		RUS	
	Rank Final	Cont Final	Rank	Cont								
parallel_changed_file_num	1	45	1	9	1	9	1	9	1	9	1	9
developer_num	2	42	2	9	2	7	3	8	3	9	2	9
commit_num	3	36	3	6	4	6	2	7	2	9	3	8
file_added	4	23	5	4	3	5	5	3	4	8	5	3
line_added	4	27	5	4	6	4	7	3	4	7	3	9
file_modified	6	17	7	2	9	2	4	6	-	-	6	7
duration	7	7	4	4	7	2	10	1	-	-	-	-

file_removed	8	6	8	3	-	-	6	3	-	-	-	-
commit_density	9	8	10	1	5	5	8	2	-	-	-	-
improve_frequency	10	4	-	-	-	-	8	2	6	2	-	-
change_frequency	11	1	-	-	-	-	-	-	7	1	-	-
messages_max	12	2	-	-	8	2	-	-	-	-	-	-
line_removed	13	4	9	3	-	-	10	1	-	-	-	-
document_frequency	14	2	-	-	9	2	-	-	-	-	-	-

- **Conjunto Python**

De forma análoga al conjunto de Java, primero se muestran los resultados obtenidos en importancia por permutación, Fig. 77 - Fig. 81.

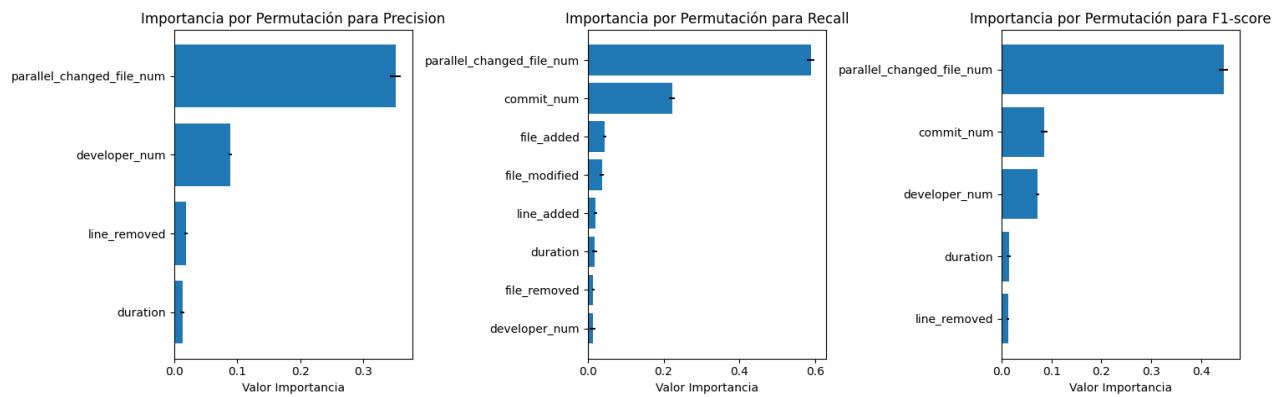


Fig. 77. Importancia por permutación Python + Random Forest

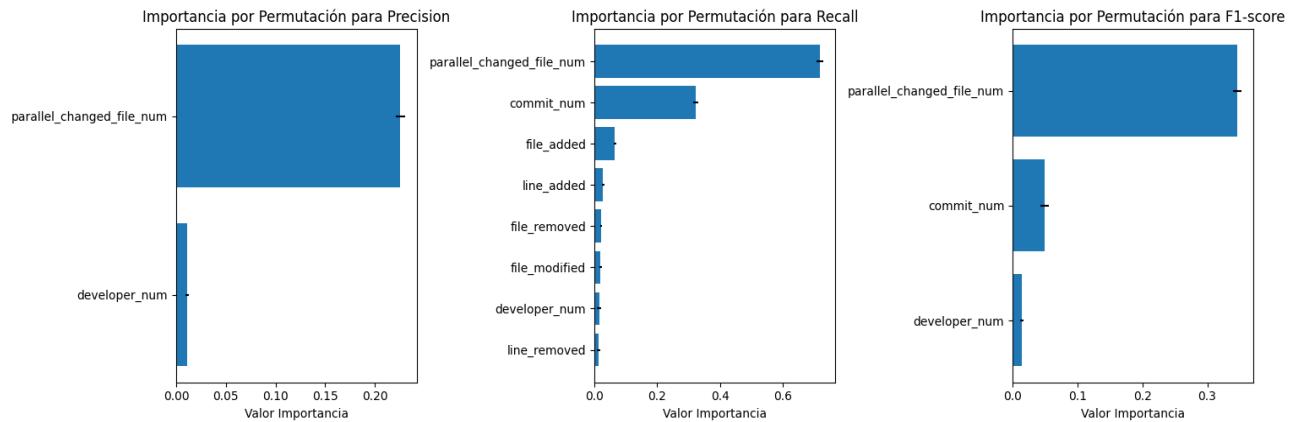
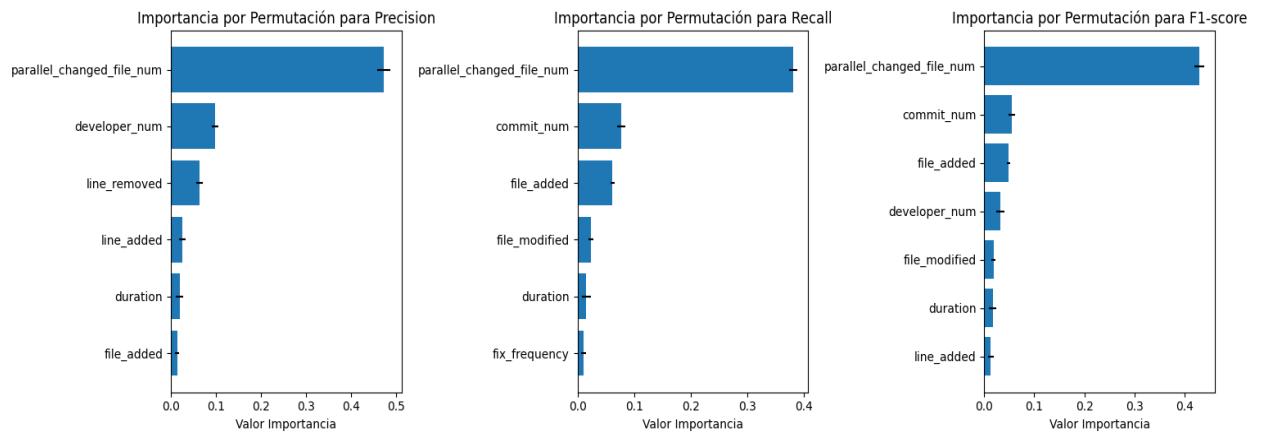
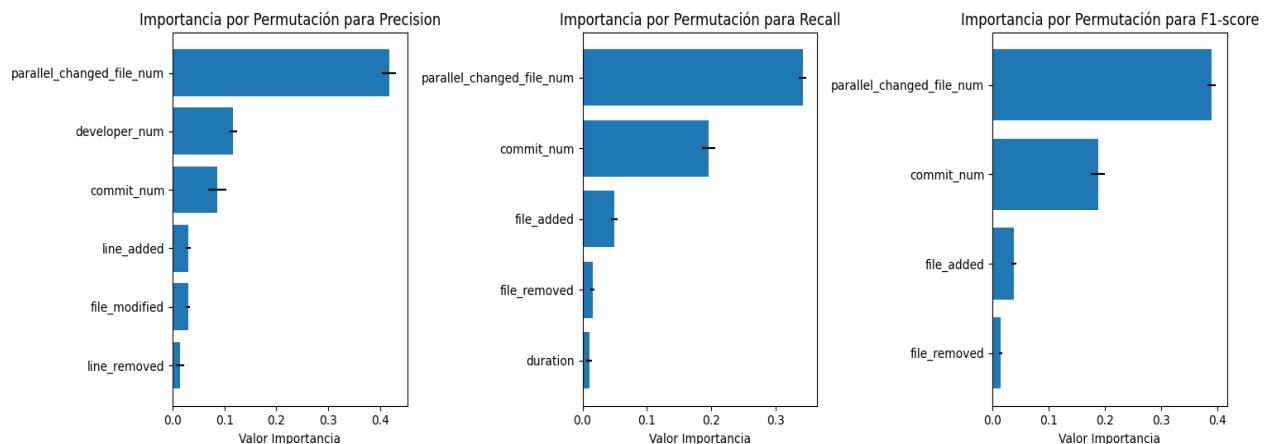


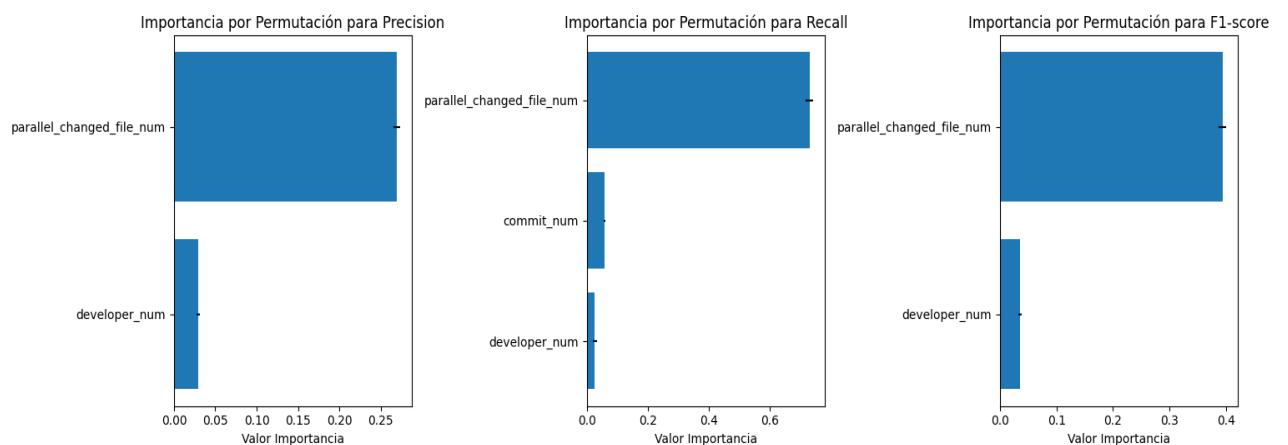
Fig. 78. Importancia por permutación Python + Balanced Random Forest



**Fig. 79. Importancia por permutación Python + Gradient Boosting**



**Fig. 80. Importancia por permutación Python + ADABoost**



**Fig. 81. Importancia por permutación Python + RUSBoost.**

La característica “*parallel\_changed\_file\_num*” es claramente la característica con mayor valor de importancia, para todos los modelos y métricas. De nuevo, “*commit\_num*” y “*developer\_num*” son las

dos siguientes para este lenguaje también. Sin embargo, tras estas tres primeras características hay diversidad, podemos encontrar las características “*file\_added*”, “*line\_added*”, “*file\_modified*”, etc. Pero la importancia de estas características es, en general, bastante inferior a las tres primeras. Los resultados en explicabilidad local para cada modelo y característica se presentan en la Tabla 30 a la Tabla 33.

**Tabla 30. Resultados explicabilidad local en el conjunto Python para verdaderos positivos.**

Característica	General		RF		BRF		GB		ADA		RUS	
	Rank Final	Cont Final	Rank	Cont								
<i>parallel_changed_file_num</i>	1	45	1	9	1	9	1	9	1	9	1	9
<i>commit_num</i>	2	43	2	9	2	9	2	7	2	9	3	9
<i>developer_num</i>	3	35	3	9	5	5	4	7	5	5	4	9
<i>file_removed</i>	4	18	6	4	4	4	7	5	3	5	-	-
<i>file_modified</i>	5	15	4	4	3	5	9	2	6	4	-	-
<i>file_added</i>	6	13	5	3	5	4	6	3	7	3	-	-
<i>line_removed</i>	7	14	8	2	8	1	5	7	4	4	-	-
<i>commit_density</i>	8	2	-	-	-	-	3	2	-	-	-	-
<i>line_added</i>	9	14	-	-	10	1	-	-	8	4	2	9
<i>messages_min</i>	10	8	-	-	8	1	-	-	-	-	5	7
<i>messages_max</i>	11	2	6	2	-	-	-	-	-	-	-	-
<i>duration</i>	12	5	9	1	7	3	10	1	-	-	-	-
<i>delete_frequency</i>	13	3	-	-	10	1	-	-	-	-	6	2
<i>messages_median</i>	14	3	9	1	-	-	-	-	9	2	-	-
<i>remove_frequency</i>	15	1	-	-	-	-	7	1	-	-	-	-
<i>bug_frequency</i>	16	1	9	1	-	-	-	-	-	-	-	-
<i>fix_frequency</i>	17	1	-	-	10	1	-	-	-	-	-	-
<i>add_frequency</i>	17	1	-	-	10	1	-	-	-	-	-	-
<i>improve_frequency</i>	17	1	-	-	-	-	10	1	-	-	-	-

**Tabla 31. Resultados explicabilidad local del conjunto Python para verdaderos negativos.**

Característica	General		RF		BRF		GB		ADA		RUS	
	Rank Final	Cont Final	Rank	Cont								
<i>parallel_changed_file_num</i>	1	44	1	9	1	9	1	8	1	9	1	9
<i>commit_num</i>	2	37	3	6	2	8	2	6	2	8	3	9
<i>developer_num</i>	3	37	2	7	3	7	3	6	4	8	2	9
<i>file_removed</i>	4	15	6	5	5	4	5	3	3	3	-	-
<i>file_added</i>	5	18	4	6	13	2	6	7	8	3	-	-
<i>messages_min</i>	6	12	5	2	-	-	11	2	7	2	5	6
<i>line_added</i>	7	14	8	3	9	1	10	1	-	-	4	9
<i>messages_max</i>	8	6	9	1	4	2	4	3	-	-	-	-
<i>improve_frequency</i>	9	6	-	-	6	2	8	2	5	2	-	-

line_removed	10	8	-	-	8	2	7	4	10	2	-	-
delete_frequency	11	3	-	-	-	-	-	-	-	-	6	3
duration	12	7	7	2	9	1	11	2	11	2	-	-
messages_median	13	4	10	1	9	1	-	-	5	2	-	-
bug_frequency	14	4	10	1	7	3	-	-	-	-	-	-
file_modified	15	6	10	1	14	1	9	1	8	3	-	-
refactor_frequency	16	1	-	-	9	1	-	-	-	-	-	-
commit_density	17	1	-	-	-	-	-	-	12	1	-	-
fix_frequency	18	1	-	-	14	1	-	-	-	-	-	-

Tabla 32. Resultado explicabilidad del conjunto Python para falso positivos.

Característica	General		RF		BRF		GB		ADA		RUS	
	Rank Final	Cont Final	Rank	Cont								
parallel_changed_file_num	1	45	1	9	1	9	1	9	1	9	1	9
commit_num	2	45	2	9	2	9	2	9	2	9	2	9
developer_num	3	34	3	8	4	5	3	7	3	5	4	9
file_added	4	21	4	5	8	4	4	5	4	7	-	-
file_removed	5	19	4	6	3	6	6	4	6	3	-	-
line_added	6	18	-	-	10	1	-	-	4	8	3	9
line_removed	7	12	7	3	5	3	5	4	6	2	-	-
file_modified	8	9	4	4	9	2	9	1	8	2	-	-
messages_min	9	9	-	-	12	1	7	3	-	-	5	5
delete_frequency	10	2	-	-	-	-	-	-	-	-	6	2
commit_density	11	1	-	-	6	1	-	-	-	-	-	-
messages_max	11	3	-	-	7	2	9	1	-	-	-	-
duration	13	3	-	-	10	1	8	2	-	-	-	-
bug_frequency	14	1	8	1	-	-	-	-	-	-	-	-
fix_frequency	15	1	-	-	12	1	-	-	-	-	-	-

Tabla 33. Resultado explicabilidad local del conjunto Python para falsos negativos.

Característica	General		RF		BRF		GB		ADA		RUS	
	Rank Final	Cont Final	Rank	Cont								
parallel_changed_file_num	1	45	1	9	1	9	1	9	1	9	1	9
developer_num	2	44	2	9	3	9	2	8	2	9	2	9
commit_num	3	35	3	7	2	8	4	3	3	8	3	9
file_modified	4	14	4	3	4	4	7	4	6	3	-	-
line_added	5	20	9	1	6	4	3	4	8	2	4	9
messages_min	6	6	-	-	-	-	-	-	-	-	5	6
file_removed	6	12	4	2	5	3	8	4	4	3	-	-
duration	8	13	4	5	8	3	6	3	5	2	-	-
file_added	9	18	4	6	10	1	4	9	10	2	-	-
delete_frequency	10	3	-	-	-	-	-	-	-	-	6	3
line_removed	11	6	8	1	7	2	9	1	11	2	-	-

<b>commit_density</b>	12	4	-	-	9	1	-	-	6	3	-	-
<b>use_frequency</b>	13	3	-	-	10	1	-	-	8	2	-	-
<b>bug_frequency</b>	14	1	9	1	-	-	-	-	-	-	-	-

- **Conjunto Ruby**

En el conjunto Ruby, al igual que en los conjuntos anteriores, la característica “*parallel\_changed\_file\_num*” ocupa el primer puesto en importancia en todos los casos. Las siguientes características en importancia por permutación están lejos del valor que obtiene “*parallel\_changed\_file\_num*” y destacando la característica “*commit\_num*” en el segundo puesto. De hecho, para los modelos *ADABoost* y *RUSBoost* el método de importancia por permutación solo dan importancia únicamente a “*parallel\_changed\_file\_num*”. Los resultados se pueden ver en la Fig. 82 - Fig. 86.

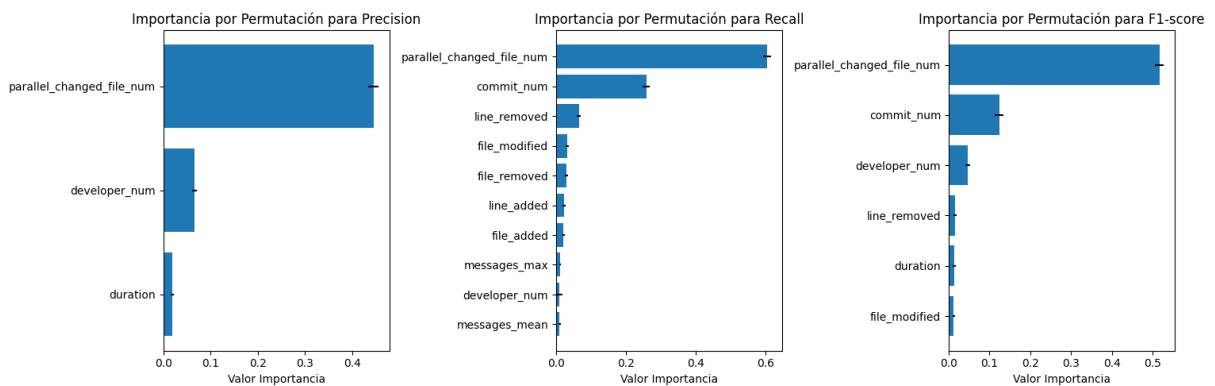


Fig. 82. Importancia por permutación Ruby + Random Forest

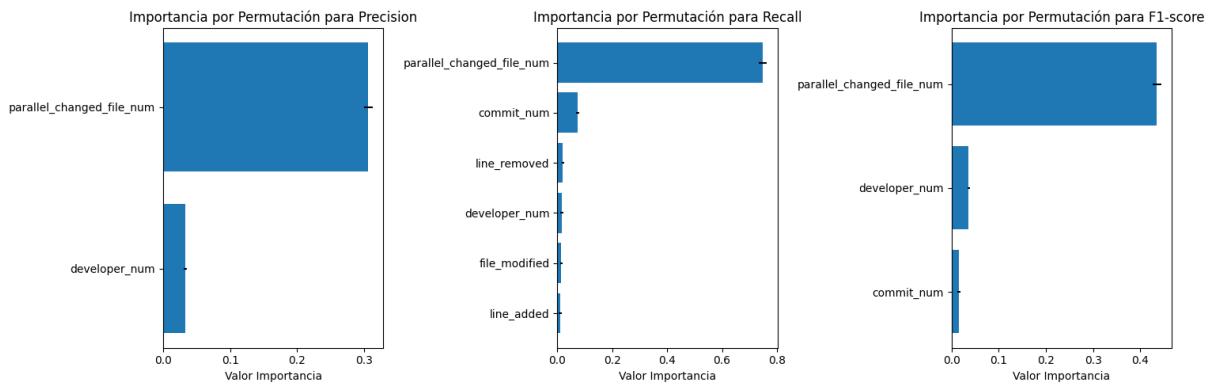
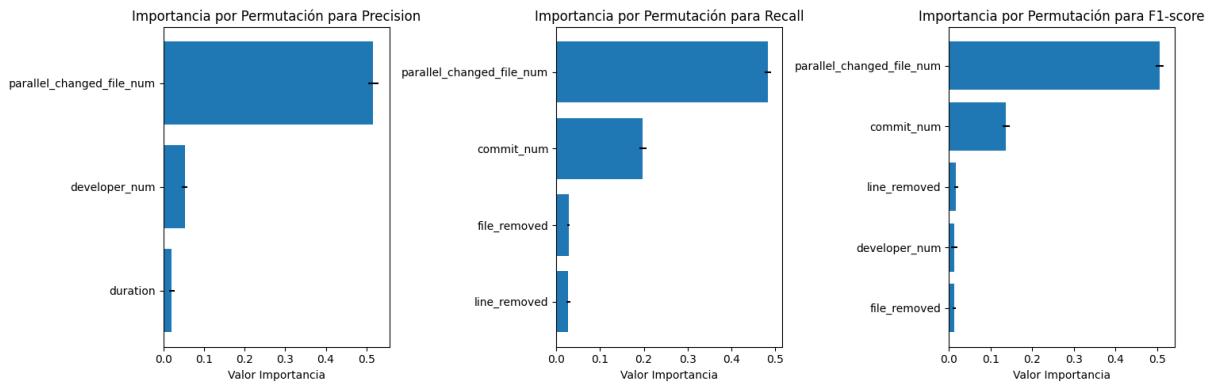
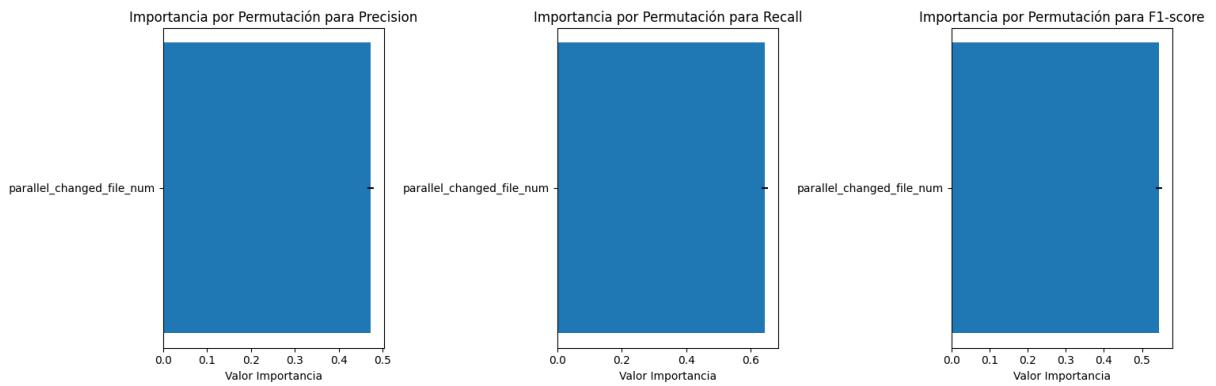


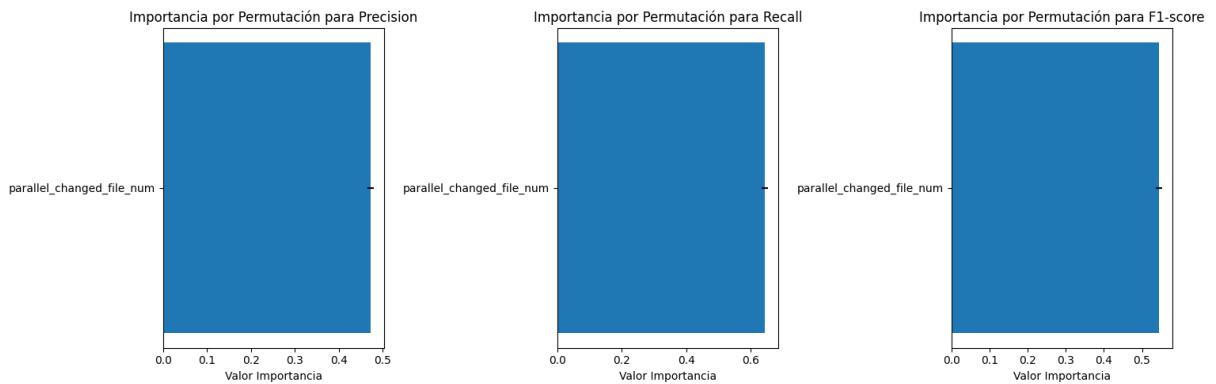
Fig. 83. Importancia por permutación Ruby + Balanced Random Forest



**Fig. 84. Importancia por permutación Ruby + Gradient Boosting**



**Fig. 85. Importancia por permutación Ruby + ADABoost**



**Fig. 86. Importancia por permutación Ruby + RUSBoost**

Al comparar los resultados de explicabilidad local, presentes en las Tabla 34, Tabla 35, Tabla 36 y Tabla 37, con los resultados de explicabilidad global vistos anteriormente observamos que coinciden al considerar “*parallel\_changed\_file\_num*” como la característica que más contribuye en las predicciones estudiadas. Por otro lado, podemos ver que las características “*developer\_num*” y “*commit\_num*” se encuentran en los siguientes puestos con un menor valor de contribución y número de apariciones en las técnicas de explicabilidad local.

Tabla 34. Resultados explicabilidad local del conjunto Ruby para verdadero positivo.

Característica	General		RF		BRF		GB		ADA		RUS	
	Rank Final	Cont Final	Rank	Cont								
parallel_changed_file_num	1	45	1	9	1	9	1	9	1	9	1	9
developer_num	2	29	3	8	2	6	3	6	2	9	-	-
commit_num	3	21	2	9	3	5	2	7	-	-	-	-
file_removed	4	21	8	3	5	4	4	5	5	6	2	3
duration	5	13	5	3	8	3	5	5	6	1	6	1
file_modified	5	13	6	6	4	4	8	3	-	-	-	-
improve_frequency	7	4	-	-	-	-	7	2	-	-	4	2
feature_frequency	8	5	-	-	11	1	-	-	4	1	3	3
line_removed	9	9	4	4	6	3	11	2	-	-	-	-
line_added	10	8	7	2	8	3	6	3	-	-	-	-
fix_frequency	10	5	-	-	11	1	-	-	3	3	6	1
use_frequency	12	3	-	-	7	1	-	-	-	-	5	2
remove_frequency	13	3	-	-	14	1	-	-	6	1	6	1
messages_max	14	3	-	-	14	1	9	2	-	-	-	-
file_added	15	2	-	-	10	2	-	-	-	-	-	-
messages_mean	16	1	9	1	-	-	-	-	-	-	-	-
commit_density	17	2	-	-	11	1	10	1	-	-	-	-

Tabla 35. Resultados explicabilidad local del conjunto Ruby para verdaderos negativos.

Característica	General		RF		BRF		GB		ADA		RUS	
	Rank Final	Cont Final	Rank	Cont								
parallel_changed_file_num	1	45	1	9	1	9	1	9	1	9	1	9
developer_num	2	30	2	9	5	6	2	6	2	9	-	-
commit_num	2	21	3	8	2	7	3	6	-	-	-	-
file_modified	4	16	5	4	4	6	4	6	-	-	-	-
line_removed	5	15	4	5	3	4	7	6	-	-	-	-
file_removed	6	16	10	1	6	3	5	3	5	6	2	3
use_frequency	7	5	6	2	-	-	-	-	7	1	7	2
fix_frequency	8	3	7	1	-	-	-	-	3	1	5	1
remove_frequency	9	4	-	-	10	2	-	-	3	1	4	1
duration	10	11	7	3	9	4	6	4	-	-	-	-
feature_frequency	10	5	-	-	11	1	-	-	7	1	3	3
improve_frequency	12	1	-	-	-	-	-	-	-	-	5	1
commit_density	13	5	10	1	-	-	-	-	6	2	7	2
messages_max	14	3	7	1	-	-	8	2	-	-	-	-
bug_frequency	15	1	-	-	-	-	-	-	7	1	-	-
messages_mean	15	1	-	-	7	1	-	-	-	-	-	-
line_added	17	3	10	1	8	1	9	1	-	-	-	-
messages_min	18	2	-	-	-	-	10	2	-	-	-	-
add_frequency	19	1	-	-	-	-	-	-	-	-	9	1

<b>update_frequency</b>	20	1	-	-	11	1	-	-	-	-	-	-
-------------------------	----	---	---	---	----	---	---	---	---	---	---	---

Tabla 36. Resultados explicabilidad local del conjunto Ruby para falsos positivos.

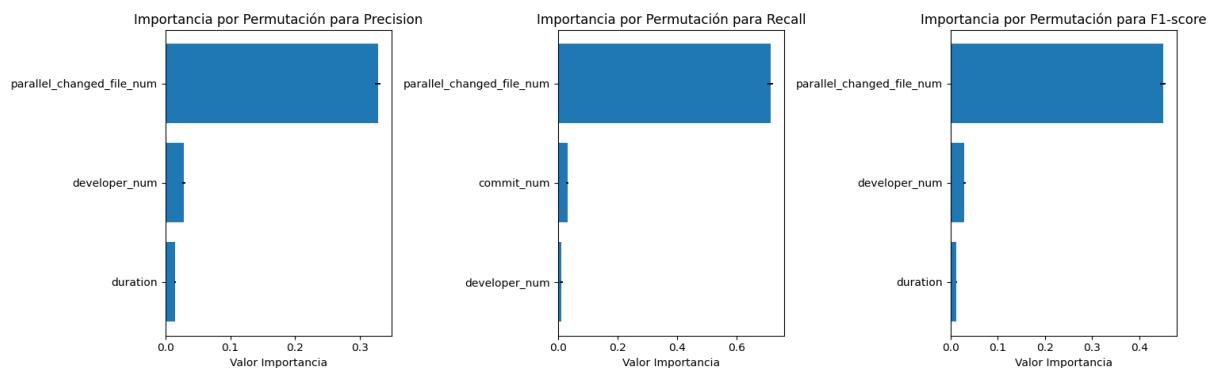
Característica	General		RF		BRF		GB		ADA		RUS	
	Rank Final	Cont Final	Rank	Cont								
<b>parallel_changed_file_num</b>	1	45	1	9	1	9	1	9	1	9	1	9
<b>commit_num</b>	2	24	2	9	2	8	2	7	-	-	-	-
<b>developer_num</b>	3	28	3	9	8	3	4	7	2	9	-	-
<b>line_removed</b>	4	18	5	7	2	7	8	4	-	-	-	-
<b>file_removed</b>	5	20	9	2	5	6	7	3	4	6	2	3
<b>file_modified</b>	6	8	4	2	7	1	3	5	-	-	-	-
<b>fix_frequency</b>	7	4	-	-	-	-	-	-	3	3	5	1
<b>improve_frequency</b>	8	4	-	-	-	-	-	-	5	1	4	3
<b>duration</b>	9	9	5	3	10	2	5	3	-	-	7	1
<b>commit_density</b>	10	10	10	1	9	2	5	3	7	2	6	2
<b>file_added</b>	11	4	8	1	4	3	-	-	-	-	-	-
<b>line_added</b>	12	6	-	-	5	3	9	3	-	-	-	-
<b>bug_frequency</b>	13	1	-	-	-	-	-	-	5	1	-	-
<b>feature_frequency</b>	14	4	-	-	11	1	-	-	-	-	3	3
<b>messages_max</b>	15	2	7	2	-	-	-	-	-	-	-	-
<b>messages_min</b>	16	1	-	-	-	-	10	1	-	-	-	-

Tabla 37. Resultados explicabilidad local del conjunto Ruby para falsos negativos.

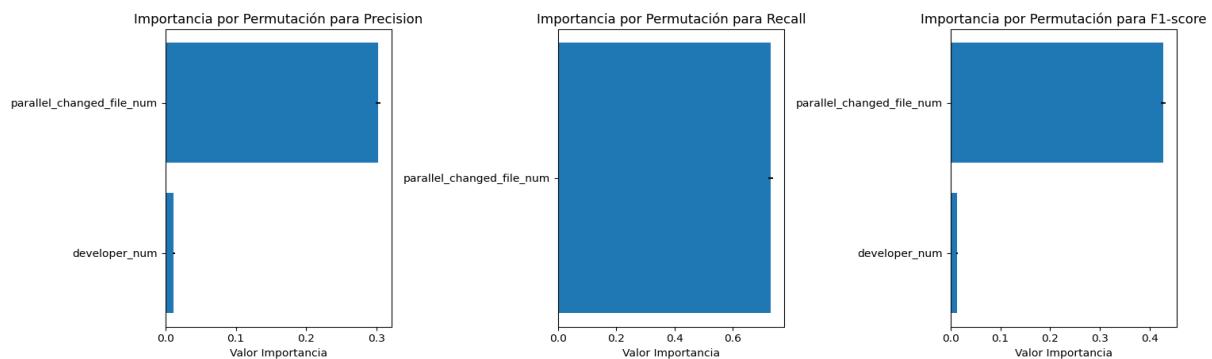
Característica	General		RF		BRF		GB		ADA		RUS	
	Rank Final	Cont Final	Rank	Cont								
<b>parallel_changed_file_num</b>	1	45	1	9	1	9	1	9	1	9	1	9
<b>commit_num</b>	2	24	2	9	3	7	2	8	-	-	-	-
<b>developer_num</b>	2	29	3	7	2	6	3	7	2	9	-	-
<b>file_removed</b>	4	20	9	3	9	5	4	3	4	6	2	3
<b>fix_frequency</b>	5	5	-	-	-	-	-	-	3	3	4	2
<b>commit_density</b>	6	12	4	4	5	2	5	2	7	2	7	2
<b>line_removed</b>	7	12	5	4	4	3	8	5	-	-	-	-
<b>improve_frequency</b>	8	4	-	-	-	-	-	-	6	1	3	3
<b>duration</b>	9	12	7	3	6	5	6	4	-	-	-	-
<b>file_modified</b>	10	11	6	3	7	4	7	4	-	-	-	-
<b>remove_frequency</b>	11	2	-	-	-	-	-	-	5	1	6	1
<b>feature_frequency</b>	12	4	-	-	10	2	-	-	-	-	4	2
<b>messages_max</b>	13	3	10	1	7	2	-	-	-	-	-	-
<b>messages_mean</b>	14	3	-	-	-	-	9	3	-	-	-	-
<b>line_added</b>	15	2	8	2	-	-	-	-	-	-	-	-

- **Conjunto PHP**

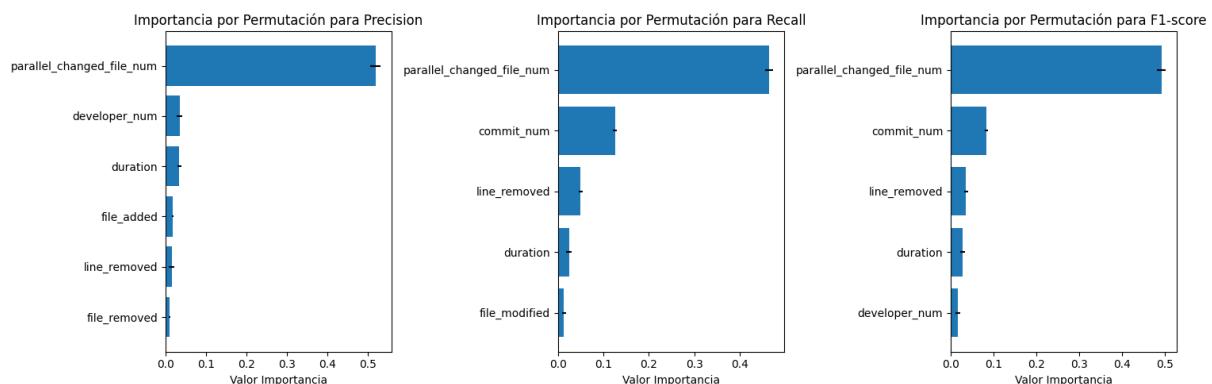
El conjunto PHP sigue el mismo patrón que los conjuntos anteriores. Esto significa que en explicabilidad global, (Fig. 87 - Fig. 91) “parallel\_changed\_file\_num” lidera en valores de importancia por amplia diferencia, mientras que “commit\_num” y “developer\_num” la siguen en importancia pero con valores muy bajos. En ocasiones aparecen características como “duration”, “file\_added”, “line\_removed”, “file\_removed” o “file\_modified” pero con valores de importancia por permutación poco relevantes.



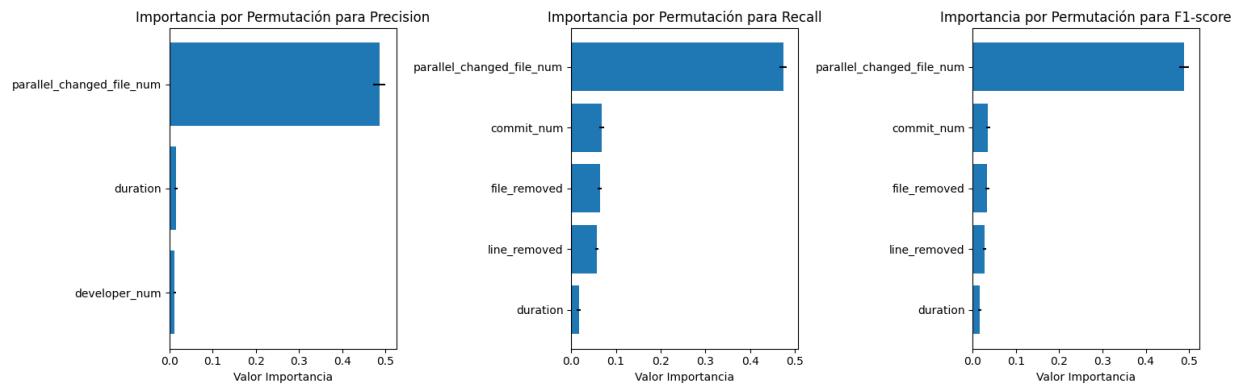
**Fig. 87. Importancia por permutación PHP + Random Forest**



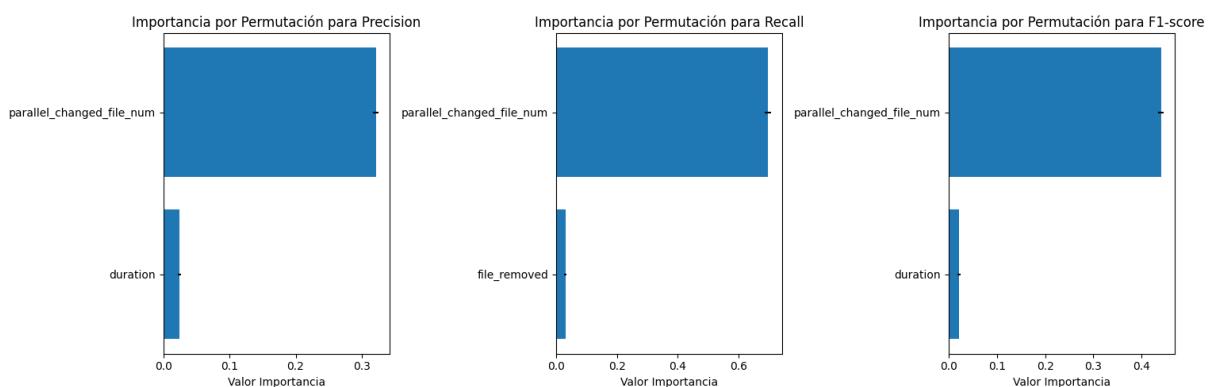
**Fig. 88. Importancia por permutación PHP + Balanced Random Forest**



**Fig. 89. Importancia por permutación PHP + Gradient Boosting.**



**Fig. 90. Importancia por permutación PHP + ADABoost**



**Fig. 91. Importancia por permutación PHP + RUSBoost**

Las deducciones sacadas de los resultados en explicabilidad global quedan confirmados por los resultados obtenidos con las técnicas de explicabilidad local (Tabla 38 - Tabla 41), donde “parallel\_changed\_file\_num” es la característica con mayor contribución en todos los modelos y con el mayor número de apariciones en los resultados con diferencias respecto al resto. En cuanto al resto de características en este caso “commit\_num” y “developer\_num” a pesar de que siguen estando por encima de otras características en rango y número de apariciones están igualadas con otras como “line\_removed”, “duration” y “file\_removed”.

**Tabla 38. Resultados explicabilidad local del conjunto PHP para verdadero positivos.**

Característica	General		RF		BRF		GB		ADA		RUS	
	Rank Final	Cont Final	Rank	Cont								
parallel_changed_file_num	1	45	1	9	1	9	1	9	1	9	1	9
commit_num	2	29	2	8	2	8	3	5	3	8	-	-
line_removed	3	26	3	5	4	4	2	9	2	8	-	-
developer_num	4	34	6	6	7	5	6	6	4	8	3	9
duration	5	25	4	4	6	3	4	4	5	6	4	8
bug_frequency	6	7	4	4	3	2	-	-	-	-	6	1
commit_density	7	5	6	2	5	3	-	-	-	-	-	-
file_removed	8	19	10	2	10	4	8	1	6	3	2	9
file_modified	9	9	9	2	11	3	5	4	-	-	-	-
improve_	10	3	6	1	8	1	-	-	-	-	6	1

frequency													
document_frequency	11	1	-	-	-	-	-	-	-	-	5	1	
messages_max	12	3	-	-	-	-	7	3	-	-	-	-	-
line_added	13	4	12	1	8	1	-	-	7	2	-	-	-
delete_frequency	14	1	-	-	-	-	-	-	-	-	6	1	
file_added	15	3	-	-	12	1	10	2	-	-	-	-	-
refactor_frequency	16	1	-	-	-	-	8	1	-	-	-	-	-
add_frequency	17	2	-	-	13	1	11	1	-	-	-	-	-
fix_frequency	18	1	11	1	-	-	-	-	-	-	-	-	-

Tabla 39. Resultados explicabilidad local del conjunto PHP para verdaderos negativos.

característica	General		RF		BRF		GB		ADA		RUS	
	Rank Final	Cont Final	Rank	Cont								
parallel_changed_file_num	1	45	1	9	1	9	1	9	1	9	1	9
developer_num	2	41	2	9	4	6	2	8	3	9	3	9
duration	3	38	4	6	2	6	3	9	2	9	3	8
commit_num	4	22	3	6	3	6	6	3	4	7	-	-
file_removed	5	23	8	2	5	7	-	-	7	5	2	9
line_removed	6	17	5	7	6	4	4	6	-	-	-	-
file_modified	7	10	6	5	7	2	-	-	5	3	-	-
file_added	8	5	-	-	-	-	6	4	-	-	6	1
document_frequency	9	1	-	-	-	-	-	-	-	-	5	1
line_added	10	7	7	1	9	1	8	2	6	3	-	-
messages_max	11	3	-	-	9	1	5	2	-	-	-	-
bug_frequency	12	1	-	-	-	-	-	-	-	-	6	1
messages_median	13	4	-	-	8	2	9	2	-	-	-	-
fix_frequency	14	1	-	-	-	-	-	-	-	-	8	1
messages_min	15	1	-	-	9	1	-	-	-	-	-	-

Tabla 40. Resultado explicabilidad local del conjunto PHP para falsos positivos.

característica	General		RF		BRF		GB		ADA		RUS	
	Rank Final	Cont Final	Rank	Cont								
parallel_changed_file_num	1	45	1	9	1	9	1	9	1	9	1	9
commit_num	2	26	2	7	2	6	5	5	2	8	-	-
developer_num	2	35	4	6	3	5	2	7	3	8	3	9
line_removed	4	20	3	9	4	4	4	4	8	3	-	-
duration	5	20	7	3	8	1	2	5	7	4	3	7
file_modified	6	14	5	4	7	6	6	2	5	2	-	-
file_removed	7	23	8	1	9	5	9	2	6	6	2	9
line_added	8	11	6	3	10	3	-	-	4	5	-	-
document_frequency	9	4	-	-	-	-	8	2	-	-	5	2
feature_frequency	9	2	-	-	5	2	-	-	-	-	-	-

messages_max	11	4	-	-	-	-	7	3	-	-	7	1
add_frequency	11	4	-	-	5	2	9	2	-	-	-	-
delete_frequency	13	1	-	-	-	-	-	-	-	-	6	1
fix_frequency	14	3	9	2	-	-	12	1	-	-	-	-
bug_frequency	15	1	-	-	-	-	-	-	-	-	7	1
improve_frequency	15	2	10	1	-	-	12	1	-	-	-	-
file_added	17	2	-	-	12	1	11	1	-	-	-	-
messages_median	18	1	-	-	11	1	-	-	-	-	-	-
messages_min	19	1	-	-	-	-	12	1	-	-	-	-

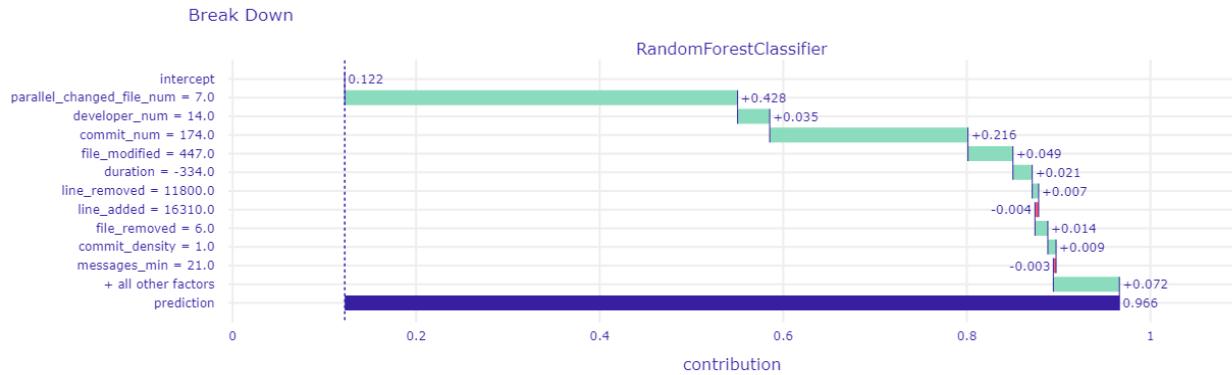
Tabla 41. Resultados explicabilidad local del conjunto PHP para falsos negativos.

característica	General		RF		BRF		GB		ADA		RUS	
	Rank Final	Cont Final	Rank	Cont								
parallel_changed_file_num	1	45	1	9	1	9	1	9	1	9	1	9
developer_num	2	42	2	8	3	8	2	9	4	8	4	9
commit_num	3	25	3	5	2	7	5	6	3	7	-	-
file_removed	3	29	4	3	4	7	6	2	5	8	2	9
duration	5	28	5	5	8	3	7	3	2	9	3	8
line_removed	6	18	6	6	7	5	3	6	7	1	-	-
messages_max	7	2	-	-	-	-	4	2	-	-	-	-
file_modified	8	9	8	2	5	1	7	2	6	3	5	1
file_added	9	9	7	2	6	3	7	4	-	-	-	-
document_frequency	10	2	-	-	-	-	-	-	-	-	6	2
bug_frequency	11	5	9	3	9	1	-	-	-	-	7	1
line_added	12	4	10	1	9	1	7	2	-	-	-	-
improve_frequency	13	1	10	1	-	-	-	-	-	-	-	-

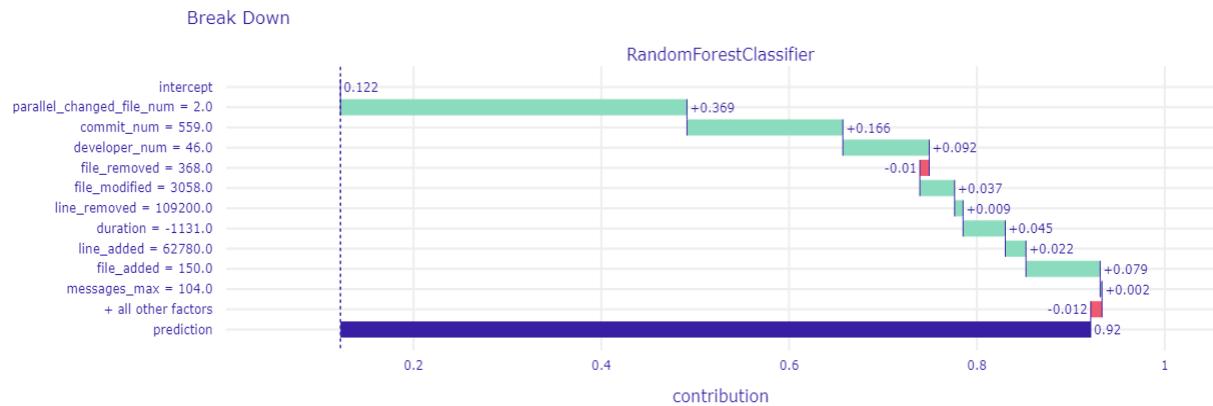
Al igual que en la parte de estimación de esfuerzo al comparar los resultados a nivel de instancia, es decir, dentro de un mismo resultado de clasificación pero distinto nivel de probabilidad mediante los gráficos de las técnicas de explicabilidad local se han encontrado similitudes en todos los modelos y conjuntos de datos. Por lo tanto, en esta sección se va a realizar un tratamiento común para todos los modelos y conjuntos de datos mediante unos ejemplos representativos de los hallazgos, mientras que una colección parcial con otros gráficos obtenidos como resultado se pueden consultar en el Anexo 4.

El fenómeno observado es la relación entre el número de características que contribuyen a una determinada predicción en un determinado sentido (signo de la contribución), al igual que el valor de la contribución. La Fig. 92, Fig. 93 y Fig. 94 se corresponden con los resultados de las técnicas *Break-down* aplicadas a la instancia verdadero positivo del conjunto Python con probabilidad máxima, en la mediana y mínima, respectivamente. Como se puede ver la mayoría de las características representadas contribuyen de forma positiva. Sin embargo, cuando pasamos a la instancia verdadero positivo con probabilidad en la mediana del mismo modelo y conjunto de datos podemos ver que o bien empiezan a aparecer características que contribuyen de forma negativa o características que antes lo hacían de forma positiva ahora lo hacen de forma negativa. Además, aquellas características que contribuyen de

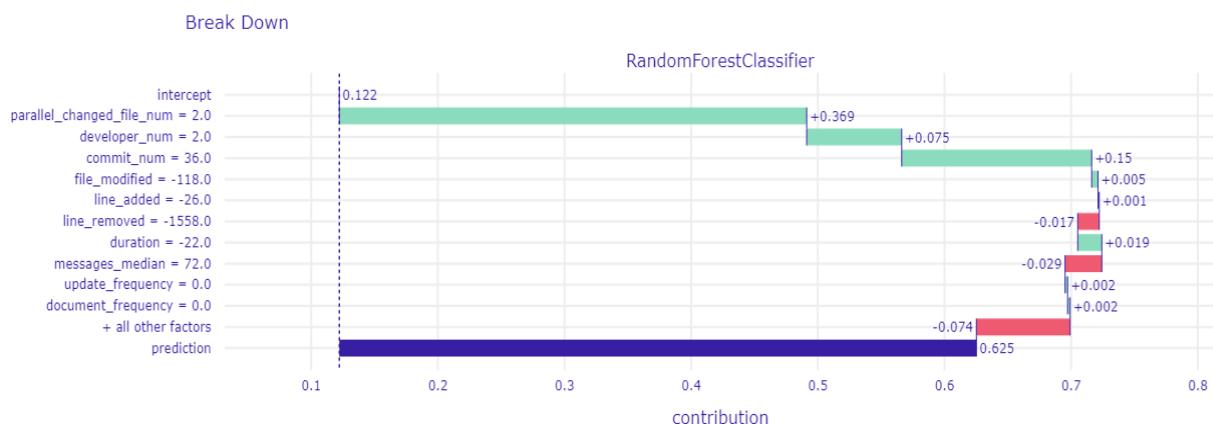
forma positiva lo hacen con un valor menor comparado con el caso de probabilidad máxima. Para el caso de la instancia de probabilidad mínima este hecho se acentúa y se hace más evidente.



**Fig. 92. Break-down en el conjunto Python para el modelo Random Forest y la instancia verdadero positivo de probabilidad máxima.**



**Fig. 93. Break-down en el conjunto Python para el modelo Random Forest y la instancia verdadero positivo con probabilidad en la mediana.**



**Fig. 94. Break-down en el conjunto Python para el modelo Random Forest y la instancia verdadero positivo con probabilidad mínima.**

Por otro lado, para los casos de resultados negativos (verdadero negativo y falso negativo) este comportamiento también tiene lugar pero en sentido contrario al expuesto, lo cual se puede observar mediante la técnica LIME del conjunto Java para el modelo *Balanced Random Forest* y la instancia falso negativo que se recoge en la Fig. 95 y Fig. 96.

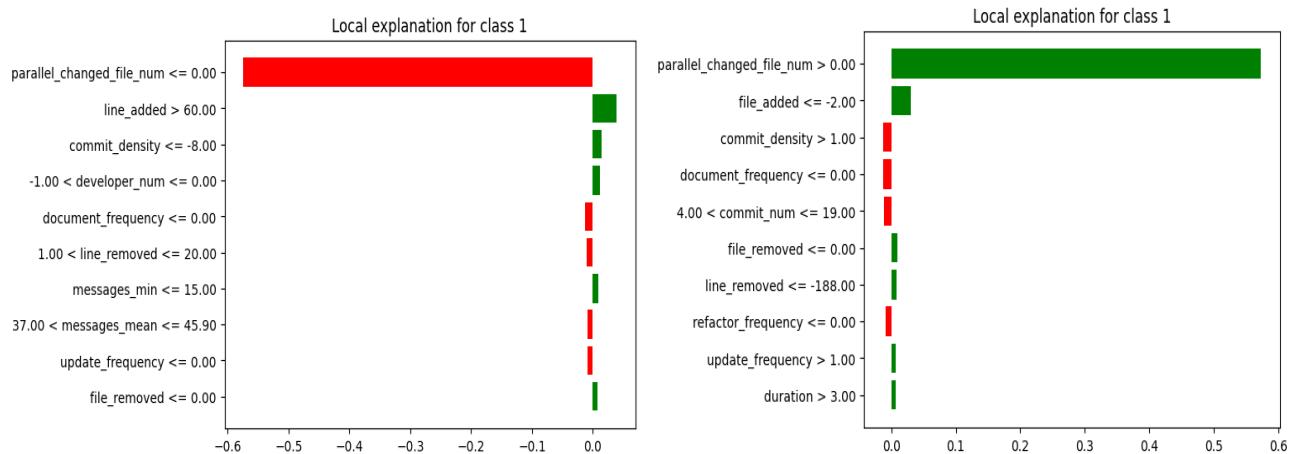


Fig. 95. En la izquierda se representa la técnica LIME en el conjunto Java para el modelo *Balanced Random Forest* y la instancia falso negativo con probabilidad mínima. A la derecha esa misma técnica, modelo y conjunto de datos pero para la instancia con probabilidad en la mediana.

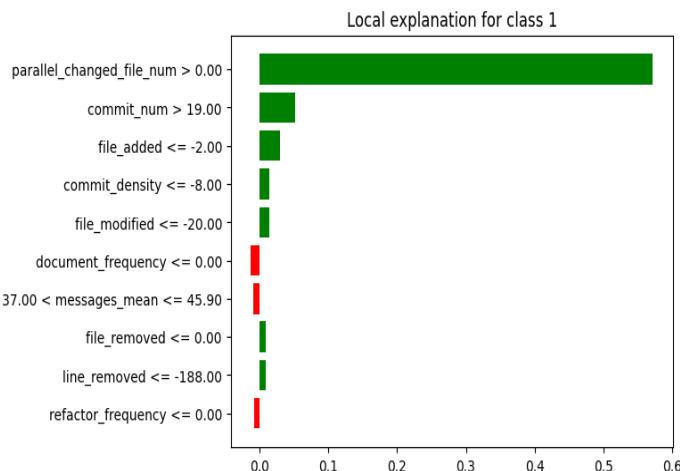


Fig. 96. LIME en el conjunto Java para el modelo *Balanced Random Forest* y la instancia falso negativo con probabilidad máxima.

Tras la comparación a nivel de instancias y observando las figuras anteriores se puede inferir cómo es la comparación a nivel del tipo de resultados (verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos). Cuando estamos estudiando instancias que son “positivos” hay un mayor influjo de contribuciones positivas, especialmente en los casos de probabilidad máxima y que decrece al disminuir la probabilidad. Para el caso de los “negativos” es al contrario, existe un mayor influjo de contribuciones negativas. Como se ha comentado previamente se pueden consultar otros gráficos que avalan estos hallazgos el Anexo 4. En las Fig. 97 - Fig. 100 solo se presenta una demostración del este comportamiento.

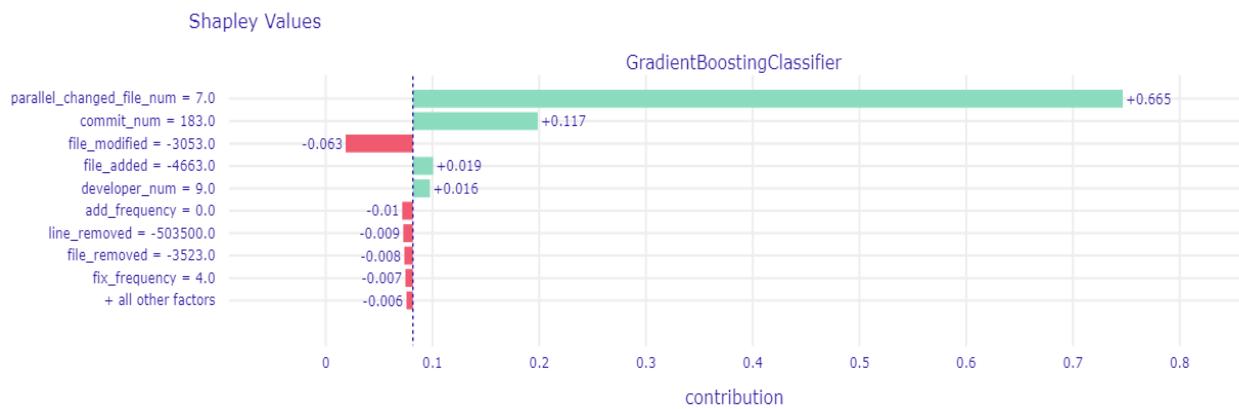


Fig. 97. Shapley Values en el conjunto Java para el modelo Gradient Boosting y la instancia verdadero positivo en la mediana.

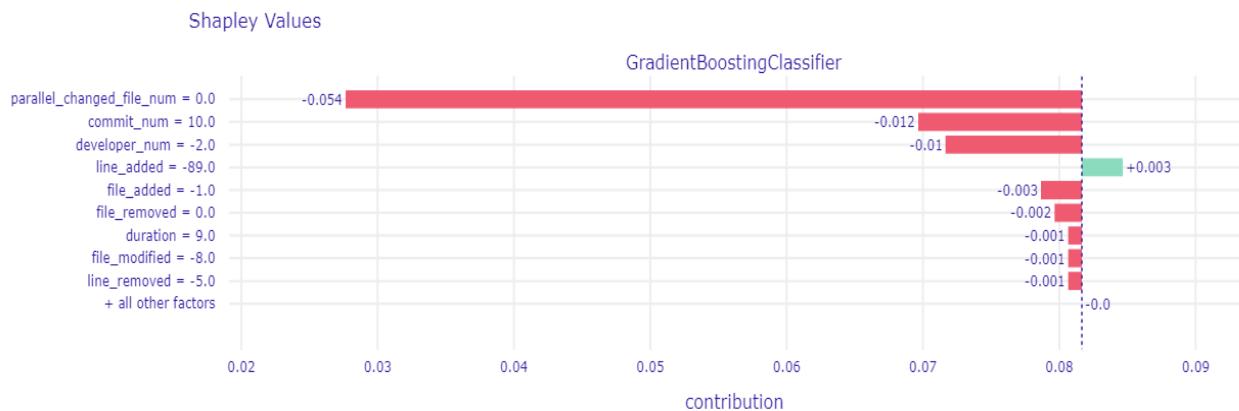


Fig. 98. Shapley Values en el conjunto Java para el modelo Gradient Boosting y la instancia verdadero negativo en la mediana.

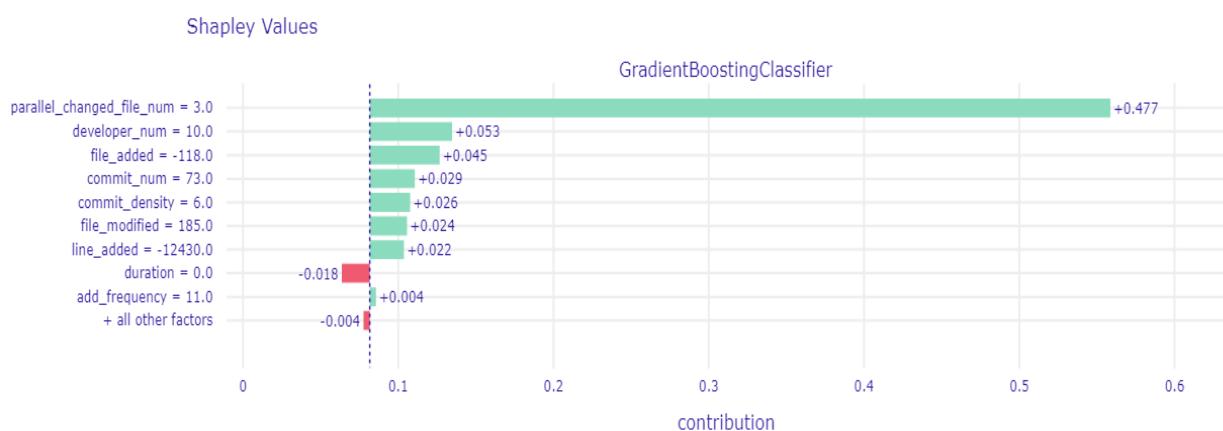
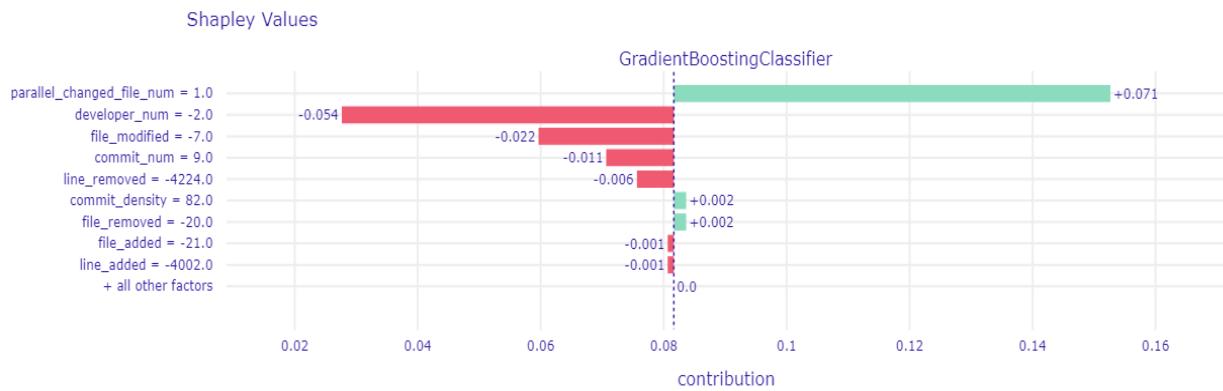
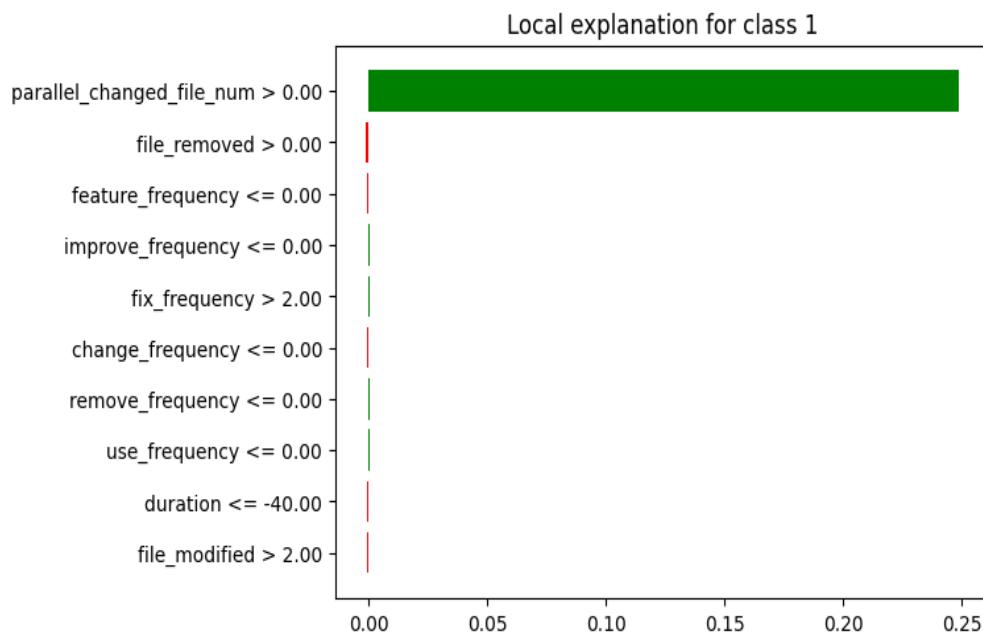


Fig. 99. Shapley Values en el conjunto Java para el modelo Gradient Boosting y la instancia falso positivo en la mediana.



**Fig. 100.** Shapley Values en el conjunto Java para el modelo Gradient Boosting y la instancia falso negativo en la mediana.

Por último, un hecho destacable que se ha observado en los conjuntos de esta parte es cómo los modelos, especialmente *ADABoost* y *RUSBoost* (con *ADABoost* como base), reducen la contribución en las técnicas de explicabilidad local de la mayoría de las instancias estudiadas prácticamente a una única característica, siendo lo esperado si observamos los resultados de explicabilidad global de los modelos en estos conjuntos de datos (Fig. 86, Fig. 85, Fig. 90 y Fig. 91) donde la característica “*parallel\_changed\_file\_num*” se muestra como la única con importancia o acompañada por otra características que tiene un valor ínfimo. A continuación se presentan unos ejemplos de este hecho en la Fig. 101, Fig. 102 y Fig. 103.



**Fig. 101.** LIME en el conjunto Ruby para el modelo RUSBoost y la instancia verdadero positivo con probabilidad máxima.

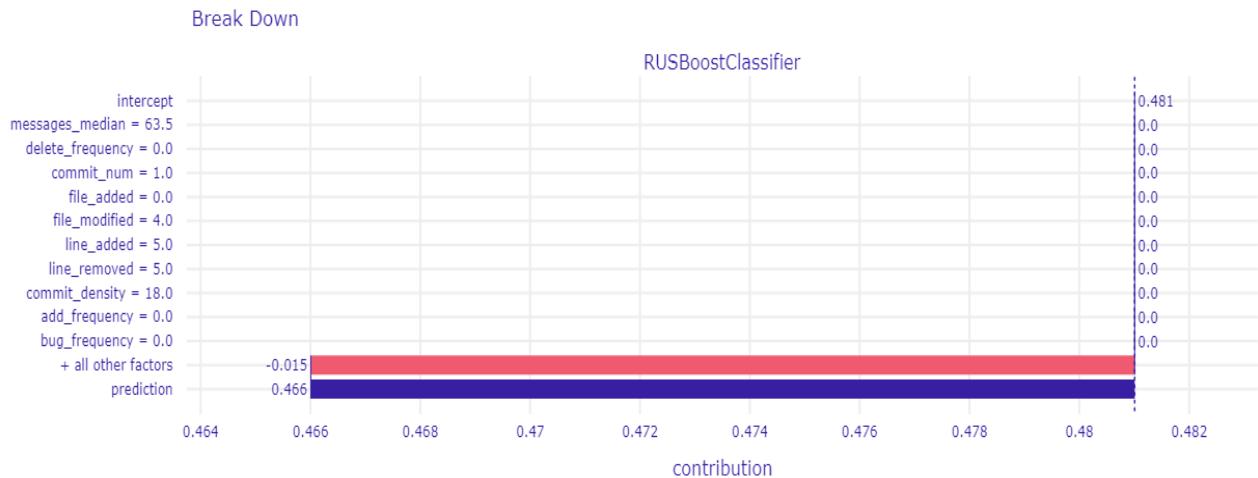


Fig. 102. Break-down en el conjunto PHP para el modelos RUSBoost y la instancia falso negativo con probabilidad en la mediana.

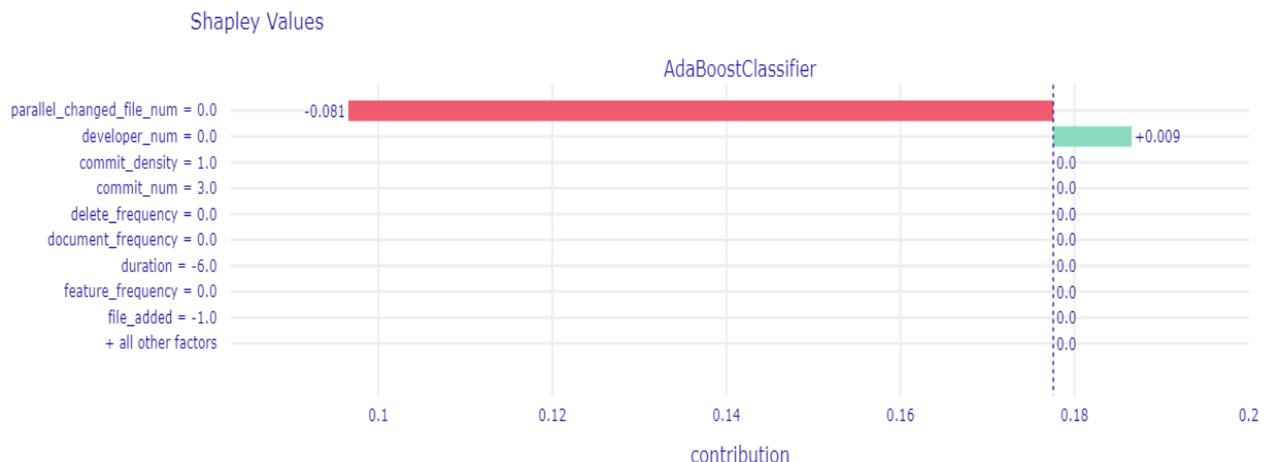


Fig. 103. Shapley Values en el conjunto Ruby para el modelos ADABoost y la instancia falso negativo con probabilidad máxima.

## 6.3. Conclusiones del análisis de resultados

Debido a la alta densidad de gráficas y tablas incluidas en la secciones de análisis de resultados anteriores puede resultar difícil tener claro que hallazgos se han extraído en dichas secciones. Por este motivo, en esta sección se recogen esos hallazgos y deducciones para permitir una visión rápida y clara de lo conseguido.

### 6.3.1. Análisis de datos

En el análisis de los datos para el problema de estimación de esfuerzo es complejo sacar conclusiones generales que abarquen todos los conjuntos, debido a que cada conjunto tiene una procedencia y características distintas. Pero se puede observar características que mantienen una alta correlación con aquellas de similar significado, como es el caso de características que hacen referencia a la experiencia del equipo y la experiencia del *manager* o aquellas correspondientes al número de entradas o salidas.

Otro ejemplo son las características del número de consultas, número de ficheros o de contenido modificado. Otro caso ya comentado son los conjuntos incluyen una característica y esa misma característica normalizada. En estos casos, como es lógico la correlación es muy alta ya que se trata de la misma característica, y este hecho ha llevado a su eliminación durante el preprocesado.

En el análisis de datos en el problema de clasificación se pueden hacer deducciones generales, ya que el método de recopilación de los datos y características es idéntico para todos los conjuntos. En este caso, es más evidente la correlación entre características de procedencia similar. Tenemos como ejemplo las características correspondientes a la media de mensajes, mediana de mensajes y máximo de mensajes. Aquellas características que hacen referencia a los ficheros o líneas de código también muestran alta correlación entre ellas. Un hecho que llama la atención es qué a pesar de contar con muchas características relacionadas con la frecuencia, éstas no muestran una correlación especialmente alta entre ellas.

### 6.3.2. Rendimiento predictivo

En la parte de rendimiento predictivo para la estimación de esfuerzo no hemos obtenido buenos resultados en el rendimiento de los modelos. También, hemos comprobado como el tamaño de los conjuntos de datos puede influir en los resultados dificultando obtener un rendimiento alto en mayoría de casos. Al no contener suficientes datos, estos conjuntos no permiten que los modelos extraigan información y aprender patrones o relaciones de forma adecuada. Tenemos como ejemplo que el conjunto CHINA, es más grande empleado en esta parte, consigue un rendimiento aceptable. Además, la mejora obtenida al aumentar el número de instancias del conjunto Atkinson apoya esta deducción. Otra causa puede ser la baja correlación de algunas características en los conjuntos de datos con respecto a la variable objetivo. Este hecho puede provocar que proporcionen poca información relevante, en algunos casos incluso pueden llegar a considerarse ruido y por lo tanto dificultan el buen desempeño de los modelos.

Por otro lado, en el ámbito de la predicción de conflictos los algoritmos de clasificación demostraron obtener resultados aceptables en la detección de casos afirmativos. Además, hemos confirmado que la métrica *Accuracy* no es adecuada en escenarios con gran desbalanceo de clases. Esto se debe a que consigue valores muy altos por el excesivo número de la clase mayoritaria mientras que el resto de las métricas, incluyendo *F1-score* que se considera adecuada estos escenarios, obtienen valores más bajos. Por otro lado, un hecho destacable es que no hemos conseguido mejorar el rendimiento mediante los algoritmos centrados en escenarios de desbalanceo los algoritmos convencionales. Vemos como los modelos “ADABOOST” y “Gradient Boosting” destacan en la métrica *F1-score*, mientras “Random Forest” y “RUSBoost” en la métrica *Recall*. Cuando se trató este mismo problema con las clases invertidas, es decir, centrándonos en la predicción de casos de “No conflicto” los resultados obtenidos fueron realmente buenos en todas las métricas. Esto era lo esperado debido principalmente al desbalanceo de clases que existía en todos los conjuntos de datos a favor de los casos de “No conflicto” y al construir los modelos para detectar los casos de “No conflicto” las métricas se disparan. Los resultados obtenidos son similares a los que recogen otros autores en su estudio [7].

### 6.3.3. Explicabilidad del modelo

Los resultados en explicabilidad cumplen con la hipótesis inicial. Las características que mostraban una mayor correlación coinciden con aquellas que consiguen valores altos de importancia por permutación (explicabilidad global). Además, esas características aparecen con frecuencia entre las

características destacadas por las técnicas *Break-down*, *Shapley Values* y LIME (explicabilidad local). Por lo tanto, los resultados son coherentes entre sí y coinciden que los esperado antes analizar los resultados de la experimentación. Centrándonos en las técnica de explicabilidad local, los resultados, en general, muestran una alta coincidencia entre las tres técnicas estudiadas. Esto se deduce al comprobar qué características ocupan los primeros puestos en el valor de contribución a la predicción en cada técnica, así como el signo de su contribución.

Otro hallazgo obtenido a partir de la técnicas de explicabilidad locales es cómo cambia la contribución de las características a nivel de instancia, es decir, en función del valor de la estimación en el caso de regresión o del valor de probabilidad en el caso de clasificación. Se ha observado que existe una relación entre el nivel de estimación o probabilidad y el número de características que contribuyen a la predicción en un determinado sentido (signo de la contribución), además del valor de esa contribución. Los casos de instancias con estimación máxima llevan asociado un mayor número de características contribuyendo de forma positiva y con valores mayores que los casos de instancia estudiadas con un valor de estimación menor. Este resultado era lo esperado ya que si la estimación para una instancia determinada es alta, se debe esperar que las contribuciones de sus características sean la consecuencia del impulso en el valor de la estimación de forma positiva para superar la predicción media de un modelo. Como es lógico a mayor valor de estimación en una instancia más se distancia de la predicción media en sentido positivo. De la misma forma, se espera que a medida que se reduce el nivel de estimación pasando a la instancia en la mediana, se reduce el valor de la contribución de las características que eran positivas, incluso características que antes contribuían de forma positiva ahora lo hacen de forma negativa. Entonces, se reduce el número de características que contribuyen de forma positiva y pueden aparecer características con contribuciones en sentido negativo que antes no tenían valores significativos. Este fenómeno se acentúa hasta llegar al caso de la instancia de estimación mínima donde la mayoría de las contribuciones de valor alto son negativas.

De forma análoga, en la parte de predicción de conflictos se observa el mismo comportamiento cuando dentro de un valor de predicción (verdadero positivo, verdadero negativo, falso positivo o falso negativo) se comparan los resultados en función de la probabilidad de la predicción. A mayor probabilidad de predicción mayor número de características en ese sentido y con mayor valor, las cuales disminuyen al disminuir la probabilidad de predicción. El razonamiento es similar al comentado previamente para los valores de estimación de esfuerzo: una mayor probabilidad para instancias del mismo resultado de predicción conlleva que los valores en sus características impulsen la predicción media del modelo hacia el sentido correspondiente, es decir, si es un “positivo” contribuyendo de forma positiva y si es un “negativo” contribuyendo de forma negativa. Cuando la probabilidad baja debe reducirse las contribuciones en ese sentido o aparecer nuevas características que contribuyen con signo contrario. Por otro lado, en el caso de la predicción de conflictos podemos encontrar otro fenómeno destacable que guarda relación con el comportamiento recién comentado. Sin embargo, tiene lugar al comparar los resultados entre tipos de resultado en la clasificación. Cuando la predicción del modelo pertenece a los “positivo” (verdadero positivo o falso positivo) también se observan un mayor número de características influyendo de forma positiva a la predicción media del modelo, y en mayor valor. Esto es más evidente en los casos de probabilidad máxima y que decae al disminuir la probabilidad como hemos visto. Debemos tener en cuenta que para que una instancia sea clasificada como positiva, en un escenario de clasificación binaria como el que se presenta aquí, debe de superar el umbral del 0.5 de probabilidad para esa clase. Es lógico esperar que los casos positivos presenten más contribuciones positivas pues deben de superar dicho umbral. De igual forma en los “negativos” (verdadero negativo y

falso negativo) pero estos deben de superar el mismo umbral pero en el sentido contrario para ser clasificados como negativos.

A pesar de información proporcionados por las técnicas de explicabilidad empleadas en el análisis de los resultados obtenidos en nuestro problema de clasificación, nos encontramos con un desafío que ha demostrado ser esquivo: la identificación del origen de los falsos positivos y falsos negativos. Estas técnicas Las técnicas de explicabilidad están diseñadas principalmente para arrojar luz sobre cómo el modelo de clasificación toma decisiones y cuál es la contribución relativa de cada característica en una predicción específica. Sin embargo, no se centran en explicar directamente el porqué el modelo ha cometido un error específico en el problema de clasificación. Los FP y FN no se han podido atribuir de manera directa a una única característica o patrón evidente, y en muchos casos el valor de probabilidad que los modelos dan a los FP o FN es mayor en los VP o VN. Por último, la experimentación arroja otro hecho destacable. En la explicabilidad de los modelos en el problema de predicción de conflictos, el número de características relevantes es muy bajo, prácticamente la contribución queda reducida a una única característica. Aunque aparecen otras, tanto en explicabilidad global como local, pero con valores mucho menores en contribución. En cambio, en el problema de estimación de esfuerzo aparecen un mayor número con valores relevantes. Esto llama la atención si tenemos en cuenta que, en general, el número de características de los conjuntos empleados en la parte de estimación es menor que en los conjuntos usados en clasificación.

## 7. Conclusiones

En el transcurso de este trabajo, hemos explorado y abordado dos desafíos críticos en el campo de la ingeniería de software: la estimación de esfuerzo y la predicción de conflictos durante la fusión de código. Nuestro enfoque se centró en la aplicación de técnicas de ML para resolver estos problemas fundamentales.

En primer lugar, nos propusimos evaluar la viabilidad del ML para abordar estos desafíos mediante el uso de diversos algoritmos de regresión, para la parte de estimación de esfuerzo, y de clasificación para la parte de predicción de conflictos. Esto acompañado del uso de conjuntos de datos relevantes en cada área y tomados de proyectos reales. Para la estimación de esfuerzo, los resultados no siempre alcanzaron buenos niveles en las métricas. Como se ha comentado, los motivos de estos resultados pueden deberse al tamaño y la baja correlación de algunas características con respecto a la variable objetivo en los conjuntos de datos con los que hemos trabajado. Sin embargo, hay que tener en cuenta que hemos tratado con conjuntos de datos procedentes de proyectos reales, lo que conlleva que son problemas complejos de tratar ya que suelen incluir factores muy complejos o ruido en los datos. Aun así, se ha comprobado que en casos con suficiente volumen de datos, el uso de ML para estimar el esfuerzo puede dar buenos resultados.

Los resultados proporcionados en este trabajo proporcionan información valiosa para el tratamiento de la estimación de esfuerzo y la predicción de conflictos mediante ML, hay que reconocer las limitaciones que presenta y que abre la puerta a futuras investigaciones y mejoras. Según avanzamos en el trabajo continuamos enfrentando desafíos que debíamos de superar para mejorar nuestros métodos y enfoques. Por un lado, el desbalanceo de clases en la parte de predicción de conflictos presentó un desafío que tiene evidente repercusión en los resultados. Aunque como se ha comprobado si se invierten las clases y se buscan predecir los casos de “No conflicto” los resultados son excelentes eso no

quita la necesidad de desarrollar sistemas capaces de detectar eficientemente los casos de positivos de conflicto, ya que esos casos son los que realmente generan problemas y retrasos en los proyectos. Por lo tanto, hay que resaltar la importancia de abordar el problema de desbalanceo de clases en la predicción de conflictos en futuras investigaciones. Otro de los aspectos susceptibles de mejora del presente trabajo, es la necesidad de contar con conjuntos de datos más grandes y diversos, particularmente en el contexto de la estimación de esfuerzo. Los resultados de la regresión pueden estar influidos por la escala de los datos y la inclusión de proyectos más grandes y variados podría llevar a una mejora significativa en la precisión de la estimación. Además, sería interesante explorar un espectro más amplio de algoritmos de ML u otro tipo de métodos como el aprendizaje profundo y las redes neuronales, con la intención de ofrecer una visión más completa de las posibles soluciones a estos dos problemas que enfrenta la ingeniería de software y permitiendo una comparación entre las capacidades y limitaciones de los distintos métodos.

Un elemento clave de nuestro trabajo fue el uso de técnicas de explicabilidad tanto a nivel global como local, en respuesta a la creciente demanda de comprender las decisiones de los modelos de ML, que a menudo se consideran cajas negras por la dificultad al ser interpretados. Estas técnicas proporcionaron una visión coherente y consistente de las características influyentes en nuestros modelos, y respaldaron la validez de nuestros resultados. Además, nos proporcionaron información sobre las relaciones que existen entre la contribución de cada características y el valor de la estimación (problema de regresión) o el valor de la probabilidad (problema de clasificación). De esta forma, se ha demostrado el potencial de las técnicas de explicabilidad y su valor como herramienta para comprender e interpretar nuestros modelos de ML. Además, es evidente lo valiosa que es la información que proporcionan para los profesionales de cualquier sector que use ML a la hora de tomar decisiones más sólidas al comprender cómo influyen en los resultados las características específicas y las relaciones que existen entre ellas. Sin embargo, existen otras técnicas de explicabilidad más allá de las empleadas en este trabajo y por lo tanto ofrecen la posibilidad de explorar esas técnicas de explicabilidad en trabajos futuros. Por ejemplo sería interesante estudiar los métodos contrafactuales para ver cómo un "conflicto" se traduciría en "no conflicto", pues no hemos conseguido conocer el motivo por el cuál un modelo de ML en el problema de clasificación comete errores. A pesar de que las técnicas de explicabilidad nos han proporcionado una comprensión más amplia de cómo funciona el modelo en general, aún debemos combinarlas con enfoques adicionales, como el análisis de errores y revisión exhaustiva de las instancias FP y FN. De esta forma, podemos identificar patrones ocultos o relaciones complejas que podrían estar contribuyendo a estos errores. Esto una cuestión fundamental para comprender las debilidades del modelo y tomar medidas correctivas. Es una tarea difícil ya que los FP y FN pueden deberse a una combinación de factores complejos en los datos (como interacciones no lineales, ruido en los datos o limitaciones de las características disponibles). Este enigma hace evidente que, aunque las técnicas de explicabilidad empleadas son herramientas valiosas para el análisis, hay aspectos de la toma de decisiones de los modelos que son un desafío y necesitan una exploración más profunda y detallada.

En resumen, este trabajo ha abordado desafíos importantes que enfrenta en la actualidad el campo de la ingeniería de software mediante el uso de ML y técnicas de explicabilidad. Aunque se han logrado resultados significativos, es evidente la necesidad de un trabajo continuo en esta área para avanzar en la mejora de las prácticas de desarrollo de software y esperamos que este trabajo consiga contribuir a crear un camino para futuras investigaciones que podrían profundizar aún más en el uso de inteligencia artificial explicable en este contexto.

## Referencias

- [1] Wen, J., Li, S., Lin, Z., Hu, Y., & Huang, C. (2012). Systematic literature review of machine learning based software development effort estimation models. *Inf. Softw. Technol.*, 54, 41-59.
- [2] Biecek, P., & Burzykowski, T. (2019). *Explanatory Model Analysis: Explore, Explain, and Examine Predictive Models*. Boca Raton, FL: CRC Press.
- [3] Guidotti, R. , Monreale, A., Turini, F., Pedreschi, D. & Giannotti, F. (2018). A Survey of Methods for Explaining Black Box Models. *ACM Computing Surveys*. 51. 10.1145/3236009.
- [4] Mahmood, Y., Kama, N., Azmi, A., Khan, A. S. & Ali, M. (2021). Software effort estimation accuracy prediction of machine learning techniques: A systematic performance evaluation. *Software Practice and Experience*. 52. 10.1002/spe.3009.
- [5] Yang, Y., Xia, X., Lo, D., Bi, T., Grundy, J.C., & Yang, X. (2020). Predictive Models in Software Engineering: Challenges and Opportunities. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 31, 1 - 72.
- [6] Cabral, J.T., Oliveira, A. & Silva, F. (2022). Ensemble Effort Estimation: An updated and extended systematic literature review. *Journal of Systems and Software*. 195. 111542. 10.1016/j.jss.2022.111542.
- [7] Owhadi-Kareshk, M., Nadi, S. & Rubin, J. (2019). Predicting Merge Conflicts in Collaborative Software Development. 1-11. 10.1109/ESEM.2019.8870173.
- [8] Repositorio de referencia para los conjuntos de estimación de esfuerzo. Disponible en: <https://github.com/danrodgar/DASE/tree/master/datasets/effortEstimation> (último acceso: 18 de septiembre de 2023)
- [9] Repositorio de referencia para los conjuntos de predicción de conflictos. Disponible en: <https://github.com/ualberta-smr/conflict-prediction> (último acceso: 18 de septiembre de 2023)
- [10] Repositorio Zenodo del conjunto de estimación de esfuerzo Miyazaki. Disponible en: <https://zenodo.org/record/268473> (último acceso: 18 de septiembre de 2023)
- [11] Repositorio Zenodo del conjunto de estimación de esfuerzo Kitchenham. Disponible en: <https://zenodo.org/record/268457> (último acceso: 18 de septiembre de 2023)
- [12] Repositorio Zenodo del conjunto de estimación de esfuerzo ISBSG. Disponible en: <https://zenodo.org/record/268485> (último acceso: 18 de septiembre de 2023)
- [13] Repositorio Zenodo del conjunto de estimación de esfuerzo Albretch. Disponible en: <https://zenodo.org/record/268467> (último acceso: 18 de septiembre de 2023)
- [14] Repositorio Zenodo del conjunto de estimación de esfuerzo CHINA. Disponible en: <https://zenodo.org/record/268446> (último acceso: 18 de septiembre de 2023)
- [15] Repositorio Kaggle del conjunto de estimación de esfuerzo Desharnais. Disponible en: <https://www.kaggle.com/datasets/toniesteves/desharnais-dataset> (último acceso: 18 de septiembre de 2023)

[16] Repositorio Zenodo del conjunto de estimación de esfuerzo Maxwell. Disponible en: <https://zenodo.org/record/268461> (último acceso: 18 de septiembre de 2023)

[17] Murugesan, S. K. & Balasubramanian, C. (2015). An Accurate FFPA-PSR Estimator Algorithm and Tool for Software Effort Estimation. *The Scientific World Journal*. 2015. 1-5. 10.1155/2015/919825.

[18] Zakrani, A. & Idri, A. (2010). Applying radial basis function neural networks based on fuzzy clustering to estimate web applications effort. *International Review on Computers and Software*. 5. 516-524.

## Anexos

### Anexo 1: Preprocesado estimación de esfuerzo

- **Desharnais**

Inicialmente, este conjunto de datos contaba con 13 características y no contiene ningún valor perdido. De sus características iniciales se han eliminado “*id*” y “*Project*” debido a que eran los identificadores de cada instancias y proyecto respectivamente y por lo tanto no aportaban información relevante al modelo. Al estudiar la correlación entre características se observa una alta correlación (0.98) entre las características “*PointsNonAdjust*” y “*PointsAjust*”. Tras evaluar el desempeño del modelo siguiendo el procedimiento descrito previamente se ha eliminado la característica “*PointsNonAdjust*”. Las características finales y su descripción estadísticas se presenta en la tabla La variable “*Effort*” es la variable dependiente que será estimada por cada uno de los modelos. Para la evaluación del modelo final se han empleado validación cruzada con 5 divisiones.

- **China**

Este conjunto de datos no presenta valores perdidos y contiene 18 características iniciales, siendo ‘*Effort*’ la variable dependiente. Por otro lado, contiene una característica relacionada con la estimación del esfuerzo normalizado, ‘*N\_effort*’. Por los motivos expuestos previamente esta característica ha sido eliminada. Las características ‘*Id*’ y ‘*Dev\_type*’ han sido eliminadas ya que la primera es el identificador, mientras la segunda corresponde con el tipo de software desarrollado sin embargo siempre tiene el mismo valor para todas las instancias. En el análisis de correlación se han encontrado varias características con valores por encima del umbral considerado: ‘*PDR\_AFP*’, ‘*PDR\_UFP*’, ‘*NPDR\_AFP*’ y ‘*NPDU\_UFP*’. De las cuales tras evaluar los modelos han sido eliminadas ‘*PDR\_AFP*’, ‘*NPDR\_AFP*’ y ‘*NPDU\_UFP*’. Por lo tanto, el conjunto final de datos conta de 12 características. El valor de divisiones empleada en la validación cruzada para evaluar el modelo ha sido establecido en 5.

- **Maxwell**

Conjunto de datos con 28 características numéricas y sin valores perdidos. La variable dependiente que contiene el valor del esfuerzo es ‘*Effort*’. Se observa una alta correlación entre las características ‘*Syear*’ y ‘*Time*’, que tras ser evaluadas se decide eliminar ‘*Syear*’. Por lo tanto, el número final de características empleadas es 27. Por el reducido número de instancias se han empleado validación cruzada de 4 divisiones en este caso.

- **Miyazaki**

Contiene 9 características inicialmente de las cuales la variable dependiente correspondiente al valor del esfuerzo es ‘*MM*’. Este conjunto de datos no tiene valores perdidos, y como el conjuntos de datos anteriores hemos eliminado la característica ‘*ID*’ ya que no proporciona información relevante. Tras analizar la correlación entre características se descubre que ‘*EFILE*’ y ‘*FILE*’ tienen una correlación muy alta, siendo prácticamente la misma variable. Tras evaluar los modelos, se concluye que mantener ‘*EFILE*’ influye de manera positiva en mayor medida a los modelos. Al eliminar ‘*FILE*’ nos quedamos

finalmente con 6 características de tipo numérico. Se ha empleado validación cruzada con 5 divisiones del conjunto de datos completo.

- **Albretch**

No presenta valores perdidos en ninguna de las 8 características. Siendo '*Effort*' la variable dependiente que buscamos estimar. '*RawFPcounts*' y '*AdjFP*' presentan demasiada correlación, por lo tanto se ha eliminado '*RawFPcounts*' ya que tiene una menor contribución para estimar la variable de salida en los modelos evaluados. Debido a que solo contiene 24 instancias los resultados de rendimiento obtenidos eran muy bajos, por este motivo se han empleado técnicas de *bootstrapping* para aumentar el número de instancias al doble del número de instancias original. Con este número de instancia ha sido posible nuevas instancias emplear validación cruzada para evaluar los modelos finales con 5 divisiones.

- **Atkinson**

Incluye 15 variables numéricas sin ningún valor perdido y siendo '*Act Effort*' la variable dependiente. En primer lugar, se elimina la característica '*Name*' ya que es un identificador que no aporta información relevante. Por otro lado, la característica '*Est Effort*' se corresponde con una estimación del esfuerzo y por lo tanto debe de ser eliminada como se ha comentado previamente. Se ha encontrado correlaciones que superan el umbral en la siguientes características: '*EA-RTFP*', '*UA-RTFP*', '*IAT*', '*IT*', '*OMT*', '*OAT*', '*OT*', '*ER*', '*EA*', '*ERA*'. Al evaluar los modelos se concluye que las características que aportan menos a los modelos son '*OMT*', '*IT*', '*UA-RTFP*' y '*EA*'. Tras eliminar estas características tenemos un conjunto de datos con 9 características finales. Se ha empleado validación cruzada con 5 divisiones como método de evaluación de los modelos.

- **Kitchenham**

Este conjunto de datos contiene 8 características de las cuales dos son categóricas ('*Project.type*' y '*First.estimate.method*'). Debido a que '*First.estimate.method*' hace referencias al método empleado para hacer una estimación del esfuerzo y la característica '*First.estimate*' es el valor de esfuerzo estimado ambas características son eliminadas. Este conjunto de datos contiene valores perdidos, concretamente 10 valores perdidos pertenecientes a la característica '*Project.type*'. Siguiendo el procedimiento expuesto para tratar valores perdidos en características de tipo categórico, se ha evaluado el rendimiento de los modelos sustituyendo la moda (previamente se ha identificado que la moda en este caso tiene valor 'P') y eliminando las instancias que presentan estos valores perdidos. El resultado de la comparación de rendimiento nos dice que los modelos obtienen mejores resultados eliminando las instancias con valores perdidos directamente.

Para '*Project.type*' se ha seguido el procedimiento para tratar características de tipo categórico descrito previamente. De esta forma, se han originado 6 características binarias: '*Project.type\_A*', '*Project.type\_C*', '*Project.type\_D*', '*Project.type\_P*', '*Project.type\_Pr*', '*Project.type\_U*'. Por otro lado, las características '*Project*' y '*Client.code*' han sido eliminadas por no aportar información relevante al ser el identificador del proyecto y el identificador de cliente. Debido a que la característica '*Adjusted.function.points*' presenta una correlación muy elevada (0.98) con la variable dependiente de este conjunto de datos, '*Actual.effort*', se ha eliminado dicha característica. De esta forma el número final de 7 características.

- **ISBSG**

Este conjunto de datos se ha obtenido realizando un filtrado del conjunto de datos original siguiendo las recomendaciones del propio repositorio, las cuales se exponen a continuación:

- Las características recomendadas para realizar un estudio de estimación de esfuerzo son los atributo relacionados con el tamaño del proyecto (*'Input count'*, *'Output count'*, *'Enquiry count'*, *'File count'*, *'Interface count'*, *'Adjusted Function Points'*) y los atributos *'Development Type'*, *'Primary programming language'*, *'Development platform'*.
- Otros atributos que pueden ser interesantes son: *'Organisation type'*, *'Business area type'*, *'Application type'* y *'Development techniques'*.
- En cuanto al filtrado de instancias se recomienda: Seleccionar proyectos con calificación A o B en el atributo *'Data Quality'*, así como calificación A en *'UFP'*. Dejar fuera proyectos con una valor de relación normalizada (definido como el atributo *'Normalized Effort'* dividido por el atributo *'Summary Effort'*) superior a 1,2. Centrarse en proyectos con valor nivel de recursos igual a '1', que significa que solo se registra el esfuerzo del equipo de desarrollo. Como ultimo filtro recomendado sugieren seleccionar proyectos que utilizan IFPUG como método de dimensionamiento funcional, que se utiliza para calcular el número de AFP (puntos de función ajustados), además no debemos de mezclar proyectos que usen IFPUG anterior a la versión 4 con los proyectos que usan IFPUG V4/V4+. Por lo tanto, solo mantenemos proyectos dimensionados usando IFPUG V4/V4+.

Teniendo estos filtros en cuenta y recopilando las 14 instancias que recomiendan mencionadas previamente, obtenemos un conjunto de datos con 722 instancias. Sin embargo, este subconjunto de datos presenta muchos valores perdidos, concretamente 4530 valores perdidos distribuidos entre distintas característica. Las características *'Organisation type'*, *'Application Type'*, *'Business Area'* y *'Development Techniques'* que presentaban más de 500 casos de valores perdidos han sido eliminadas. Por otro lado, dado que las características relacionadas con el tamaño del proyecto son consideradas es especial valor se ha realizado un filtrado para mantener aquellas instancias con dos o menos valores perdidos en cualquiera de las características relacionadas con el tamaño del proyecto. Tras este filtro nos quedan un subconjunto de datos formado por 209 instancias y 12 características, además de la variable dependiente, *'Normalised Work Effort Level 1'*. Gracias a este filtro tambien hemos conseguido que no haya instancias con más de dos valores perdidos en el total de características.

Tras evaluar el rendimiento eliminando las instancias o sustituyendo los valores por la moda o la media se ha comprobado que eliminar las instancias con valores perdidos conlleva una mejora en el rendimiento. De esta forma, se ha reducido el número de instancias a 197 sin contener ningún valor perdido. Las variables categóricas *'Development Type'*, *'Development Platform'* y *'Primary Programming Language'* han sido transformadas a variables binarias. Finalmente el análisis de correlación ha concluido que las características *'Primary Programming Language\_ASP'* y *'Input count'* son de escasa relevancia para los modelos y por lo tanto han sido eliminadas. Dando como resultado un subconjunto de 197 instancias y 37 características.

## Anexo 2: Preprocesado clasificación de conflictos

Debido a que los cuatro conjuntos de datos correspondientes a la parte de predicción de conflictos son similares y contienen las misma características, hay hechos que son comunes: en todos ellos las características “*file\_renamed*” y “*file\_copied*” han sido eliminadas de cada conjunto durante el preprocesado debido a que eran característica constantes, es decir, solo tenían un valor para todas las instancias. La característica “*is\_conflict*” contiene la clase que se busca predecir. Posteriormente para cada conjunto de datos se ha estudiado la correlación y se han eliminados aquellas instancias que tienen una correlación superior al umbral. La gran cantidad de instancias de todos los conjuntos estudiados en clasificación de conflictos permite el uso de validación cruzada con 10 divisiones en todos estos conjuntos.

- **Conjunto Java**

Constaba de 28 características inicialmente, tras eliminar “*file\_renamed*” y “*file\_copied*” por ser constantes, y “*messages\_median*” por tener una correlación excesiva con “*messages\_mean*”, siendo esta última más importante para el rendimiento de los clasificadores. Por lo tanto, finalmente quedan 25 características.

- **Conjunto Python**

El preprocesado es similar al conjunto Java, con la diferencia que en este caso se ha eliminado la característica “*messages\_mean*” en lugar de “*messages\_median*” por ser ahora menos valiosa para el rendimiento. Al igual que el conjunto anterior nos quedamos con 25 características.

- **Conjunto PHP y Conjunto Ruby**

El preprocesado para el conjunto Ruby es idéntico al llevado a cabo en el conjunto de datos Java, comentado anteriormente. Por otro lado, el preprocesado del conjunto PHP es idéntico al realizado sobre el conjunto de datos Python.

## Anexo 3: Explicabilidad estimación de esfuerzo

A continuación, se presentan una colección de gráficas obtenidas para todos los conjuntos de datos, modelos y técnicas de explicabilidad estudiadas: El modelo SVR del conjunto Albretch se recoge en la Fig. 104 - Fig. 111; La técnica *Break-down* para *Random Forest* en el conjunto Atkinson en la Fig. 112, Fig. 113 y Fig. 114; La técnica LIME para *Gradient Boosting* en el conjunto Miyazaki en la Fig. 115 y Fig. 116; La técnica *Shapley Values* para *Random Forest* en el conjunto Kitchenham en la Fig. 117, Fig. 118 y Fig. 119; La técnica *Break-down* para *Voting* en el conjunto ISBSG en la Fig. 120, Fig. 121 y Fig. 122.

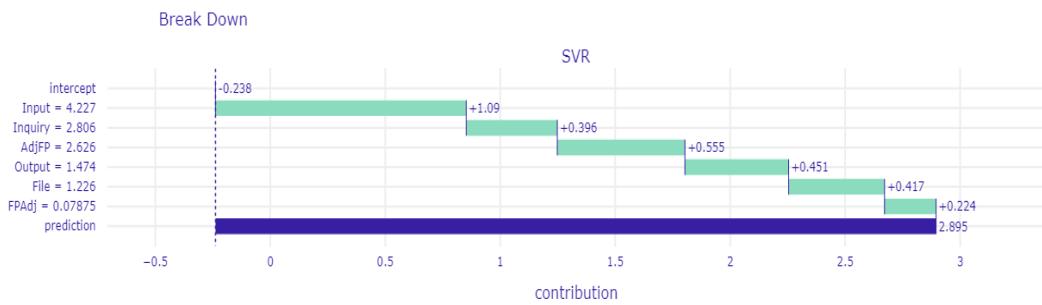


Fig. 104. Break-down en el conjunto Albretch para SVR y la instancia de estimación máxima.

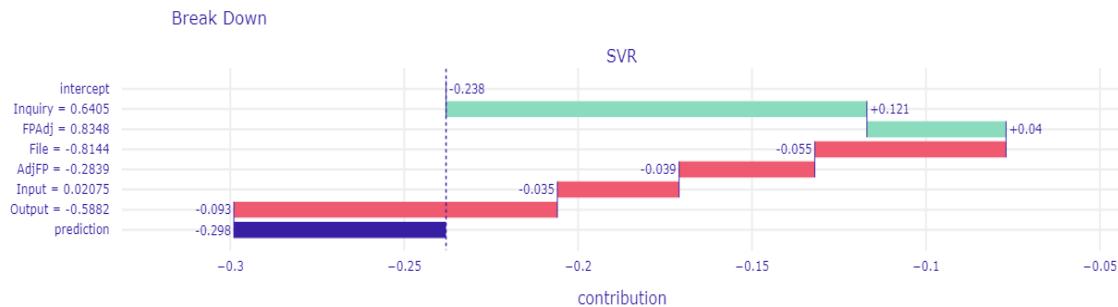


Fig. 105. Break-down en el conjunto Albretch para SVR y la instancia de estimación en la mediana.

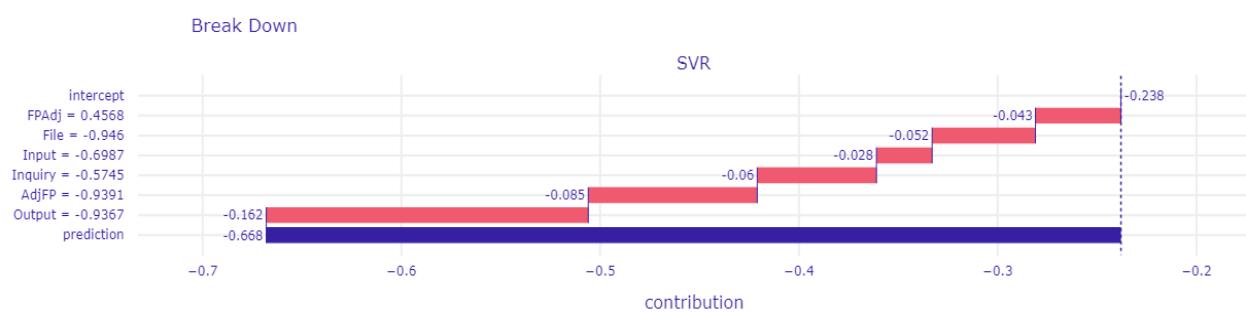


Fig. 106. Break-down en el conjunto Albretch para SVR y la instancia de estimación mínima.

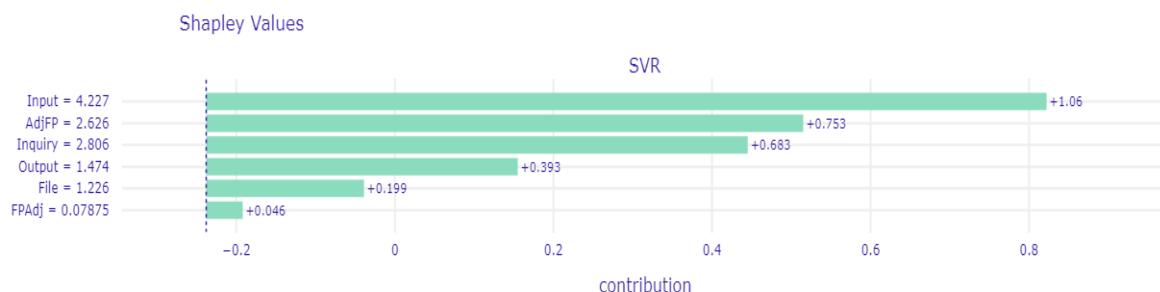


Fig. 107. Shapley Values en el conjunto Albretch para SVR y la instancia de estimación máxima.

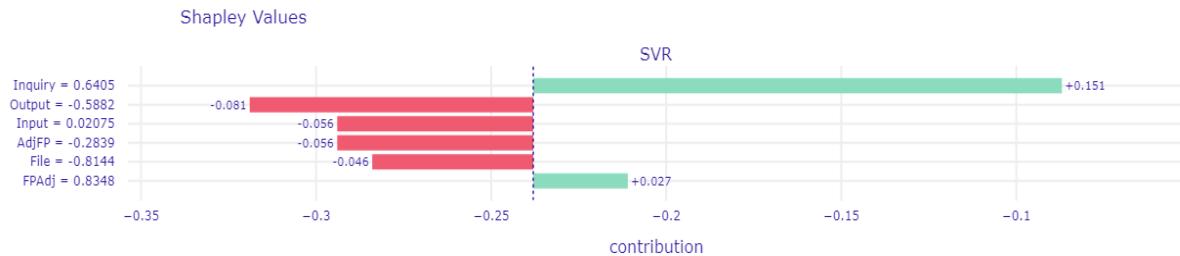


Fig. 108. Shapley Values en el conjunto Albretch para SVR y la instancia de estimación en la mediana.

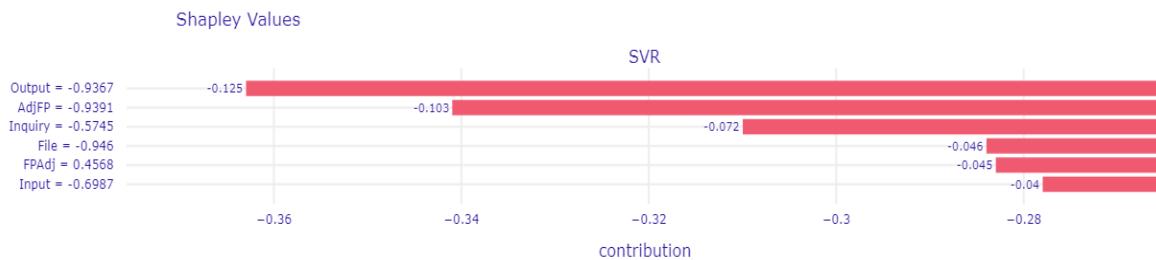


Fig. 109. Shapley Values en el conjunto Albretch para SVR y la instancia de estimación mínima.

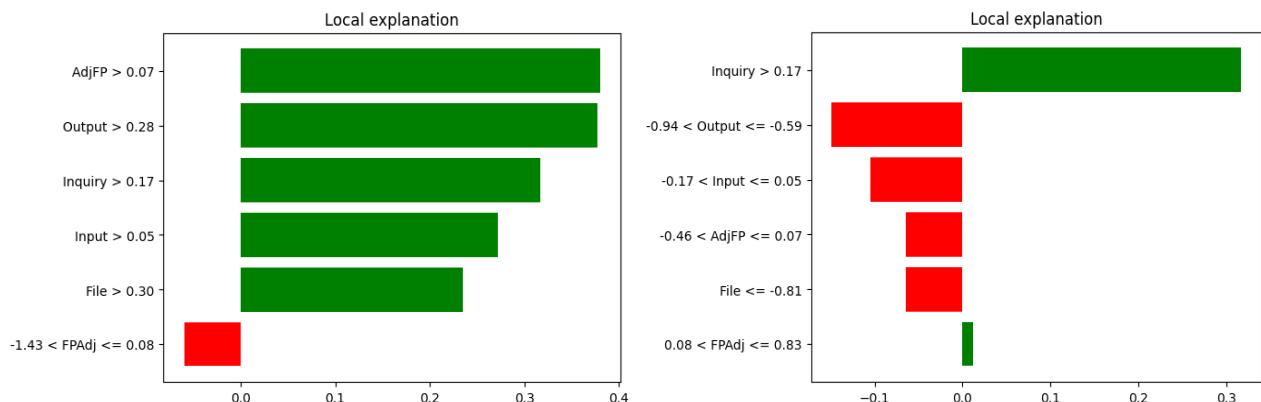


Fig. 110. En la izquierda se representa la técnica LIME en el conjunto Albretch para el modelo SVR y la instancia de estimación máxima. A la derecha esa misma técnica, modelo y conjunto de datos pero para la instancia en la mediana.

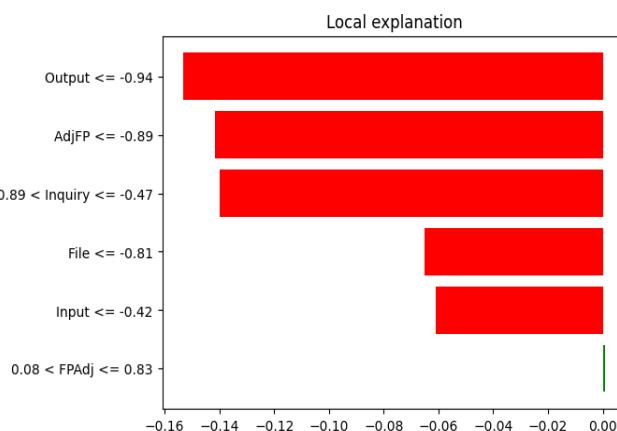


Fig. 111. LIME en el conjunto Albretch para SVR y la instancia de estimación mínima.

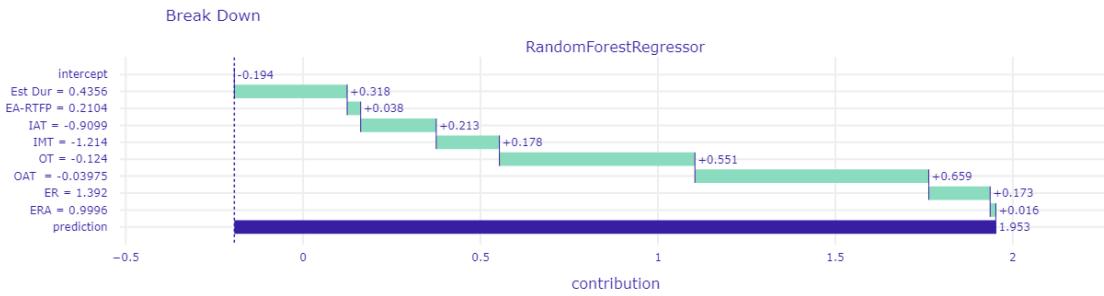


Fig. 112. Conjunto Atkinson: Break-down para el modelo Gradient Boosting e instancia máxima.

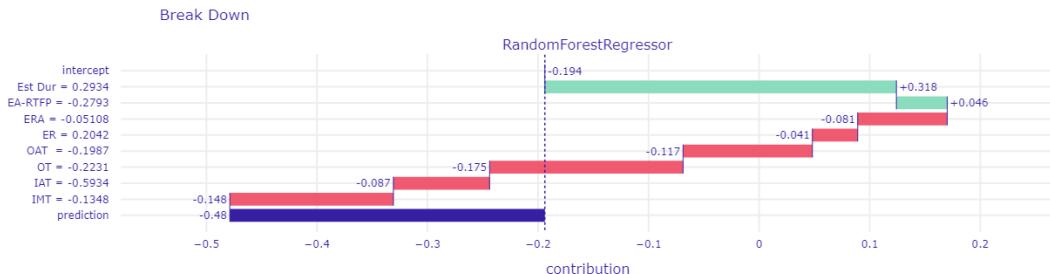


Fig. 113. Conjunto Atkinson: Break-down para el modelo Gradient Boosting e instancia en la mediana.

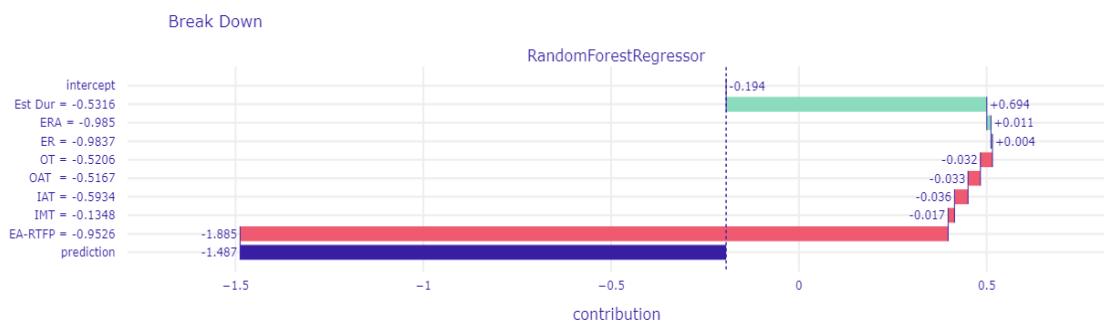


Fig. 114. Conjunto Atkinson: Break-down para el modelo Gradient Boosting e instancia mínima.

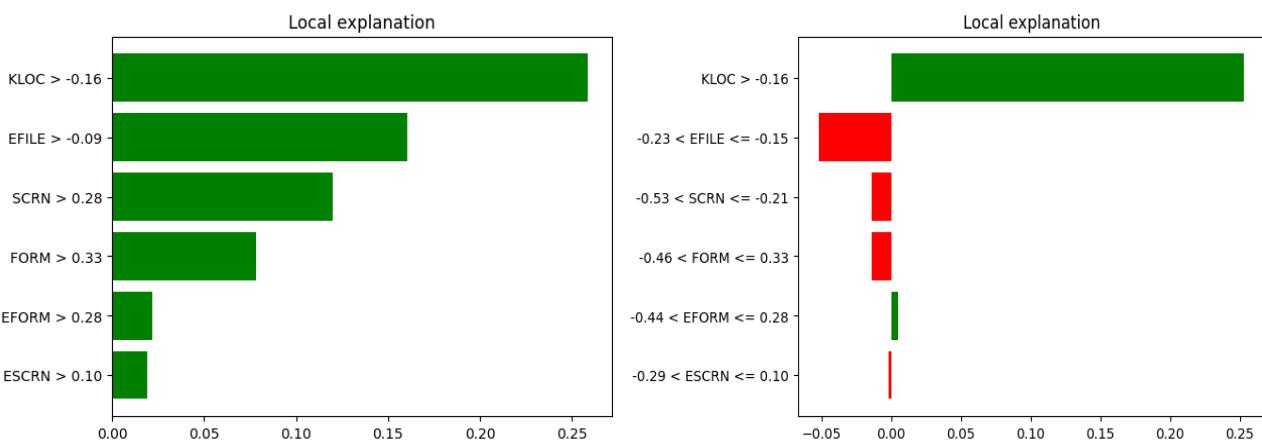
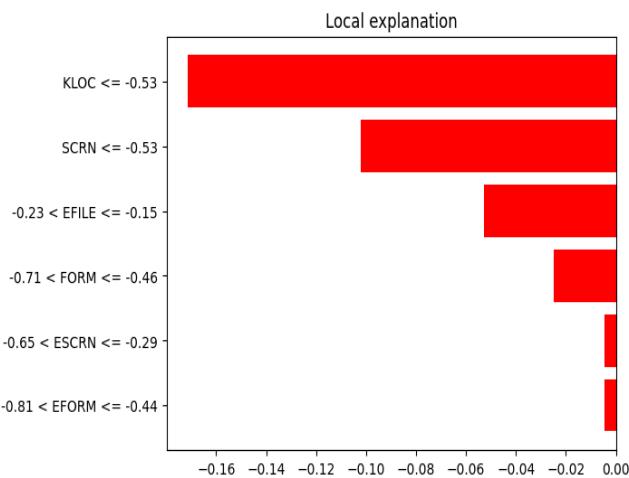
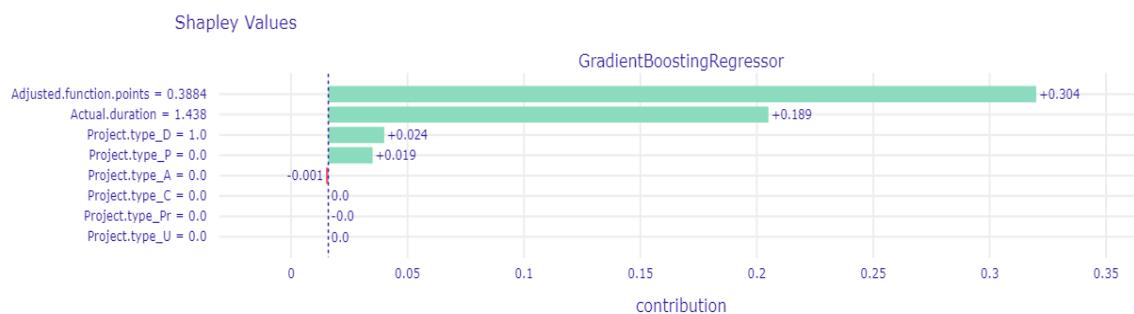


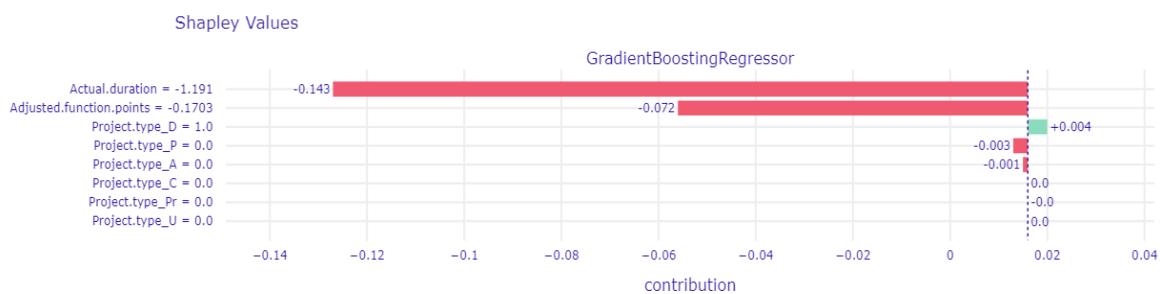
Fig. 115. En la izquierda se representa la técnica LIME en el conjunto Miyazaki para el modelo Gradient Boosting y la instancia de estimación máxima. A la derecha esa misma técnica, modelo y conjunto de datos pero para la instancia en la mediana.



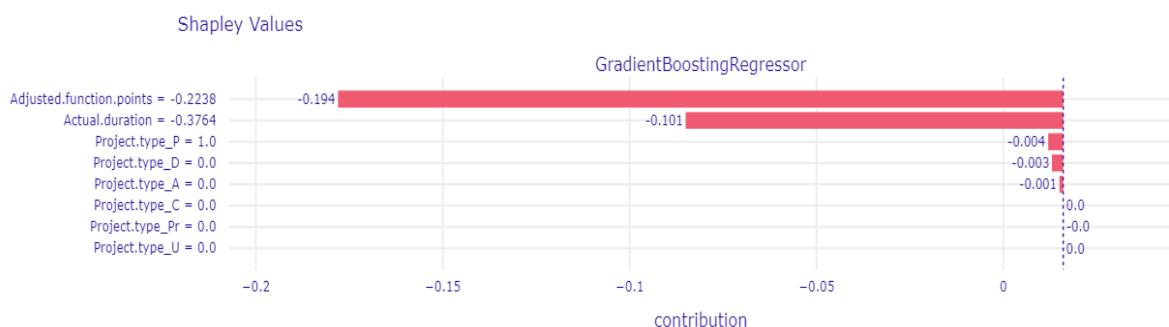
**Fig. 116. LIME en el conjunto Miyazaki para Gradient Boosting y la instancia de estimación mínima.**



**Fig. 117. Shapley Values en el conjunto Kitchenham para Gradient Boosting y la instancia de estimación máxima.**



**Fig. 118. Shapley Values en el conjunto Kitchenham para Gradient Boosting y la instancia de estimación en la mediana.**



**Fig. 119. Shapley Values en el conjunto Kitchenham para Gradient Boosting y la instancia de estimación mínima.**

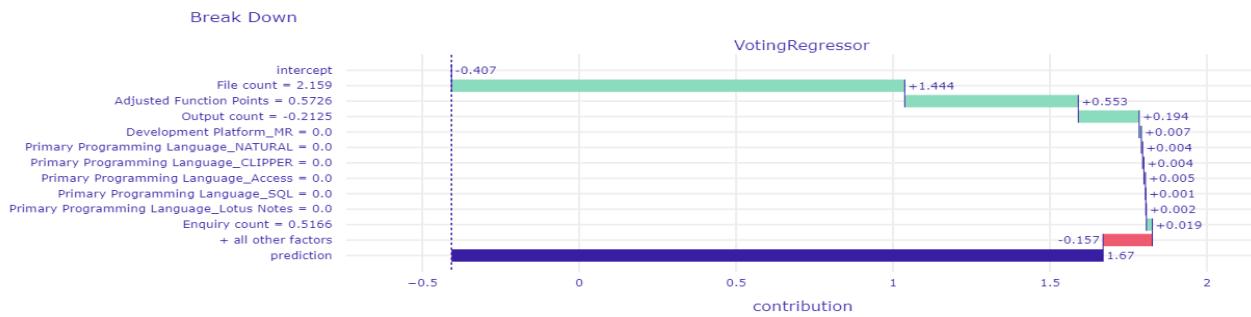


Fig. 120. Break-down en el conjunto ISBSG para Voting y la instancia de estimación máxima.

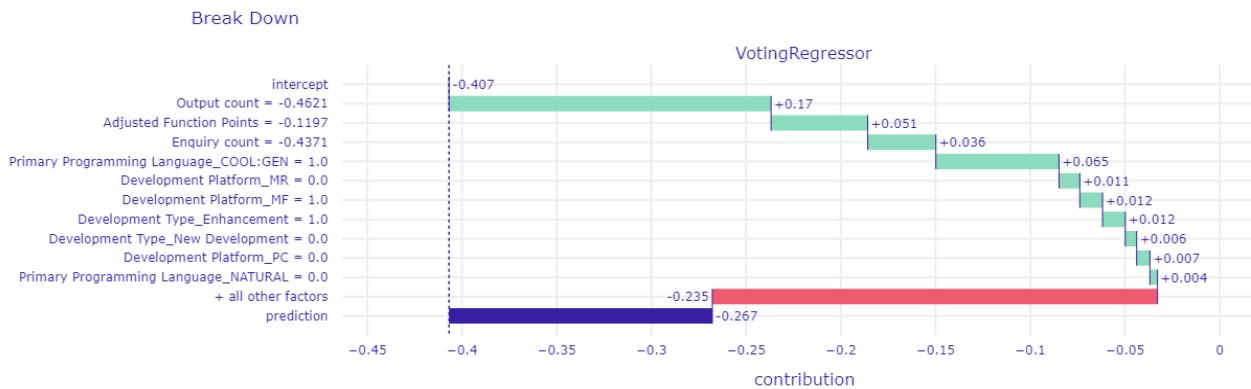


Fig. 121. Break-down en el conjunto ISBSG para Voting y la instancia de estimación en la mediana.

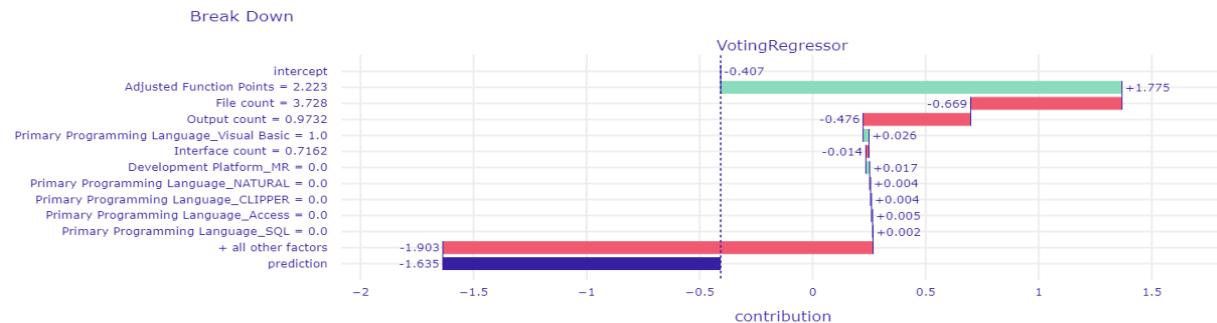


Fig. 122. Break-down en el conjunto ISBSG para Voting y la instancia de estimación mínima.

## Anexo 4: Explicabilidad predicción de conflictos

En este anexo se recoge la colección parcial de las gráficas obtenidas para explicabilidad en la experimentación del problema de predicción de conflictos. Se presentan resultados de todos los conjuntos de datos, modelos y técnicas de explicabilidad estudiadas: El modelo *Gradient Boosting* del conjunto Java para la instancia falso positivo se recoge en la Fig. 123 - Fig. 130; El modelo *Random Forest* en el conjunto PHP para la instancia verdadero negativo en la Fig. 131 - Fig. 138; La técnica *Shapley Values* para *Balanced Random Forest* en el conjunto Ruby para la instancia verdadero positivo se presenta en la

Fig. 139, Fig. 140 y Fig. 141; La técnica LIME para ADABoost en el conjunto Python para la instancia falso negativo en la Fig. 142 y Fig. 143.

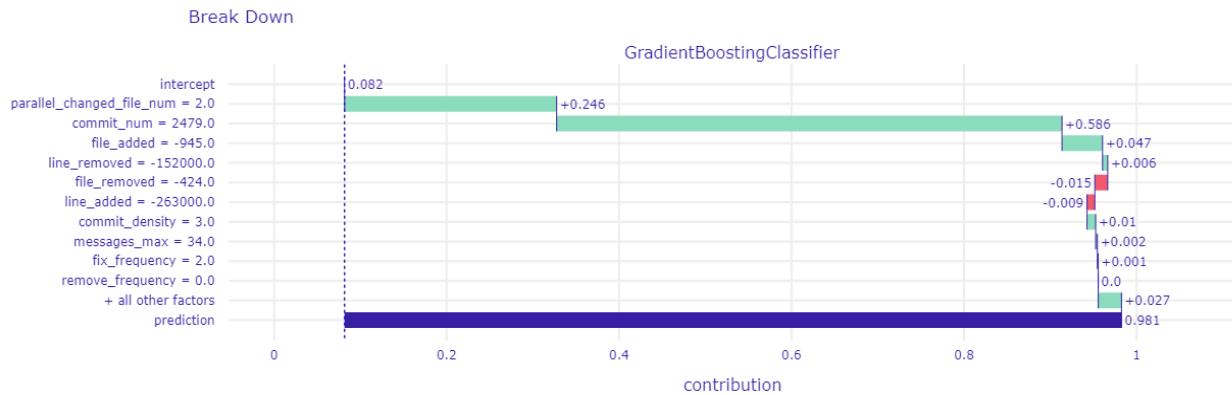


Fig. 123. Break-down en el conjunto Java para Gradient Boosting en la instancia falso positivo con probabilidad máxima.

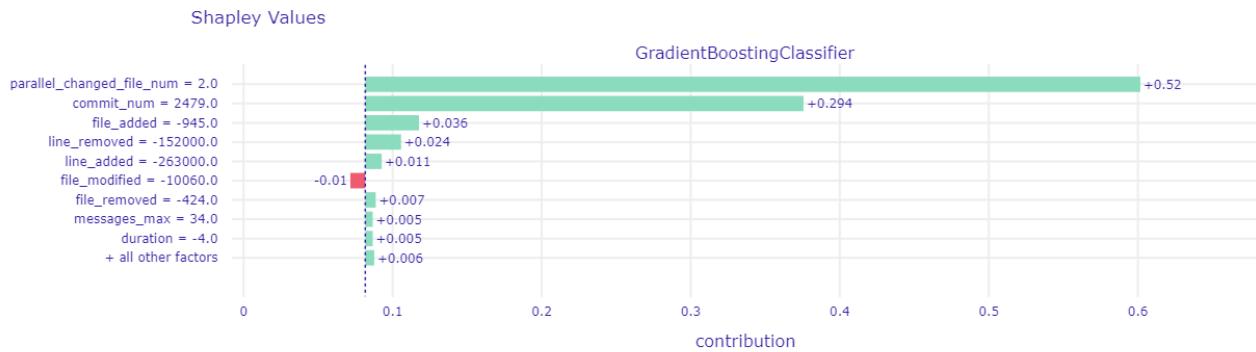


Fig. 124. Shapley Values en el conjunto Java para Gradient Boosting en la instancia falso positivo con probabilidad máxima.

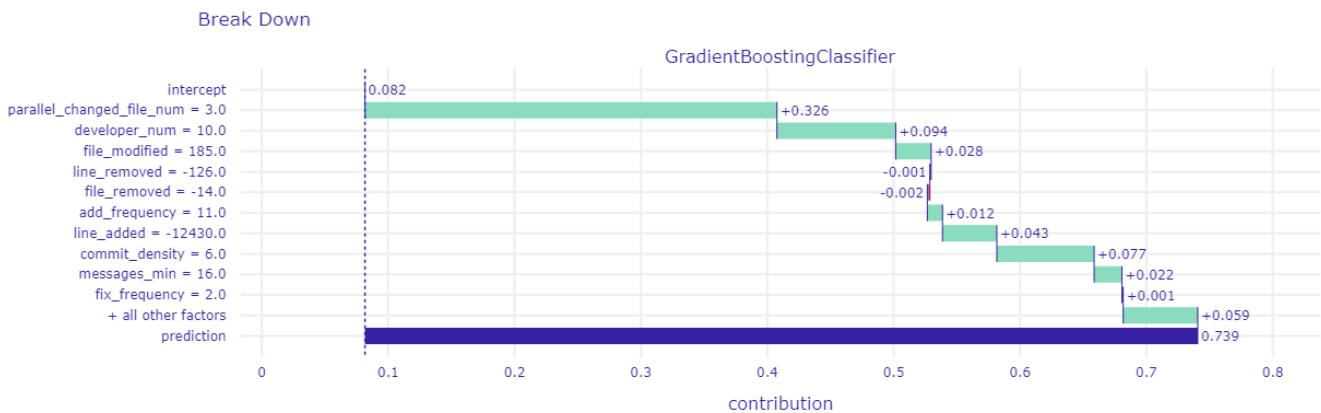


Fig. 125. Break-down en el conjunto Java para Gradient Boosting en la instancia falso positivo con probabilidad en la mediana.

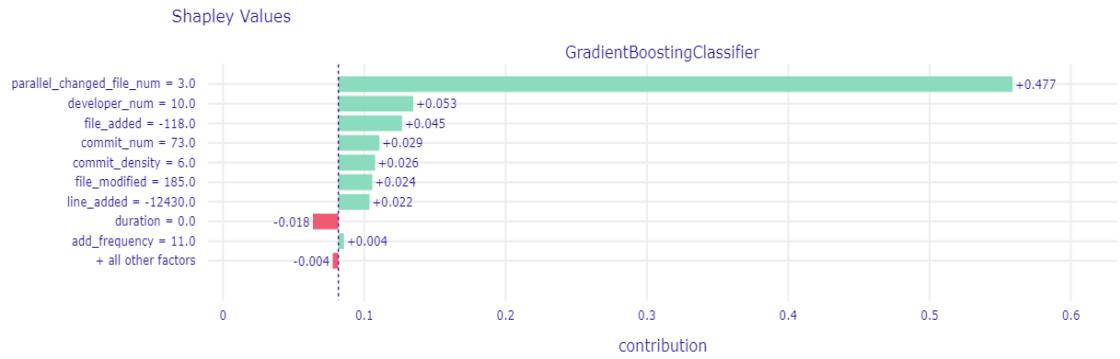


Fig. 126. Shapley Values en el conjunto Java para Gradient Boosting en la instancia falso positivo con probabilidad en la mediana.

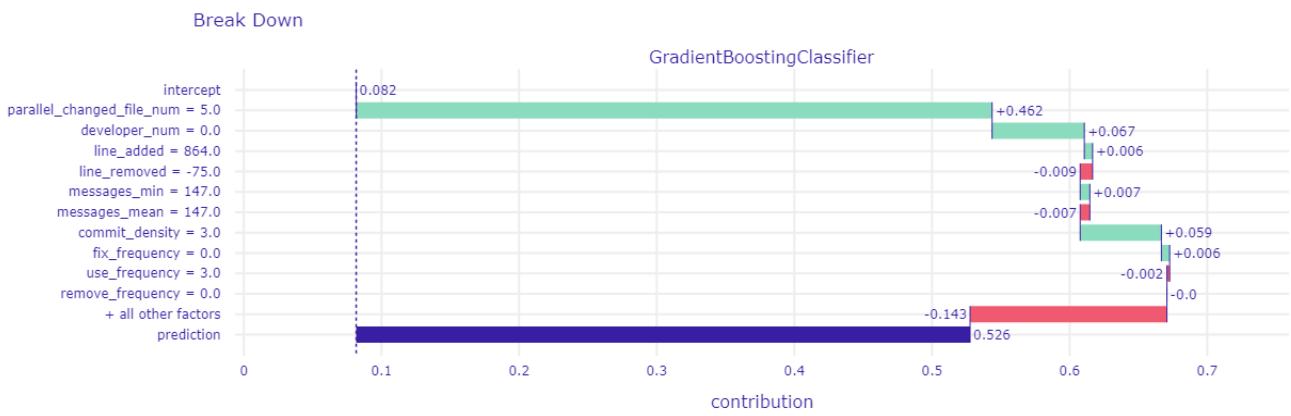


Fig. 127. Break-down en el conjunto Java para Gradient Boosting en la instancia falso positivo con probabilidad mínima.

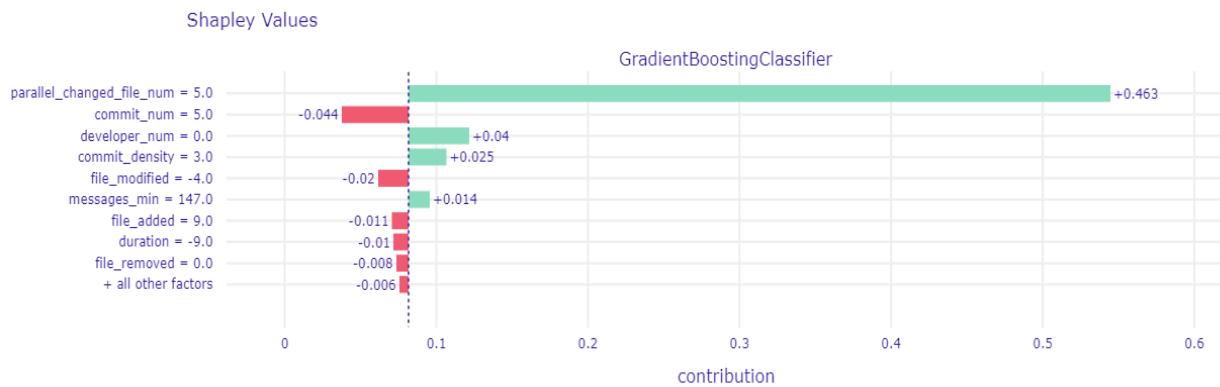
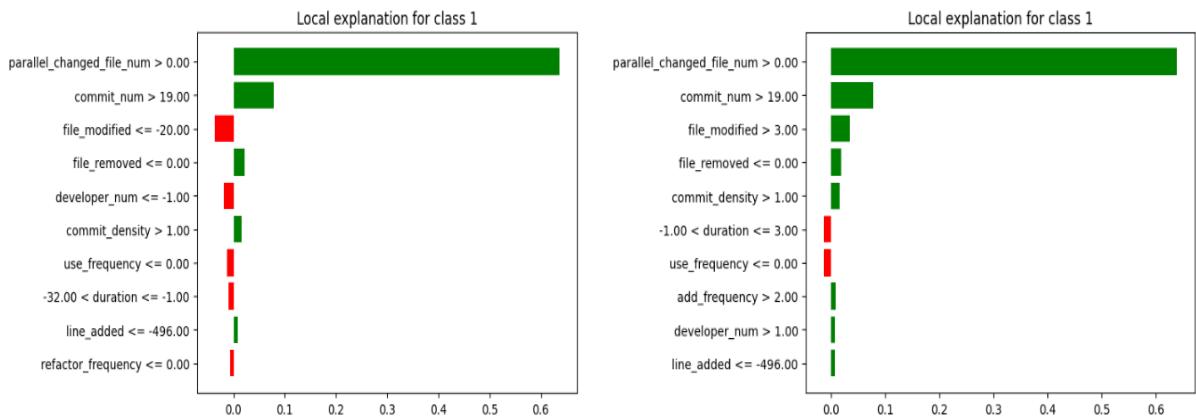
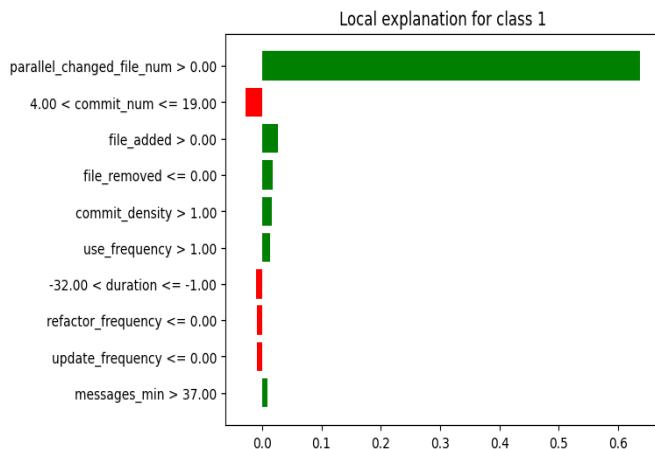


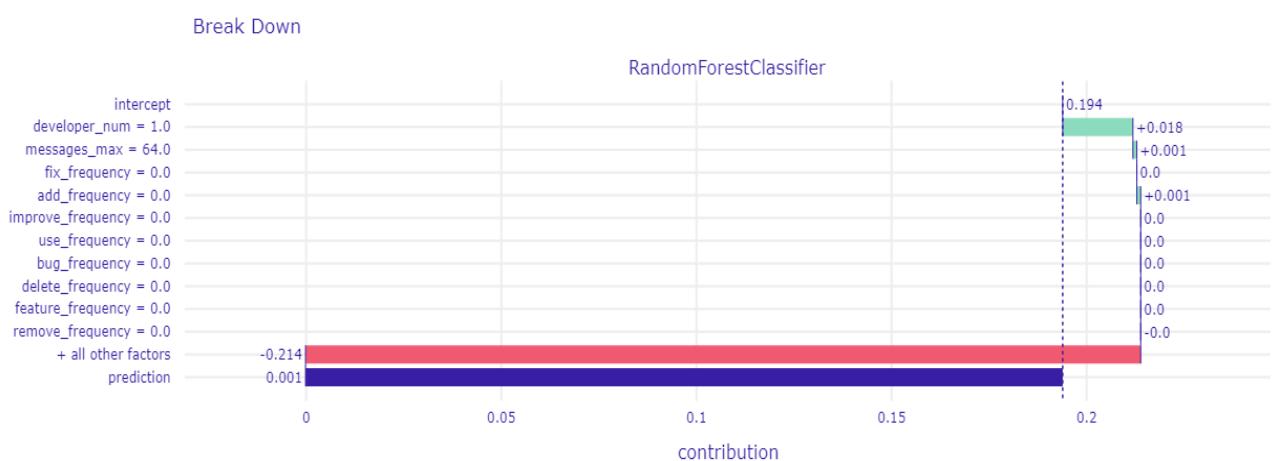
Fig. 128. Shapley Values en el conjunto Java para Gradient Boosting en la instancia falso positivo con probabilidad mínima.



**Fig. 129.** En la izquierda se representa la técnica LIME en el conjunto Java para el modelo Gradient Boosting en la instancia falso positivo de probabilidad máxima. A la derecha esa misma técnica, modelo y conjunto de datos pero con probabilidad en la mediana.



**Fig. 130.** LIME en el conjunto Java para Gradient Boosting en la instancia falso positivo con probabilidad mínima.



**Fig. 131.** Break-down en el conjunto PHP para Random Forest en la instancia verdadero negativo con probabilidad máxima.

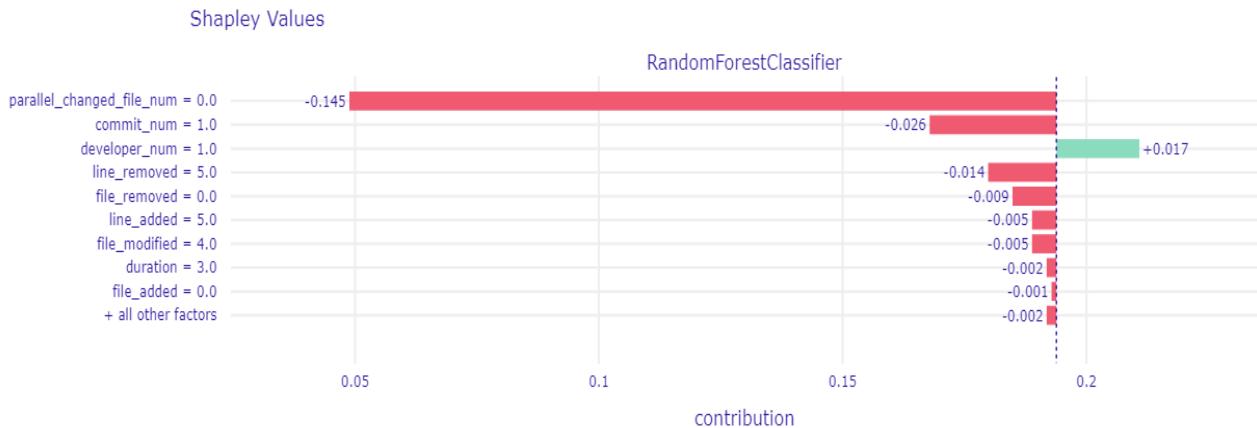


Fig. 132. Shapley Values en el conjunto PHP para Random Forest en la instancia verdadero negativo con probabilidad máxima.

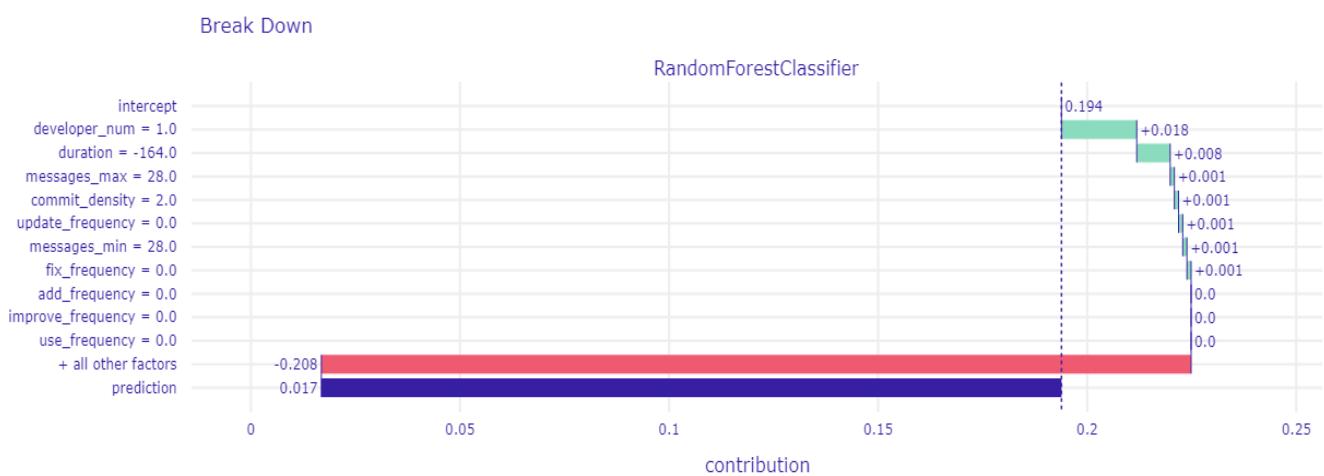


Fig. 133. Break-down en el conjunto PHP para Random Forest en la instancia verdadero negativo con probabilidad en la mediana.

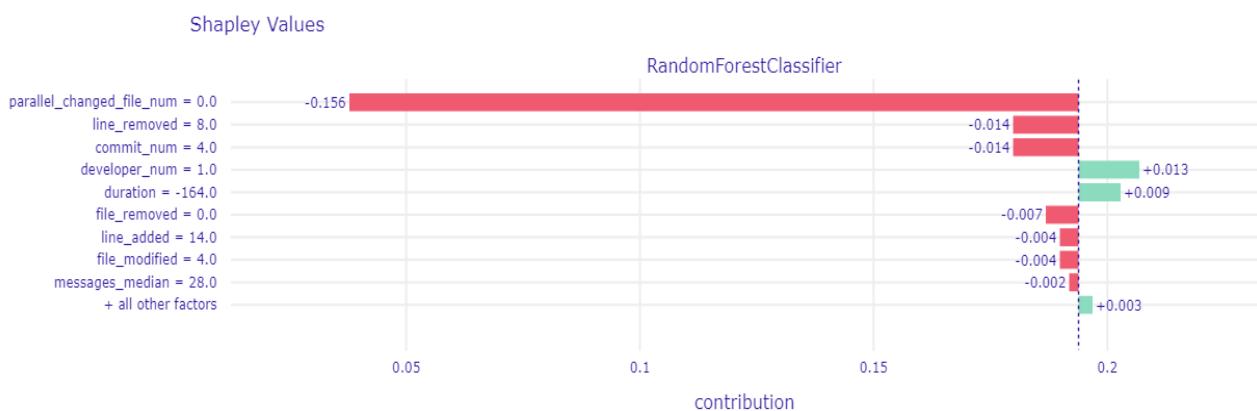


Fig. 134. Shapley Values en el conjunto PHP para Random Forest en la instancia verdadero negativo con probabilidad en la mediana.

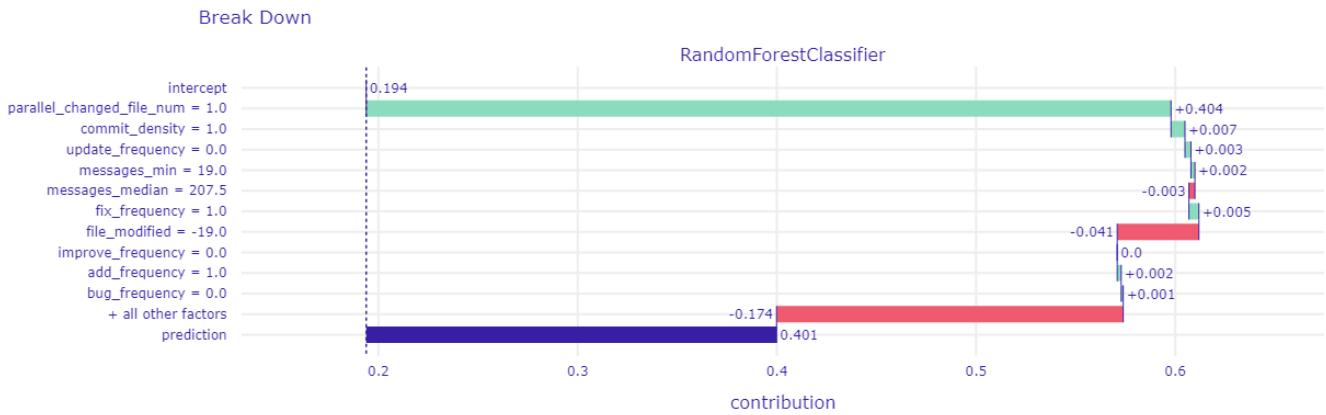


Fig. 135. Break-down en el conjunto PHP para Random Forest en la instancia verdadero negativo con probabilidad mínima.

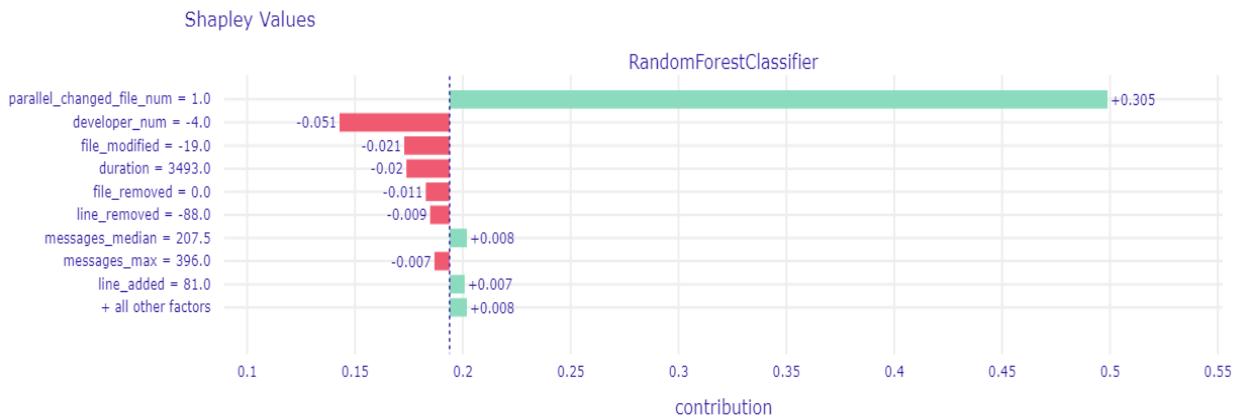


Fig. 136. Shapley Values en el conjunto PHP para Random Forest en la instancia verdadero negativo con probabilidad mínima.

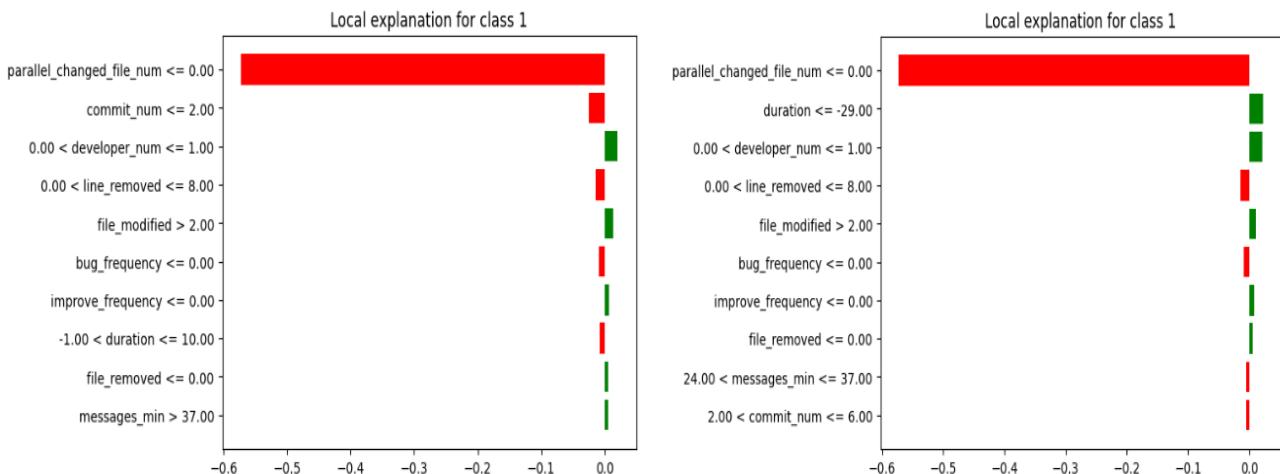
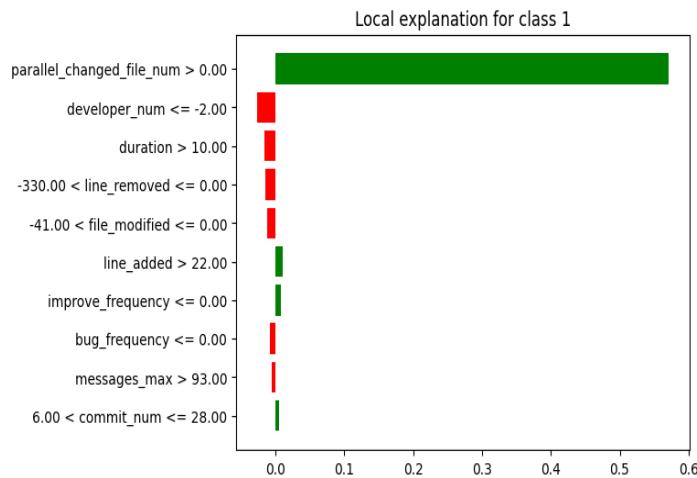
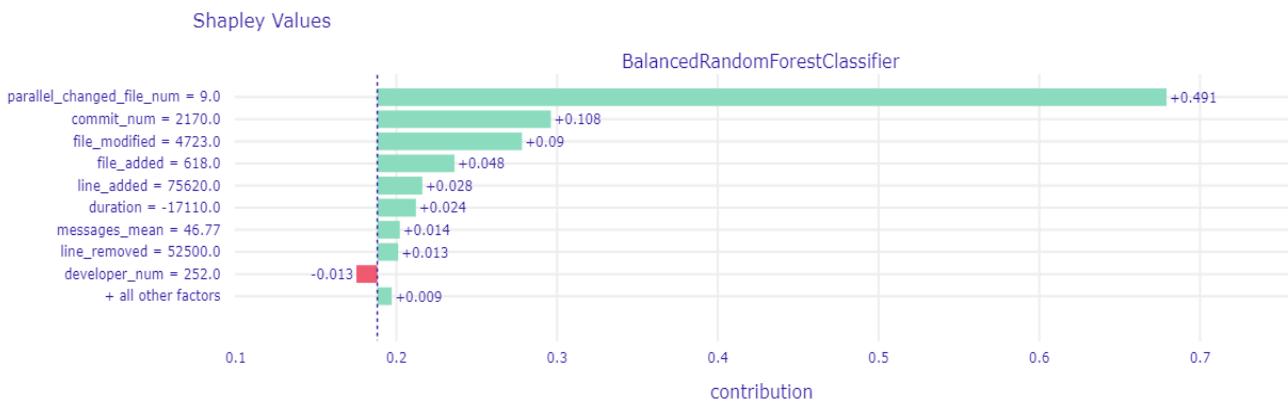


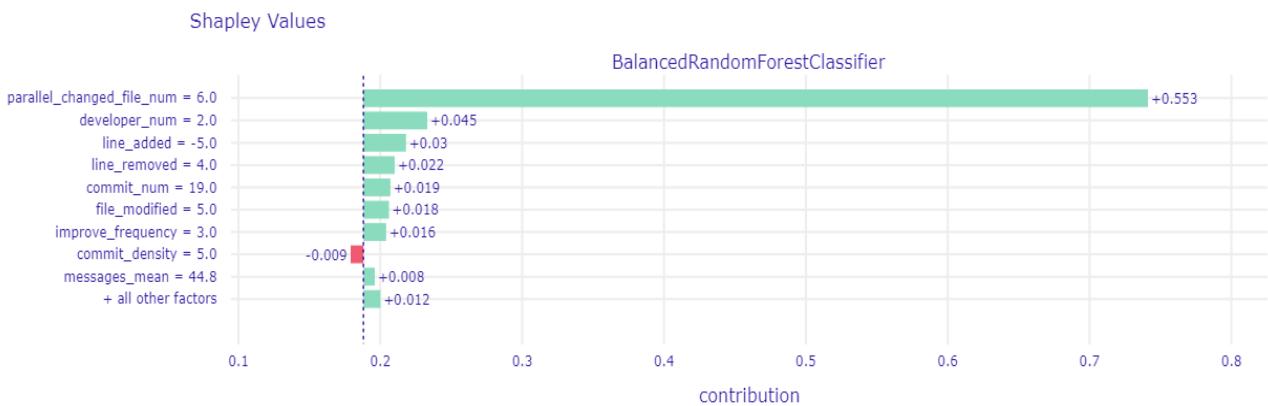
Fig. 137. En la izquierda se representa la técnica LIME en el conjunto PHP para el modelo Random Forest en la instancia verdadero negativo de probabilidad máxima. A la derecha esa misma técnica, modelo y conjunto de datos pero con probabilidad en la mediana.



**Fig. 138. LIME en el conjunto PHP para Random Forest en la instancia verdadero negativo con probabilidad mínima.**



**Fig. 139. Shapley Values en el conjunto Ruby para BRF en la instancia verdadero positivo con probabilidad máxima.**



**Fig. 140. Shapley Values en el conjunto Ruby para BRF en la instancia verdadero positivo probabilidad en la mediana.**

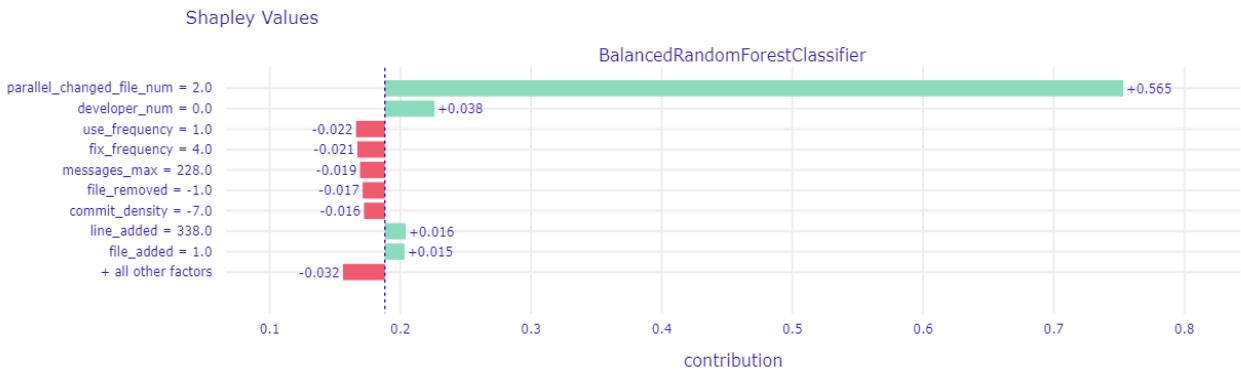


Fig. 141. Shapley Values en el conjunto Ruby para BRF en la instancia verdadero positivo con probabilidad mínima.

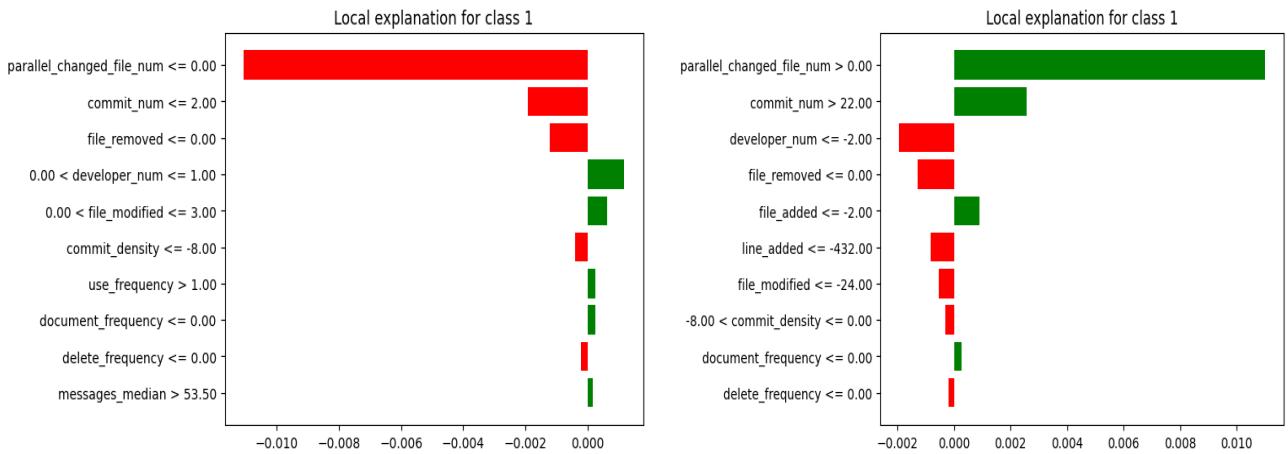


Fig. 142. En la izquierda se representa la técnica LIME en el conjunto Python para el modelo ADABoost en la instancia falso negativo de probabilidad máxima. A la derecha esa misma técnica, modelo y conjunto de datos pero con probabilidad en la mediana.

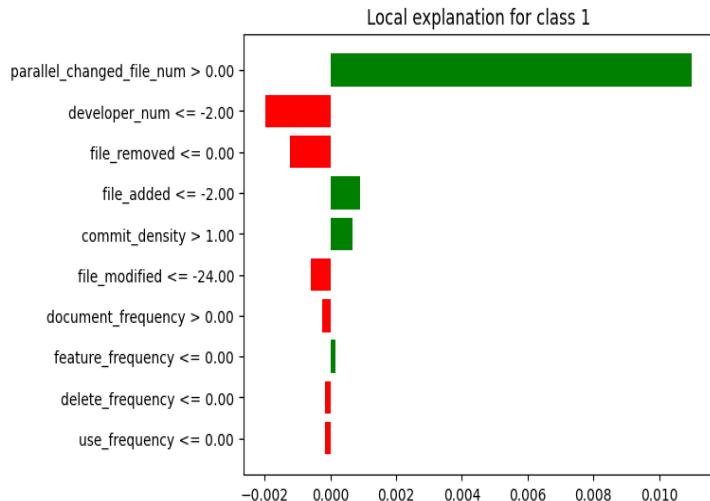


Fig. 143. LIME en el conjunto Python para ADABoost en la instancia falso negativo con probabilidad mínima.