Universidade Vigo

Traballo Fin de Mestrado

Aprendizaxe dinámica de selección de variable nun algoritmo de ramificación e limitación

Cintia Riveiro Rojo

Mestrado en Técnicas Estadísticas

Curso 2022-2023

Proposta de Traballo Fin de Mestrado

Título en galego: Aprendizaxe dinámica de selección de variable nun algoritmo de ramificación e limitación

Título en español: Aprendizaje dinámico de selección de variable en un algoritmo de ramificación y acotación

English title: Dynamic learning of variable selection in an branch-and-bound algorithm

Modalidade: Modalidade A

Autor/a: Cintia Riveiro Rojo, Universidade de Santiago de Compostela

Director/a: Beatriz Pateiro López, Universidade de Santiago de Compostela; Brais González Rodríguez, Universidade de Santiago de Compostela

Breve resumo do traballo:

Os algoritmos de ramificación e acoutamento son unha das técnicas más empregadas á hora de resolver problemas de optimización. Dita técnica basease en considerar relaxacións do problema orixinal que simplifican a resolución do mesmo. Durante o desenvolvemento do algoritmo, hai que tomar diversas decisións sobre o proceso de ramificación. En particular, este traballo focalízase na seleción da variable de ramificación en cada unha das iteracións do algoritmo RLT aplicado á resolución de problemas de optimización polinómica, escollendo para iso entre distintos criterios de ramificación. Ata o de agora, foi estudada a incorporación de técnicas de aprendizaxe estatística neste contexto, co obxectivo de seleccionar a mellor regra de ramificación para a resolución dun determinado problema de optimización. Sen embargo, este procedemento lévanos a resolver un problema concreto utilizando sempre o mesmo criterio de ramificación en todos os nodos do avance do algoritmo. O obxectivo deste traballo é refinar esta técnica, propoñendo un método de aprendizaxe estática (aprendizaxe offline) con variables explicativas de carácter dinámico, de forma que, en lugar de fixar un único criterio para cada problema, poidamos ir seleccionando o mellor criterio de ramificación a medida que crece a árbore, o que podería derivar nun mellor rendemento do proceso de aprendizaxe e da resolución do problema, pois teríase en conta información adicional en cada iteración do algoritmo RLT.

Doña Beatriz Pateiro López, de la Universidade de Santiago de Compostela y don Brais González Rodríguez, de la Universidade de Santiago de Compostela, informan que el Trabajo Fin de Máster titulado

Aprendizaxe dinámica de selección de variable nun algoritmo de ramificación e limitación

fue realizado bajo su dirección por doña Cintia Riveiro Rojo para el Máster en Técnicas Estadísticas. Estimando que el trabajo está terminado, dan su conformidad para su presentación y defensa ante un tribunal.

En Santiago de Compostela, a 27 de Xaneiro de 2023.

La directora:

PATEIRO LOPEZ BEATRIZ - 46898344X
Firmado digitalmente por PATEIRO LOPEZ BEATRIZ - 46898344X
Fecha: 2023.01.25 09:07:44 +01'00'

Doña Beatriz Pateiro López

El director:

GONZALEZ RODRIGUEZ BRAIS - 44661382M
Digitally signed by GONZALEZ RODRIGUEZ BRAIS - 44661382M
Date: 2023.01.24 22:50:32 -05'00'

Don Brais González Rodríguez

RIVEIRO ROJO CINTIA - 54152409J
Firmado digitalmente por RIVEIRO ROJO CINTIA - 54152409J
Fecha: 2023.01.25 22:59:08 +01'00'

La autora:
Doña Cintia Riveiro Rojo

Declaración responsable. Para dar cumplimiento a la Ley 3/2022, de 24 de febrero, de convivencia universitaria, referente al plagio en el Trabajo Fin de Máster (Artículo 11, Disposición 2978 del BOE núm. 48 de 2022), **la autora declara** que el Trabajo Fin de Máster presentado es un documento original en el que se han tenido en cuenta las siguientes consideraciones relativas al uso de material de apoyo desarrollado por otros/as autores/as:

- Todas las fuentes usadas para la elaboración de este trabajo han sido citadas convenientemente (libros, artículos, apuntes de profesorado, páginas web, programas,...)
- Cualquier contenido copiado o traducido textualmente se ha puesto entre comillas, citando su procedencia.
- Se ha hecho constar explícitamente cuando un capítulo, sección, demostración,... sea una adaptación casi literal de alguna fuente existente.

Y, acepta que, si se demostrara lo contrario, se le apliquen las medidas disciplinarias que correspondan.

Agradecementos

Cerrar unha etapa é, ás veces, complicado. Este traballo marca exactamente iso, a fin dunha etapa, pechando un ciclo de sete anos de formación e aprendizaxe constante, polos cales non podería estar máis agradecida, malia que por momentos resultaran difíciles.

Primeiramente, agradecer á Universidade de Santiago de Compostela e, en particular, á Facultade de Matemáticas e ao seu persoal docente, pola formación recibida e tamén por todo aquilo que vai máis alá das paredes da aula. Grazas por facerme crecer a nivel académico e persoal. En especial, destacar aos directores desta memoria, Beatriz Pateiro e Brais Rodríguez, pola súa completa dispoñibilidade e os seus consellos, sen os cales, este traballo non houbera chegado a ser o que é. Agradecer tamén a Julio González, pola comprensión e o apoio recibido. Grazas a todos por acollerme no proxecto de `Raposa`.

Por outra parte, dar grazas á miña compañeira e amiga, Sofía Rodríguez, por escoitarme, pensar en min para continuar co seu traballo e polo constante apoio día tras día.

Destacar tamén á miña familia, por proporcionarme os medios necesarios para formarme naquilo que quixen, sen ningún tipo de preocupación. A todas as miñas amizades, en especial, aos 3 piares que me deu a facultade. Lorena, por ser o mellor exemplo de constancia e loita. Aida, matemática á fuga, polo bo humor e preocuparte sempre polos teus. María, a miña xemelga de distinta nai, a miña man forte, que leve me fixeches o camiño. Grazas por tirar de min cando pensei en deixalo. Que sorte a miña.

E a Armando, por ser a persoa máis afastada do mundo matemático, e que mellor finxe entender o que digo.

Índice xeral

Resumo IX

1. Optimización polinómica 1
1.1. Conceptos previos . . . 2
1.2. Problemas de optimización polinómica . . . 3
1.2.1. Algoritmo RLT . . . 5

2. Modelos de Aprendizaxe Estatística 11
2.1. Árbores de decisión . . . 11
2.2. Boosting . . . 13
2.3. Regresión cuantil . . . 15

3. Metodoloxía da aprendizaxe 18
3.1. Criterios de ramificación . . . 18
3.1.1. Pseudocost Branching . . . 20
3.2. Descrición do conxunto de datos . . . 21
3.2.1. Proceso de xeración . . . 22
3.2.2. Variables explicativas . . . 23
3.2.3. Variable resposta . . . 23
3.2.4. Depuración do conxunto de datos e selección de variables . . . 24
3.2.5. Transformación das variables . . . 25

4. Resultados prácticos 26
4.1. Proceso de aprendizaxe . . . 29
4.1.1. Validación cruzada . . . 30
4.1.2. Mostra aleatoria sobre o conxunto orixinal . . . 30
4.1.3. Mostra con $k = 500$ nodos . . . 40
4.1.4. Mostra con $k = 10000$ nodos . . . 47
4.2. Análise de empates . . . 54

5. Conclusións 59

A. Variables explicativas 61
A.1. Variables estáticas . . . 61
A.2. Variables dinámicas . . . 62

Bibliografía 65

Resumo

Resumo en galego

Os problemas de programación polinómica son un campo de gran interese na actualidade, e por ende, é de gran importancia dispoñer de técnicas adecuadas para a súa resolución. O algoritmo RLT é unha destas propostas, cuxas bases se fundamentan sobre os algoritmos de ramificación e limitación. Debido a isto, as regras de ramificación consideradas en cada iteración do algoritmo adquiren especial relevancia, buscando unha partición óptima que proporcione o maior incremento na cota inferior. Idealmente, empregaríanse todos os criterios de ramificación posibles en cada nodo e seleccionaríamos aquel que resultase óptimo, pero, computacionalmente, isto é intratable. Neste punto xorde a necesidade dun modelo que seleccione, en cada nodo, o criterio óptimo a partir de técnicas de aprendizaxe estatística, concretamente, os modelos Boosting baseado en regresión cuantil, que semellan ter un bo comportamento en conxuntos de datos con fortes asimetrías e alto número de atípicos. Os resultados obtidos presentaranse de xeito gráfico e de xeito facilmente interpretable, realizando unha análise detallada dos mesmos, de xeito progresivo, xustificando a motivación das diferentes probas executadas.

English abstract

Nowadays, polynomial optimization problems are of great interest, and therefore, it is important to have adequate techniques for their resolution. The RLT algorithm is one of these proposals, whose fundamentals are based on branch and bound algorithms. Due to this, the branching rules considered at each iteration of the algorithm acquire special relevance, it is needed find the optimal partition that provides the greatest increase in the lower bound.

Ideally, we would use all possible branching criteria at each node and we would select the optimal one, but this is computationally intractable. At this point, there is thus a need for a model that selects, at each node, the optimal criterion employing statistical learning techniques, specifically, Boosting models based on quantile regression, which seem to have a good behavior in data sets with strong asymmetries and a high number of outliers. The results obtained will be presented graphically and, in an easy interpretable way, carrying out a detailed analysis of them, in a progressive way, justifying the motivation of the different executions performed.

Capítulo 1

Optimización polinómica

Un dos primeiros pasos que damos cando comezamos a introducirnos no mundo da programación matemática é o de diferenciar entre os distintos tipos de problemas cos que nos podemos atopar. Así, unha primeira diferenciación clásica é a de problemas lineais fronte a problemas non lineais.

Rememorando os nosos coñecementos de optimización matemática, todos concordaremos en que existen multitude de técnicas algorítmicas que abordan a resolución dos problemas lineais, garantindo a optimalidade global das solucións obtidas, como por exemplo, o popular método símplex. Malia que os problemas de optimización lineal xorden en multitude de ocasións, tamén son moi habituais, máis do que quizais nos gustaría, os problemas de optimización non lineal. Sen embargo, aínda que indubidablemente é un campo amplamente investigado, non existe tanta diversidade de técnicas para levar a cabo a resolución dos mesmos, debido principalmente á súa dificultade.

A resolución dos problemas de optimización non lineal, é, polo xeral, moito máis complicada que a dos problemas lineais, no sentido de que atopar unha solución e garantir ademais a súa optimalidade global, pode chegar a ser unha tarefa realmente complicada. A estreita relación entre a convexidade e a optimización facilita en gran medida estas labores, ata tal punto en que distinguir entre problema convexo ou non convexo, pode chegar a ser tan crucial como distinguir se é, ou non, lineal, xa que, en liñas xerais e sen entrar en detalles, para un problema convexo, calquera óptimo local será tamén un óptimo global.

A convexidade é unha propiedade altamente desexable, malia que en algunhas ocasións está ausente. Sen embargo, podemos dicir que senta as bases cara outros enfoques de resolución. Abordar a resolución dos problemas non lineais dividíndoos en subgrupos con características comúns, como por exemplo, problemas con función obxectivo bilineal ou sen restricións, facilitará a busca dunha posible solución, o que suporía, ademais, un avance no marco xeral.

Os problemas de optimización polinómica son un bo exemplo de problemas non lineais que presentan gran interese a nivel teórico, mais tamén na práctica, xa que xorden en multitude de situacións, como por exemplo, no procesamento de sinais, Aittomäki et al. (2009), nas finanzas, Zeng et al. (2017), González et al. (2020), ou no sector enerxético (*pooling problems*), Kimizuka et al. (2019), entre outras, Ghaddar et al. (2016). A principal característica destes problemas é a forma polinómica das funcións obxectivo e das restricións. Ao longo do século XXI, a investigación neste tipo de problemas incrementouse amplamente, o que tivo como consecuencia o desenvolvemento de varios algoritmos que permiten atopar unha solución global a este tipo de problemas.

Neste capítulo presentaremos os problemas de programación polinómica. Comezaremos con algúns conceptos moi básicos e intuitivos, como os de monomio ou polinomio, e outros que non o son tanto, como os multiconxuntos. Será necesario asimilar correctamente todos eles, para comprender a formulación matemática dos problemas polinómicos. Posteriormente, introduciremos unha relaxación lineal destes problemas, que, xunto coa amplamente coñecida técnica de *branch and bound*, empregada para a resolución de problemas enteiros e mixtos, conforman os ingredientes clave para a resolución dos problemas polinómicos. Finalmente, e sen entrar demasiado en detalle, introduciremos o algoritmo RLT

(Reformulation-Linearization Technique), co que resolveremos os distintos problemas involucrados no desenvolvemento deste traballo. Para unha descrición máis ampla e detallada dos contidos do capítulo pode consultarse González-Rodríguez (2017), referencia empregada para a redacción do mesmo.

1.1. Conceptos previos

Dende que comezamos a dar os nosos primeiros pasos na álxebra, estivemos manexando case acotío monomios e polinomios dunha forma moi natural e intuitiva, pois son conceptos, que como ben vimos dicindo, se caracterizan pola súa sinxeleza. Malia ser moi intuitivos, recordaremos unha vez máis a súa definición, prestando especial atención á expresión alxébrica que os caracteriza. Unha vez familiarizados con ela, veremos os problemas de optimización polinómica dunha maneira máis amigable.

Definición 1.1. Sexa $\boldsymbol{x} = (x_1, \ldots, x_n)^\top$ un vector de n variables. Un *monomio*, $m(\boldsymbol{x})$, é unha expresión alxébrica da forma

$$m(\boldsymbol{x}) = \alpha \prod_{j=1}^{n} x_j^{\beta_j}, \quad \alpha \in \mathbb{R}, \beta_j \in \mathbb{N} \cup \{0\}, \forall j \in \{1, \ldots, n\}.$$

Ademais, defínese o *grao do monomio* $m(\boldsymbol{x})$ como a suma dos expoñentes de cada unha das variables, isto é, $\sum_{j=1}^{n} \beta_j$. De maneira similar, definiremos un *polinomio*, $\phi(\boldsymbol{x})$, como a suma dun número finito de monomios multiplicados por uns determinados coeficientes reais, é dicir, calquera expresión alxébrica que presente a seguinte forma:

$$\phi(\boldsymbol{x}) = \sum_{t \in T} \alpha_t \prod_{j=1}^{n} x_j^{\beta_{tj}},$$

sendo $T \subset \mathbb{N}$ un conxunto de índices, $\alpha_t \in \mathbb{R}$, para todo $t \in T$ e $\beta_{tj} \in \mathbb{N} \cup \{0\}$, para todo $j \in \{1, \ldots, n\}$, $t \in T$. Ademais, definiremos o *grao do polinomio* $\phi(\boldsymbol{x})$ como o máximo grao dos monomios que o conforman, isto é, $\max_{t \in T} \{\sum_{j=1}^{n} \beta_{tj}\}$, sempre e cando $\alpha_t \neq 0$.

Ambos son conceptos que calquera persoa coñece e domina, mais convén ter clara a súa definición formal. Isto é debido a que, aínda que os problemas de optimización polinómica presentan función obxectivo e restricións polinómicas, non adoita presentarse a súa formulación matemática empregando as expresións que vimos de definir, senón dun xeito alternativo, que, aínda que equivalente, é lixeiramente menos intuitivo, empregando o que se coñece como multiconxuntos. Para comprender a equivalencia entre as expresións orixinais, vistas anteriormente na definición de polinomio, e as novas, obtidas mediante multiconxuntos, é necesario ter en mente a estrutura matemática das primeiras.

No que segue, pasaremos a introducir brevemente os multiconxuntos así como algúns exemplos e aclaracións, que poden resultar clarificadores na posterior formulación dos problemas de optimización polinómicos.

Definición 1.2. Dados un conxunto finito N e unha aplicación $p : N \to \mathbb{N}$, referirémonos ao par (N, p) como *multiconxunto*, sendo p unha aplicación que indica a multiplicidade de cada un dos elementos do conxunto N. Nótese que $|(N, p)| := \sum_{j \in N} p(j)$.

Como vimos de comentar, o motivo de introducir o concepto de multiconxunto neste traballo débese unicamente a facilitar a comprensión da formulación alternativa que presentaremos para os problemas de programación polinómica, evitando afondar máis no tema de xeito innecesario. Por tal motivo, permitirémonos ademais, un pequeno abuso de notación. Así, dado un conxunto $N = \{1, \ldots, n\}$, e unha aplicación $p : N \to \mathbb{N}$, suporemos que o correspondente multiconxunto se pode expresar como $(N, p) = \{1, \overset{p(1)}{\ldots}, 1, \ldots, n, \overset{p(n)}{\ldots}, n\}$. Un caso particular que xurdirá ao longo deste capítulo, será aquel no que a aplicación p é constante, é dicir, $p(j) = \delta \in \mathbb{N}$, para todo $j \in N$, denotando ao conxunto $(N, p) = \{1, \overset{\delta}{\ldots}, 1, \ldots, n, \overset{\delta}{\ldots}, n\}$ como N^δ.

A continuación presentaremos varios exemplos a modo ilustrativo, para comprender mellor todo o que vimos de comentar, e que, á súa vez, serán interesantes de cara a posteriores definicións.

Exemplo 1.3. Sexan $N = \{1, 2, 3, 4\}$ un conxunto finito e $p : N \to \mathbb{N}$ unha aplicación tal que $p(1) = 1$, $p(2) = 3$, $p(3) = 2$ e $p(4) = 1$. Tendo en conta a anterior definición, o multiconxunto asociado a N e p é $(N, p) = \{1, 2, 2, 2, 3, 3, 4\}$.

Exemplo 1.4. Sexa $N = \{1, 2, 3\}$ e supoñamos que $p : N \to \mathbb{N}$ é unha aplicación constante igual a 2, é dicir, $p(i) = 2$, para todo $i \in \{1, 2, 3\}$. Entón, o correspondente multiconxunto é $N^2 = (N, p) = \{1, 1, 2, 2, 3, 3\}$.

Observemos que, empregando os multiconxuntos, podemos reescribir as expresións alxébricas de monomio e polinomio vistas anteriormente. Así, dado un monomio $m(x) = \alpha \prod_{j=1}^{n} x_j^{\beta_j}$, $\alpha \in \mathbb{R}$, de grao $\delta = \sum_{j=1}^{n} \beta_j$, podemos expresalo de xeito equivalente como

$$m(x) = \alpha \prod_{j \in J} x_j, \;\; J = \{1, \overset{\beta_1}{\ldots}, 1, \ldots, n, \overset{\beta_n}{\ldots}, n\} \subset N^{\delta}.$$

De maneira similar, podemos expresar un polinomio da forma $\phi(x) = \sum_{t \in T} \alpha_t \prod_{j=1}^{n} x_j^{\beta_{tj}}$, e grao δ, como

$$\phi(x) = \sum_{t \in T} \alpha_t \prod_{j \in J_t} x_j, \;\; J_t = \{1, \overset{\beta_{t1}}{\ldots}, 1, \ldots, n, \overset{\beta_{tn}}{\ldots}, n\} \subset N^{\delta}, \forall t \in T.$$

1.2. Problemas de optimización polinómica

Os problemas de programación polinómica son problemas de optimización onde tanto a función obxectivo coma as restricións son funcións polinómicas. Este tipo de problemas xorden de xeito natural en multitude de disciplinas e ámbitos, o que os converte nun campo de gran interese dentro do mundo da investigación operativa. Numerosos autores trataron de abordar a resolución dos mesmos empregando diferentes técnicas e focalizándose en casos particulares diversos, como por exemplo os problemas de programación polinómica enteiros binarios. Nesta memoria traballaremos cunha versión máis xeral deste tipo de problemas, considerando como suposición adicional que as variables son continuas e están limitadas, o que nos permitirá definir os *bound factors* e garantir a converxencia do algoritmo, Sherali e Tuncbilek (1991). Deseguido presentamos a definición formal dos mesmos.

Definición 1.5. Un *problema de optimización polinómica*, $PP(\Omega)$, é un problema de programación matemática que presenta a seguinte estrutura:

$$\begin{aligned} \text{minimizar} \quad & \phi_0(\boldsymbol{x}) \\ \text{suxeito a} \quad & \phi_r(\boldsymbol{x}) \geq \beta_r, \;\; r \in \{1, \ldots, R_1\}, \\ & \phi_r(\boldsymbol{x}) = \beta_r, \;\; r \in \{R_1 + 1, \ldots, R\}, \\ & \boldsymbol{x} \in \Omega \subset \mathbb{R}^n, \end{aligned}$$

onde

1. Para cada $r \in \{0, \ldots, R\}$, $\phi_r(\boldsymbol{x}) = \sum_{t \in T_r} \alpha_{rt} \prod_{j \in J_{rt}} x_j$ é un polinomio de grao δ_r, T_r é un conxunto de índices que identifica o número de monomios polos que está formado o polinomio, α_{rt} son os coeficientes asociados aos mesmos e $J_{rt} \subset N^{\delta_r}$ son multiconxuntos con cardinal $|J_{rt}| \leq \delta_r$.

2. $\Omega = \{\boldsymbol{x} \in \mathbb{R}^n : 0 \leq l_j \leq x_j \leq u_j < \infty, j \in \{1, \ldots, n\}\}$. Referirémonos aos u_j como *cotas superiores* e aos l_j como *cotas inferiores*.

Ademais, definiremos o *grao do problema* como o maior dos graos dos polinomios que o conforman, isto é, $\delta = \max_{r \in \{0, \ldots, R\}} \delta_r$.

Observemos que se $\delta = 1$, estamos ante un problema de optimización lineal, para os que xa existen multitude de técnicas de resolución. De agora en diante, suporemos que o grao do problema é como mínimo 2, isto é, $\delta \geq 2$.

Unha pregunta razoable inherente a calquera problema que se nos presente é se é posible atopar unha solución para el. No campo da optimización matemática é habitual dar resposta a esta cuestión empregando algoritmos que devolvan solucións, desexablemente globais, aínda que en ocasións tan só se pode garantir a optimalidade local da solución. Un dos algoritmos máis coñecidos é o de *branch and bound* empregado habitualmente para resolver problemas de programación lineal enteira e mixta. O bo comportamento deste tipo de algoritmos, convérteo nunha opción interesante para a creación de novos algoritmos. As técnicas de *branch and bound*, combinadas cunha relaxación lineal do problema orixinal, dan lugar ao denominado algoritmo RLT, que será o empregado neste traballo para a resolución dos distintos problemas de programación polinómica. Co obxectivo de comprender paso a paso a evolución deste algoritmo, debemos introducir primeiro algúns conceptos. Comezaremos presentando os *bound factors*.

Definición 1.6. Sexa $PP(\Omega)$ un problema de optimización polinómica de grao δ. Os *bound factors* defínense como un conxunto de restricións de desigualdade que veñen definidas polas cotas inferior e superior de cada variable, isto é,

$$x_j - l_j \geq 0, \quad u_j - x_j \geq 0, j \in N.$$

Tal e como comentamos con anterioridade, o algoritmo RLT caracterízase pola sinerxía entre as técnicas de *branch and bound* e certa relaxación lineal do problema polinómico orixinal. Para a definición do problema relaxado serán esenciais as variables RLT.

Definición 1.7. Dado un problema de programación polinómica de grao δ, $PP(\Omega)$, coa estrutura habitual, definiremos as *variables RLT* como

$$X_J = \prod_{j \in J} x_j, \text{para todo } J \subset N^\delta, 2 \leq |J| \leq \delta.$$

Observación 1.8. Observemos que varios conxuntos de índices J do multiconxunto N^δ representan ao mesmo monomio, por exemplo, $N_1 = \{1, 2, 1\}$ e $N_2 = \{2, 1, 1\}$ dan lugar ao monomio $x_1^2 x_2$. Para evitar os posibles problemas orixinados por esta diversificación, suporemos que os elementos de J están en orde crecente.

Chegados a este punto, tan só resta un ingrediente por presentar antes de detallar o algoritmo RLT, a relaxación lineal do problema polinómico considerada para a resolución do mesmo. A continuación detallamos a súa formulación matemática.

Definición 1.9. Sexa $PP(\Omega)$ un problema de programación polinómica como o da Definición 1.5. Definiremos a súa relaxación lineal como o seguinte problema de optimización

$$\begin{aligned}
\text{minimizar} \quad & [\phi_0(\boldsymbol{x})]_L \\
\text{suxeito a} \quad & [\phi_r(\boldsymbol{x})]_L \geq \beta_r, \ r \in \{1, \ldots, R_1\}, \\
& [\phi_r(\boldsymbol{x})]_L = \beta_r, \ r \in \{R_1 + 1, \ldots, R\}, \\
& \left[\prod_{j \in J_1} (x_j - l_j) \prod_{j \in J_2} (u_j - x_j) \right]_L \geq 0, \forall J_1 \cup J_2 \subset N^\delta, |J_1 \cup J_2| = \delta, \\
& \boldsymbol{x} \in \Omega \subset \mathbb{R}^n,
\end{aligned}$$

e denotarémola por $LP(\Omega)$.

Como podemos observar, ademais das restricións linearizadas do problema orixinal, resultantes de substituír os monomios de grao maior que 1, isto é, distintos de $x_1, \ldots, x_n$, polas correspondentes variables RLT, tamén se incorporan novas restricións linearizadas, orixinadas a partir das distintas combinacións do produto de δ bound factors.

1.2.1. Algoritmo RLT

Os problemas de optimización polinómicos son unha xeneralización doutros ben coñecidos como poden ser os problemas de programación lineal enteira ou os problemas cadráticos. Sen embargo, a súa resolución pode chegar a ser realmente complexa se a función obxectivo ou o conxunto factible non son convexos. O algoritmo RLT foi introducido por primeira vez en Sherali e Tuncbilek (1991) como unha proposta para resolver este tipo de problemas e será tamén o que empregaremos para o desenvolvemento deste traballo. A partir do problema orixinal, modifícase a súa estrutura, sustituíndo os monomios compostos por varias variables por novas variables, coñecidas como variables RLT, obtendo unha relaxación linear como na Definición (1.6). Iteración a iteración, empregando técnicas de ramificación e limitación, particionarase a rexión factible de orixe noutras máis pequenas a partir das solucións obtidas para cada subproblema, que á súa vez, tamén serán relaxacións lineais do problema polinómico de partida.

Habitualmente, antes de inicializar o algoritmo fíxase un criterio de ramificación, que será empregado para determinar a variable ramificadora a partir das solucións das relaxacións lineais, tendo en conta en que medida cada solución viola as restricións do problema orixinal mediante as violacións RLT. Unha das regras de ramificación máis coñecidas é o criterio do máximo, que ramifica por aquela variable que produce unha maior infactibilidade no problema polinómico inicial. Matematicamente, dado un subproblema, ramificaremos pola variable x_p, cando

$$p \in \arg\max_{j \in N} \{\theta_j\}, \text{ onde}$$

$$\theta_j = \max_{J \subset N^\delta, |J| \leq \delta - 1} \{\bar{X}_{J \cup j} - \bar{x}_j \bar{X}_J|\}, \forall j \in N,$$

sendo $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{X}})$ a solución óptima obtida para o problema relaxado en cuestión. Como veremos no seguinte capítulo, existen outras moitas regras de ramificación que, dependendo das características do problema, poden actuar mellor ou peor.

Retomando o fío inicial, ao resolver cada unha das relaxacións lineais existentes, obtemos unha nova cota inferior da solución óptima do problema polinómico orixinal. Tal e como ocorre cando aplicamos o algoritmo de *branch and bound* a problemas enteiros, en cada iteración do algoritmo RLT levarase a cabo unha mellora progresiva da mesma. Cando, para a solución dunha determinada relaxación linear, as correspondentes violacións RLT sexan nulas para cada unha das variables, isto é, $\theta_j = 0$ en (3.1), $j \in \{1, \ldots, n\}$, o valor asociado á función obxectivo proporciona unha cota superior da solución óptima do problema polinómico de partida, inicializada, en principio, a infinito [1]. Deste xeito, acharemos a solución óptima do problema cando ambas cotas coincidan, ou, dito doutro xeito, o algoritmo deterase cando non teñamos en cola ningún problema cuxa solución mellore a cota superior existente, inicializada a cero, e tomarase como solución óptima esta mesma.

Chegar a un punto no que ambas cotas coincidan de xeito exacto pode ser custoso en tempo, posto que, cabe a posibilidade de que existan problemas en cola para os que a mellora na cota superior, aínda que ínfima, sexa posible. Para evitar este tipo de situacións, que poden chegar a ser computacionalmente moi custosas, é habitual definir unha certa tolerancia que permita garantir a optimalidade dunha solución cando a diferenza entre ditas cotas sexa menor que unha determinada cantidade próxima a cero.

De seguido presentamos unha descrición máis detallada do algoritmo que vimos de presentar tomada das referencias Rodríguez-Ballesteros (2022) e González-Rodríguez (2017).

[1] En problemas complexos, é aconsellable empregar algún solver non lineal, coma por exemplo Knitro, para determinar unha cota superior máis adecuada que infinito.

Algoritmo 1 Algoritmo Reformulation-Linearization Tecnique (RLT)

- **Inicialización.**
  - Inicializamos a mellor solución como $\boldsymbol{x}^* = \emptyset$ e o mellor valor obxectivo como $v^* = \infty$. Así mesmo, tomamos $k = 1$, $T_1 = \{1\}$ e $\Omega^{1,1} = \Omega$.
  - Resolvemos a relaxación linear $LP(\Omega^{1,1})$, denotando á solución óptima $(\boldsymbol{x}^{1,1}, \boldsymbol{X}^{1,1})$.
  - Determinamos a variable ramificadora, x_p, $p \in \{1, \ldots, n\}$ empregando o criterio de ramificación previamente fixado. Se $\theta_p = 0$ para todo $p \in \{1, \ldots, n\}$, o algoritmo detense posto que $\boldsymbol{x}^{1,1}$ resolve o problema orixinal $PP(\Omega)$. Deste xeito, $\boldsymbol{x}^* = \boldsymbol{x}^{1,1}$ e $v^* = v[LP(\Omega^{1,1})]$. En caso contrario, tomamos $t = 1$ e pasamos a Etapa 1.
- **Etapa 1 (Ramificación).** Supoñemos resolto o problema $LP(\Omega^{k,t})$, para o cal se seleccionou como variable de ramificación x_p e $\theta > 0$. Dado que $\theta > 0$, tense que $l_p^{k,t} < x_p^{k,t} < u_p^{k,t}$ sendo $l_p^{k,t}, u_p^{k,t}$ as cotas inferior e superior de x_p, respectivamente. Pode consultarse a demostración desta desigualdade no Lema 3 de Sherali e Tuncbilek (1991). De seguido, consideramos a seguinte partición de $\Omega^{k,t}$:

$$\Omega^{k,t_1} = \Omega^{k,t} \cap \{\boldsymbol{x} \in \mathbb{R}^n : l_p^{k,t} \leq x_p \leq x_p^{k,t}\}$$
$$\Omega^{k,t_2} = \Omega^{k,t} \cap \{\boldsymbol{x} \in \mathbb{R}^n : x_p^{k,t} \leq x_p \leq u_p^{k,t}\},$$

  onde $t_1, t_2 \neq T_k$. A continuación, actualizamos $T_k = (T_k - \{t\}) \cup \{t_1, t_2\}$.
- **Etapa 2 (Limitación).** Resolvemos as relaxacións lineais obtidas a partir das particións xeradas pola rexión factible. Comezamos con $LP(\Omega^{k,t_1})$, denotando a súa solución por $(\boldsymbol{x}^{k,t_1}, \boldsymbol{X}^{k,t_1})$, con valor obxectivo $LB_{k,t_1} = v[LP(\Omega^{k,t_1})]$. Posteriomente, determinamos a variable ramificadora e distinguimos a seguinte casuística:
  - Se $\theta_p = 0$, entón $\boldsymbol{x}^{k,t_1}$ resolve $PP(\Omega^{k,t_1})$ e proporciona, a súa vez, unha cota superior para o problema orixinal, $PP(\Omega)$. Se ademais, $v[LP(\Omega^{k,t_1})] < v^*$, é dicir, dita cota é mellor que a actual, procederemos coa actualización da mellor solución e do mellor valor obxetivo, $\boldsymbol{x}^* = \boldsymbol{x}^{k,t_1}$ e $v^* = LB_{k,t_1}$.
  - Se $\theta_p > 0$, gardamos o índice da variable ramificadora, p.

  Repetimos o mesmo procedemento con $LP(\Omega^{k,t_1})$ e pasamos á Etapa 3.
- **Etapa 3 (Poda).** Definimos o conxunto $T_{k+1} = T_k \backslash \{t \in T_k : LB_{k,t} \geq v^*\}$, é dicir, eliminando os nodos que dan lugar a unha maior cota superior que a actual.
  - Se $T_{k+1} = \emptyset$, paramos.
  - Noutro caso, actualizamos $\Omega^{k+1,t} = \Omega^{k,t}$, $(\boldsymbol{x}^{k+1,t}, \boldsymbol{X}^{k+1,t}) = (\boldsymbol{x}^{k,t}, \boldsymbol{X}^{k,t})$, $LB_{k+1,t} = LB_{k,t}$, $t \in T_{k+1}$ e $k = k + 1$.
- **Etapa 4 (Selección do nodo).** Seleccionamos aquel nodo (k, t) de T_k con valor mínimo da cota inferior, isto é, $t \in T_k$ tal que

$$t \in \arg\min_{t \in T_k} \{LB_{k,t}\}.$$

  Posteriomente, volvemos á Etapa 1.

Chegados a este punto, deixaremos a un lado os detalles teóricos do algoritmo, que poden consultarse en Sherali e Tuncbilek (1991), e centrarémonos en ilustrar a súa evolución de maneira sinxela, paso a paso.

Exemplo 1.10. Consideremos o seguinte problema de optimización polinómica (PP):

$$\begin{aligned} \text{minimizar} \quad & x_1^2 + x_2^2 + x_1x_2 + x_1x_2^2 + x_2 \\ \text{suxeito a} \quad & x_1x_2 \geq 1, \\ & x_1^2 + x_2 \geq 5, \\ & 2 \leq x_1 \leq 7, \\ & 0 \leq x_2 \leq 3. \end{aligned}$$

Como podemos observar, o problema ten $n = 2$ variables, x_1 e x_2, e o seu grao é 3, posto que o grao do polinomio de maior grao $x_1^2 + x_2^2 + x_1x_2 + x_1x_2^2 + x_2$ é 3. Para a súa resolución empregaremos o algoritmo RLT, pero primeiro é necesario calcular as variables RLT asociadas ao problema. A partir da Definición 1.7, definimos as variables RLT como

$$\begin{aligned} X_{11} &= x_1^2, & X_{112} &= x_1^2x_2, \\ X_{12} &= x_1x_2, & X_{122} &= x_1x_2^2, \\ X_{22} &= x_2^2, & X_{222} &= x_2^3. \\ X_{111} &= x_1^3, \end{aligned}$$

No que segue, especificaremos todos os pasos levados a cabo polo algoritmo RLT para resolver o problema orixinal.

- Inicialización.
  - Inicializamos a mellor solución como $\boldsymbol{x}^* = \emptyset$ e o mellor valor obxectivo como $\boldsymbol{v} = \infty$. Ademais, tomamos $k = 1$, $T_1 = \{1\}$ e $\Omega^{1,1} = \Omega$.
  - A continuación, resolvemos a relaxación lineal do problema orixinal, $LP(\Omega^{1,1})$,

$$\begin{aligned} \text{minimizar} \quad & X_{11} + X_{22} + X_{12} + X_{122} + x_2 \\ \text{suxeito a} \quad & X_{12} \geq 1, \\ & X_{11} + x_2 \geq 5, \\ & \left[\prod_{j \in J_1} (x_j - l_j) \prod_{j \in J_2} (u_j - x_j) \right]_L \geq 0, \ \forall J_1, J_2 \in N^\delta, |J_1 \cup J_2| = 3, \\ & \boldsymbol{x} \in \Omega^{1,1} \subset \mathbb{R}^3, \end{aligned}$$

  As restricións orixinadas a partir dos multiconxuntos, $J_1, J_2 \subset N^3 = \{1, 1, 1, 2, 2, 2\}$, correspóndense coa linearización dos bound factors a partir das cotas inferiores e superiores de cada variable orixinal, $l_1 = 2, l_2 = 0, u_1 = 7, u_2 = 3$, substituíndo os produtos de x_1 e x_2 polas correspondentes variables RLT. Para a resolución de $LP(\Omega^{1,1})$ empregouse a linguaxe de programación AMPL xunto co solver Gurobi, obtendo como solución:

$$(\boldsymbol{x}^{1,1}, \boldsymbol{X}^{1,1}) = (2.06, 0.5, 4.5, 1, 0, 0), v[LP(\Omega^{1,1})] = \ 6.$$

  Dita solución proporciona unha cota inferior inicial para o óptimo do problema orixinal, $LB(PP) = LB_{1,1} = 6$, que, nas sucesivas iteracións, será actualizada.

- O seguinte paso sería determinar a variable de ramificación, x_p, $p \in \{1,2\}$, para o que é necesario establecer un criterio de ramificación. Este é quizais o punto máis importante do exemplo, que servirá para motivar o desenvolvemento desta memoria. Sobre el volveremos máis adiante, polo momento, suporemos que o criterio de ramificación será o criterio do máximo, visto con anterioridade. Así, para o problema $LP(\Omega^{1,1})$, tense que

$$\theta_1 = \text{máx}\{|X_{11} - x_1^2|, |X_{12} - x_1x_2|, |X_{122} - x_1X_{22}|\} = 0.27.$$
$$\theta_2 = \text{máx}\{|X_{12} - x_2x_1|, |X_{22} - x_2^2|, |X_{122} - x_2X_{12}|\} = 0.5.$$

  Ambos son non nulos, polo tanto, a variable ramificadora é x_2.

- Etapa 1 (Ramificación). Chegados a este punto, xa resolvemos a relaxación lineal do problema orixinal e tamén determinamos a variable ramificadora empregando o criterio do máximo. Sexa $t = 1$. Dado que $\theta_2 > 0$, verifícase que $l_2^{k,t} < x_2^{k,t} < u_2^{k,t}$, é dicir, $0 < x_2^{1,1} < 3$. Polo tanto, consideraremos a seguinte partición da rexión factible $\Omega^{1,1}$:

$$\Omega^{1,2} = \Omega^{1,1} \cap \{\boldsymbol{x} \in \mathbb{R}^n : l_p^{1,1} \leq x_p \leq x_p^{1,1}\} = \Omega^{1,1} \cap \{\boldsymbol{x} \in \mathbb{R}^n : 0 \leq x_2 \leq 0.5\},$$
$$\Omega^{1,3} = \Omega^{1,1} \cap \{\boldsymbol{x} \in \mathbb{R}^n : x_p^{1,1} \leq x_p \leq u_p^{1,1}\} = \Omega^{1,1} \cap \{\boldsymbol{x} \in \mathbb{R}^n : 0.5 \leq x_2 \leq 3\}.$$

  A continuación, actualizamos $T_k = \{2,3\}$.

- Etapa 2 (Limitación). Nesta nova etapa resolveremos os problemas $LP(\Omega^{1,2})$ e $LP(\Omega^{1,3})$ e, a partir da solución obtida, determinaremos as variables ramificadoras candidatas, para continuar particionando a rexión factible, de ser necesario. Os resultados obtidos recóllense na seguinte táboa:

Problema	Solución óptima	Valor óptimo	Variable Ramificadora
$LP(\Omega^{1,2})$	$(\boldsymbol{x}^{1,2}, \boldsymbol{X}^{1,2}) = (2.13, 0.47, 4.53, 1, 0.22, 0.47)$	$v[LP(\Omega^{1,2})] = 6.68$	x_1
$LP(\Omega^{1,3})$	$(\boldsymbol{x}^{1,3}, \boldsymbol{X}^{1,3}) = (2.06, 0.5, 4.5, 1.03, 0.25, 0.51)$	$v[LP(\Omega^{1,2})] = 6.79$	x_1

Táboa 1.1: Solucións obtidas para $LP(\Omega^{1,2})$ e $LP(\Omega^{1,3})$.

- Etapa 3 (Poda). O seguinte paso é eliminar de T_1 os nodos que orixinan un valor da función obxetivo máis grande que o que xa temos, $v = \infty$. Para iso, definimos $T_2 = T_1 \backslash \{t \in T_1 : LB_{1,t} \geq v\} = \{2,3\}$. Así,

$$\Omega^{2,2} = \Omega^{1,2}, \quad (\boldsymbol{x}^{2,2}, \boldsymbol{X}^{2,2}) = (\boldsymbol{x}^{1,2}, \boldsymbol{X}^{1,2}), \quad LB_{2,2} = LB_{1,2} = 6.68.$$
$$\Omega^{2,3} = \Omega^{1,3}, \quad (\boldsymbol{x}^{2,3}, \boldsymbol{X}^{2,3}) = (\boldsymbol{x}^{1,3}, \boldsymbol{X}^{1,3}), \quad LB_{2,3} = LB_{1,3} = 6.79.$$

  Ademais, actualizamos o valor de k, $k = 2$.

- Etapa 4 (Selección do nodo). Seleccionamos aquel nodo activo, $(k,t) \in \{(2,2),(2,3)\}$, que minimice o valor da función obxectivo, proporcionando a mellor cota superior en caso de que $\theta_1 = \theta_2 = 0$, isto é,

$$t \in \underset{t \in T_2}{\arg\min}\, \{LB_{k,t}\} = \underset{t \in \{2,3\}}{\arg\min}\, \{LB_{2,t}\} = \underset{t \in \{2,3\}}{\arg\min}\, \{6.68, 6.79\} = 2,$$

  actualizamos a cota inferior do problema, $LB(PP) = LB_{2,2} = 6.68$, e volvemos á etapa 1.

Dado que o procedemento para as seguintes iteracións é idéntico ao que vimos de describir, non continuaremos co seu desenvolvemento paso por paso (Figura 1.1), limitarémonos a detallar o momento de parada do algoritmo.

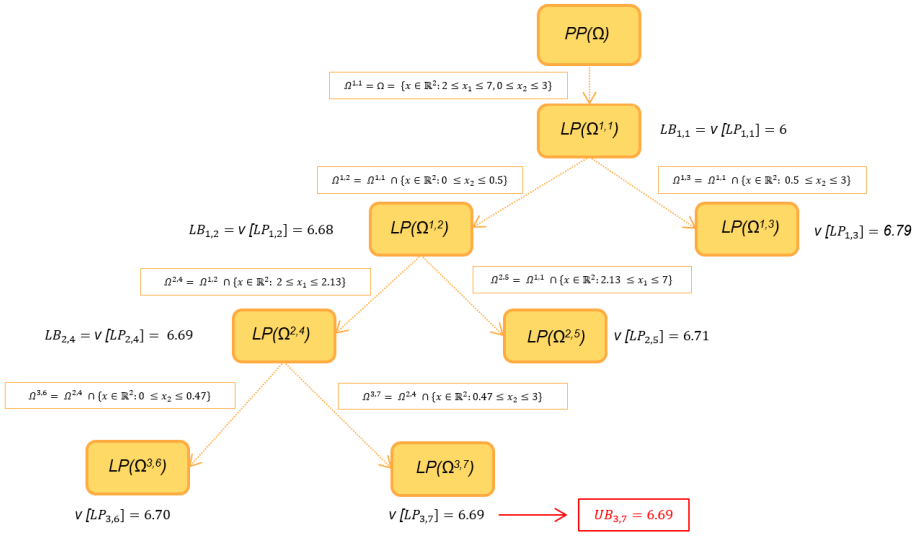


Figura 1.1: Evolución do algoritmo RLT para o Exemplo 1.10.

Durante a terceira iteración do algoritmo, na Etapa 2, debemos determinar novamente as variables ramificadoras para os problemas $LP(\Omega^{3,6})$ e $LP(\Omega^{3,7})$. Sen embargo, para este último obtense que $\theta_1, \theta_2 \approx 0$, polo tanto, podemos concluír que a solución do problema $LP(\Omega^{3,7})$,

$$(\boldsymbol{x}^{3,7}, \boldsymbol{X}^{3,7}) = (2.13, 0.47, 4.53, 1, 0.22, 0.47),$$

é unha solución factible para $PP(\Omega^{3,7})$, proporcionando á súa vez, unha cota superior para $PP(\Omega)$, $v[LP(\Omega^{3,7})] = 6.69$. Dito doutro xeito, esta solución proporciona un menor valor na función obxectivo que o definido no inicio do algoritmo, $v^* = \infty$, co cal, actualizaremos tanto a mellor solución como o mellor obxectivo a $\boldsymbol{x}^* = (2.13, 0.47)$ e $v^* = 6.69$. Ningún dos nodos restantes proporcionará unha cota superior mellor que a que xa temos, é dicir, máis pequena, polo tanto, é innecesaria a ramificación dos problemas restantes, e o algoritmo detense.

Un aspecto clave no desenvolvemento e velocidade de converxencia do algoritmo RLT é a forma na que se realizan as particións, ou, dito doutro xeito, como se seleccionan as variables ramificadoras. Parece razoable pensar que, dependendo do problema considerado, o criterio do máximo pode ser máis ou menos adecuado, seleccionando nalgúns casos a variable ramificadora de xeito óptimo, e noutros non tanto. Isto abre a porta a novas metodoloxías como por exemplo o criterio dos pesos constantes ou dos valores duais, entre outros.

Malia que o abanico de posibilidades é amplo, segue sendo fundamental determinar cal é o criterio que se adecúa mellor a cada problema. A raíz disto, xurdiron numerosas propostas para abordar dita cuestión. Unha das máis populares, e tamén sobre a que traballaremos nesta memoria, é a aplicación de técnicas de aprendizaxe estatística. En Rodríguez-Ballesteros (2022), empregáronse este tipo de técnicas para seleccionar de forma automática o método de ramificación óptimo, dentro dun conxunto de criterios predefinidos, para cada problema de optimización polinómica en base as súas características. Os modelos analizados no estudo presentaron, polo xeral, un bo comportamento, e indubidablemente unha mellor maneira de proceder que a que resulta dunha selección arbitraria. Cabe destacar que o

criterio seleccionado é único en todo o problema, isto é, para determinar a variable ramificadora en cada unha das relaxacións lineais, o criterio empregado é sempre o mesmo. Por exemplo, se decidimos empregar o criterio do máximo, este non cambiará ao longo de todo o problema. Sen embargo, as distintas relaxacións que teñamos que resolver, poden chegar a ser moi distintas entre sí, e en particular, da relaxación lineal inicial, facendo que nos cuestionemos se quizais a selección única dun criterio encaixa neste tipo de situacións.

Os desenvolvementos realizados neste traballo enfócanse coma unha extensión natural dos resultados obtidos en Rodríguez-Ballesteros (2022), co fin de determinar o criterio de ramificación máis adecuado, dun xeito personalizado para cada un dos distintos subproblemas resultantes a partir do problema de programación polinómica orixinal. Así mesmo, abordaremos esta cuestión empregando diversas metodoloxías de aprendizaxe supervisada, soportándonos nos resultados existentes para realizar as comparativas e, precisar se a selección dun criterio distinto para cada subproblema resulta ou non vantaxosa.

Capítulo 2

Modelos de Aprendizaxe Estatística

No eido das matemáticas, dependendo do tipo de problema tratado, ben de regresión ou ben de clasificación, existe un amplo número de metodoloxías de aprendizaxe estatística para tratar de predicir o valor que da variable de interese a partir da información dun conxunto de variables explicativas. Exemplos disto poden ser desde o clásico modelo de regresión linear ata modelos máis complexos como as redes neuronais ou as máquinas de soporte vectorial (SVM), entre outros. De maneira xeral, o proceso de aprendizaxe basease no particionamento do conxunto de datos, habitualmente en dúas submostras: a mostra de adestramento e a mostra de test. A primeira, empregarase para a construción dos modelos, e a mostra de test, para analizar o comportamento do modelo, de cara a obter unha previsión do seu desempeño sobre novas observacións, mellorando certos aspectos cando sexa oportuno. Sobre este tema volveremos máis adiante cando se describa en detalle a metodoloxía seguida para o aprendizaxe.

Os modelos empregados nesta memoria resultan da aplicación do algoritmo Stochastic Gradient Boosting a diferentes conxuntos de datos. É amplamente coñecido que o Boosting é un método estreitamente relacionado coas árbores de decisión, co que, como primera aproximación para chegar a comprender o algoritmo presentaremos estas últimas, modelos sinxelos e facilmente interpretables, que sentan as bases desta metodoloxía, empregando como guía os apuntamentos da materia Aprendizaxe Estatística, Fernández-Casal et al. (2021). Posteriormente, describiremos en detalle o método Boosting e centrarémonos no caso concreto do Stochastic Gradient Boosting, que como veremos está fortemente ligado ao método de optimización do gradiente, analizando en detalle a variante baseada en regresión cuantil. [1]

2.1. Árbores de decisión

As árbores de decisión son un modelo preditivo aplicable tanto a problemas de regresión como de clasificación, caracterizado pola súa sinxeleza e fácil interpretación. Polo xeral, a súa capacidade preditiva adoita ser mediocre, en especial para problemas de regresión, polo que nestes casos soe empregarse como un método meramente descritivo. En calquera caso, cabe destacar a súa importancia pois sentan as bases de metodoloxías máis complexas e amplamente empregadas coma por exemplo o Boosting, empregado neste traballo.

A idea subxacente é simple. Consiste en dividir o espazo preditor en distintos subconxuntos empregando particións que orixinan rexións rectangulares definidas polas distintas variables explicativas, que poden ser tanto numéricas como categóricas. Tras un número finito de particións, a árbore xerada estará composta dun certo número de nodos terminais ou follas, que se empregarán para realizar a predición. Deste xeito, para os problemas de regresión, dada unha nova observación, localízase o nodo terminal ao que pertence e devólvese como predición unha constante que coincide coa media mostral

[1] Neste capítulo destacaremos as variables aleatorias multidimensionais en negriña e en maiúscula, por exemplo, $\boldsymbol{X}$. Así mesmo, identificaremos os vectores como $\boldsymbol{x}$.

das observacións que conforman ese nodo. É dicir, a predición é constante en cada nodo terminal, motivo que xustifica que o seu poder preditivo sexa especialmente limitado nos casos de regresión. No caso dos problemas de clasificación, empregarase no seu lugar a moda. Como podemos imaxinarnos, para aqueles espazos preditores que non resultan sinxelos de particionar en rexións rectangulares, as árbores de decisión non proporcionarán bos resultados.

Posto que no noso caso traballaremos con un problema de regresión, introduciremos, aínda que non en detalle, o fundamento das árbores de regresión baseadas na metodoloxía CART, Breiman et al. (1984).

Un aspecto clave para a construción dunha árbore de regresión como modelo preditivo é a maneira de definir as sucesivas particións que darán lugar ás distintas rexións, R_j, $j \in \{1, \ldots, k\}$. En cada nodo terminal R_j, a predición que daremos para a variable resposta será constante, co que, unha boa forma de proceder é seleccionar as rexións de tal maneira que se minimice un termo de erro, por exemplo, a suma residual de cadrados. Sexan $(\boldsymbol{x}_i, y_i)_{i=1}^n$ o conxunto de observacións co que estamos a traballar. Así, na rexión R_j, interesa devolver como predición aquel c_j tal que

$$\min_{c_j} \sum_{\boldsymbol{x}_i \in R_j} (y_i - c_j)^2.$$

A anterior expresión acada o seu valor mínimo na media mostral de $\{y_i\}_{i \in \{1,\ldots,n\}}$ na rexión R_j, que denotaremos por $\hat{y}_{R_j}$. Deste xeito, a selección óptima das rexións $R_1, \ldots, R_J$ obterase minimizando a seguinte expresión:

$$\sum_{j=1}^{J} \sum_{\boldsymbol{x}_i \in R_j} (y_i - \hat{y}_{R_j})^2.$$

Na práctica, o anterior problema de minimización é realmente complexo e require certa simplificación. Aquí é onde se introduce a metodoloxía CART, que, basicamente, aborda o problema buscando particións adecuadas mediante cortes binarios, e non considerando todas as posibilidades, o que é computacionalmente intratable. Supoñamos que o nodo inicial está formado por todas as observacións. Unha variable explicativa X_j e un punto de corte s, definen dous subconxuntos:

$$R_1 = \{X : X_j \leq s\},$$

$$R_2 = \{X : X_j > s\}.$$

Seleccionaremos a variable e o punto de corte de tal maneira que se minimice a seguinte expresión:

$$\sum_{\boldsymbol{x}_i \in R_1} (y_i - \hat{y}_{R1})^2 + \sum_{\boldsymbol{x}_i \in R_2} (y_i - \hat{y}_{R2})^2.$$

De seguido, aplicarase o mesmo proceso nas rexións resultantes, R_1 e R_2, e continuaremos de forma iterativa ate establecer un criterio de parada, ben, alcanzando certa profundidade máxima, ou ben, exixindo un número mínimo de observacións nos nodos terminais.

Notemos que se construímos unha árbore demasiado grande, corremos o risco de obter un modelo sobreaxustado ou cunha interpretación complexa. Pola contra, se a árbore é demasiado pequena, as predicións serán malas, aínda que a súa interpretabilidade sinxela é evidente. Esta balanza lévanos a considerar a seguinte estratexia. Farase crecer unha árbore grande e posteriormente colapsaranse algúns dos seus nodos. Isto dará lugar a unha árbore máis pequena, cun maior erro sobre a mostra de adestramento que a árbore completa, pero que actúa mellor sobre a mostra de test e sobre novas observacións. Sen embargo, analizar unha por unnha cal das subárbores presenta menor erro é un problema complexo. Neste caso o que se fai é introducir un hiperparámetro, que controla a complexidade do modelo e selecciona a subárbore óptima cos datos de adestramento. O seu valor pode seleccionarse empregando a mostra de validación ou validación cruzada, entre outros.

2.2. Boosting

En aprendizaxe estatística existen numerosas técnicas estatísticas que abordan a resolución de problemas de aprendizaxe supervisada, tanto para regresión coma para clasificación. O obxetivo que perseguen todas elas é común: a partir dun conxunto de observacións proporcionar un modelo que sexa quen de predicir o valor da variable resposta ao aplicalo sobre novos conxuntos datos. En torno á década de 1990, xurdiron os métodos de ensamblado. Baixo este nome englóbase unha serie de técnicas preditivas cuxa formulación se basea na combinación de múltiples preditores sinxelos e cunha alta varianza, *base learners*. Este tipo de metodoloxías son moi comúns e entre elas atópanse o Boosting, que será na que nos focalizaremos. Pode consultarse máis información sobre os métodos de ensamblado en Dietterich (2000).

O Boosting é unha método de aprendizaxe que xurdiu fai pouco máis de 20 anos das mans de Robert Schapire en 1990, Schapire (1990). A versión inicial foi o AdaBoost Discreto e foi formulada para problemas de clasificación binarios. Esencialmente a idea é partir do conxunto de adestramento, asignar pesos a cada unha das observacións que o compoñen e empregalas para a elaboración de múltiples árbores de decisión. A medida que avanza o algoritmo, os pesos das observacións ben clasificadas manteranse constantes, e aqueles nas que se errou, modificaranse consecuentemente asignándolles un valor maior. É evidente cada iteración depende do que ocorreu na anterior, o que implica que as arbores xeradas non son independentes. Esta mesma idea foi adaptada para os problemas de regresión, dando lugar ao Real AdaBoost, Friedman et. al (2000).

O Stochastic Gradient Boosting é unha variante da metolodoxía Boosting proposta por Jerome Friedman no 2002, Friedman (2002). O obxectivo principal é obter unha función $\hat{f}$ que modele a relación existente entre as variables explicativas, $\boldsymbol{X}$, e a variable dependente, Y, a partir dun conxunto de observacións que conforman a mostra de adestramento, $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, minimizando o valor esperado dunha determinada función de perda, $L(Y, f(\boldsymbol{X}))$ [2]. Isto é,

$$\hat{f}(\boldsymbol{X}) = \arg\min_{f(\boldsymbol{X})} \mathbb{E}[L(Y, f(\boldsymbol{X}))].$$

A metodoloxía Boosting asume que a función $\hat{f}$ pode ser representada mediante unha expansión aditiva de preditores débiles, *base learners*:

$$\hat{f}(\boldsymbol{x}) = \sum_{m=0}^{M} \beta_m \boldsymbol{h}_m(\boldsymbol{x}, \boldsymbol{a}_m).$$

Para cada $m \in \{1, \ldots, M\}$, os coeficientes de expansión, β_m, e os parámetros dos preditores base, $\boldsymbol{a}_m$, poden calcularse minimizando a seguinte expresión

$$(\beta_m, \boldsymbol{a}_m) = \arg\min_{\beta, \boldsymbol{a}} \sum_{i=1}^{n} L(y_i, f_{m-1}(\boldsymbol{x}_i) + \beta h(\boldsymbol{x}_i, \boldsymbol{a})),$$

onde

$$f_m(\boldsymbol{x}) = f_{m-1}(\boldsymbol{x}) + \beta h(\boldsymbol{x}, \boldsymbol{a}),$$

inicializando $\boldsymbol{f}_0$ con certo valor prefixado. O anterior problema de minimización é bastante complexo, co que a metolodoxía Boosting propón unha estratexia de resolución alternativa, calculando primeiro os parámetros $\boldsymbol{a}_m$ por mínimos cadrados, converténdoo nun problema facilmente abordable. De forma analítica, para cada $m \in \{1, \ldots, M\}$, comezaremos calculando $\boldsymbol{a}_m$ como

$$\boldsymbol{a}_m = \arg\min_{\boldsymbol{a}, \rho} \sum_{i=1}^{n} [g_{im} - \rho h(\boldsymbol{x}_i, \boldsymbol{a})]^2,$$

[2] $\boldsymbol{X}$ denotará un vector aleatorio formado por k variables aleatorias e Y unha variable aleatoria unidimensional

sendo

$$g_{im} = -\left[\frac{\partial L(y_i, f(\boldsymbol{x}_i))}{\partial f(\boldsymbol{x}_i)}\right]_{f(\boldsymbol{x}_i)=f_{m-1}(\boldsymbol{x}_i)}.$$

Observemos que g_{im} é o gradiente negativo da función de perda avaliada en f_{m-1}, que coincide coa dirección local de máximo descenso, é dicir, a dirección para a que o valor de $L(Y, f(\boldsymbol{X}))$ descende máis rapidamente. Posteriormente, obteremos o valor de β_m como

$$\beta_m = \arg\min_{\beta} \sum_{i=1}^{n} L(y_i, f_{m-1}(\boldsymbol{x}_i) + \beta h(\boldsymbol{x}_i, \boldsymbol{a}_m)). \tag{2.1}$$

En particular, suporemos que os *base learners* son árbores de regresión. Deste xeito, en cada iteración $m \in \{1, \ldots, M\}$, unha árbore de regresión particionará o espazo preditor en L rexións disxuntas, $\{R_{lm}\}_{l=1}^{L}$ asignando unha predición constante a cada unha delas, que coincide coa media mostral das observacións que a conforman, $\bar{y}_{lm}$. Así,

$$h(\boldsymbol{x}, \{R_{lm}\}_{l=1}^{L}) = \sum_{l=1}^{L} \bar{y}_{lm} \mathbb{I}[\boldsymbol{x} \in R_{lm}].$$

Tendo en conta o anterior, o problema de minimización (2.1) é equivalente a:

$$\gamma_{lm} = \arg\min_{\gamma} \sum_{\boldsymbol{x}_i \in R_{lm}} L(y_i, f_{m-1}(\boldsymbol{x}_i) + \gamma), \tag{2.2}$$

e finalmente,

$$f_m(\boldsymbol{x}) = f_{m-1}(\boldsymbol{x}) + \nu \sum_{l=1}^{L} \gamma_{lm} \mathbb{I}[x \in R_{lm}]$$

onde $0 < \nu \leq 1$ é un parámetro de regularización que controla a taxa de aprendizaxe do algoritmo.

Até este punto, para os desenvolvementos teóricos supuxemos que en cada iteración m do algoritmo se consideraba a mostra de adestramento completa, describindo en detalle o coñecido Gradient Boosting. Incorporando certa compoñente de aleatoriedade obtemos o Stochastic Gradient Boosting. Así, en cada iteración seleccionarase ao azar unha submostra do conxunto de adestramento, sen reemplazamento, e será a empregada para os cálculos. Sexa, $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$ o conxunto de observacións que conforman a mostra de train. Denotaremos por $\{(\tilde{\boldsymbol{x}}_{im}, \tilde{y}_{im})\}_{i=1}^{\tilde{n}}$, a submostra aleatoria de tamaño $\tilde{n} < n$ obtida en cada iteración m do algoritmo.

Algoritmo 2 Algoritmo de Stochastic Gradient Boosting

1. Inicializamos $f_0(x) = \arg\min_\gamma \sum_{i=1}^{n} L(y_i, \gamma)$. Esta constante representa a unha árbore de decisión formada por un único nodo.
2. Para cada iteración, $m \in \{1, \ldots, M\}$,
   - Extraemos unha submostra aleatoria do conxunto de adestramento, $T_m = \{(\tilde{\boldsymbol{x}}_{im}, \tilde{y}_{im})\}_{i=1}^{\tilde{n}}$.
   - Calculamos os gradientes negativos, g_{im}, e axustamos unha árbore de decisión a $\{(\tilde{\boldsymbol{x}}_{im}, g_{im})\}_{i=1}^{\tilde{n}}$ que dividirá o espazo preditor en L rexións, $R_{1m}, \ldots, R_{Lm}$.
   $$g_{im} = -\left[\frac{\partial L(\tilde{y}_{im}, f(\tilde{\boldsymbol{x}}_{im}))}{\partial f(\tilde{\boldsymbol{x}}_{im})}\right]_{f(\tilde{\boldsymbol{x}}_{im})=f_{m-1}(\tilde{\boldsymbol{x}}_{im})}, i \in \{1, \ldots \tilde{n}\}.$$
   - Para cada rexión R_{lm}, $l \in \{1, \ldots, L\}$, calculamos γ_{lm} como se indica en (2.2) e actualizamos f_m,
   $$f_m(\boldsymbol{x}) = f_{m-1}(\boldsymbol{x}) + \nu \sum_{l=1}^{L} \gamma_{lm} \mathbb{I}(\boldsymbol{x} \in R_{lm}), 0 < \nu \leq 1.$$
3. O modelo proposto será $\hat{f}(\boldsymbol{x}) = f_M(\boldsymbol{x})$.

Esta descrición xeral do algoritmo dá lugar a distintas versións dependendo da función de perda $L(y, f(\boldsymbol{x}))$ considerada. Algúns exemplos son a función de perda de mínimos cadrados, a diferenza de erros absolutos, a función de Huber e a función de perda cuantílica, sobre a que volveremos máis adiante, e que da lugar ao método Boosting baseado en regresión cuantil:

$$L(y, f(\boldsymbol{x})) = \tau|y - f(\boldsymbol{x})|\mathbb{I}[y > f(\boldsymbol{x})] + (1 - \tau)|y - f(\boldsymbol{x})|\mathbb{I}[y \leq f(\boldsymbol{x})].$$

Na seguinte sección detallaremos os fundamentos que dan lugar a esta función de perda.

2.3. Regresión cuantil

Analiticamente, os modelos de regresión poden expresarse como a suma dunha función de regresión máis unha compoñente de erro, isto é,

$$Y = m(\boldsymbol{X}) + \varepsilon, \varepsilon \sim N(0, \sigma^2).$$

Na regresión clásica é habitual considerar como función de regresión a media condicionada da variable resposta, é dicir,

$$m(\boldsymbol{x}) = \mathbb{E}[Y/\boldsymbol{X} = \boldsymbol{x}], \forall \boldsymbol{x} \in \boldsymbol{X}.$$

Unha das propiedades da esperanza dunha variable aleatoria é

$$\mathbb{E}[Y] = \arg\min_c \mathbb{E}(Y - c)^2.$$

Así, para a esperanza condicionada de Y,

$$\mathbb{E}[Y/\boldsymbol{X}] = \arg\min_{m(\boldsymbol{X})} \mathbb{E}(Y - m(\boldsymbol{X}))^2.$$

En regresión lineal, unha das hipóteses asumidas é que a función de regresión se pode expresar coma unha función lineal, isto é,

$$m(\boldsymbol{x}) = \mathbb{E}[Y/\boldsymbol{X} = \boldsymbol{x}] = \boldsymbol{x}^\intercal \boldsymbol{\beta},$$

onde

$$\boldsymbol{\beta} = \arg\min_{\boldsymbol{\beta}} \mathbb{E}[(Y - \boldsymbol{X}^\intercal\boldsymbol{\beta})^2].$$

Deste xeito, dado un conxunto de observacións, $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, estimaremos o modelo de regresión que representa a tendencia dos datos como

$$\hat{y} = \hat{m}(\boldsymbol{x}) = \boldsymbol{x}^\intercal\hat{\boldsymbol{\beta}},$$

onde,

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{b}} \sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^\intercal\boldsymbol{b})^2.$$

Este método, acompañado de certas suposicións adicionais como por exemplo a homocedasticidade ou a varianza constante nos termos de erro, coñécese como regresión mínimo cadrática. Sen embargo estas hipóteses a miúdo se incumpren na práctica. Como alternativa a estas limitacións xorde a regresión cuantil ou cuantílica.

A regresión cuantil foi introducida por primeira vez en 1978 polos autores Koenker e Bassett, Koenker y Bassett (1978), aínda que as ideas iniciais datan do século XVIII, cando Boskovich estaba estudando os parámetros que rexían a función elíptica da Terra. A principal vantaxe respecto da regresión cadrática é que non asume ningunha hipótese sobre os erros. A regresión cuantílica adoita presentar bos resultados naqueles conxuntos de datos heterocedásticos, cunha alta presenta de atípicos ou nos que os erros non seguen unha distribución normal. En xeral, naqueles casos nos que se incumpren as restricións impostas para o modelo de regresión cadrática. Malia a súa versatilidade en múltiples casos, o seu uso non está tan extendido como a regresión clásica, aínda que cada vez son máis as aplicacións nas que se emprega.

Na sección anterior puidemos ver que a regresión cadrática presenta unha estreita relación coa esperanza condicional. No caso da regresión cuantílica, como o seu propio nome indica, presenta unha estreita relación cos cuantis condicionais, polo que, un bo punto de partida sería comezar pola definición dos mesmos.

Definición 2.1. Sexa Y unha variable aleatoria e F_Y a súa función de distribución. Definimos o cuantil $Q_Y(\tau)$, $\tau \in (0,1)$, como

$$Q_Y(\tau) = \text{ínf}\{y \in Y : F_Y(y) \geq \tau\}.$$

Definición 2.2. Sexan $\boldsymbol{X}$, Y dúas variables aleatorias. O cuantil condicional de Y respecto de $\boldsymbol{X}$ é o cuantil $Q_Y(\tau)$ da función de distribución de Y condicionado a $\boldsymbol{X}$, isto é,

$$Q_{Y/\boldsymbol{X}}(\tau) = \text{ínf}\{y : F_{Y/\boldsymbol{X}}(y) \geq \tau\}.$$

Deste xeito, en regresión cuantil asumiremos que a función de regresión coincide cun determinado cuantil condicional, é dicir,

$$m(\boldsymbol{x}) = Q_{Y/\boldsymbol{X}=\boldsymbol{x}}(\tau),$$

e suporemos ademais que este admite unha expresión lineal,

$$Q_{Y/\boldsymbol{X}=\boldsymbol{x}}(\tau) = \boldsymbol{x}^\intercal\boldsymbol{\beta}_\tau.$$

Unha das propiedades dos cuantís é que

$$Q_Y(\tau) = \arg\min_{c} \mathbb{E}(p_\tau(Y - c)),$$

onde p_τ é a función de perda cuantílica, definida como

$$p_\tau(z) = \tau|z|\mathbb{I}[z > 0] + (1-\tau)|z|\mathbb{I}[z \leq 0].$$

Entón, para o cuantil condicionado,

$$Q_{Y/\boldsymbol{X}}(\tau) = \arg \min_{m(\boldsymbol{X})} \mathbb{E}(p_\tau(Y - m(\boldsymbol{X}))).$$

Asumindo que a función de regresión se pode expresar coma unha función lineal, temos que

$$m(\boldsymbol{x}) = Q_{Y/\boldsymbol{X}=\boldsymbol{x}}(\tau) = \boldsymbol{x}^\intercal \boldsymbol{\beta}_\tau.$$

onde,

$$\boldsymbol{\beta}_\tau = \arg \min_{\beta_\tau} \mathbb{E}(p_\tau(Y - \boldsymbol{X}^\intercal \boldsymbol{\beta}_\tau)).$$

Deste xeito, dado un conxunto de observacións, $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, estimaremos o modelo de regresión para o cuantil τ como

$$\hat{y} = \hat{m}(\boldsymbol{x}) = \boldsymbol{x}^\intercal \hat{\boldsymbol{\beta}}_{\boldsymbol{\tau}},$$

sendo,

$$\hat{\boldsymbol{\beta}}_{\boldsymbol{\tau}} = \arg \min_{\hat{\boldsymbol{\beta}}_{\boldsymbol{\tau}}} \sum_{i=1}^{n} p_\tau(y_i - \boldsymbol{x_i}^\intercal \boldsymbol{\beta}_{\boldsymbol{\tau}}).$$

Finalmente, destacar que se $\tau = 0.5$, estamos a traballar coa mediana. Para máis información sobre a regresión cuantil pode consultarse a referencia Koenker (2005)

Capítulo 3

Metodoloxía da aprendizaxe

No Capítulo 1 xa anticipamos que esta memoria se enfocaría como unha extensión do estudo realizado en Rodríguez-Ballesteros (2022), onde se empregan varios modelos de aprendizaxe supervisada para tentar determinar cal é o mellor criterio de ramificación para certo problema de programación polinómica, dentro dun conxunto predefinido. Na anterior referencia empréganse distintas variables estáticas, (*offline features*) para o adestramento dos modelos. Estas variables dependen unicamente do problema orixinal e, polo tanto, non varían ao largo dos distintos subproblemas resultantes do proceso de ramificación. Así, os modelos determinan un único criterio de ramificación óptimo, en base ás características do problema de partida, independentemente do que suceda no desenvolvemento da árbore orixinada ao aplicar o algoritmo RLT. Como xa comentamos, os resultados foron bastante alentadores, e indubidablemente mellor que unha selección aleatoria do criterio. Sen embargo, sería interesante ir un paso máis alá, e aproveitar a información asociada á resolución das relaxacións lineais para refinar os modelos de aprendizae estatística obtidos, propoñendo un criterio de ramificación para cada subproblema. Seguindo esta liña de traballo, a información global do problema, materializada e estruturada en forma de características estáticas, seguirá a ser relevante, mais tamén debemos considerar novas *features*, que nos permitan adaptar esta selección en cada subproblema, polo que, no noso conxunto de adestramento incluiremos tamén variables de carácter dinámico (*online features*).

O obxetivo principal deste traballo é investigar como abordar este novo enfoque e se é posible, presentar algún modelo de aprendizaxe estatística, que permita seleccionar o criterio de ramificación máis adecuado en cada unha das relaxacións lineais resultantes a partir da resolución dun determinado problema de programación polinómica mediante o algoritmo RLT, empregando, ademais das variables estáticas propostas en Rodríguez-Ballesteros (2022), novas variables de carácter dinámico, dependentes de cada nodo.

Comezaremos introducindo os criterios de ramificación seleccionados para o estudo e, posteriormente, describiremos en detalle o proceso de aprendizaxe, así como, a estrutura do conxunto de datos e as variables que o conforman.

3.1. Criterios de ramificación

A idea principal que fundamenta calquera regra de ramificación é a selección da variable ramificadora que orixina o maior problema de infactibilidade. A diferenza entre uns e outros criterios atópase, polo xeral, nos pesos asignados á violación das restricións para cada variable. Así, dada unha relaxación linear resultante de aplicar o algoritmo RLT a un determinado problema de optimización polinómica, ramificaremos pola variable x_p cando

$$p \in \underset{j \in N}{\arg\max} \{\theta_j\}, \text{ onde}$$

$$\theta_j = \sum_{J \subset N^\delta, |J| \leq \delta - 1} \omega(j, J) |\bar{X}_{J \cup j} - \bar{x}_j \bar{X}_J|, \forall j \in N, \tag{3.1}$$

sendo $\omega(j, J)$ os pesos asociados.

As regras de ramificación que empregaremos serán unha sinerxía dalgúns dos criterios que se recollen en Rodríguez-Ballesteros (2022) e no artigo resultante, Ghaddar et al. (2022), baseándonos na calidade dos resultados obtidos, e novas propostas, que se presentan como unha variación das anteriores, resultado da súa síntese co *Pseudocost Branching* e co *Reliability Branching*, metodoloxías de selección de variables habitualmente empregadas na resolución de problemas de programación enteira e mixta.

Comecemos introducindo aqueles criterios xa coñecidos empregados nas referencias bibliográficas que vimos de citar.

- **Criterio dos valores duais**. O valor dual asociado a unha restrición indica o cambio que se produce na función obxectivo se aumentamos unha unidade o valor do lado dereito da restricción en cuestión. Tendo en conta o anterior, dado un multiconxunto $J \subset N^\delta$ e unha variable x_j, seleccionaremos aquelas restricións que conteñen o conxunto $J \cup \{j\}$, e definiremos o peso correspondente, $\omega(j, J)$, como a suma dos valores duais, en valor absoluto, asociados ás restricións seleccionadas.

- **Criterio do rango da variable**. O rango dunha variable é a diferenza entre as súas cotas superior e inferior. Para cada variable x_j e cada multiconxunto $J \subset N^\delta$, consideraremos como peso o cociente

$$\omega(j, J) = \frac{\min\{\bar{u}_j - \bar{x}_j, \bar{x}_j - \bar{l}_j\}}{u_j - l_j}.$$

  Deste xeito, relacionando a diferencia das cotas inferior e superior da variable, $\bar{l}_j, \bar{u}_j$, respecto do valor da mesma no óptimo do nodo actual, $\bar{x}_j$, coa diferenza entre as cotas no nodo inicial, l_j, u_j, asignaremos maior prioridade ás variables cuxo rango se reducira en menor proporción a medida que o algoritmo avanza.

O último dos criterios que tomaremos da memoria bibliográfica mencionada anteriormente merece unha explicación á parte por ser concibido especificamente no eido dos grafos, estruturas que sempre estiveron estreitamente relacionadas cos problemas de optimización. De feito, nós mesmos neste traballo nos temos referido ás distintas relaxacións lineais resultantes da ramificación como nodos en multitude de ocasións. Así, dado un problema de optimización polinómico, o grafo de intersección monomio-restrición, *constraints-monomials intersection graph*, (CMIG), presenta un nodo asociado á función obxectivo, a cada monomio e a cada unha das restricións que hai no problema, definindo unha arista entre un nodo e unha restricción ou a función obxectivo cando certo monomio apareza na mesma. Outro grafo interesante, que non empregaremos para definir novos criterios, pero cuxas características poden resultar de utilidade para a aprendizaxe é o grafo de intersección variable-variable, *variables intersection graph*, (VIG), que presenta tantos nodos coma variables ten o problema, definindo unha arista cando ambas pertecen ao mesmo monomio. Como veremos máis adiante, algunhas das variables estáticas empregadas para o entrenamento dos modelos foron definidas en base a este tipo de grafos.

De cara a definir o criterio en cuestión, introduciremos unha das propiedades máis coñecidas e empregadas na Teoría de Grafos: a centralidade. A *centralidade de vector propio*, *eigencentrality* ou *eigenvector centrality*, é unha medida que trata de cuantificar a importancia dun nodo dentro do grafo, tendo en conta as conexións existentes entre os nodos. Analíticamente, definiremos a centralidade como o autovector asociado ao autovalor de maior valor da matriz de adxacencia asociada ao grafo, popularmente coñecido como *first eigenvalue* (Teorema Perron-Frobenius, Godsil et al. (2001)). Deste xeito, cada unha das compoñentes do autovector representará a centralidade de vector propio de cada nodo. Para coñecer máis acerca desta medida pode consultarse Latora et al. (2017).

Tendo en conta o anterior, baseado no grafo CMIG, e en combinación co concepto de centralidade de vector propio, podemos definir un novo criterio de ramificación, onde o peso asociado a cada variable

x_j se corresponde coa centralidade de vector propio do nodo asociado ao monomio x_j. No caso de que dito monomio non exista, asignaráselle o valor cero.

Até este punto, fixemos unha breve recopilación dalgunhas das regras de ramificación que mellor resultado presentaron en Rodríguez-Ballesteros (2022). Todas elas manteñen un nexo común: asignan pesos ás violacións das restricións para cada unha das variables do problema. Como novidade, nesta memoria introduciremos novos criterios cun enfoque lixeiramente distinto. Esta nova perspectiva consiste na adaptación de certa regra de ramificación, popularmente empregada en problemas de programación enteira e mixta (MILP), máis concretamente, o *Pseudocost Branching*, aproveitando que estamos a traballar con problemas con variables continuas. En Belotti et al. (2009) atopamos un exemplo deste tipo de práctica. No que segue, introduciremos brevemente esta metodoloxía.

3.1.1. Pseudocost Branching

O *Pseudocost Branching* é unha regra de ramificación que asigna en cada nodo un pseudocoste a cada unha das variables, en función das veces que se ramificou pola variable en cuestión e como de bo foi o resultado obtido, empregando a información histórica das variables ramificadoras dos nodos anteriores.

Dado un nodo Q e unha variable x_j, $j \in \{1, \dots, n\}$, se ramificamos pola mesma obteremos dous novos nodos Q_j^- e Q_j^+, asociados aos subproblemas resultantes de engadir ao problema inicial, nodo Q, as restricións $x_j \leq \lfloor x_j \rfloor$ e $x_j \geq \lceil x_j \rceil$, respectivamente. Sexa $\bar{x}_j$ a solución óptima da relaxación linear asociada ao nodo Q para a variable x_j e $\bar{c}_Q$ o valor da función obxectivo. Entón, definimos ς_j^- e ς_j^+ como

$$\varsigma_j^+ = \frac{\Delta_j^+}{f_j^+} = \frac{\bar{c}_{Q_j^+} - \bar{c}_Q}{\lceil \bar{x}_j \rceil - \bar{x}_j}, \quad \varsigma_j^- = \frac{\Delta_j^-}{f_j^-} = \frac{\bar{c}_{Q_j^-} - \bar{c}_Q}{\bar{x}_j - \lfloor \bar{x}_j \rfloor},$$

onde $\bar{c}_{Q_j^-}$ e $\bar{c}_{Q_j^+}$ denotan o valor da función obxectivo, η_j^+ o número de nodos anteriores ao nodo actual para os que a variable j resultou ser a variable ramificadora e para os que se obtivo unha solución factible ao resolver o subproblema subxacente e σ_j^+ a suma das correspondentes ς_j^+ o ς_j^-. Analogamente, definimos η_j^- e ς_j^- [1]. Así, o pseudocoste asignado no nodo Q_j^+ á variable $j-$ésima é:

$$\Psi_j^+ = \sigma_j^+ / \eta_j^+.$$

Equivalentemente, para o nodo Q_j^-,

$$\Psi_j^- = \sigma_j^- / \eta_j^-.$$

Deste xeito, para determinar a variable de ramificación debemos calcular a *score* [2] asociada a cada unha das variables candidatas, isto é,

$$s_j = score(f_j^- \Psi_j^-, f_j^+ \Psi_j^+),$$

onde $f_j^+ = \lceil \bar{x}_j \rceil - \bar{x}_j$, $f_j^- = \bar{x}_j - \lfloor \bar{x}_j \rfloor$, e seleccionar aquela con maior s_j.

Un problema que podemos percibir facilmente é que, nas primeiras etapas do algoritmo, existirá un elevado número de variables sen ningún pseudocoste asociado. Nestes casos, o máis habitual é asignarlle como pseudocoste a esas variables o valor medio dos pseudocostes das variables que sí teñen un pseudocoste asignado. Outras metodoloxías, como por exemplo o *Reliability Branching*, empregan *Strong Branching* nas primeiras iteracións de cara a solventar este problema, aínda que, por suposto, cun custo computacional máis elevado, especialmente a segunda. Pode consultarse máis información acerca de ambas en Achterberg et al. (2005).

[1] Observemos que, na primeira iteración $\sigma_j^+ = \sigma_j^- = \varsigma_j^+ = \varsigma_j^- = 0$.

[2] A *función score* defínese como $score(a, b) = (1 - \mu)\,\text{mín}\{a, b\} + \mu\,\text{máx}\{a, b\}$, onde o *score factor*, μ, é un parámetro que toma valores comprendidos entre 0 e 1 que adoita determinarse de xeito empírico.

Chegados a este punto e tendo en conta o anterior, é razoable que nos preguntemos a partir de cantas ramificacións os pseudocostes asociados ás variables son o suficientemente fiables. Con esta idea en mente, presentamos unha variación do método anterior que será a que empregaremos como base dos novos criterios de ramificación neste traballo. De xeito similar ao *Reliability Branching*, definiremos un parámetro, η_{rel}, que representará o número de ramificacións necesarias para garantir a fiabilidade dos pseudocostes na implementación realizada do método.

Outro aspecto a ter en conta, é que tal como describimos a metodoloxía, os pseudocostes actualízanse de forma indefinida até resolver o problema en cuestión, mais tamén pode ser de interese considerar outro parámetro que determine o número máximo de ramificacións para deter a actualización dos pseudocostes. Ben é certo que coa actualización sen límitacións obteremos máis información, pero cando o número de nodos sexa bastante elevado, corremos o risco de que esta mesma información sexa demasiado local, provocando un sobreaxuste nos modelos considerados.

A introdución do *Pseudocost Branching* que vimos de facer, está definida para para problemas de programación enteira e mixta. A continuación, presentaremos a adaptación proposta para traballar con problemas de optimización polinómica, campo que nos compete nesta memoria. Así, para calquera variable x_j, $j \in \{1, \dots, n\}$, definimos o pseudocoste asociado a un determinado nodo Q, como

$$\Psi_{j,J} = \frac{1}{k_j} \sum_{i=1}^{k_j} \frac{\Delta_j^i}{f_{j,J}},$$

sendo k_j o número de nodos anteriores ao nodo actual Q para os cales a variable ramificadora resultou ser x_j e Δ_j^i a diferenza dos valores da función obxectivos para os óptimos no nodo $i-$ésimo e actual, é dicir, $\Delta_j^i = \bar{c}_{Q_i} - \bar{c}_Q$. A clave das variacións propostas como criterio atópase na definición de $f_{j,J}$, definidas como as propias violacións RLT nos criterios do rango, dual e eigencentralidade CMIG, respectivamente, é dicir, o valor de θ_j para cada un dos criterios:

$$f_{j,J} = \omega(j, J)|\bar{X}_{J\cup j} - \bar{x}_j \bar{X}_J|, \forall j \in N.$$

Deste xeito, seleccionaremos a variable x_p cando

$$p \in \arg\max_{j \in N} \{\theta_j\}, \text{ onde}$$

$$\theta_j = \sum_{J \subset N^\delta, |J| \leq \delta - 1} \omega(j, J)\Psi_{j,J}|\bar{X}_{J\cup j} - \bar{x}_j \bar{X}_J|, \forall j \in N,$$

sendo $\omega(j, J)$ os pesos asociados ao criterio dual, dos rangos ou de centralidade descritos anteriormente.

Unha vez temos definidos os criterios de ramificación candidatos, o seguinte paso é describir o conxunto de datos que empregaremos para os cálculos, dende o procedemento para a súa xeración ata a definición da variable resposta e a selección das variables explicativas.

3.2. Descrición do conxunto de datos

A orixe dos problemas empregados para a construción do conxunto de datos será a mesma que en Rodríguez-Ballesteros (2022), recopilando información de tres baterías distintas:

- Dalkiran and Sherali (2016), libraría `EVRIM`: 180 problemas de optimización polinómica xerados aleatoriamente con distinto grao, número de variables e densidade.
- Bussieck et al. (2003), libraría `MINLP`: 168 problemas de programación non lineal enteira mixta, con variables acotadas e continuas, condición imposta sobre os problemas de programación polinómicos introducidos para garantir a converxencia do algoritmo RLT, Definición (1.5).
- Furini et al. (2017), libraría `QP`: 63 problemas de optimización cuadrática.

Así, obtemos un conxunto formado por 411 problemas de optimización, do que eliminaremos aqueles problemas que se resolven nunha soa iteración, porque non aportan á aprendizaxe, ou que non presentan restricións. Deste xeito, o noso estudo focalizarase unicamente sobre 354 problemas.

Ademais de clasificar as instancias por batería, considerarase unha clasificación adicional por familia. Para `EVRIM`, definiranse as familias `d2`, `d3`, `d4`, `d5`, `d6` e `d7`, onde a cifra indica o grao do problema. No relativo a `MINLP`, temos `ex`, engloba un conxunto de problemas empregados como exemplo en Floudas et. al. (2000), `ka`, con problemas xeométricos sobre seccións circulares, `po`, problemas sobre a mistura de produtos petroquímicos, `st`, distintos exemplos de Tawarmalani and Sahinidis (2002), `wa`, problemas orientados á minimización do malgasto de auga, e `other`, que engloba problemas de diferente tipoloxía, cuxas respectivas familias non acadan un número mínimo de 10 problemas. Pode consultarse máis información sobre estas instancias no seguinte enlace. Os problemas cadráticos formarán unha familia por sí mesmos, `qp`. Para maior detalle, pode consultarse a seguinte ligazón.

Nas seguintes subseccións describiremos en detalle os pasos executados para a obtención do conxunto de datos final, comezando polas especificacións técnicas consideradas para obter as observacións que conforman o dataset, e seguindo coa definición da variable resposta e a selección das variables explicativas, así como, todo o proceso de depuración que se levou a cabo.

3.2.1. Proceso de xeración

`RAPOSa`, *Reformulation Algorithm for Polynomial Optimization - Santiago*, é un solver de carácter global focalizado na resolución de problemas de programación polinómica, continuos e con rexión factible limitada, aplicando o algoritmo RLT. Dito optimizador foi implementado en `C++` por un grupo de investigadores da Universidade de Santiago de Compostela e do ITMATI, *Technological Institute for Industrial Mathematics*. Á súa vez, `RAPOSa` apóiase noutros solvers, que se deixan á elección do usuario, que colaboran para a resolución das distintas relaxacións lineais orixinadas, por exemplo, Gurobi, e tamén optimizadores locais de problemas non lineais, como Knitro, que proporcionarán unha cota superior para inicializar o algoritmo. Para maior detalle pode consultarse a referencia González-Rodríguez et. al. (2020).[3]

Os problemas de optimización polinómica presentados na sección foron resoltos empregando o solver `RAPOSa` e as respectivas execucións foron levadas a cabo no superordenador FinisTerrae II do Centro de Supercomputación de Galicia, CESGA, cun tempo máximo de execución de 10 minutos.[4] A partir da información xerada nestas execucións constituiremos o conxunto de datos. Cada un dos nodos asociados ás distintas relaxacións lineais representará unha observación no mesmo, coas correspondentes valores para as variables explicativas e a variable resposta. Así mesmo, para todos eles calcularase a variable ramificadora empregando cada un dos criterios introducidos na sección anterior, e avaliarase o seu rendemento en base á mellora das cotas inferiores, utilizando posteriormente esta información para definir a variable resposta.

Ademais de gardar a información resultante da aplicación dos criterios, para continuar coa execución do algoritmo e non deternos nun único nodo, debemos seleccionar un deles, é dicir, determinar cal será a variable escollida como ramificadora para continuar. Idealmente, sería desexable ramificar empregando cada unha das variables que se obtiveron coas distintas regras, de tal maneira que se reflexe como actúa cada criterio en cada situación e a forma que adquire a árbore xerada, pero isto, computacionalmente, é intratable. Entón, como escollemos o criterio ramificador que determinará a nova relaxación linear, e en consecuencia, a nova observación? Posto que estamos tratando de seleccionar aquela regra que presente un mellor comportamento en cada nodo, unha posible opción sería quedar sempre con aquel criterio que presenta o mellor desempeño en termos de mellora nas cotas inferiores. Proceder deste xeito pode introducir certo sesgo nos resultados. Polo tanto, as opcións baralladas neste traballo como alternativas, serán, por unha banda, seleccionar de modo aleatorio o criterio de ramificación, de xeito que, a modo xeral, o conxunto de datos conteña información de multitude de problemas en diferentes

[3] Tamén pode consultarse o seguinte enlace https://raposa.usc.es/index.html

[4] Para máis información sobre o CESGA pode consultarse a páxina web do centro: https://www.cesga.es/.

estados de resolución, fixando unha semente para garantir a reproducibilidade dos resultados obtidos e, por outra, seguir sempre un mesmo criterio dos introducidos na sección anterior.

Deste xeito, cada un dos 354 problemas, será resolto oito veces, seguindo dúas ordes aleatorias que combinan a totalidade das regras de ramificación consideradas, e seis máis, seguindo sempre un mesmo criterio prefixado. Así, o noso conxunto de datos terá unha observación para cada nodo de cada problema resolto con cada unha das oito ordes consideradas. Por exemplo, dado un problema, se supoñemos que con cada unha das ordes se orixinan 15 nodos (non ten por que ser o mesmo número), o número de observacións correspondentes ascenderá a $8 \cdot 15 = 120$. Esta información foi almacenada por `RAPOSa` en distintos arquivos con formato `json`, cada un deles asociado a un problema e a unha orde concreta. Deste xeito, para cada problema teremos 8 arquivos `json` asociados, é dicir, un total de 2832.

A transformación desta información nun formato adecuado para o seu manexo en R foi levada a cabo coa libraría `RJSONIO`. Malia que pode parecer sinxelo, este procedemento deu lugar a un conxunto de datos de gran tamaño, con $12,984,303$ millóns de observacións, 350 problemas e 83 variables iniciais, cuxa manipulación resultou moi complicada debido á elevada volumetría dos datos e á insuficiencia da memoria RAM do ordenador do alumno. Por tal motivo, o conxunto problemas tivo que ser partillado en 3 máis pequenos, cada un deles asociado a unha libraría. Para cada un, foron converténdose os arquivos `json` correspondentes en observacións, que á súa vez, foron almacenadas en distintos *dataframes*. Os tres *dataframes* resultantes foron posteriormente unificados, empregando un nodo de MARISCAL, clúster da Facultade de Matemáticas da USC. O conxunto xerado foi gardado como un *datatable*, que en principio é un formato máis adecuado que o *dataframe* para traballar con conxuntos de gran tamaño.

3.2.2. Variables explicativas

As variables explicativas que empregaremos serán unha mestura de variables de carácter estático, empregadas en Rodríguez-Ballesteros (2022), e dinámico.

As variables estáticas (31) proporcionan información referente ao problema de optimización de partida, sen focalizarse nas distintas relaxacións lineais e mantendo inmutable os seus valores ao largo das distintas iteracións do algoritmo para un mesmo problema. Este xeito de proceder, que nunha primeira instancia resultara adecuado na mencionada referencia bibliográfica, agora quédase curto, dado que, desta quenda, o obxectivo é determinar aquela regra que mellor acae á hora de ramificar para cada unha das relaxacións lineais.

Dada esta modificación no nivel de granularidade, para adestrar os modelos de forma adecuada debemos recoller, ademais de información xeral do problema de partida, variables específicas sobre as características dos subproblemas asociados a cada nodo, ás que nos referiremos, como xa anticipamos, como variables dinámicas (52).

Sobre a selección das variables volveremos na subsección 3.2.4.

3.2.3. Variable resposta

Vimos de introducir as variables explicativas coas que adestraremos os modelos para tratar de predicir cal é o criterio de ramificación que mellor actúa en cada nodo. Sen embargo, falta outro ingrediente indispensable no conxunto de datos: a variable resposta ou variable dependente.

Un procedemento habitual para analizar o rendemento dun criterio no ámbito da optimización matemática, é o uso de KPIs (*Key Performance Indicators*). Algúns exemplos son o *gap óptimo*, o *número de iteracións*, o *tempo de execución* ou a *profundidade* da árbore asociada ao problema. No noso caso definiremos unha medida propia a partir das reducións das cotas inferiores.

Dado un nodo i orixinado a partir do algoritmo RLT, o valor da función obxectivo no óptimo proporciona unha cota inferior para a solución do problema orixinal, e denotarémola por LB_i. Para cada criterio de ramificación k, $k \in \{1, \ldots, 6\}$, obtemos unha variable ramificadora que orixina dous novos subproblemas, que á súa vez proporcionan novas cotas inferiores, maiores ou iguais cás do nodo

i posto que a rexión factible se reduciu. Así mesmo, denotarémolas por $LB_{i_1}^k$ e $LB_{i_2}^k$, respectivamente. Deste xeito, para cada criterio de ramificación k, definimos como medida do desempeño do criterio de ramificación:

$$\text{KPI}_k = \text{mín}\{LB_{i_1}^k - LB_i, LB_{i_2}^k - LB_i\}, \quad k \in \{1, \ldots, 6\}.$$

A partir deste KPI_k, definiremos unha variable resposta multidimensional $\mathbf{Y} = (Y_1, \ldots, Y_6)$, onde, para cada $k \in \{1, \ldots, 6\}$,

$$Y_k = \frac{\text{KPI}_k}{\text{KPI}_{best}}, \tag{3.2}$$

sendo KPI_{best} o KPI_k asociado ao criterio de mellor desempeño, isto é,

$$\text{KPI}_{best} = \max_{k \in \{1,\ldots,6\}} \text{KPI}_k.$$

Ademais, se KPI_{best} é nulo significa que non se produce melloría algunha ao ramificar con ningún dos criterios, polo tanto, neste tipo de situación eliminaremos dita observación do conxunto de datos. Observemos que, para cada $k \in \{1, \ldots, 6\}$, $Y_k \in [0, 1]$. Así, canto máis próximas a 1 se atopen as compoñentes de $\mathbf{Y}$, mellor será o desempeño do criterio asociado ás mesmas, o que nos proporciona como vantaxe adicional a facilidade na interpretación dos resultados obtidos cos modelos, tal e como veremos máis adiante.

3.2.4. Depuración do conxunto de datos e selección de variables

Do conxunto de datos inicial foi necesario suprimir aquelas observacións nas que todas as medidas de rendemento resultaron nulas, 4983471, é dicir, nodos nos que ningún criterio resultou óptimo, feito que ocasionaría certo problema na definición da variable resposta, como vimos de comentar, (3.2). Así mesmo, tamén descartaremos aqueles subproblemas que presenten duplicados, concretamente 12380, casuísticas nas que todas as regras de ramificación presentan un rendemento idéntico e as variables explicativas toman os mesmos valores. A maiores, eliminaremos as observacións relativas a problemas dos que descoñecemos o valor das variables explicativas de carácter estático, 131733. Estas consideracións inducen unha redución no conxunto de datos, composto finalmente por 7856719 millóns de observacións, 328 problemas e 83 variables, 52 dinámicas e 31 estáticas.

Tal e como comentamos no Capítulo 2, no noso caso, adestraremos modelos boosting baseados en regresión cuantil. Os modelos de regresión vense fortemente afectados polo fenómeno de colinealidade dificultando a estimación dos coeficientes do modelo, que presentan altas varianzas, e onde pequenas modificacións no conxunto de datos pode dar lugar a un modelo totalmente distinto. Unido a isto, tamén temos un elevado número de variables explicativas que é desexable reducir. Polo tanto, de cara a solventar ambas problemáticas tomouse a decisión de realizar unha selección de variables en base a un estudo das correlacións lineais existentes entre as mesmas. Sen embargo, novos problemas de memoria fixeron necesaria a extracción dunha mostra representativa do conxunto composta por 550062 observacións e o mesmo número de variables. Dita mostra foi seleccionada preservando as proporcións de cada batería e cada familia, de tal maneira que se conservase a distribución do conxunto orixinal depurado. No Capítulo 4 realizaremos unha análise descriptiva de ambos conxuntos, estudando a súa distribución e volumetría, así como o comportamento da variable resposta nos mesmos, o que xustificará gráfica e analiticamente a selección da mesma.

A partir da mostra, realizouse a correspondente selección de variables en base ao estudo da correlación existente entre as mesmas, dous a dous, despois da supresión de variables con variabilidade case nula, que non aportan no proceso de aprendizaxe por presentar valores case constantes. Servíndonos da función `findCorrelation` da libraría `caret`, seleccionamos aqueles pares de variables con correlación maior que 0.75 e, para cada un deles, eliminamos aquela variable que presenta, en media, unha maior correlación co resto de variables. Como resultado deste procedemento, o dataset considerado pasará a estar formado por 40 variables, 14 estáticas e 26 dinámicas. Pode consultarse un listado das mesmas no Apéndice A.

3.2.5. Transformación das variables

No noso caso, todas as variables preditoras son de tipo numérico e polo tanto, a escala na que se miden así como a súa varianza poden influír en gran medida no modelo proposto, especialmente cando coexisten variables con diferencias moi destacables en escala. Aquelas variables que presenten unha escala maior, independentemente do grao de relación que presenten coa variable resposta, presentarán certa predisposición a ter maior peso no modelo fronte ás que presentan unha escala máis reducida. Co obxectivo de paliar os problemas que a diferenza de escala poida supoñer, empregaremos unha transformación nas variables aplicando logaritmos.

A transformación logarítmica é unha transformación que pertence á familia das transformacións BoxCox, onde o parámetro λ determina a transformación escollida. En esencia, as transformacións BoxCox xurdiron co obxectivo de modificar a distribución dun conxunto de datos que presenta certa asimetría, maneira que se asemelle máis ao comportamento dunha normal. Analiticamente, esta familia de transformacións pode expresarse como

$$t_\lambda(y) = \begin{cases} \frac{y^\lambda - 1}{\lambda}, & \text{se} \quad \lambda \neq 0, \\ \\ \ln(y), & \text{se} \quad \lambda = 0, \end{cases}$$

onde para todo $y > 0$. Cada valor de λ produce unha transformación distinta, co que debemos ter coidado á hora de seleccionar o parámetro máis conveniente. Observemos un punto importante, ditas transformacións aplícanse sobre variables estritamente positivas. No noso caso isto non supón ningún problema, pero de ser necesario, deberíamos sumar unha pequena cantidade para garantir que a variable en cuestión verifica esta restricción.

No noso caso, aplicaremos unha transformación logarítmica a todas as variables como paso previo ao adestramento dos modelos. Para garantir que todas elas acadan valores estritamente positivos sumarémoslle a cantidade 10^{-6} ás que sexa necesario. Deste xeito, aquelas variables con valores moi pequenos, verán aumentada a súa escala, e obterase o efecto contrario coas que tomen valores elevados, reducindo as elevadas diferenzas existentes. Ademais, no noso conxunto de datos existe un alto número de atípicos, algo que podemos apreciar nos múltiples boxplots do seguinte capítulo, co que se desvirtúa aínda máis a situación. Mediante esta transformación podemos ver que tamén se mingua o seu efecto, corrixindo a elevada variabilidade das variables.

Capítulo 4

Resultados prácticos

Tras a depuración que vimos de describir no capítulo anterior, o número inicial de problemas considerados reduciuse de 354 (56 da libraría `QPLib`, 127 da libraría `MINLPLib` e 171 de `EVRIM`) a 328, onde 47 pertencen a `QPLib`, 111 a `MINLPLib` e 170 foron tomados de `EVRIM`. Tras aplicar o algoritmo RLT a cada un deles e analizar o número de subproblemas xerados, observamos que este varía moito dependendo do problema considerado e da batería á que pertenza o mesmo. O gráfico de barras que se mostra a continuación, Figura 4.1, representa a distribución do número de nodos para cada problema. Apréciase claramente que aqueles asociados á libraría `MINLPLib` son tamén os que teñen, con moita diferenza, un maior número de nodos, representando un total $6,508,264$ millóns de observacións no conxunto total, aproximadamente un $82.84\,\%$ dos datos, fronte a $694,498$ de `QPLib`, un $8.84\,\%$ e $653,957$ de `EVRIM`, cun $8.32\,\%$.

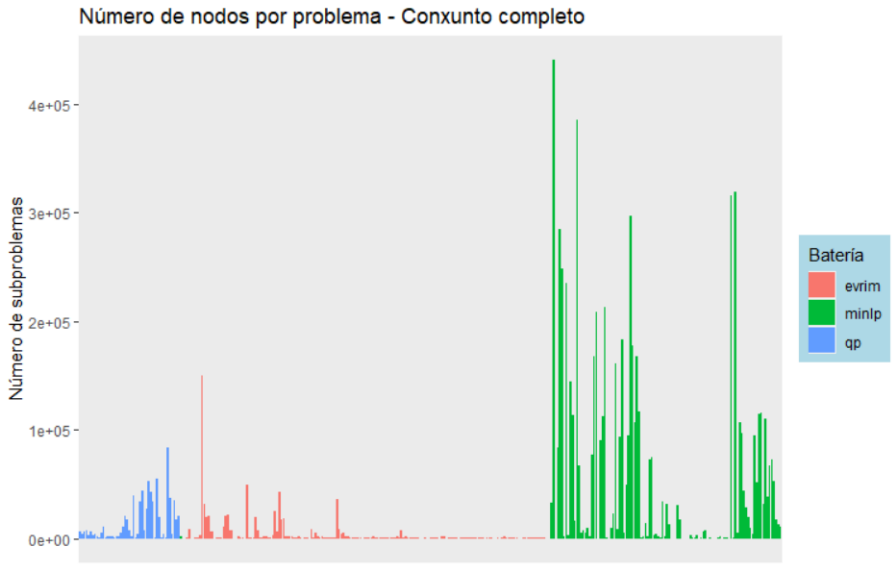


Figura 4.1: Distribución do número de observacións por problema.

Á vista do anterior, é razoable preguntarse como se distribúen estas porcentaxes en cada unha das familias, especialmente, naquelas asociadas á batería `MINLPLib`. É un reparto equitativo, ou, pola contra, novamente unhas subfamilias destacan sobre as outras? A Figura 4.2 confirma claramente esta segunda hipótese, sendo `ex`, `ka` e `wa` as que orixinan a meirande parte da volumetría de datos da batería `MINLPLib`, ás que lles segue, aínda que ao lonxe, `QPLib`.

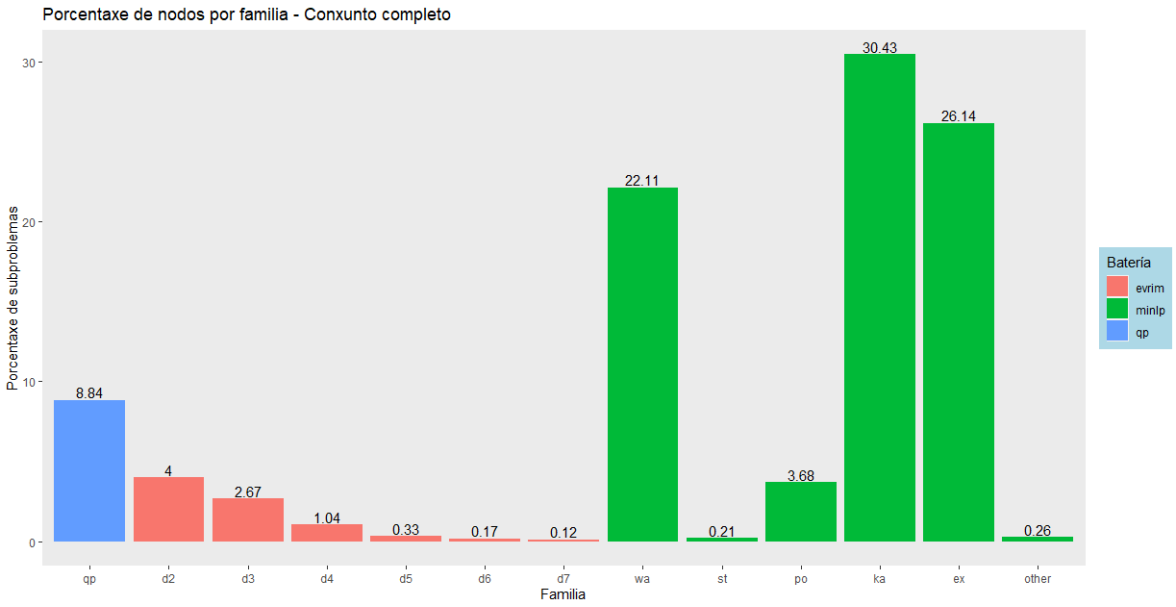


Figura 4.2: Porcentaxe de observacións por familia no conxunto orixinal.

Os resultados deste estudo analítico son un claro indicativo de que á hora de tomar as mostras de train e test debemos ter especial coidado para manter as proporcións por familias, de tal maneira que teñamos una visión representativa do conxunto de datos co que traballamos, aspecto importante en calquera proceso de adestramento de modelos. Doutro xeito, correríamos o risco de que a meirande parte das observacións consideradas en ambos conxuntos pertencesen a estas librarías e introduciríase un sesgo na aprendizaxe.

Outro aspecto fundamental que debemos abordar previo ao desenvolvemento dos modelos, é a análise do comportamento da variable resposta no conxunto de datos inicial, definida anteriormente en (3.2). É de esperar que observemos certa variabilidade entre os distintos criterios de ramificación. Noutro caso, significaría que é irrelevante seleccionar un ou outro, posto que todos devolverán resultados semellantes, e a busca de modelos de aprendizaxe estatística para a selección dunha regra apropiada resultaría innecesaria. Así, agárdase que certos criterios resulten adecuados para algúns subproblemas pero para outros moitos non, facendo necesaria unha discriminación de en que casos aplicar uns ou outros en base ás características das propias instancias, motivando deste xeito o estudo que estamos a desenvolver.

Co obxectivo de ratificar que variable de interese verifica as anteriores suposicións, representaremos a distribución da mesma para cada un dos criterios de ramificación empregando diagramas de caixas e bigotes a distintos niveis: considerando inicialmente o conxunto completo, posteriormente, dividíndoo por baterías e finalmente, partillándoo por familias. Isto axudaranos a interpretar os datos dunha forma visual e sinxela.

Tal e como podemos ver na Figura 4.3, ningún criterio parece destacar en desempeño respecto do resto. Polo xeral, o conxunto de datos presenta bastante variabilidade, que é, en esencia, o que esperábamos. Observamos, por exemplo, que existen numerosos problemas para os que o criterio dual, en azul, resulta óptimo ou que ten un desempeño próximo ao do criterio óptimo, cun valor do KPI case unitario, pero para outros moitos os resultados son realmente malos en comparación coa regra óptima, tomando valores inferiores a 0.25. Estes últimos casos son os que nos interesan especialmente. O obxectivo que perseguimos é ser quen de propoñer un modelo que discrimine dos seis criterios, aquel que presenta o mellor desempeño, ou, no seu defecto, algún cun rendemento próximo a este, tentando mellorar os tempos de execución do algoritmo RLT, e consecuentemente, o número de iteracións necesarias do mesmo, se considerásemos cunha regra fixa.

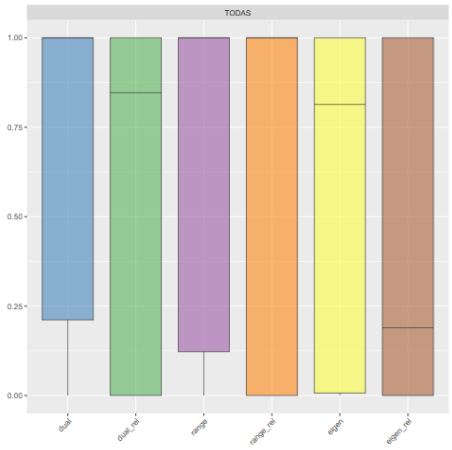


Figura 4.3: Distribución da variable resposta para cada criterio no conxunto orixinal.

Vexamos agora que ocorre se analizamos esta mesma información para cada batería.

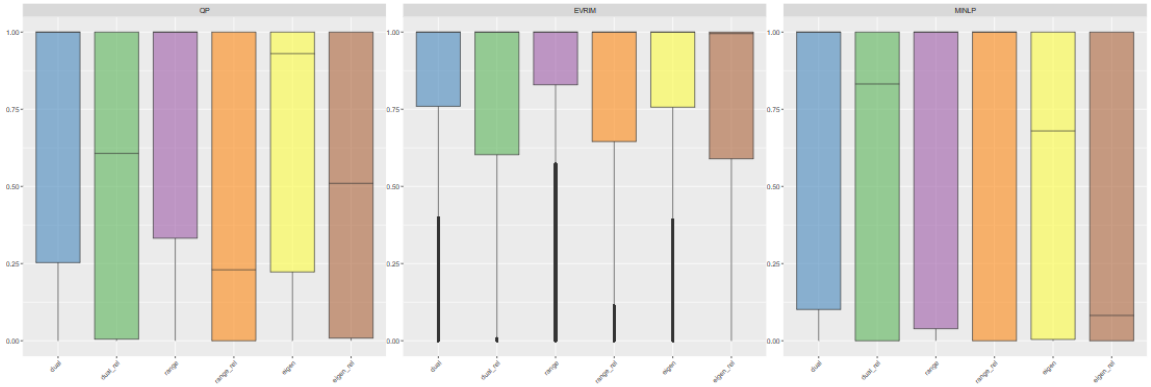


Figura 4.4: Distribución da variable resposta para cada criterio no conxunto orixinal por batería.

Primeiramente, non é de estrañar, tendo en conta a volumetría da libraría `MINLPLib`, que o seu comportamento sexa semellante ao do conxunto total, observándose unha ampla variabilidade, aspecto que comparte á súa vez con `QPLib`. Así, podemos inferir que existen subproblemas para os que cada regra presenta bos resultados de xeito individualizado, pero outros moitos nos o seu desempeño deixa bastante que desexar en comparación co criterio de óptimo. Pola contra, a batería `EVRIM` presenta un comportamento peculiar. Polo xeral, parece que calquera que sexa o criterio de ramificación considerado presenta bo desempeño, aínda que cun bo número de excepcións que se reflexan na alta presenza de *outliers*. Isto quere dicir, que aínda que o criterio seleccionado para ramificar non sexa o óptimo, calquera dos outros representará tamén unha boa aproximación. En calquera caso, ningunha regra destaca sobre as demais de xeito común a todas as baterías, xa que, malia que o criterio do rango semella presentar bos resultados para un bo número de problemas de `EVRIM`, para `MINLP` non sería unha gran elección nun alto número de casos.

De seguido analizaremos como se comporta a variable resposta por familia. Cabe destacar que a libraría `QPLib` representa unha familia por sí mesma, polo tanto, os gráficos asociados, así como as conclusións, son idénticos aos anteriores. Por tal motivo, e para tentar reducir os elevados tempos computacionais na xeración dos gráficos co conxunto de datos inicial, que presenta unha alta volumetría, unicamente analizaremos as familias asociadas a `MINLPLib` e `EVRIM`.

Na Figura 4.5, observamos que o comportamento de `ka` e `wa`, dúas das familias con maior número de subproblemas da batería `MINLPLib`, é semellante ao da mesma, presentando unha alta variabilidade. Ademais, para `ex` e `st`, aparentemente, o criterio dual é o que mellor actúa, aínda que con numerosas excepcións, reflectidas na presenza de múltiples atípicos. Por outra parte, chaman a atención os problemas das familias `d6` e `d7`. De xeito xeralizado, aínda que existe un alto número de *outliers*, son instancias para as que a maioría dos criterios presentan empates en optimalidade. Por exemplo, dos 13009 subproblemas pertencentes a `d6`, só 379 teñen un único criterio óptimo, e 10769, presentan tres ou máis empates, o que explica a situación reflectida na gráfica. Sobre este tema volveremos máis adiante cando analicemos os problemas encontrados durante o proceso de aprendizaxe. A todo o anterior, tamén se suma que, polo xeral, os criterios baseados en pseudocostes, en laranxa, amarelo e marrón, adoitan ter un comportamento peor que a súa versión en bruto.

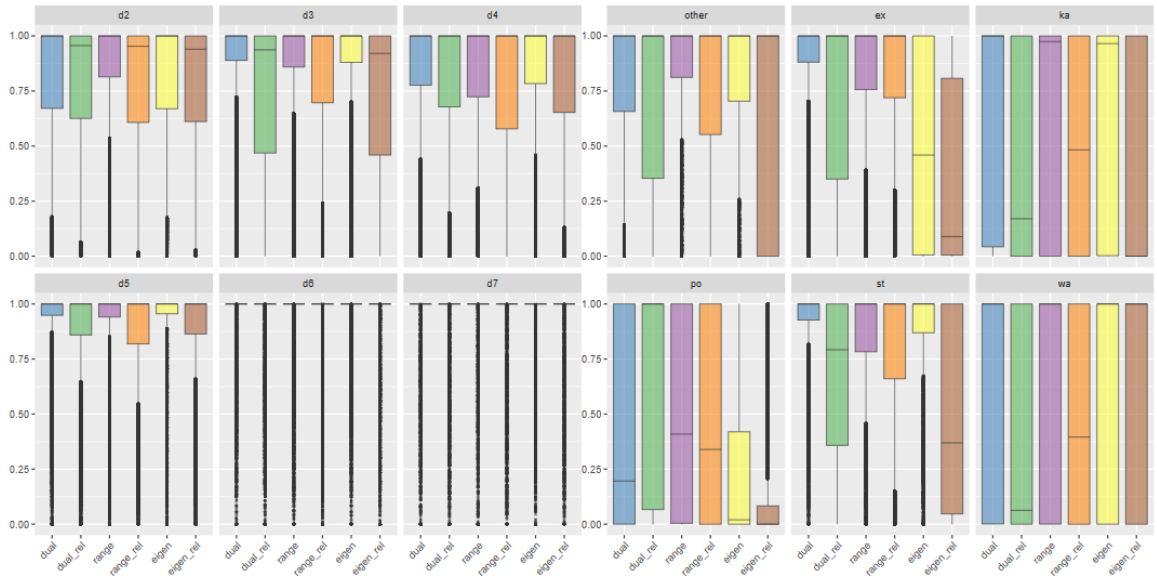


Figura 4.5: Distribución da variable resposta para cada criterio no conxunto orixinal por familia.

Unha conclusión que é evidente á vista do anterior é que non parece existir un criterio óptimo, común para todas as familias. Ademais, a presenza dun alto número de atípicos, así como, de certos boxplots con asimetría negativa, motiva aínda máis necesidade dun modelo preditivo que axude a determinar se o criterio que maioritariamente actúa mellor dentro dunha mesma familia, é realmente a mellor elección, e en caso negativo, que sexa quen de propoñer un cun desempeño aceptable.

Chegados a este punto, no que temos perfilado o comportamento do conxunto de datos que estamos a manexar, podemos proceder co proceso de aprendizaxe, descrito máis en detalle na seguinte sección.

4.1. Proceso de aprendizaxe

En aprendizaxe estatística, o procedemento habitual para a modelización dun conxunto de datos consiste en dividir dito conxunto en dous: a mostra de adestramento, habitualmente formada por un 70 % ou 75 % das observacións totais, e a mostra de test, coa porcentaxe restante. Nalgúns casos considérase unha terceira mostra, coñecida como mostra de validación.

A mostra de adestramento emprégase para a construción dos modelos e a de test, para avaliar o rendemento dos mesmos, e proporcionar estimacións do erro cometido ao aplicar os modelos sobre novas observacións. En que punto entra entón a mostra de validación? Pois ben, os modelos de aprendizaxe

estatística gozan de hiperparámetros que establecen limitacións durante o proceso de aprendizaxe, e, en moitos casos, axudan tamén a controlar a complexidade do modelo, aspecto que implica un alto risco de sobreaxuste. A mostra de test non debe empregarse para seleccionar os valores máis adecuados destes parámetros, soamente sobre o modelo final, xa que do contrario, estariamos axustando os seus valores para que o modelo "funcionase" adecuadamente en test. Recordemos que este non é o obxectivo, senón a súa aplicación sobre novas observacións non involucradas no proceso de aprendizaxe. Polo tanto, a mostra de validación será empregada para validar o rendemento dos distintos modelos adestrados coa mostra de *train*, permitíndonos analizar unha estimación do erro cometido e posibilitando a selección dun mellor modelo modificando os seus parámetros, que posteriormente será testado coa mostra de test.

Con todo, o máis habitual é traballar só coas dúas mostras de adestramento e test, aplicando o método de validación cruzada para a selección dos hiperparámetros óptimos, e será esta a aproximación que seguiremos para obter os modelos presentados ao longo deste traballo.

4.1.1. Validación cruzada

A validación cruzada, *cross-validation*, é unha técnica amplamente empregada para avaliar os modelos xerados en certos proxectos de aprendizaxe estatística, proporcionando unha medida de erro, sen necesidade de acudir á mostra de test. Existen multitude de variantes, como por exemplo, a validación cruzada en k grupos, *k-fold cross-validation*, ou a validación cruzada deixando un fóra, *Leave-one-out cross-validation*, entre outras, sen embargo, a idea subxacente é similar en todas elas. No noso caso, empregaremos validación cruzada en k grupos e será esta a metodoloxía que describiremos a continuación.

Inicialmente, particionamos o conxunto de adestramento en k subconxuntos, axustando un novo modelo por cada un deles. Para o i-ésimo grupo, G_i, $i \in \{1, \ldots, k\}$, consideraremos como mostra de adestramento todas as observacións do conxunto de adestramento orixinal, a excepción das que pertencen a G_i. Deste xeito, para cada un dos k modelos xerados, calcularemos unha medida de erro, empregando cada modelo sobre as observacións excluídas nos correspondentes subgrupos. Finalmente, promediando estas k medidas de erro obtemos unha estimación do erro obtido co modelo xerado coa mostra total de adestramento o que nos sirve como orientación para avaliar a selección dos hiperparámetros involucrados. Se o modelo se compón de diferentes parámetros, como é o habitual, dispoñer dunha medida de erro para cada modelo con distintos valores dos parámetros, permitiranos seleccionar o seu valor óptimo, buscando o modelo con menor erro, e testealo posteriormente sobre a mostra de proba de xeito independente.

Sentadas as bases teóricas do proceso de aprendizaxe, o seguinte paso é trasladalas á práctica. Nas seguintes seccións describiremos as catro aproximacións consideradas no desenvolvemento deste traballo e as conclusións obtidas que motivaron cadansúa. As principais diferenzas entre cada unha delas radican esencialmente na maneira de extraer as mostras de adestramento e test. Tratamos de presentar os resultados prácticos de xeito natural e por orde cronolóxica, de tal maneira, que as conclusións obtidas a partir dun modelo motiven as modificacións consideradas na seguinte proposta.

4.1.2. Mostra aleatoria sobre o conxunto orixinal

A elevada volumetría do conxunto de datos inicial dificulta enormemente os cálculos no computador do alumno. Por tal motivo, para a selección das variables involucradas na construción do modelo decidiuse realizar a extracción dunha mostra representativa do conxunto orixinal, de xeito aleatorio, mantendo as proporcións de cada familia e cada batería. Como resultado, obtívose un novo conxunto formado por 317 problemas, 165 da libraría `EVRIM`, 105 de `MINLP` e 47 de `QP`, cun total de 550062 observacións. Once dos problemas orixinais non teñen representación nesta mostra. As instancias en cuestión son problemas cun número de nodos moi reducido respecto do total, 80 fronte a case oito millóns de rexistros, co que o impacto da supresión dos mesmos é mínimo.

Como primeira aproximación, particionaremos esta mostra en dous conxuntos, o de adestramento e o de test, que empregaremos para construír e avaliar un primeiro modelo, ao tempo que tiramos certas conclusións.

Teñamos en conta que estamos levando a cabo unha enorme redución no conxunto de datos inicial, co que, antes de proseguir sería prudente corroborar que o comportamento da mostra é similar ao do conxunto orixinal. Repetindo a análise que vimos de desenvolver na sección anterior, as conclusións obtidas con este novo conxunto reducido deberían ser semellantes. De seguido, presentamos unha serie de gráficos que nos servirán para ratificar que a forma de proceder non desvirtúa o comportamento de `datos_orixinal`.

A mostra obtida, á que por comodidade nos referiremos como, `datos_reducido`, está composta por 550062 observacións das cales 455667 pertencen á libraría `MINLP`, representando un 82.84 % do total, fronte a 48615 de `QP`, un 8.84 %, e 45780 de `EVRIM`, un 8.32 %. Fixémonos en que estas porcentaxes son prácticamente idénticas ás do conxunto orixinal, atopándose as diferenzas nas milésimas. Na Figura 4.6, presentamos a distribución destas porcentaxes por familia.

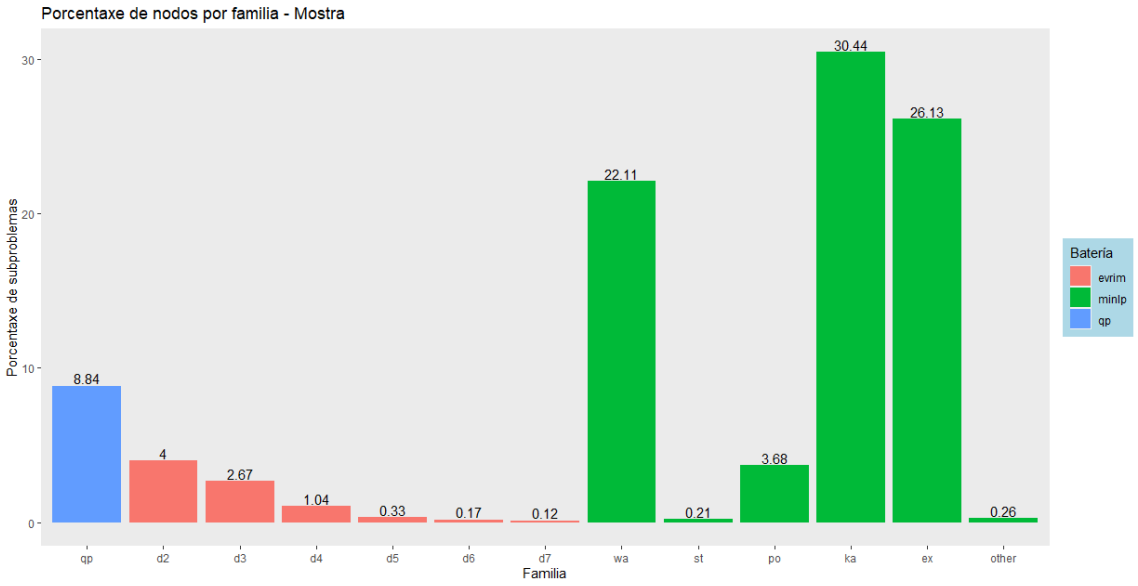


Figura 4.6: Porcentaxe de observacións por familia na mostra reducida.

De igual xeito, a porcentaxe de subproblemas para cada familia é case igual á de `datos_orixinal`, tal e como podemos ver na Figura 4.2. Polo tanto, podemos concluír que a selección da mostra non modificou a distribución das observacións a nivel numérico, tan só reduciu, considerablemente, a súa cantidade.

Recordemos que o noso obxectivo é construír un modelo que sexa capaz de predicir o criterio óptimo, ou, no seu defecto, propoñer un cun desempeño aceptable, á hora de ramificar en cada un dos nodos resultantes de aplicar o algoritmo RLT a un determinado problema de programación polinómica. No noso conxunto de datos a información sobre o desempeño de cada regra é resumida en forma KPIs que compoñen a variable resposta, xogando un papel clave no proceso de aprendizaxe. Ademais de garantir que porcentualmente a distribución das observacións non varía demasiado por batería e familia, tamén debemos asegurar que a redución do conxunto inicial á mostra, non supoña unha perda de información ou unha alteración no relativo ó comportamento da variable dependente.

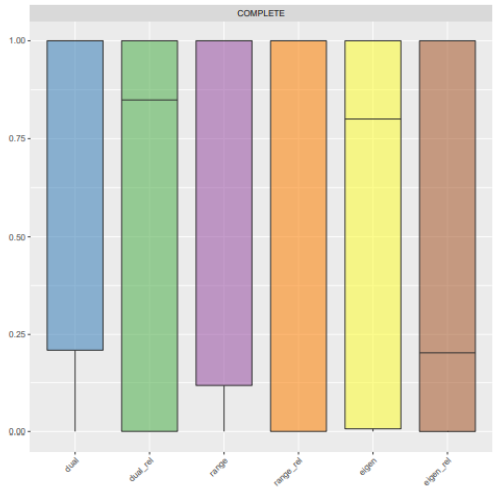


Figura 4.7: Distribución da variable resposta para cada criterio na mostra reducida.

Como podemos ver na Figura 4.7, considerando o total de observacións que compoñen a mostra `datos_reducido`, o comportamento mantense igual ao observado na Figura 4.3, tanto en termos de variabilidade, coma respecto dos máximos e mínimos, ou as medianas. A continuación, presentamos a mesma información baixando o nivel de análise ás baterías.

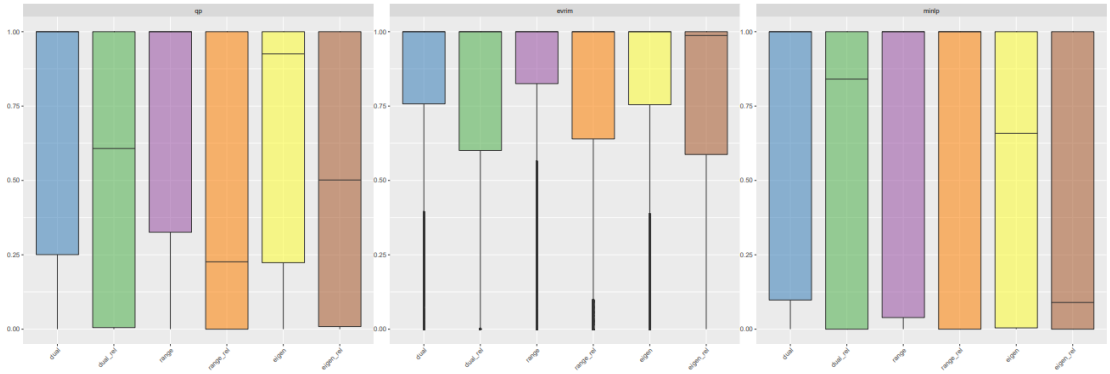


Figura 4.8: Distribución da variable resposta para cada criterio na mostra reducida por batería

De igual maneira, para cada libraría a distribución da variable resposta asociada a cada criterio é moi similar no conxunto orixinal e na mostra seleccionada. Na Figura 4.9 observamos que por familia aplican as mesmas conclusións, co que non nos deteremos a facer maiores consideracións.

Chegados a este punto, podemos retomar o fío inicial da sección. Agora temos máis ou menos clara a distribución dos subproblemas en función da batería e da familia á que pertencen, e tamén como se comporta a variable resposta, o que supón un punto de apoio á hora de interpretar os gráficos asociados aos modelos presentados neste traballo. A mostra seleccionada permitiranos realizar as probas pertinentes e determinar se a información da que dispoñemos e a forma na que a estamos tratando é adecuada para propoñer un modelo relativamente bo, evitando os problemas computacionais detectados inicialmente.

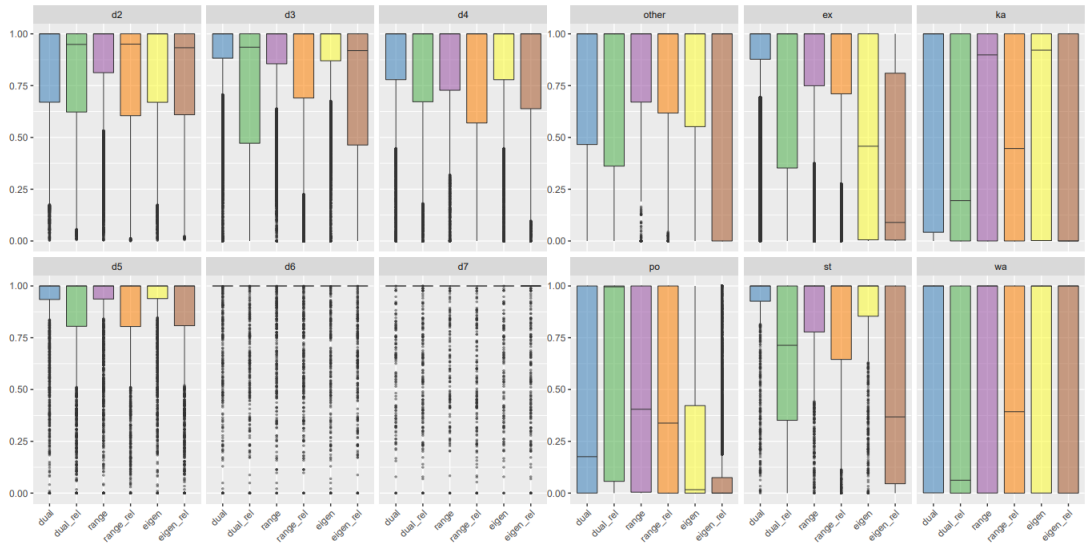


Figura 4.9: Distribución da variable resposta para cada criterio na mostra reducida por familia.

Comecemos dividindo a mostra reducida, `datos_reducido`, en dous subconxuntos: train e test. A mostra de adestramento estará composta por un 70 % das observacións totais e a de test, pola porcentaxe restante, seleccionadas de xeito aleatorio, ao tempo que se manteñen as proporcións orixinais por batería e familia. Ademais, cabe destacar que se unha observación dun problema pertenece a unha das dúas mostras, por exemplo, á de adestramento, o resto de observacións asociadas a ese problema formarán tamén parte da mesma mostra. Este xeito de proceder vén motivado porque, para un mesmo problema, os rexistros asociados a cada subproblema poden ser moi similares, especialmente nas primeiras iteracións do algoritmo, o que introduciría certo sesgo á hora de analizar o desempeño do modelo coa mostra de test se mesturásemos rexistros en ambas mostras. Os conxuntos de adestramento e test resultantes presentan 325595 e 224497 observacións, respectivamente, e 51 variables seleccionadas en base ao estudo de correlacións descrito no capítulo anterior. Nas seguintes gráficas podemos observar as porcentaxes de cada batería e familia en train e test, así como, en `datos_reducido`, que se refire como "Total".

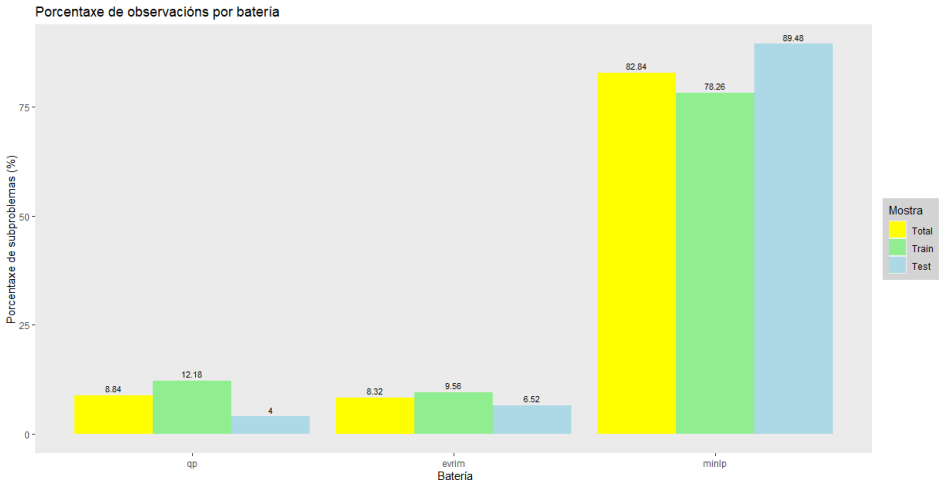


Figura 4.10: Porcentaxe de subproblemas por batería.

As proporcións presentan bastantes diferenzas, especialmente na libraría `MINLP`, Figura 4.10, o que supón un indicativo de que o procedemento escollido para a selección das mostras quizais non é o máis adecuado. Analizando esta mesma información por familia, Figura 4.11, as conclusións son similares, con diferenzas acusadas especialmente en `ka`, `ex` e `qp`.

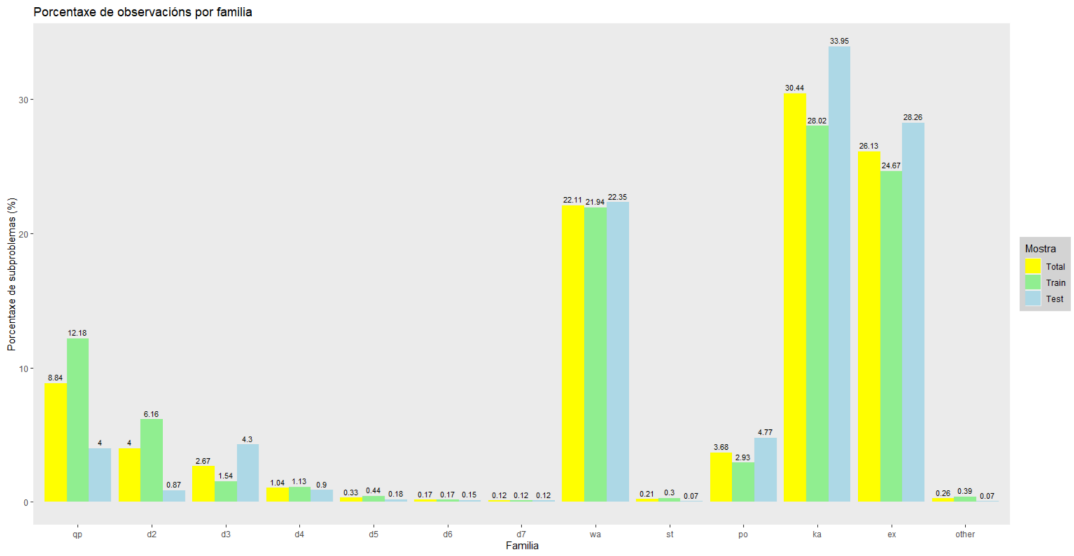


Figura 4.11: Porcentaxe de subproblemas por familia.

A selección dos nodos que forman parte das mostras de train e test realizouse, como xa comentamos, de xeito aleatorio. Sen embargo, para garantir a reproducibilidade dos datos, fixouse unha semente. Podemos pensar que o efecto da mesma é a causa destas discrepancias que vimos de observar, e quizais se a modificamos a situación se normalice. Malia que non engadiremos as correspondentes gráficas neste traballo por extensión e engadir ruído innecesario, o feito é que tras a modificación da mesma, os resultados reflectidos eran similares. Na seguinte táboa poden consultarse os resultados numéricos obtidos por batería variando as mostras consideradas.

Mostra	`MINLPLib`	`QPLib`	`EVRIM`
Train_1	78.26	12.18	9.56
Test_1	89.48	4	6.52
Train_2	83.96	8.48	7.56
Test_2	78.88	10.11	11.01
Train_3	79.61	10.59	9.81
Test_3	89.81	5.06	5.13
Train_4	82.43	8.54	9.03
Test_4	84.24	9.87	5.88
Train_5	85.14	7.12	7.74
Test_5	75.37	14.42	10.22
Conxunto reducido total	82.84	8.84	8.32

Táboa 4.1: Porcentaxes de cada libraría nas submostras de adestramento e test.

As gráficas anteriores indican que as porcentaxes orixinais do conxunto de partida se viron afectadas en ambas mostras, o que provoca que neste sentido non sexan de todo representativas. En calquera caso, o que nos interesa é que o comportamento da variable resposta sexa similar, entre train e test, e tamén respecto da mostra reducida, e é o que verificaremos a continuación. As seguintes figuras presentan unha comparativa de frecuencias para cada regra de ramificación óptima tanto no conxunto de adestramento, coma no conxunto de test.

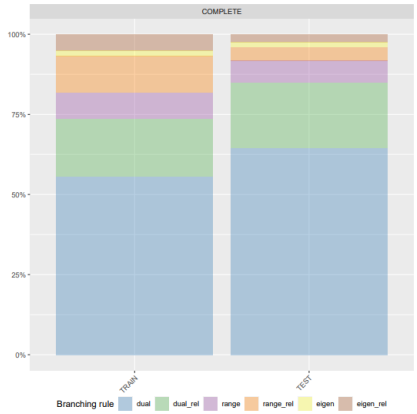


Figura 4.12: Porcentaxe de subproblemas para os que cada regra é optima.

De xeito global, podemos concluír que se seguen observando diferenzas notorias entre ambas mostras. O criterio dual, en azul, semella ser o que maioritariamente resulta óptimo, ao que lle segue o dual baseado en pseudocostes. Sen embargo, na mostra de test a porcentaxe de problemas nas que o criterio dual é óptimo é maior que en train, e de igual xeito ocorre co dual ou co dos rangos baseados en pseudocostes, en laranxa. Ademais do anterior, este é un bo punto para facer certa apreciación sobre os empates en canto a optimalidade. Como podemos observar na gráfica, o dual é indiscutiblemente o criterio que adoita ser óptimo. Sen embargo, debemos ter en conta que se nun nodo, varios criterios resultan óptimos, pola forma na que está programado o noso código en R, escóllese sempre como resultado o primeiro que aplica en caso de que existan empates, aplicando a función máx, o que explica que o criterio dual destaque tanto. Vexamos que ocorre analizando este mesmo gráfico discriminando por batería.

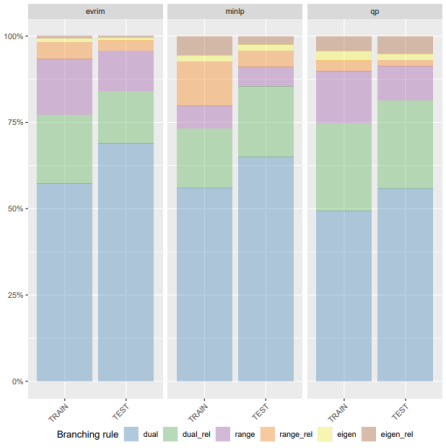


Figura 4.13: Porcentaxe de subproblemas para os que cada regra é optima por batería.

Como podemos observar na Figura 4.13, as conclusións son similares ás que vimos de comentar. Novamente, vemos que maioritariamente o criterio dual domina en máis da metade dos nodos, ao que lle segue o dual baseado en pseudocostes, en verde. Pola contra, os criterios de centralidade están practicamente ausentes. Omitiremos os resultados obtidos por familia, pola similitude cos anteriores.

Asumindo as diferenzas observadas, introducirémonos no proceso de aprendizaxe. Como comentamos no Capítulo 2, empregaremos a metolodoxía boosting combinada coa regresión cuantil, recomendada para aqueles conxuntos de datos que presentan asimetrías, atípicos ou ausencia de normalidade, considerando distintos valores para os cuantís, $\tau \in (0.1, 0.3, 0.5)$.

Os modelos propostos serán obtidos empregando a función `gbm` de R. Tras unha revisión exhaustiva das funcións dispoñibles para o modelado boosting baseado en regresión cuantil, non se atopou ningunha alternativa para o caso multivariante, que é exactamente o que aplica ao noso caso, xa que a variable resposta é multidimensional. Por tal motivo, optouse por adestrar seis modelos distintos, considerando cada unha das compoñentes da variable dependente como unha única variable en sí mesma, e combinar posteriormente os resultados coma se fose un só. Debemos ter en conta que un dos inconvenientes desta forma de proceder é que estaremos omitindo as posibles relacións entre as compoñentes pero é algo que debemos asumir. De seguido describiremos brevemente algúns dos parámetros máis relevantes e os valores finalmente considerados tras varias probas:

- `n.trees`: Número total de árbores que serán axustadas que coincide co número de iteracións realizadas, tendo en conta a definición do algoritmo. Os resultados que se mostrarán ao longo do traballo correspóndense con `n.trees=250`.

- `shrinkage`: Parámetro de regularización aplicado a cada árbore que controla a velocidade na aprendizaxe do algoritmo. O ideal é que tome valores pequenos, para garantir unha aprendizaxe lenta, pero tampouco demasiado, xa que canto máis pequeno sexa o valor deste parámetro maior número de árbores será necesario. No noso caso consideramos 0.1, valor por defecto.

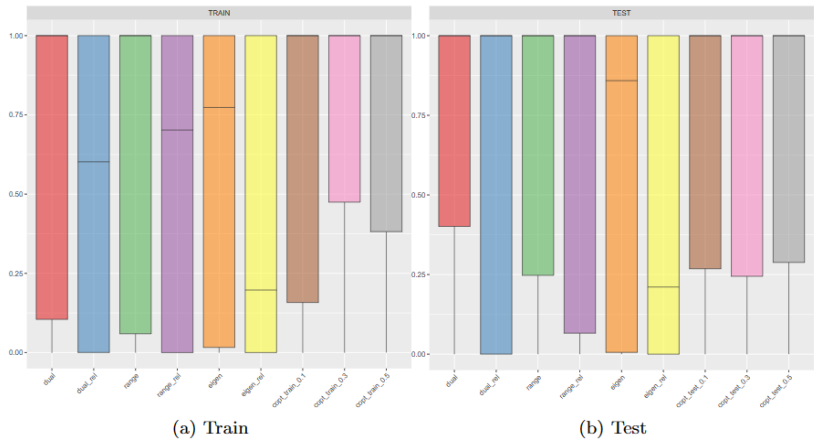
- `cv.folds`: Número de grupos considerados na validación cruzada. O seu valor foi fixado en `cv.folds=5`.

- `n.minobsinnode`: Número mínimo de observacións en cada nodo das árbores de regresión. No noso caso `n.minobsinnode=10`.

- `interaction.depth`: Máximo nivel de profundidade en cada árbore. Nas probas executadas, tomouse `interaction.depth=2`.

No que segue presentaremos unha serie de gráficos nos que se amosan os resultados obtidos con cada un dos modelo, facilitando así a súa interpretación.

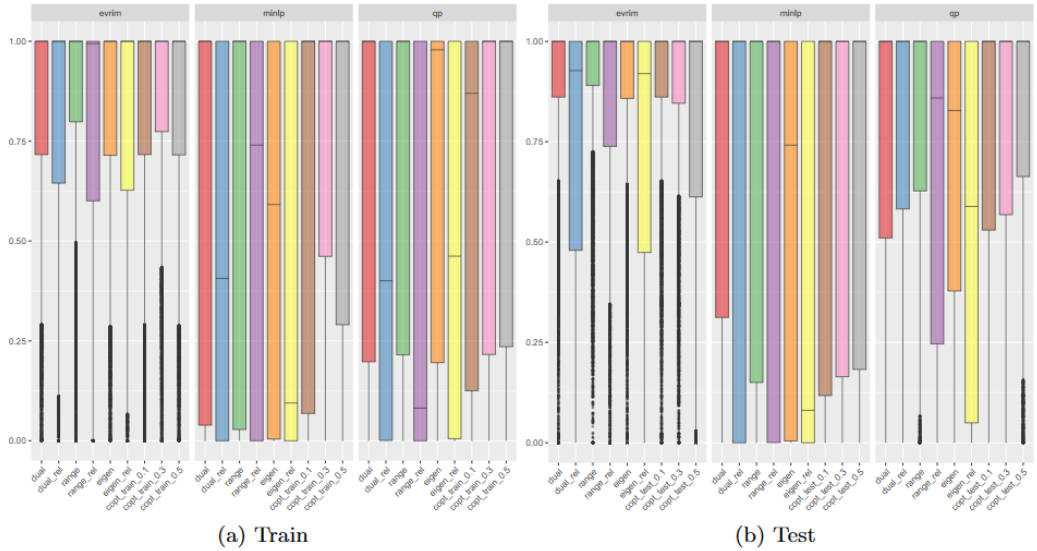
Nos boxplots da Figura 4.14 amósase o comportamento da variable resposta para cada criterio en ambas mostras así como, o valor de KPI asociado ao criterio predito como óptimo con cada un dos tres modelos xerados, dependendo do cuantil considerado. Como xa viñamos anticipando, o comportamento de train e test semella ser lixeiramente distinto, apreciándose no primeiro conxunto unha maior variabilidade para os criterios dual e dos rangos, en vermello e verde, que no segundo. Esta diferenza non fai máis que incrementarse no modelado. O modelo proposto para $\tau = 0.3$, en cor rosa, parece ser o que presenta mellor comportamento na mostra de adestramento, incluso mellor que os criterios propostos, aínda que cun rango intercuartílico demasiado amplo, que denota un exceso de variabilidade. Pola contra, na mostra de test observamos que a situación reflectida non é a mesma, presentando o criterio dual mellores resultados que o mesmo. Situación semellante ocorre para $\tau = 0.5$ (en gris).



(a) Train (b) Test

Figura 4.14: Resultados do modelo boosting obtido a partir da mostra reducida.

Analizando a información por batería, Figura 4.15, poden detallarse máis os anteriores comentarios. Para a libraría `EVRIM` observamos tanto en train coma en test que calquera dos criterios presenta, polo xeral, un bo desempeño, salvando certos casos atípicos. Sen embargo, en test acentúanse máis as diferenzas entre uns e outros, destacando especialmente os criterios baseados en pseudocostes polo seu comportamento anómalo. No tocante aos modelos propostos, ningún parece destacar, e ademais, en test, o modelo para $\tau = 0.5$ sofre un empeoramento notable respecto dos outros dous. Para `MINLP` segue a apreciarse unha enorme variabilidade, algo que caracteriza a esta batería. Sen embargo, aínda que en adestramento aparentemente o modelo para $\tau = 0.3$ é o mellor, e ademais, resulta vantaxoso respecto dos criterios prefixados, en test semella ter un desempeño similar aos mesmos e, noutras mostras observouse que incluso peor. Os problemas cadráticos presentan unha situación similar á de `MINLP` aínda que con menor variabilidade para os criterios en bruto.

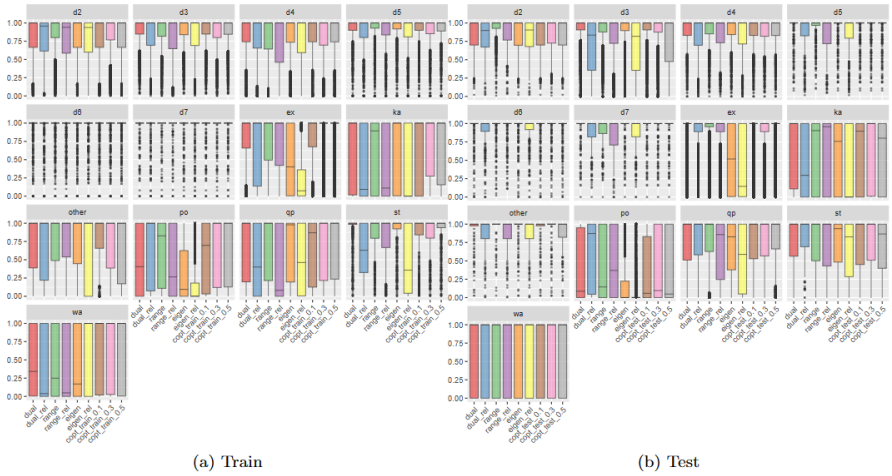


(a) Train (b) Test

Figura 4.15: Resultados do modelo boosting obtido a partir da mostra reducida por batería.

No seguinte gráfico pódese observar esta mesma información baixando un nivel máis de granularidade.

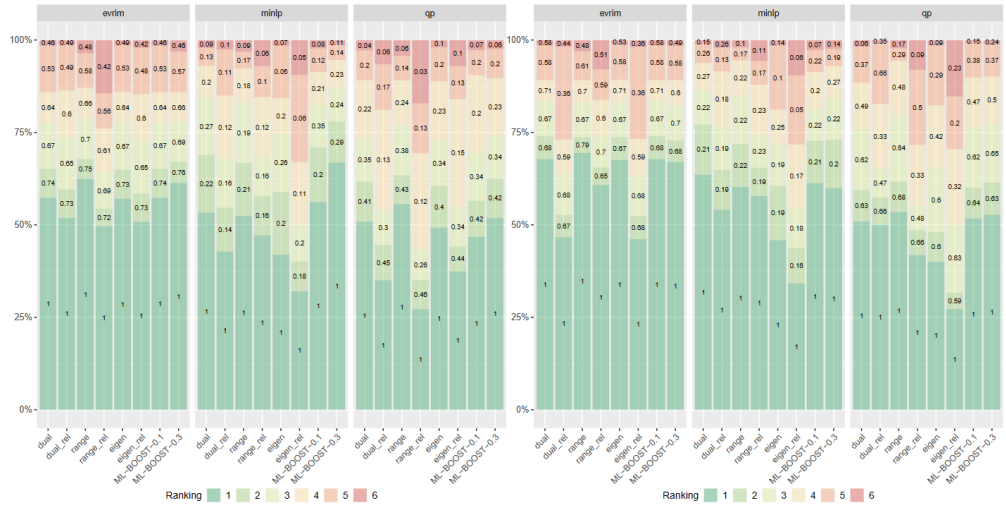
Omitindo a familia `other`, que engloba problemas de diferente tipoloxía e ata certo punto, pode xustificar as diferenzas observadas, a situación non é moi diferente das anteriores. Os modelos propostos, en cor marrón, rosa e gris, non son capaces de propoñer un criterio de ramificación para cada nodo, presentando mellor desempeño, en xeral, que un dos criterios fixado, feito que pode vir motivado polas diferenzas entre ambas mostras, especialmente notorias en familias como `ex` ou `st`.



(a) Train (b) Test

Figura 4.16: Resultados do modelo boosting obtido a partir da mostra reducida por familia.

Á vista do anterior barallamos dúas opcións. A primeira delas é que o que estamos a ver nas gráficas se debe a un comportamento patolóxico das mostras seleccionadas e considerando outra semente a situación se normalizaría. Malia que non se engaden as gráficas correspondentes, executáronse as probas pertinentes con outras mostras e foi comprobado que a situación é semellante. A segunda opción supón que a forma de seleccionar a mostra reducida orixina as diferenzas observadas entre adestramento e test afectando a súa vez á aprendizaxe do modelo. Neste caso será a opción que cremos máis probable.



(a) Train (b) Test

Figura 4.17: Ranking dos criterios de ramificación e dos modelos xerados a partir da mostra reducida.

Nas gráficas anteriores focalizámonos na análise do criterio óptimo seleccionado polos modelos e a súa correspondente distribución. Sen embargo, naqueles casos nos que o criterio proposto non é realmente o óptimo, non somos quen de avaliar se os resultados son próximos ao mesmo ou o criterio seleccionado é o peor de todos. Idealmente o modelo proposto debería seleccionar o criterio de ramificación óptimo sempre. Sen embargo, esta é unha aspiración moi ambiciosa e tamén improbable. Tratando de dar unha solución máis realista, podemos idear a busca dun modelo que, aínda que erre, propoña unha regra cun rendemento aceptable. Para tratar de ilustrar isto empregaremos os gráficos da Figura 4.17.

Para cada subproblema, establécese un ranking de 1 a 6 para cada un dos criterios de ramificación, do mellor ó peor, cuantificando o seu desempeño a partir do KPI estandarizado definido en (3.2), asignando cores de verde a vermello para facer máis visual a análise. Ademais de proporcionarnos información acerca do número de problemas para os que un criterio resulta óptimo, tamén se acompañan as gráficas con certos valores numéricos, que representan o promedio dos valores do KPI para ese conxunto de problemas. Por exemplo, para todos os problemas da mostra de adestramento para os que o criterio dual é o segundo mellor criterio, o promedio do KPI é 0.74, do que podemos deducir que, aínda que non sexa o óptimo, para os problemas de `EVRIM`, o criterio dual é unha boa elección como regra ramificadora posto que o valor promedio é próximo a 1. Nos sucesivos promedios, 0.67, 0.64 non decae demasiado, co que o seu desempeño é aceptable, pero por suposto moi mellorable, o que motiva novamente o noso traballo.

A conclusións que podemos tirar son similares ás das anteriores gráficas. Para os problemas de `EVRIM`, tanto en adestramento coma en test todos os criterios presentan polo xeral bo desempeño, o que se reflexa nos valores altos do KPI. Sen embargo, hay problemas para os que o promedio acada valores próximos ao 0.4, o que fai evidente a necesidade dunha alternativa para estes casos. Para `MINLP` a situación é distinta. Para cada problema semella existir un criterio óptimo claro, salvando empates, e o resto de criterios presentan un desempeño moito peor, algo que apreciamos no decrecemento abrupto do KPI de 1 a 0.2 ou a 0.1. Aínda que en `QP` a situación non é tan radical, sí que se aprecia este empeoramento que vimos de comentar no KPI, en ambas mostras.

No relativo aos modelos propostos, para `EVRIM` o criterio dos rangos actúa lixeiramente mellor que os modelos propostos, cun 62.45 % de problemas para os que o primeiro é o criterio óptimo en train, fronte a 61.37 % do ML-BOOST-0.3 e 57.36 % para ML-BOOST-0.1. En test a situación é similar, cun 62.45 % de problemas para os que o primeiro é o criterio óptimo en train, fronte a 61.37 % do ML-BOOST-0.3 e 57.36 % para ML-BOOST-0.1. Para `MINLP` comezamos a observar as mencionadas discrepancias entre adestramento e test. A train parece suxerir que o modelo para $\tau = 0.3$ é mellor que o resto de criterios, cun 66.88 % para os que é o mellor, aínda que naqueles nos que non é o caso, erra enormemente, con valores do KPI próximos a 0.29. Pola contra, en test non é maioritariamente o mellor, de feito a porcentaxe de acertos en optimalidade é maior para o modelo con $\tau = 0.1$. Para `QP`, non apreciamos unha discrepancia tan evidente coma no caso anterior, sen embargo é o criterio dos rangos o que parece resultar o que ten mellor rendemento, cun 55.64 % dos problemas para os que é óptimo en train, ao que lle segue o modelo ML-BOOST-0.3, cun 51.9 %. En test a situación é semellante.

Das anteriores gráficas pódense tirar varias conclusións. A primeira, que existe un alto número de empates posto que as porcentaxes da primeira posición do ranking de todos os criterios non suman 100. O segundo, en ningunha das baterías hay un criterio nin modelo que destaque especialmente por ser o mellor, ningunha das porcentaxes para as primeiras posicións dos rankings sobrepasa o 70 %, o que é un indicativo de que debemos replantexar os modelos obtidos. E terceira, e última, omitindo o caso da libraría `EVRIM`, para a cal, os criterios, polo xeral, actúan todos máis ou menos ben, os modelos propostos non son quen de proporcionar unha regra alternativa á óptima, cando erran, presentando un desempeño aceptable, do que podemos concluír, que o algoritmo non é quen de caracterizar ben os problemas.

A modo de comparativa, en Rodríguez-Ballesteros (2022), non se apreciaban diminucións tan abruptas no KPI en media, do que se conclúe que ao traballar nodo a nodo, é difícil propoñer un criterio que localmente presente un desempeño igual de aceptable que o da regra de ramificación óptima, a

excepción da librería `EVRIM`, na que se aprecia a existencia dun "plan B" cun KPI non demasiado malo, en media. Debemos ter en conta tamén que, na anterior referencia bibliográfica, o KPI era una medida distinta á actual, e o criterio era o mesmo para todos os subproblemas, de tal maneira que aínda que o seu desempeño fora moi malo nun determinado nodo, os outros "compensaban" o KPI dando lugar a valores relativamente bos. Isto agora non ocorre, estamos a traballar cun KPI de carácter local, o que xustifica en certo sentido o que observamos nas gráficas.

Á vista de todo o anterior, é claro que os modelos obtidos non están a presentar o comportamento esperado. Amosan unha ampla variabilidade en termos de KPI óptimo, cando en realidade, o ideal sería que o rango intercuartílico se movese entre 0.8 e 1, o que suxire que erra bastante nas súas predicións. Ademais, nestes casos, analizando os rankings vemos que os criterios propostos presentan, en promedio, valores de KPI bastante baixos. Como xa adiantamos, unha posible causa que xustifica esta situación poden ser as diferenzas entre as mostras de adestramento e test, debido ao xeito de xerar a mostra reducida. Tomando como punto de partida esta hipótese, na seguinte sección, propoñemos unha nova aproximación baseada na modificación do proceso de extracción das mostras de train e test co obxectivo de minimizar as discrepancias entre ambas.

4.1.3. Mostra con $k = 500$ nodos

Durante a análise descriptiva inicial realizada na sección anterior, observáronse diferenzas entre as mostras de adestramento e test anticipando, en certo modo, as conclusións finalmente obtidas. Debemos ter en conta que o conxunto empregado na construción do modelo e a mostra de test deben presentar un comportamento semellante, posto que, en caso contrario, os coñecementos adquiridos non aplicarían e os resultados serían aleatorios e sen fundamento orixinando as discrepancias observadas. Isto non só ocorre cos modelos obtidos, senón tamén cos criterios prefixados, onde se aprecian resultados estraños e diverxentes en train e test. Ante as diferenzas evidentes entre ambas mostras, proporemos unha nova alternativa para a selección das mesmas tratando de buscar unha solución a esta problemática.

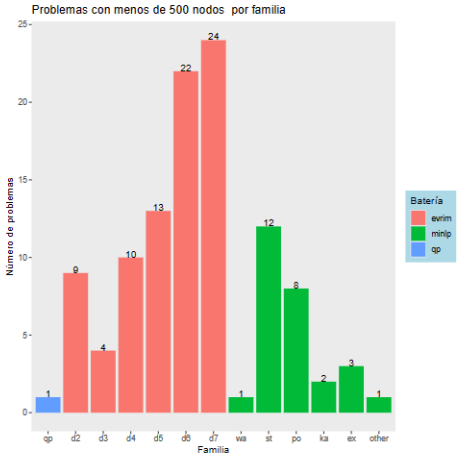


Figura 4.18: Número de problemas con menos de 500 nodos por familia no conxunto orixinal.

A libraría que presenta maior volumetría respecto do número de nodos é `MINLPLib`, e tamén é a segunda con maior número de problemas, despois de `EVRIM`. Dentro desta batería, destacan as familias `wa`, `ka` e `ex`. Como podemos ver na Figura 4.11, o procedemento empregado na sección anterior para a obtención das mostras de train e test desvirtuou esta situación, o que pode deberse á enorme diferenza no número de nodos dos problemas considerados. Así, pode ocorrer que algún problema cun alto número de subproblemas "caese" na mostra de proba, e desvirtuase o resultado. O principal problema é que considerando outras mostras distintas de train e test, podería ocorrer o mesmo, e as discrepancias

continuarían a existir. Este feito supón o punto de partida para a nova selección que describiremos a continuación.

A batería `EVRIM` é a que aporta o maior número de problemas no conxunto orixinal, concretamente, 170. Sen embargo, estes problemas resólvense, polo xeral, en poucas iteracións, en contraste cos problemas de `QP` ou `MINLP` que presentan maior volumetría, Figura 4.1. Dos 328 problemas considerados, 110 resólvense en menos de 500 iteracións: 1 deles pertecence á libraría `QP`, 27 a `MINLP` e 82 a `EVRIM`. Na Figura 4.18, analizamos esta mesma información por familia.

Para a libraría `EVRIM`, vemos que, a medida que consideramos problemas con maior grao, a porcentaxe dos mesmos que se resolven en menos de 500 nodos vaise incrementando. Isto non implica que ditas instancias sexan menos complexas, nin necesariamente de resolución máis rápida en canto a tempo computacional, senón que se resolven nun menor número de iteracións. En canto a `MINLP`, `po` e `st` son as familias que maior número de problemas con menos de 500 nodos.

A nova aproximación proposta para minimizar as diferenzas atopadas entre train e test consiste en, a partir do conxunto inicial, con 328 problemas e 7856719 observacións, tomar unha nova mostra á que nos referiremos como `mostra_500`, onde, para aqueles problemas con menos de 500 nodos, se manteñan todos os rexistros, e para os problemas restantes se considere un número fixo k de 500 subproblemas. Así, `mostra_500` estará formada por 328 problemas e 128432 observacións, onde 45058 forman parte da batería `MINLP`, aproximadamente un 35.08 % dos datos, fronte a 23242 de `QPLib`, un 18.1 % e 60132 de `EVRIM`, cun 46.82 %. É de destacar a diferenza na distribución das porcentaxes de observacións que representa cada batería respecto do conxunto inicial, até o punto en que `EVRIM` pasa a ser a que maior volumetría presenta.

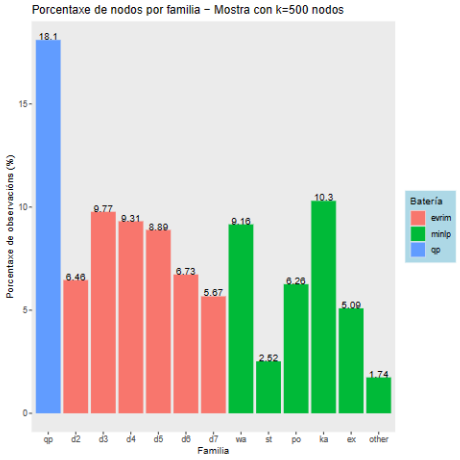


Figura 4.19: Porcentaxe de observacións por familia na mostra xerada para k =500.

Comparemos a distribución do número de nodos no conxunto inicial, `datos_orixinal` e `mostra_500`, Figuras 4.2 e 4.19. Como podemos observar, no primeiro caso existía unha evidente disparidade entre as porcentaxes, con diferenzas considerables. Pola contra, a estratexia empregada para a obtención desta nova mostra, permitiu igualar as porcentaxes por cada familia, sen presentarse diferenzas tan acusadas. En efecto, este xeito de proceder modificou a distribución inicial do conxunto, sen embargo, non ten por que ser un aspecto negativo. Pode ocorrer que exista un exceso de observacións que realmente non aportan máis información ao conxunto, todo o contrario, contribúen a confusión do modelo, engadindo ruído á aprendizaxe debido ao desbalanceo que producen.

No que segue, analizaremos en detalle o comportamento da variable resposta nesta nova mostra e compararémolo cos do conxunto orixinal, Figura 4.20. Respecto da Figura 4.3 observamos numerosas diferenzas. A máis destacable, é que a mediana da variable resposta asociada a cada criterio é igual a 1, ou, no caso, do criterio de centralidade baseado en pseudocostes, `eigen_rel`, moi próxima á unidade.

Isto significa que o 50 % das observacións, presentan un KPI igual a 1 para cada regra de ramificación. O feito de que isto ocorra en todos os criterios fainos pensar na existencia de empates, nos que varios poden ser óptimos. Debemos ter en conta este aspecto e analizar con detemento como afecta aos modelos obtidos. Ademais, podemos destacar a enorme reducción da variabilidade. Esta continúa a apreciarse, especialmente nos criterios baseados en pseudocostes, cuxo comportamento semella ser, polo xeral, peor que a súa versión en bruto, pero non dun xeito tan notorio. O criterio do dual, `dual`, e do ranking, `range` son os que presentan mellores resultados en termos de KPI. Novamente podemos pensar que é un efecto da sementa fixada para a selección dos k nodos, mais comprobouse que non era así; as conclusións con outras sementes eran semellantes.

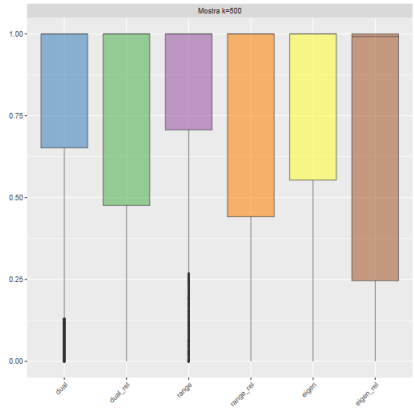


Figura 4.20: Distribución da variable resposta para cada criterio en `mostra_500`.

Vexamos que ocorre cando analizamos esta mesma información por batería, Figura 4.21. En `EVRIM` a situación é similar á do conxunto total, Figura 4.8, e polo tanto, tamén as conclusións. Isto ten certa lóxica, se temos en conta que esta libraría representa case o 47 % das observacións totais de `datos_orixinal`. Aínda así, apreciamos unha menor variabilidade e un lixeiro aumento no número de atípicos. Fixándonos en `MINLP`, observamos que continúa a ser unha batería con alta variabilidade onde ningún criterio destaca especialmente, presentando tamén un comportamento similar ao orixinal. No relativo a `QP`, segue a mesma liña que as baterías que vimos de comentar.

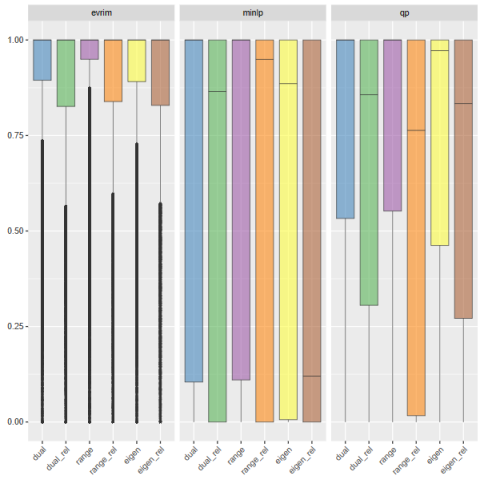


Figura 4.21: Distribución da variable resposta para cada criterio en `mostra_500` por batería.

Analizando a distribución da variable resposta por familia, cuxos gráficos se omiten pola similitude cos anteriores, observamos que, de novo, aquelas familias asociadas á libraría `EVRIM` presentan un comportamento similar ao do conxunto orixinal, observándose unha lixeira diminución da variabilidade, como xa comentamos con anterioridade. Para `MINLP` é onde se aprecian menores diferenzas, a variabilidade é similar e aínda que algunhas medianas diminúen o seu valor, outras auméntano, como é o caso do criterio dos rangos baseados en pseudocostes para a familia `ka`, o que ben pode ser efecto da semente para a selección da mostra.

Asumindo a redución observada na variabilidade, especialmente en `EVRIM`, podemos concluír que o comportamento desta nova mostra non dista demasiado do do conxunto de partida, co que, o seguinte paso é proceder á construción do modelo. A tal fin, xeraremos as mostras de adestramento e test a partir de `mostra_500`, considerando, respectivamente, un 70 % e un 30 % das observacións, tentando manter as proporcións da mesma mostra, que como xa vimos, foron alteradas respecto do conxunto inicial.

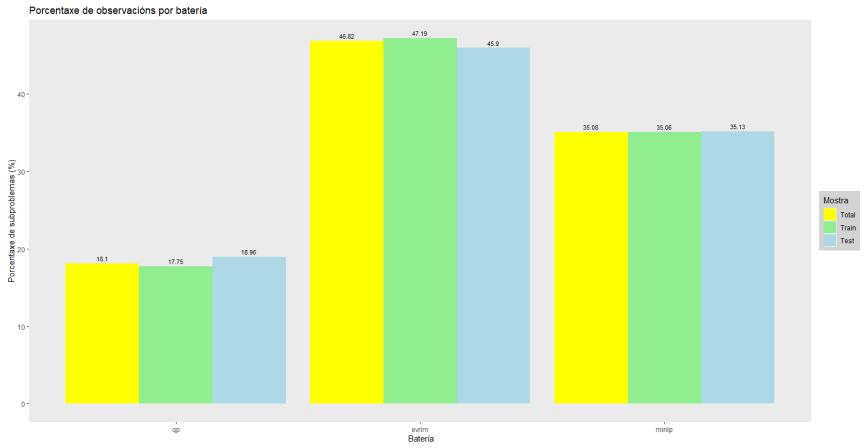


Figura 4.22: Porcentaxe de subproblemas por batería para `mostra_500`.

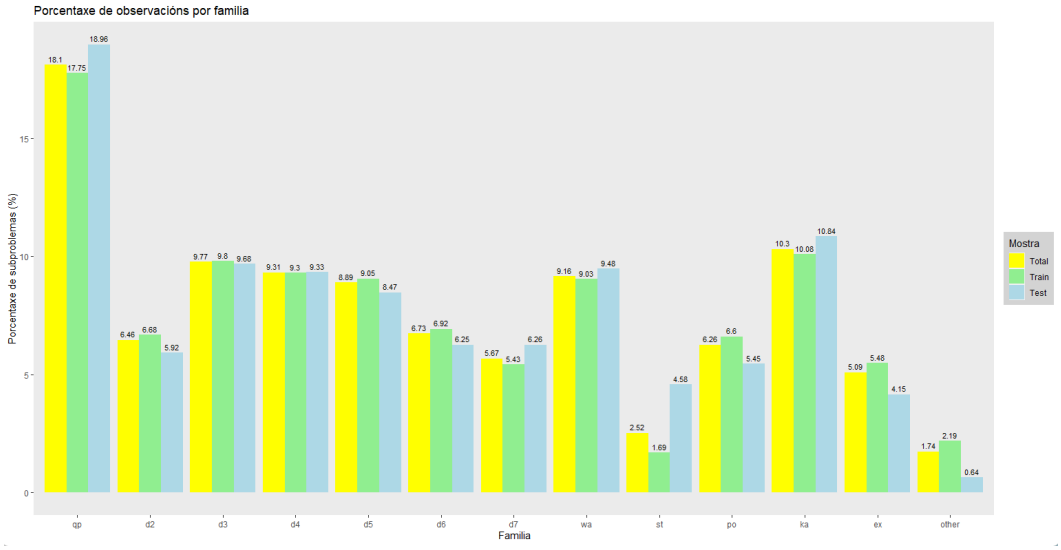


Figura 4.23: Porcentaxe de subproblemas por familia para `mostra_500`.

Os conxuntos resultantes presentan 91518 e 36914 observacións, e 51 variables. A mostra de train está formada por 235 problemas, 33 de `QP`, 81 de `MINLP`, 121 de `EVRIM` e a de test por 93, onde 14 pertencen a `QP`, 30 a `MINLP`, 49 a `EVRIM`. Nas Figuras 4.22 e 4.23, podemos ver unha comparativa da porcentaxe de observacións nas mostras de adestramento e test, respecto de `mostra_500`, por batería e por familia. En comparación co observado nas Figuras 4.10 e 4.11 para `datos_reducido`, as porcentaxes variaron cuantitativamente. Sen embargo, acadouse o obxectivo que buscabamos, equiparar a situación entre train e test para cada batería, suavizando o predominio de `MINLP` cuxa elevada volumetría podía exercer un efecto de apantallamento respecto das outras librarías. As mesmas conclusións aplican se realizamos o estudo por familia, co que non nos deteremos a facer maiores consideracións. No que segue, estudaremos o comportamento de ambas mostras respecto da variable resposta.

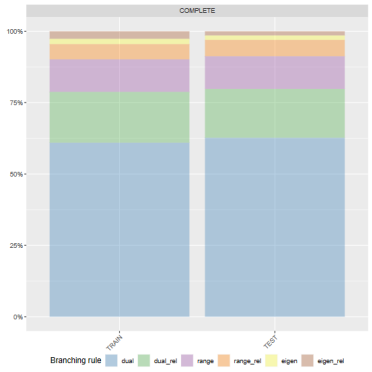


Figura 4.24: Porcentaxe de subproblemas por regra óptima en `mostra_500`.

Analizando globalmente ambos conxuntos vemos que, efectivamente, as diferenzas entre as dúas mostras no relativo á porcentaxe de problemas por criterio óptimo diminúen, en contraste co observado na Figura 4.12. Así mesmo, continúa a apreciarse o predominio do criterio dual, pero como xa puntualizamos en anteriores ocasións, a existencia de empates pode ser a causa.

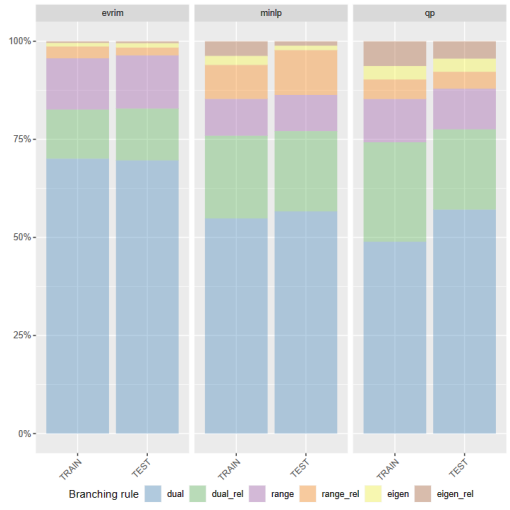
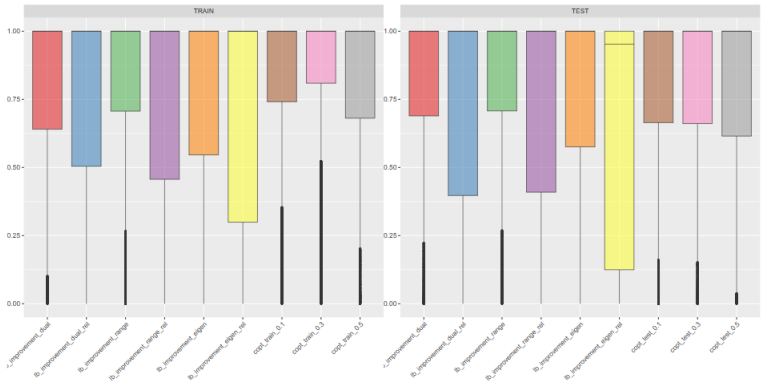


Figura 4.25: Porcentaxe de subproblemas por regra óptima en `mostra_500` para cada batería.

Por batería as conclusións son similares tal e como podemos ver na Figura 4.25. As mostras de

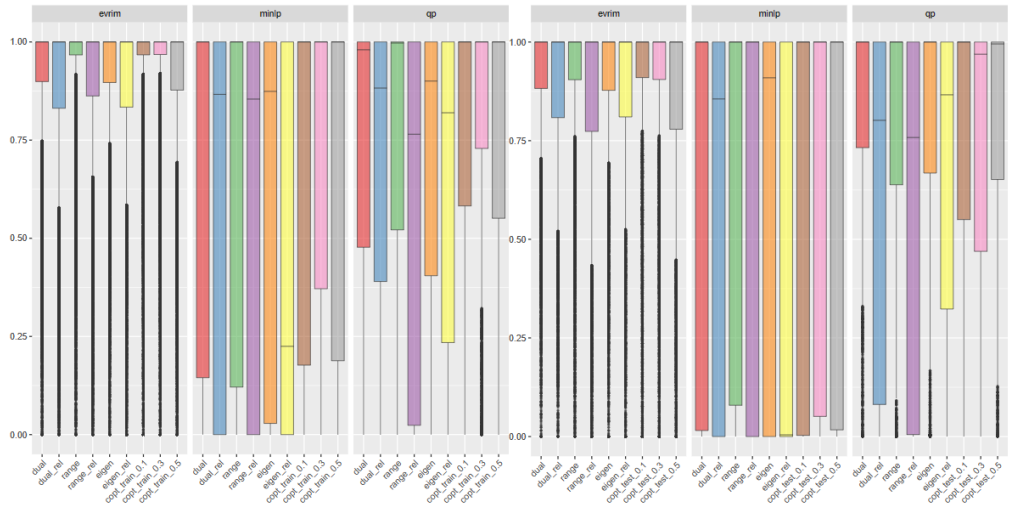
adestramento e test son especialmente parecidas en canto a porcentaxes para `EVRIM`, e, en menor medida, para `MINLP`. As maiores discrepancias atópanse en `QP`, o que tamén pode deberse á selección da mostra e ao efecto da semente empregada. En calquera caso, debemos ter en conta que os problemas cadráticos difiren bastante uns doutros o que, en certo sentido, pode xustificar esta situación. Omitiremos as gráficas por familia por ser unha extensión do que vimos de comentar.



(a) Train (b) Test

Figura 4.26: Resultados do modelo boosting obtido a partir de `mostra_500`.

Atendendo ao anterior, podemos concluír que a estratexia empregada para igualar train e test en termos de KPI deu resultado. Vexamos agora se á hora de adestrar o modelo se aprecian algunhas melloras. O proceso de aprendizaxe realizarase nas mesmas condicións que na proba anterior, adestrando 6 modelos, un para cada criterio, con tres cuantís distintos, $\tau \in (0.1, 0.3, 0.5)$, e combinando a predición de todos eles considerando aquela que acada o valor máximo para a selección do criterio óptimo. Este último resultado é o que se representa na Figura 4.26.



(a) Train (b) Test

Figura 4.27: Resultados do modelo boosting obtido a partir de `mostra_500` por batería.

Respecto da Figura 4.14, o aspecto que máis chama a atención, tanto en train coma en test, é a

enorme reducción na variabilidade. Polo xeral, semella que os criteros dual e dos rangos son os que mellores resultados presentan en termos de KPI, e os modelos para $\tau = 0.1$ e $\tau = 0.3$ parecen imitar o comportamento de ambos, sendo unha especie de sinerxía ambas regras.

Por batería, Figura 4.27, ao igual que con `datos_reducido`, os problemas de `EVRIM` presentan, polo xeral, bos resultados, independentemente da regra de ramificación empregada. Isto é positivo e negativo á vez, no sentido de que, a escasa variabilidade e a similitude nos resultados orixinados por cada regra ao ramificar son moi parecidos, o que fai que, independentemente do criterio que seleccione o modelo, aínda que o faga de forma azarosa, será unha boa proposta pola natureza deste tipo de problemas. Esta peculiaridade dos mesmos dificulta a aprendizaxe e a distinción daquel criterio que é óptimo. Nesta mostra concreta tanto o criterio con $\tau = 0.1$ coma $\tau = 0.3$ parecen imitar o comportamento do criterio dos rangos, que polo xeral, resulta o mellor en termos de KPI, tanto en adestramento coma en test.

Nas outras dúas librarías a situación é totalmente contraria. A variabilidade é moito maior, en especial para `MINLP`, e ningún dos modelos propostos presenta bos resultados. Para os problemas cadráticos, aínda que con menor variabilidade, a situación é similar e observamos un comportamento anómalo dos modelos, xa que, en train parece que o modelo con $\tau = 0.3$ é o óptimo pero en test a situación e lixeiramente distinta, cedendo o posto a $\tau = 0.1$. Esta mesma situación xa a viñamos observando na mostra da sección anterior e atribuímolo ás diferenzas entre as mostras, pero semella que esta non era a causante. No caso desta batería podemos pensar que son problemas cadráticos e, polo tanto, de difícil resolución, pero para os problemas de `MINLP` a situación é estraña, da a sensación de que o modelo non está a aprender correctamente e simplemente trata de imitar o comportamento dos criterios que, a modo xeral actúan mellor, aínda que posteriormente o resultado non fose o correcto.

No que segue, realizaremos esta mesma análise por familia. Omitiremos os gráficos asociados por cuestións de visualización, mais resumiremos brevemente os aspectos observados. As familias de problemas asociadas á batería `EVRIM` presentan todas elas un comportamento similar e pódense barallar as mesmas conclusións que vimos de comentar no parágrafo anterior. Desta quenda, faremos especial antención ás familias de `MINLP`. Todas elas son moi diferentes entre sí. A familia `other` podemos, en certo modo, xustificala porque engloba problemas de diferente tipoloxía, pero para o resto a situación é un tanto peculiar. Por exemplo, `ka` e `wa`, dúas das baterías con maior volumetría e sobre as que a redución no número de observacións foi máis abrupta, presentan unha alta variabilidade tanto en train coma en test. Sen embargo, familias coma `ex` presentan un comportamento totalmente distinto en ambas mostras, o que pode estar afectando ao desempeño dos modelos no conxunto da propia batería. Que pode estar ocorrendo cos problemas desta familia para que as diferenzas sexan tan notorias? Recordemos que `ex` era a segunda familia con maior volumetría e a redución ao seleccionar a mostra foi moi elevada.

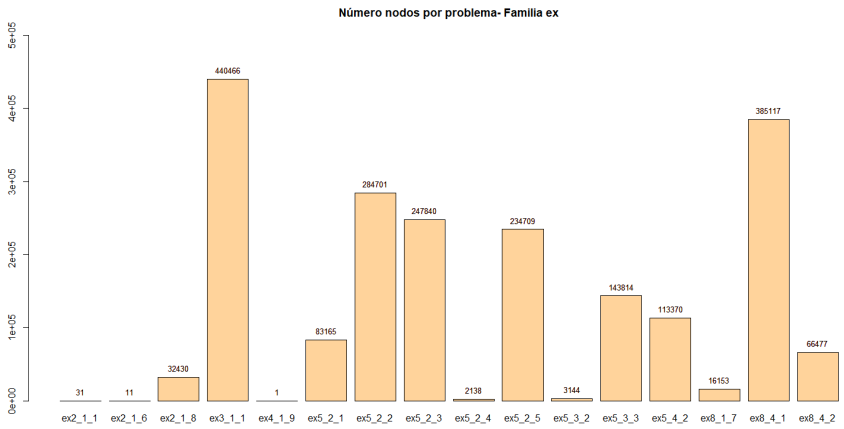


Figura 4.28: Número de nodos de cada problema da familia `ex` no conxunto orixinal.

A familia `ex` está composta por 16 problemas empregados como exemplo en distintos capítulos de Bussieck et al. (2003), co que, en principio, non teñen por que presentar características comúns e explica a disparidade observada neste conxunto. Ademais, como podemos ver na Figura 4.28, o número de iteracións necesarias para resolvelos pode variar moito duns a outros, o que non fai máis que deixar clara as diferenzas.

Dos 16 problemas pertencentes á familia `ex`, 12 forman parte da mostra de adestramento: `ex2_1_6`, `ex2_1_8`, `ex4_1_9`, `ex5_2_1` , `ex5_2_2`, `ex5_2_4` , `ex5_2_5` , `ex5_3_2`, `ex5_3_3` , `ex8_1_7` , `ex8_4_1` , `ex8_4_2`. A cuestión é se estes problemas están e influír no proceso de aprendizaxe dos modelos e crean confusión no resto de familias. Para comprobalo, executáronse os mesmos cálculos suprimindo as observacións desta familia e os resultados foron practicamente idénticos, motivo polo cal se omitiron neste traballo.

As diferenzas nos resultados e a peculiaridade dos mesmos para a familia `ex` poden xustificarse por ser problemas cun número de nodos moi distintos, tendo en conta ademais, que para os que tiñan máis de 500 nodos se prefixou esta mesma cantidade de observacións para incorporalos á mostra. Sen embargo, para o resto de familias non existe unha xustificación aparente. Unha opción que barallamos é que a información que estamos proporcionando ao modelo non sexa suficiente. Recordemos que partiamos dun conxunto don case oito millóns de observacións e, a mostra, aínda que aparentemente representativa en termos de KPI, é moito menor, con 128432 rexistros. Con motivo de paliar os posibles efectos da redución nos datos, a seguinte proba que realizaremos será aumentar o número fixo de nodos a 10000.

4.1.4. Mostra con $k = 10000$ nodos

Os resultados observados nas anteriores mostras deixan constancia de que os modelos obtidos non están a asimilar o proceso de aprendizaxe de xeito adecuado. Intentan imitar o comportamento dos criterios que maioritariamente actúan mellor, mais semella que neste sentido as características do problema non están a aportar demasiado. Barallamos dúas opcións. Por unha banda, pode que sexa necesaria maior cantidade de información que a que estamos a empregar, debemos pensar que os conxuntos utilizados até o momento representan menos dun 10 % das observacións do conxunto orixinal. Por outra banda, pode que os problemas que estamos a considerar sexan moi diferentes entre sí, aínda pertencendo á mesma familia, e isto pode estar entorpecendo o proceso de aprendizaxe.

Recordemos que as familias foron unha clasificación definida por nós, e por exemplo, no caso de `ex`, estamos ante unha das familias de maior volumetría no conxunto orixinal nas que hai enormes diferenzas no número de iteracións necesarias para resolver cada un dos problemas que a conforman e ademais, non presentan un nexo común en canto a tipoloxía ou características, que en principio poden tomar valores totalmente contrarios, simplemente son exemplos dun mesmo libro que decidimos agrupar de maneira artificiosa. Esta situación pode estar creando confusión ao modelo.

Seguindo a liña de conclusións que vimos de expoñer, comezaremos analizando os resultados obtidos se fixamos un maior número de nodos, por exemplo, $k = 10000$. No conxunto inicial, `datos_orixinal`, tiñamos 328 problemas dos cales 240 se resolven en menos de 10000 iteracións: 29 deles pertencen á libraría `QP`, 55 a `MINLP` e 156 a `EVRIM`. Fixémonos en que máis do 90 % dos problemas de `EVRIM` presentan menos de 10000 nodos, á que lle segue `QPLib` cun 60 %, e `MINLP`, con algo menos do 50 %.

Na Figura 4.29, podemos observar que para as familias asociadas á batería `EVRIM` practicamente todos os problemas se resolven en menos de 10000 iteracións, ao igual que ocorre coas familias `st`, `po` e `other` de `MINLP`. Os problemas máis volumétricos encóntranse nesta libraría, o que concorda co que estamos a observar na gráfica anterior.

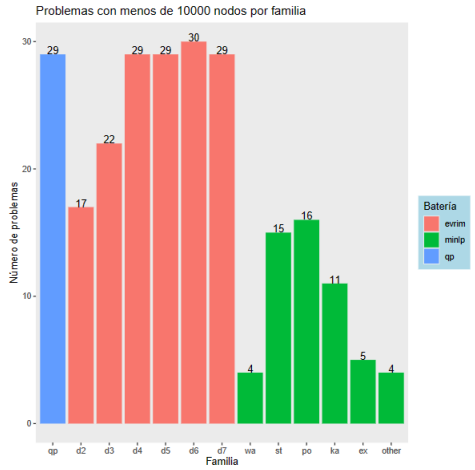


Figura 4.29: Número de problemas con menos de 10000 nodos por familia no conxunto orixinal.

Así, seleccionando todas as observacións asociadas a aqueles problemas con menos de 10000 nodos, e un número fixo k para os que superan este límite, obtemos unha nova mostra, `mostra_10000`, composta por 1267034 observacións, onde 314087 son de `EVRIM`, 677684 de `MINLP` e 275263 de `QP`, representando o 24.79 %, o 53.49 % e o 21.72 % dos rexistros totais, respectivamente. Deste xeito, temos 10 veces máis observacións que no caso anterior para `mostra_500`. Cabe puntualizar que a medida que o número de rexistros considerados aumenta, é dicir, máis alto é o valor de k, máis se parecerán estas porcentaxes ás existentes no conxunto orixinal. Na seguinte imaxe, podemos ver esta mesma distribución porcentual por familia:

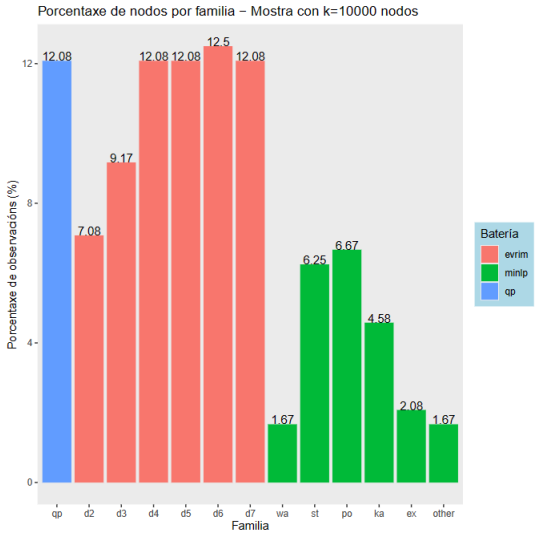


Figura 4.30: Porcentaxe de problemas por familia na mostra xerada para k=10000.

Malia que numericamente a diferenza entre as porcentaxes no conxunto inicial e o actual seguen sendo evidentes, a consecuencia da fixación dun número k de nodos para certos problemas. Vexamos como se comporta a variable resposta neste conxunto e analicemos as súas similitudes respecto do de orixe.

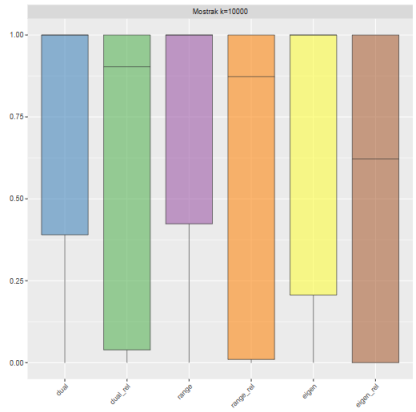


Figura 4.31: Distribución da variable resposta para cada criterio en `mostra_10000`.

Tal e como podemos apreciar na Figura 4.3, a variabilidade nos criterios dual e dos rangos diminuiu, pero non tanto coma na probas anterior con $k = 500$. Observemos tamén que a mediana non acada valores iguais á unidade en todos os criterios, como ocorría na mostra `mostra_500`, pero sí nas versións puras que non se basean en pseudocostes. En calquera caso, toma valores altos, o que supón un indicativo da existencia de empates e tamén de que, cada regra resultará óptima para a metade das observacións. Para a metade restante existe unha ampla variabilidade, e é precisamente nestes problemas onde debemos fixar o noso foco e tentar propoñer un modelo que seleccione o criterio óptimo para estas casuísticas.

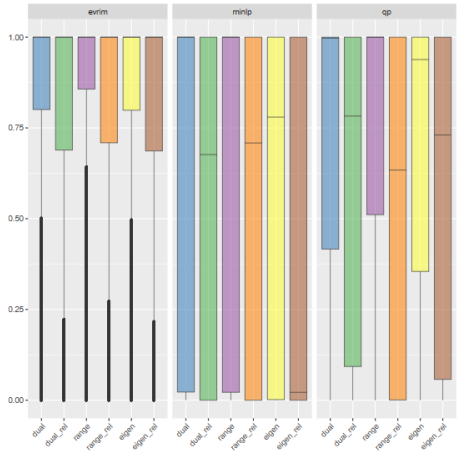


Figura 4.32: Distribución da variable resposta para cada criterio en `mostra_10000` por bateria.

Para a batería `EVRIM`, a distribución é máis ou menos similar á do conxunto orixinal, destacando as versións puras lixeiramente sobre as baseadas en pseudocostes. Ademais, as medianas continúan a tomar valores unitarios, e segue existindo un amplo número de atípicos. É a libraría que máis se asemella en termos de KPI ao conxunto orixinal. De igual xeito, observamos un comportamento similar para `MINLP` e `QP` en `conxunto_orixinal` e `mostra_10000`. En calquera das gráficas, obsérvase unha lixeira diminución na variabilidade, aínda que non resulta tan evidente coma nas probas realizadas para $k = 500$. Ao igual que no caso anterior, omitiremos os gráficos por familia e non nos deteremos a facer maiores consideracións posto que as conclusións baralladas son as mesmas que vimos de referir.

Así, á vista do anterior, podemos concluír que a distribución da variable resposta, aínda que presenta menor variabilidade, é moi similar á do conxunto orixinal.

Tal e como fixemos anteriormente, o seguinte paso é dividir a mostra xerada en dous novos subconxuntos: adestramento e test, formados por 875478 e 391556 observacións, respectivamente. O conxunto de adestramento está composto por 235 problemas, onde 121 pertencen á libraría `EVRIM`, 81 a `MINLP` e 33 a `QP`. No tocante á mostra de test, está formada por 93 problemas, onde 49 pertencen á libraría `EVRIM`, 30 a `MINLP` e 14 a `QP`.

Na seguinte gráfica representaremos a porcentaxe de subproblemas de cada batería en cada unha das mostras, train e test, así como, no total de `mostra_100000`.

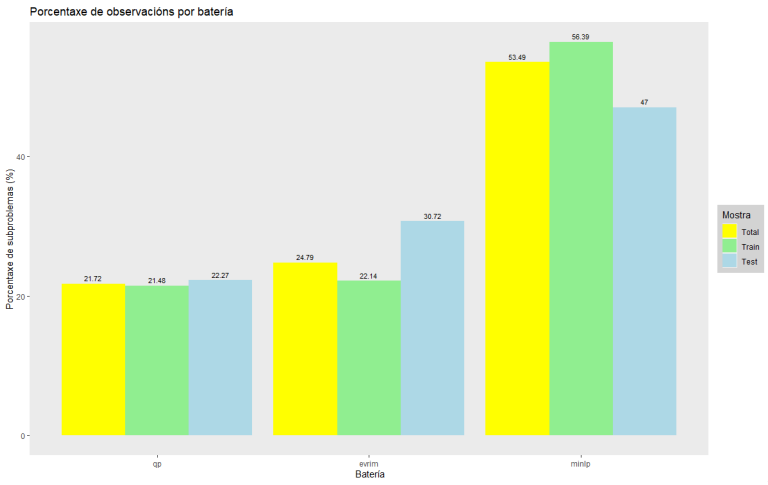


Figura 4.33: Porcentaxe de subproblemas por batería para `mostra_10000`.

Como podemos ver, aumentar o número de nodos de $k = 500$ a 10000 implicou un incremento nas diferenzas destas porcentaxe, especialmente nas baterías `MINLP` e `EVRIM`. Isto faise evidente se comparamos a gráfica anterior coa Figura 4.22.

Baixando o nivel de granularidade do gráfico, podemos analizar en detalle como se comportan estas porcentaxes para cada familia.

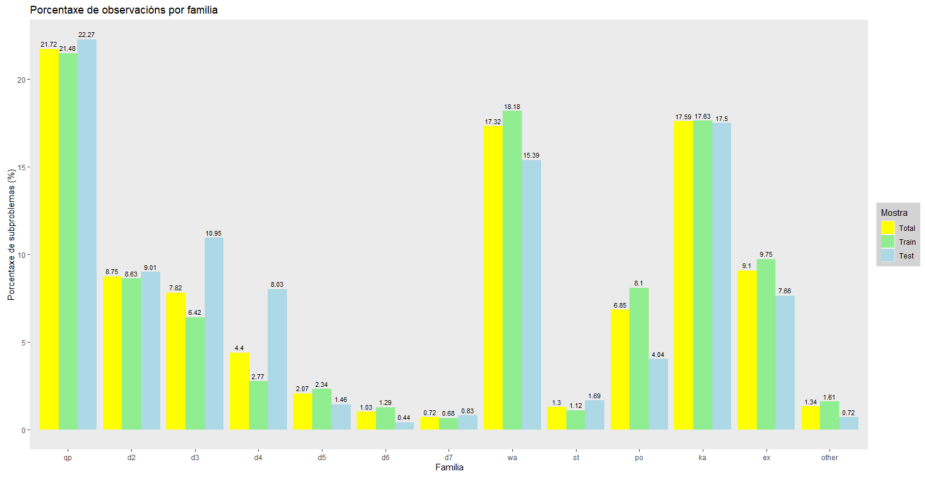


Figura 4.34: Porcentaxe de subproblemas por familia para`mostra_10000`.

Observamos que non todas as familias da batería `EVRIM` presentan as mesmas diferenzas entre adestramento e test. Por exemplo, `d2` ou `d7` son máis ou menos similares, pero isto pode deberse á selección das mostras. O mesmo podemos comentar sobre `MINLP`. En calquera caso, debemos ter en coidado se seguimos aumentando o número de nodos considerado, porque entón, a mostra seleccionada tenderá a semellarse ao conxunto orixinal e voltaremos ás diferenzas de comportamento observadas. Vexamos como se comportan estas mostras respecto da variable resposta.

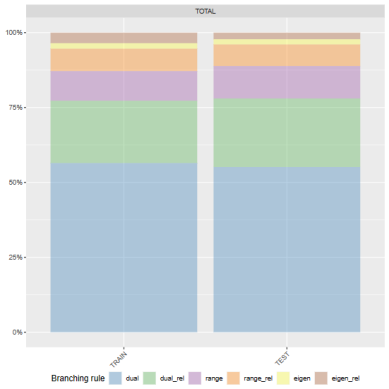


Figura 4.35: Porcentaxe de subproblemas por regra óptima en `mostra_10000`.

Malia que respecto da porxentaxe de observacións as diferenzas entre train e test aumentaron respecto dos outros casos, o comportamento da variable resposta segue a ser semellante, no conxunto global.

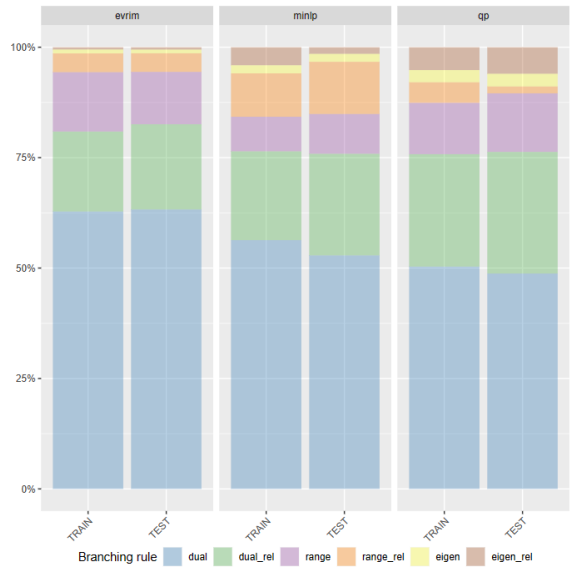


Figura 4.36: Porcentaxe de subproblemas por regra óptima en `mostra_10000` para cada batería.

Por batería a situación non é moi distinta, quizais as maiores discrepancias se observan en `MINLP`, pero encontrámonos dentro dunha marxe de aceptación, tendo en conta o observado na proba da sección anterior.

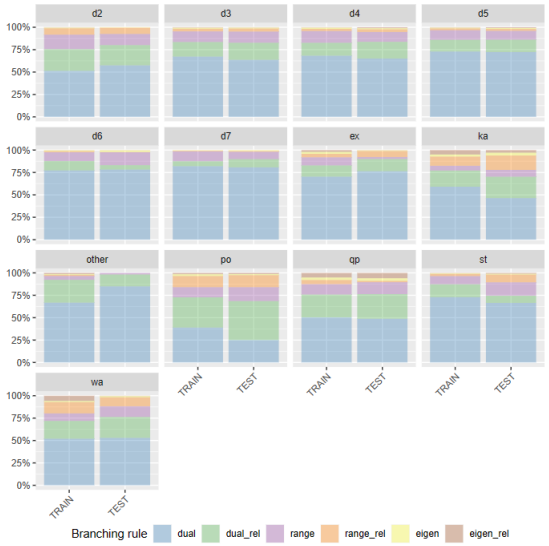
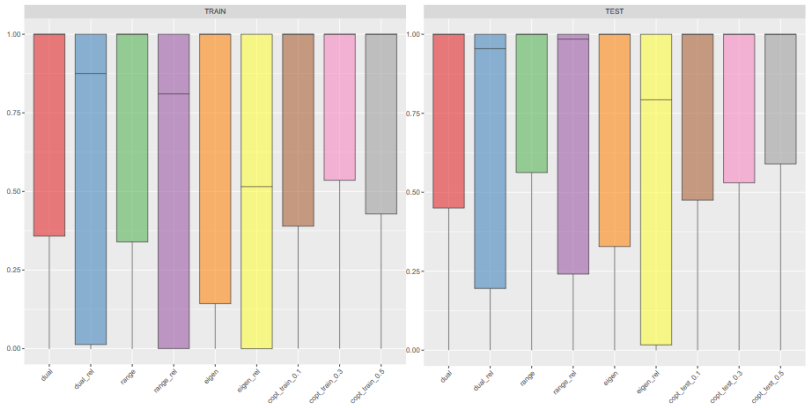


Figura 4.37: Porcentaxe de subproblemas por regra óptima en `mostra_10000` para cada familia.

A nivel de familia hai variedade de comentarios, atopando as maiores diferenzas en `po`, ou `st`. Asumiremos que as mostras son o suficientemente parecidas e a continuación abordaremos o modelado desta información.

O proceso de aprendizaxe realizarase en idénticas condicións que os anteriores, coa única modificación do conxunto de datos empregado para o adestramento. Así mesmo, empregaremos o mesmo tipo de gráficas para analizar os resultados obtidos.



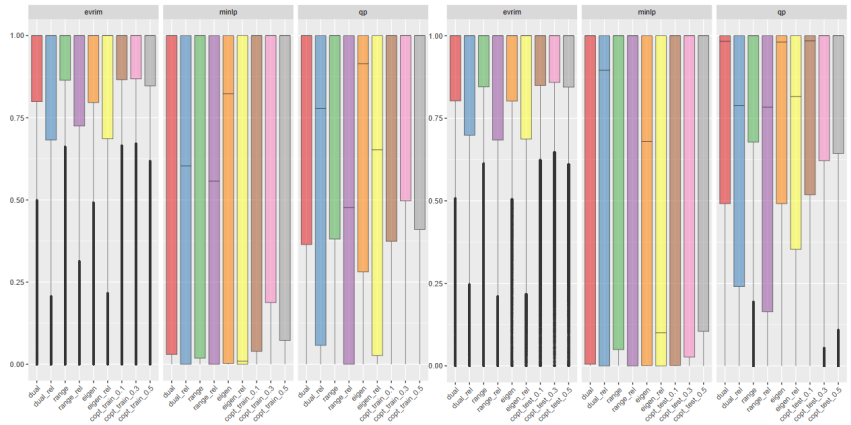
(a) Train (b) Test

Figura 4.38: Resultados do modelo boosting obtido a partir de `mostra_10000`.

A alto nivel, comparando estes gráficos cos obtidos para a mostra con $k = 500$, vemos que a variabilidade aumentou, parecéndose máis ambas submostras ao conxunto orixinal. Ademais, tanto en train coma en test, os criterios que mellor resultado presentan, polo xeral, son o dual e o dos rangos. Fixémonos en que as versións baseadas en pseudocostes, aínda que presentan unha ampla variabilidade, amosan unha mediana cun valor bastante elevado, do que podemos concluír que para a metade das observacións, os valores do KPI empregando ditas regras de ramificación son altos, superiores ao 0.80.

Aquí vemos lixeiras diferenzas entre ambas mostras, pero isto pode deberse a que os problemas son distintos entre ambas e as observacións tamén poden presentar discrepancias, como é de esperar.

No relativo aos resultados dos modelos, podemos ver que o criterio para $\tau = 0.3$ presenta mellores resultados na mostra de train que o resto, pero existe moi pouca diferenza, que incluso pode deberse á selección da mostra, co que non resulta determinante á hora de avaliar o desempeño do proceso de aprendizaxe. En test, polo xeral, os resultados de todos os criterios son mellores que en train, tamén hai que ter en conta que temos menos observacións. Sen embargo, aquí o modelo que mellor resultado presenta é aquel que foi adestrado con $\tau = 0.5$, e ademais, cun comportamento moi similar ao criterio dos rangos, o que non fai máis que ratificar que o modelo non está aprendendo correctamente, e os resultados están moi influenciados pola mostra seleccionada. En calquera caso, analizaremos está información máis en detalle, baixando ao nivel de batería.



(a) Train (b) Test

Figura 4.39: Resultados do modelo boosting obtido a partir de `mostra_10000` por batería.

Ao igual que ocorría anteriormente, na batería `EVRIM` o desempeño das regras de ramificación é polo xeral, bo, aínda que vemos un alto número de atípicos. Tendo en conta isto, non é de estrañar que os modelos adestrados presenten tamén un bo desempeño, de feito, para calquera valor de τ, vemos que o KPI se move entre 0.8 e 1. Isto non é unha evidencia clara de que o modelo actúe correctamente, senón que é un indicativo de que intenta imitar algunha das regras consideradas, o unha combinación de varias, e como estas presentan bos resultados, o modelo aparentemente tamén. Esta conclusión aplica tanto en train coma en test. `MINLP` é unha libraría composta por problemas de tipoloxía moi diversa con comportamentos moi distintos, o que se reflexa na ampla variabilidade observada. Nesta libraría é precisamente onde un bo modelo resultaría de gran utilidade. Na mostra de adestramento semella que o modelo con $\tau = 0.3$ é quizais o que mellor actúa, aínda que para nada con bos resultados (presenta unha variabilidade demasiado elevada), incluso mellor que os criterios prefixados. Comparando como se comporta na mostra de test, podemos concluír que algo está fallando no proceso de aprendizaxe. Non nos deteremos a facer maiores consideracións sobre a batería `QP`, posto que as conclusións son as mesmas. Ademais, ao igual que coa `mostra_500`, omitiremos as gráficas por familia.

Facendo recompilación de todo o observado até este punto, podemos concluír que algo está fallando no proceso de aprendizaxe. Conseguimos igualar o comportamento das mostras de train e test, en termos de porcentaxe de observacións e de KPI. Ademais, aumentamos o número de observacións de tal maneira que a falta de información non fose un limitante neste sentido, sen embargo, non se apreciaron maiores diferenzas que coas mostras iniciais, o que fai que nos preguntemos que pode estar ocorrendo. Por unha banda, é probable que as variables consideradas non estean aportando a información suficiente para discernir un criterio óptimo. Por outra, como xa comentamos en anteriores ocasións, no conxunto inicial, existen un alto número de observacións para os que non existe un

único criterio óptimo, é dicir, prodúcense empates entre as regras de ramificación, especialmente nos primeiros nodos. Unha hipótese que debemos considerar é que estes empates estean entorpecendo o proceso de aprendizaxe, enmascarando un conxunto enormemente desbalanceado neste sentido. Na seguinte sección analizaremos este último aspecto.

4.2. Análise de empates

Executando unha sinxela consulta en R, vemos que das 7856719 observacións totais, só 1806078 presentan un único criterio óptimo, o resto dos casos teñen algún empate.

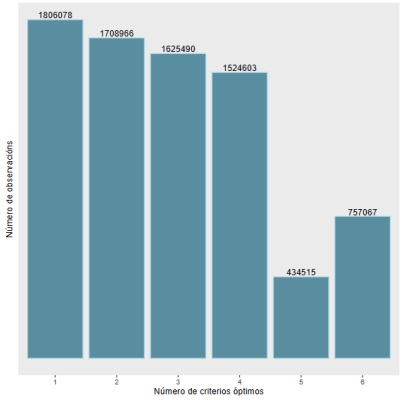


Figura 4.40: Número de observacións por número de criterios óptimo.

No gráfico anterior podemos ver para cantas observacións existe un único criterio óptimo, dous, tres e así sucesivamente ata seis, que é o total de regras consideradas. Recordemos que aqueles problemas para os que todos os criterios eran óptimos foron eliminados das mostras anteriores porque non aportaban ao proceso de aprendizaxe.

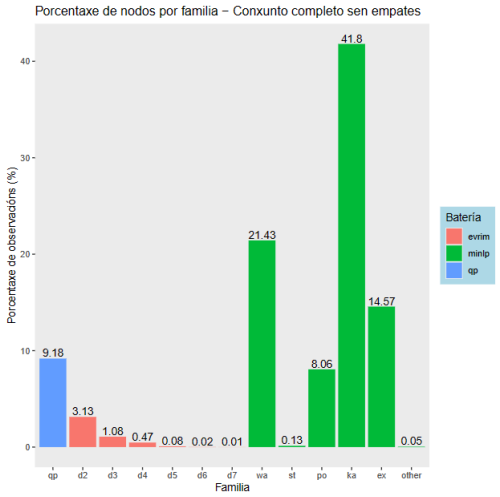


Figura 4.41: Porcentaxe de subproblemas por familia en `datos_orixinal` sen empates.

Centrémonos neste novo conxunto de observacións suprimindo os posibles empates. Está formado

por 274 problemas distintos, onde 136 pertencen a `EVRIM`, 91 a `MINLP` e 47 a `QP`. É dicir, para as dúas primeiras baterías, hai problemas para os que ningún dos seus nodos presenta un criterio óptimo determinante, senón que todos eles teñen algún empate. Analizando estes problemas, observamos que son instancias cun número de nodos bastante baixo, a excepción de tres problemas da familia `ex` (`ex5_2_1`, `ex5_2_2` e `ex5_2_3`), e, como xa comentamos, nas primeiras iteracións do algoritmo é máis probable que se produzan empates entre os criterios. Das observacións que o compoñen, 86466 pertencen á batería `EVRIM`, representando un 4.79 % dos datos, 1553798 a `MINLP`, cun 86.03 % e o restante, 165814, a `QP`, 9.18 % das observacións totais. Na Figura 4.41, podemos ver a distribución do número de nodos por familia en termos de porcentaxe, observando que a diferenza é mínima respecto da Figura 4.2.

Tal e como vimos comentando, neste conxunto de datos, para cada subproblema existe un único criterio óptimo, é dicir, cuxo KPI asociado é idéntico á unidade. Tendo isto en conta, pode ser interesante analizar se hai un criterio que destaque en canto a desempeño respecto do resto, ou existe bastante variabilidade. Na seguinte gráfica podemos ver como o criterio de centralidade de vector propio, parece ser o que peor desempeño presenta, a grandes rasgos. Ademais, observamos que destaca o criterio dos rangos baseado en pseudocostes, que resulta óptimo, pero polo xeral, existe bastante variabilidade.

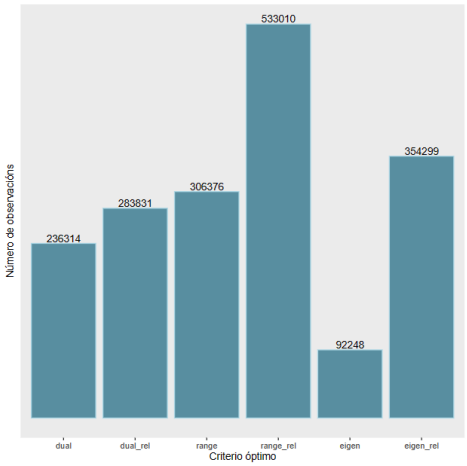


Figura 4.42: Número de observacións por criterio óptimo.

O ideal para avaliar se os empates están afectando ao proceso de aprendizaxe é seleccionar estas observacións, e adestrar de novo o modelo nas mesmas condicións para analizar cómo se comporta. Dado que o conxunto de datos é bastante grande e de cara a axilizar os cálculos no computador do alumno, tomouse como decisión fixar novamente un número fixo $k = 500$ de tal maneira que, para aqueles problemas con menos subproblemas que esta cantidade, se seleccionasen todas as observacións e para aqueles que a superasen, seleccionariase aleatoriamente un número fixo k. Así, dos 274 problemas, 151 presentan menos de 500 nodos, onde 15 pertencen a libraría `QP`, 22 a `MINLP` e 114 a `EVRIM`. De seguido, amósase este mesmo desglose por familia:

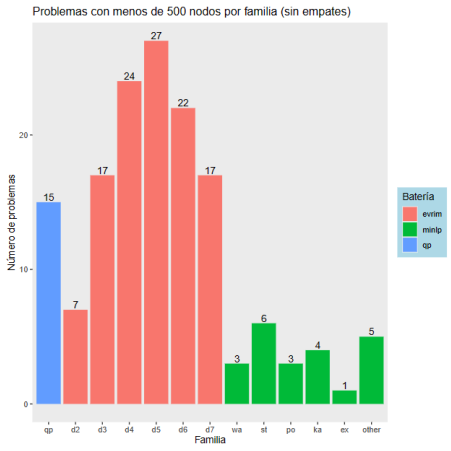
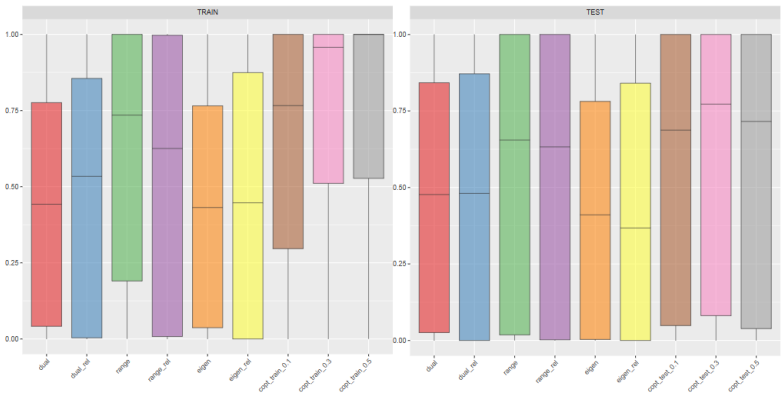


Figura 4.43: Número de problemas con menos de 500 nodos en `datos_orixinal_sen_empates`.

A partir desta mostra, `mostra_500_sen_empates`, particionaremos o espazo en dous subconxuntos: adestramento e test, coa mesma porcentaxe de observacións considerada nas anteriores seccións, 70 % e 30 %, respectivamente, fixando semente para garantir a reproducibilidade dos resultados. As mostras xeradas presentan 59823 e 16059 observacións. O conxunto de adestramento está formado por 58 problemas, onde 30 pertencen a `EVRIM`, 20 a `MINLP` e 10 a `QP`. Así mesmo, en test, temos 214 problemas, onde 106 pertencen a `EVRIM`, 71 a `MINLP` e 37 a `QP`.

Chama a atención que, por primeira vez no desenvolvemento deste traballo, hai máis problemas na mostra de test que na de adestramento, isto é debido a que os problemas que conforman a mostra de test teñen un número de observacións baixo, en contraste cos seleccionados para train. Poderíamos tratar de solventalo tomando outras mostras, pero son moito máis abundantes este tipo de problemas que os que contan cun alto número de instancias. En calquera caso, asumiremos este feito e executaremos o proceso de aprendizaxe nas mesmas condicións descritas en anteriores ocasións. Nos seguintes gráficos de caixas e bigores preséntanse os resultados obtidos para facilitar a súa interpretación. Comezaremos analizando o comportamento dos modelos adestrados en comparación cos valores da variable resposta para cada criterio de xeito global, e posteriormente iremos reducindo a granularidade a batería e familia.



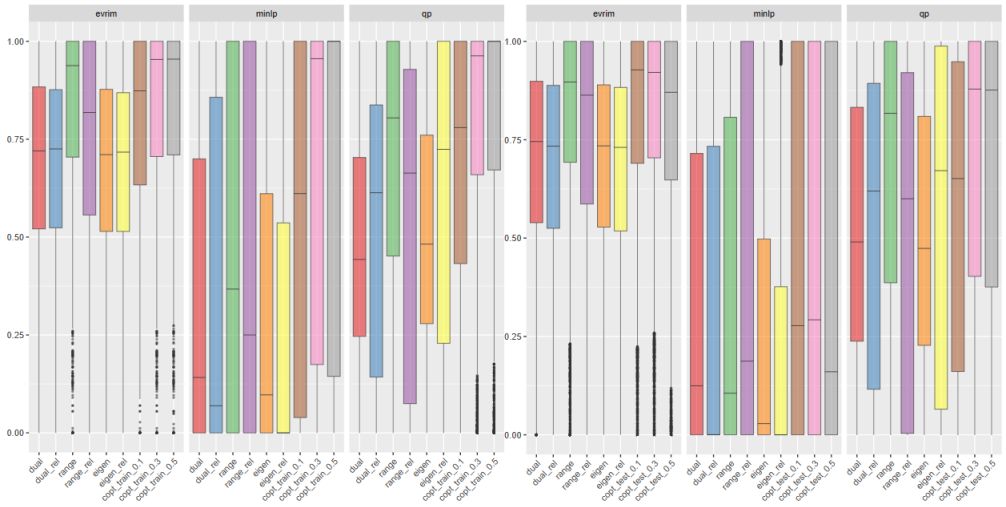
(a) Train (b) Test

Figura 4.44: Resultados dos modelos boosting obtidos a partir de `mostra_500_sen_empates`.

En termos de KPI, ambas mostras presentan un comportamento similar en canto a variabilidade para todos criterios. Obsérvase tamén unha lixeira asimetría negativa en gran parte dos boxplots, tanto en train coma en test, agás nos criterios dos rangos e rangos baseados en pseudocostes. Ademais, as medianas non acadan valores demasiado próximos á unidade. Observemos o que ocorre cos modelos adestrados. Se nos focalizamos na mostra de adestramento, podemos chegar a pensar que, restrinxindo a aprendizaxe a aqueles problemas onde existe un único criterio óptimo, o proceso mellora, de feito, os modelos obtidos para $\tau = 0.3$ e $\tau = 0.5$ resultan esperanzadores. Por exemplo, o primeiro caso, presenta unha mediana cun valor próximo a 0.90, do que inducimos que para a metade dos subproblemas, o modelo propón un criterio con resultados de KPI entre 0.9 e 1, é dicir, bastante bos. Sen embargo para o 50 % restante, os valores poden chegar a variar bastante. Sería esperable que na mostra de test ocorre algo similar, pero o feito é que o que visualizamos é moi distinto. Os modelos presentan unha elevada variabilidade. O primeiro cuartil toma un valor moito menor, o que conleva uns amplos rangos intercuartílicos, reflectindo unha enorme variabilidade nas predicións dos modelos, independentemente do valor de τ considerado.

Novamente, repetindo a mesma análise por batería, Figura 4.45, o comportamento de ambas mostras en termos de KPI non é moi distinto, do que en principio, podemos concluír como aspecto positivo que estamos a seleccionar a partición de forma adecuada.

Centrémonos nos modelos. Na batería `EVRIM` o criterio que semella presentar, polo xeral, mellor resultado é o dos rangos, onde o rango intercuartílico non é demasiado amplo e os primeiro e terceiro cuartil toman, polo xeral, valores elevados. Os modelos semellan imitar esta tendencia con medianas bastante altas. Na Figura 4.47, represéntase o número de observacións para os que cada criterio é óptimo discriminando por batería para a mostra de train. Botando man do seguinte gráfico, ratificamos que efectivamente o modelo está predicindo maioritariamente o criterio dos rangos e dos rangos baseado en pseudocostes.



(a) Train (b) Test

Figura 4.45: Resultados dos modelos boosting obtidos a partir de `mostra_500_sen_empates` por batería.

Na batería `MINLP` as mostras tamén presentan un comportamento máis ou menos similar, e ademais, cunha enorme variabilidade que se traslada aos modelos, que non logran capturar a tendencia dos datos para ningunha das dúas, o que pode deberse a gran cantidade de problemas de diferente tipoloxía que abarca este conxunto.

En canto aos problemas cadráticos, os modelos con $\tau = 0.3$ e $\tau = 0.5$ parecen ser unha boa opción na mostra de train, pero en test, presentan un desempeño moi similar ao dos rangos, aínda que iso sí,

as medianas toman valores próximos a 0.80.

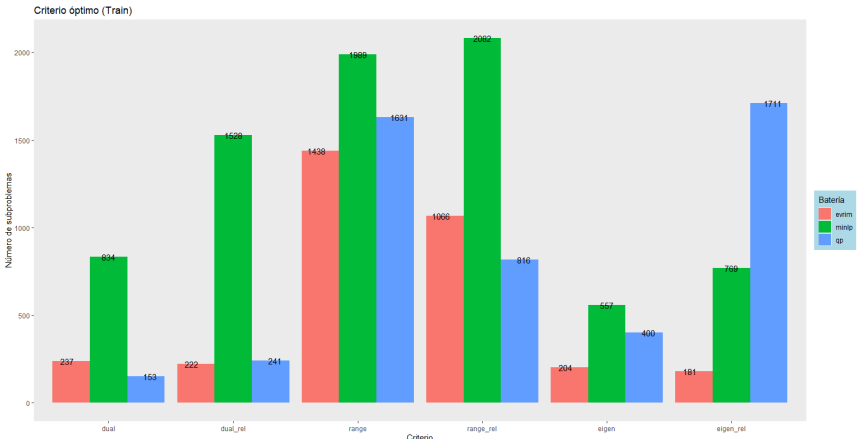


Figura 4.46: Número de subproblemas por criterio óptimo en train de `mostra_500_sen_empates`.

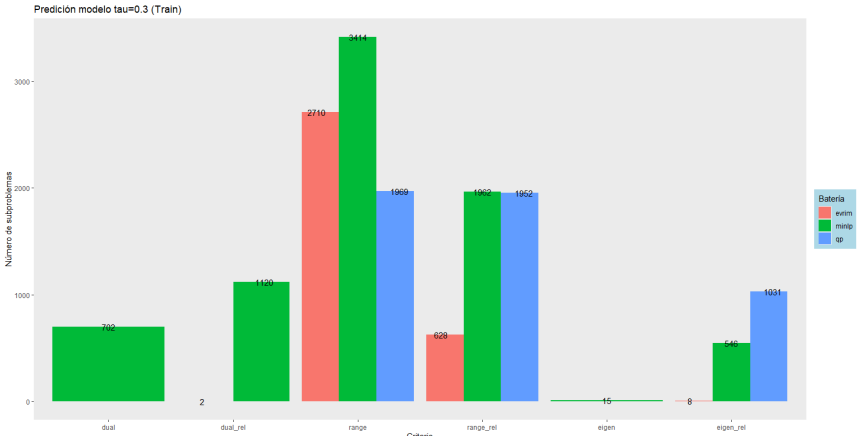


Figura 4.47: Número de subproblemas por criterio óptimo en train predito polos modelos.

Omitiremos os gráficos por familia por non aportar máis información que a que vimos de comentar. Neste conxunto, `mostra_500_sen_empates`, no que non temos ningunha observación que presente empates en optimalidade para varias regras de ramificación, podemos representar nun diagrama de barras o número de observacións para as que cada criterio resulta óptimo, Figura 4.46. Neste caso só representamos as observacións da mostra de train para tratar de comparalas coas respectivas predicións que devolve o modelo para $\tau = 0.3$, tratando de facer visibles as problemáticas detectadas, Figura 4.47. Tal e como podemos ver, para os problemas da libraría `EVRIM` notamos que o modelo tan só predice como óptimo os criterios maioritarios, ignorando practicamente aqueles que resultan óptimos para un número menor de observacións. Algo similar ocorre cos problemas cadráticos. Na batería `MINLP`, a variabilidade é tal que o modelo sí que asigna predicións a todas as regras, pero tendo en conta o observado nos boxplots, nun elevado número de casos, erra nestas asignacións.

Capítulo 5

Conclusións

Inicialmente, esta memoria foi concibida como una ampliación do traballo presentado na referencia Rodríguez-Ballesteros (2022), onde se analizan os resultados obtidos con varios modelos de aprendizaxe estatística, cuxo obxectivo era a elección dun criterio ramificador óptimo, inmutable durante as distintas iteracións do algoritmo RLT para un determinado problema de programación polinómica. O enfoque proposto preséntase como unha extensión natural do traballo anterior, particularizando a decisión da regra de ramificación en cada nodo.

En programación enteira e mixta, a aplicación de técnicas de aprendizaxe para a escolla do mellor criterio en cada iteración dun determinado algoritmo de ramificación e limitación é un campo de investigación aberto e bastante recente, e resulta aínda máis novo no ámbito da optimización polinómica. É de destacar a revisión bibliográfica levada a cabo, como por exemplo, Marcos-Álvarez et. al. (2017), Khalil et. al. (2016), adquirindo cultura sobre esta nova temática, resultando especialmente importante á hora de definir as variables explicativas.

O conxunto de datos considerado neste traballo presenta unha estructura diferente ao empregado en Rodríguez-Ballesteros (2022), destacando especialmente pola súa enorme volumetría. É importante notar que na memoria referenciada anteriormente, ningunha das variables do conxunto de datos almacenaba información sobre o número de nodos dos problemas, equiparando instancias cuxa cantidade de nodos é moi dispar. Pola contra, na presente memoria trátase de analizar en detalle cada relaxación linear, almacenando información en variables de carácter local e representando unha observación no conxunto de datos asociada a cada nodo. Este procedemento deu lugar a un alto número de rexistros, o que implicou elevados tempos de cómputo, facendo necesaria a busca de alternativas para o tratamento dos datos e a exploración diferentes metodoloxías para a extracción de submostras, minimizando a perda de información.

Os modelos Boosting baseados en regresión cuantil obtidos ao longo deste traballo supoñen unha primeira aproximación para a selección dun criterio de ramificación óptimo nodo a nodo de xeito automático. Esta é unha tarefa, cando menos, ambiciosa, e pouco explorada até o momento, polo que podemos considerar os resultados obtidos como un medio de análise sobre o proceso de aprendizaxe, que nos permitiu detectar aspectos que deberían ser tratados con maior coidado en propostas futuras.

Tal e como comentabamos, o conxunto inicial está formado por un elevado número de observacións, das cales, aproximadamente o 75 % presenta algún empate entre criterios óptimos, do que podemos concluír que é habitual que non exista unha única regra óptima, senón varias. Esta situación orixina un enorme desbalanceo entre observacións sen empates, nas que hai un criterio óptimo claro, e observacións con empates, que poden ser dous ou varios, dependendo do problema considerado. Este feito pode estar a influír na aprendizaxe e é algo que nos modelos propostos non se está a ter en conta.

Outro aspecto que debemos ter en conta é que os problemas considerados presentan un número moi diferente de observacións. Tratamos de solucionar isto de maneira artificiosa fixando un número k de observacións para minimizar as discrepancias entre adestramento e test, pero unha solución máis rigorosa e adecuada, sería empregar unha regresión con pesos para tratar de solucionar esta

problemática.

Ademais, non esquezamos que estamos a formular a selección dun criterio óptimo como un problema de regresión en base a un KPI definido por nós mesmos, sendo realmente un problema de clasificación encuberto. A variable resposta, tal e como está definida, só recompila información sobre a diminución das cotas inferiores ao ramificar, sen embargo, existen outros aspectos que se poderían ter en conta, por exemplo, o tempo de resolución de cada subproblema.

Así mesmo, as variables explicativas foron seleccionadas en base a un estudo de correlacións dous a dous, suprimindo de cada par aquela que presentaba maior relación linear co resto de variables. Sen embargo, tendo en conta o alto número de atípicos do noso conxunto, quizais sería prudente considerar métodos máis robustos.

En próximos pasos é requirido analizar con detemento posibles solucións ás limitacións que vimos de expoñer, e tamén explorar novos modelos que sexan quen de adaptarse ás mesmas. Os avances neste campo poden supoñer unha mellora notoria no rendemento do algoritmo RLT, e en xeral, na resolución de problemas de diversa tipoloxía, dende problemas sobre aforro de recursos naturais até os do sector enerxético, establecendo ademais unha estreita relación entre a estatística clásica e a computacional, resaltando a importancia de ambas de xeito individual e tamén en estreita colaboración.

Tras a revisión que vimos de facer dos resultados obtidos durante o proceso de aprendizaxe ao longo deste traballo, cómpre destacar certos aspectos relevantes do mesmo. Primeiramente, realízase unha presentación con certo grao de detalle dos problemas de programación polinómica, así como, unha revisión do algoritmo RLT, ilustrando cun exemplo práctico os pasos do mesmo, facilitando a comprensión da súa evolución. Ademais, revísanse diferentes criterios de ramificación e propóñense novas aproximacións baseadas no *Pseudocost Branching*, orixinalmente ideado para problemas de optimización enteira e mixta, co que, por esta parte, levouse a cabo unha importante tarefa de busca e revisión de información. Por outra banda, focalizándonos na aprendizaxe, fíxose unha recolección de diversas variables de carácter dinámico, ampliando o número de variables explicativas do problema respecto de Rodríguez-Ballesteros (2022). Como xa comentamos con anterioridade, desta quenda o conxunto de datos orixinado foi considerablemente máis complexo, ocasionando importantes problemas computacionais, o que nos levou á busca de novos métodos para o seu tratamento.

Finalmente e a modo conclusión, malia que como vimos de expoñer, os resultados non foron de todo satisfactorios, o traballo realizado durante estes meses, resumido nesta memoria, supón unha formación adicional nos problemas de programación polinómica e no algoritmo RLT, permitindo ademais identificar puntos claves sobre os que continuar a traballar para a proposta dun modelo predictivo adecuado, tendo en conta ademais que é un problema aberto dentro da optimización matemática e do que non temos constancia que abordado até o momento.

Apéndice A

Variables explicativas

Neste apéndice describiremos as variables explicativas empregadas para levar a cabo o adestramento dos modelos presentados no Capítulo 4, cuxa selección foi levada a cabo a partir dun estudo de correlacións. Podemos distinguir dous tipos de variables: estáticas e dinámicas. As variables estáticas son variables relativas ao problema de programación polinómica que queremos resolver, é dicir, non aportan información relativa a cada nodo en particular resultante de aplicar o algoritmo RLT. Polo tanto, a información destas variables para todos os nodos dun mesmo problema será idéntica. As variables consideradas neste traballo son un subconxunto das empregadas en Rodríguez-Ballesteros (2022). Por outra banda, as variables dinámicas proporcionan información de carácter local para cada subproblema, co que está servirá para diferenciar as distintas instancias asociadas a un mesmo problema. Combinando ambos tipos de variables, poderemos caracterizar os subproblemas orixinados na execución do algoritmo e trataremos de seleccionar o criterio de ramificación máis adecuado de entre todos os expostos na sección 3.1 do Capítulo 3.

A.1. Variables estáticas

Nome	Descrición
`degree`	Grao do problema de optimización polinómica.
`n.constr`	Número de restricións do problema de optimización polinómica.
`n.equality`	Número de restricións de igualdade entre o número de restricións totais.
`n.quadratic`	Número de restricións cadráticas entre o número de restricións totais.
`n.varcons`	Número de variables do problema entre o número de restricións máis un, este última correspondente á función obxectivo.
`range.var`	Varianza dos rangos das variables.
`n.monom`	Número de monomios do problema.

Táboa A.1: Variables estáticas.

Nome	Descrición
`diff.quadratic`	Número de monomios cadráticos distintos entre o número total de monomios.
`coef.average`	Promedio dos coeficientes presentes no problema.
`density`	Densidade do problema, isto é, o número de monomios distintos presentes no problema entre o número total de monomios que poderían aparecer no poblema tendo en conta o grao e o número de variables do mesmo.
`i.density`	Densidade do grafo intersección variable-variable.
`i.transit`	Transitividade do grafo intersección variable-variable.
`a.mod`	Modularidade do grafo intersección monomio-restricción.
`a.tree`	Ancho da árbore no grafo intersección monomio-restricción.

Táboa A.1: Variables estáticas.

A.2. Variables dinámicas

Nome	Descrición
`lb_improv_last_10_tree`	Mellora da cota inferior nos últimos 10 % de nodos anteriores do total da árbore.
`lb_improv_last_20_branch`	Mellora da cota inferior nos últimos 20 % de nodos anteriores pertencentes á mesma rama que o nodo actual.
`lb_improv_last_100_branch`	Mellora da cota inferior en todos os nodos anteriores pertencentes á mesma rama que o nodo actual.
`avg_var_red_last_10_branch`	Promedio da redución do rango da variable ramificadora no 10 % de nodos anteriores pertencentes á mesma rama que o nodo actual.
`avg_var_red_last_100_branch`	Promedio da redución do rango da variable ramificadora no total de nodos anteriores pertencentes á mesma rama que o nodo actual.

Táboa A.2: Variables dinámicas.

Nome	Descrición
avg_var_red_last_10_tree	Promedio da redución do rango da variable ramificadora nos últimos 10 % de nodos anteriores do total da árbore.
avg_var_red_last_20_tree	Promedio da redución do rango da variable ramificadora nos últimos 20 % de nodos anteriores do total da árbore.
max_var_red_last_10_branch	Máximo da redución do rango da variable ramificadora nos últimos 10 % de nodos anteriores pertencentes á mesma rama que o nodo actual.
max_var_red_last_10_tree	Máximo da redución do rango da variable ramificadora nos últimos 10 % de nodos anteriores do total da árbore.
ram_most_used_var_last_10_branch	Porcentaxe de ramificacións da variable máis empregada nos últimos 10 % de nodos anteriores pertencentes á mesma rama que o nodo actual.
ram_most_used_var_last_10_tree	Porcentaxe de ramificacións da variable máis empregada nos últimos 10 % de nodos anteriores do total da árbore.
var_num_last_10_branch	Porcentaxe do total de variables empregadas nos últimos 10 % de nodos anteriores pertencentes á mesma rama que o nodo actual.
var_LBs_queue_nodes	Coeficiente de variación do vector de cotas inferiores dos nodos que están en cola para ser resoltos.
var_depths_queue_nodes	Coeficiente de variación do vector de profundidades dos nodos que están en cola para ser resoltos.
var_of_sum_of_violations	Coeficiente de variación do vector de sumas das violacións das distintas variables, sen considerar pesos.
var_variable_relative_ranges	Coeficiente de variación dos rangos relativos das variables, isto é, os rangos do nodo actual divididos entre os rangos do nodo orixinal.
rel_diff_sum_viol_top1_var_vs_top10	Diferenza relativa entre a suma das violacións da variable máis empregada e da décima variable máis empregada.

Táboa A.2: Variables dinámicas.

Nome	Descrición
`current_depth`	Profundidade do nodo.
`var_viol_zero`	Porcentaxe de variables que presentan unha violación RLT maior que cero, $\theta_j \geq 0$.
`var_dual_values`	Coeficiente de variación dos valores duais das restricións nas que polo menos existe unha variable con violación RLT non nula, $\theta_j \neq 0$.
`avg_eigen_10_var`	Promedio da centralidade de vector propio do 10 % de variables con maior violación RLT, θ_j.
`var_pseudocosts_right`	Coeficiente de variación dos pseudocostes asociados a cada variable, Ψ_j^+.
`var_rel_left`	Coeficiente de variación da fiabilidade dos pseudocostes.
`b.ineq_constr`	Porcentaxe de restricións vinculantes de desigualdade.
`b.linear`	Porcentaxe de restricións vinculantes lineares.
`b.nonlinear`	Porcentaxe de restricións vinculantes non lineais.

Táboa A.2: Variables dinámicas.

Bibliografía

[1] Achterberg, T. et al. (2005). Branching rules revisited. *Operations Research Letters*, 33:42-54.

[2] Aittomäki, T. and Koivunen, V. (2009). Beampattern Optimization by Minimization of Quartic Polynomial. 2009 IEEE/SP 15th Workshop on Statistical Signal Processing, 437-440.

[3] Belotti, P. et al. (2009). Branching and bounds tightening techniques for non-convex MINLP. *Optimization Methods and Software*, 24:597?634.

[4] Breiman, L. et al. (1984). Classification and Regression Trees. Routledge.

[5] Bussieck, M. R., Drud, A. S. and Meeraus, A. (2003). Minlplib-a collection of test models for mixedinteger nonlinear programming. *INFORMS J. on Computing*, 15:114-119.

[6] Dalkiran, E. and Sherali, H. D. (2016). Rlt-pos: Reformulation-linearization technique-based optimization software for solving polynomial programming problems. *Mathematical Programming Computation, 8:337-375.*

[7] Dietterich, T. G. (2000). Ensemble Methods in Machine Learning. Multiple Classifier Systems. MCS 2000. Lecture Notes in Computer Science, vol. 1857. Springer.

[8] Fernández, R. et. al. (2021). Aprendizaje estadístico. Recurso online accedido en outubro de 2022: https://rubenfcasal.github.io/aprendizaje_estadistico/aprendizaje_estadistico.pdf.

[9] Floudas, C. A. et al. (1999). Handbook of Test Problems in Local and Global Optimization. Noncovex Optimization and Its Applications, vol. 33, Kluwer Academic Publishers, 1999. Journal of Global Optimization, vol. 16, 299?300 (2000).

[10] Friedman, J. H. (2002). Stochastic Gradient Boosting. Computational Statistics and Data Analysis, vol. 38, pp. 367-378.

[11] Friedman, J. H., Hastie, T., Tibshirani, R. (2000). Additive Logistic Regression: a Statistical View of Boosting. Annals of Statistics, vol. 28, pp. 337-407.

[12] Furini, F. et al. (2018). Qplib: a library of quadratic programming instances. *Mathematical Programming Computation*, 1:237-265.

[13] Ghaddar, B. et al. (2016). Optimal Power Flow as a Polynomial Optimization Problem. EEE Transactions on Power Systems, vol. 31, no. 1, pp. 539-546.

[14] Godsil, C. and Royle, G. F. (2001), Algebraic Graph Theory. *Graduate Texts in Mathematics*, Springer, Book 207.

[15] González-Díaz, J. et al. (2020). Global optimization for bilevel portfolio design: Economic insights from the Dow Jones index. *The International Journal of Management Science*, vol. 102, 102353.

[16] González-Rodríguez, B. (2017). Introducción a la programación no lineal. Master's thesis, Universidade de Santiago de Compostela.

[17] González-Rodríguez, B. et. al. (2020). Computational advances in polynomial optimization: Raposa, a freely available global solver.

[18] Khalil, E. B. et al. (2016). Learning to Branch in Mixed Integer Programming. Thirtieth AAAI Conference on Artificial Intelligence, vol. 30, no. 1.:

[19] Kimizuka M. et al. (2019). Solving pooling problems with time discretization by LP and SOCP relaxations and rescheduling methods. *Journal of Global Optimization*, Springer, vol. 75, pp. 631-654.

[20] Kriegler, B. and Berk, R. (2010). Small area estimation of the homeless in Los Angeles: An application of cost-sensitive stochastic gradient boosting. The Annals of Applied Statistics, vol. 4, No. 3, pp. 1234-1255.

[21] Koenker, R. (2005). Quantile Regression. Cambridge Univ. Press, New York.

[22] Koenker, R. and Basett, G. (1978). Regression Quantiles. Econometrica, vol. 46, no. 1, pp. 33-50.

[23] Latora, V., Nicosia, V. and Russo, G. (2017). Complex networks: principles, methods and applications. Cambridge University Press.

[24] Marcos Álvarez, A. et. al. (2017). A Machine Learning-Based Approximation of Strong Branching. Informs Journal on Computing, vol. 29.

[25] Rodríguez-Ballesteros, S. (2022) Técnicas de aprendizaje supervisado aplicadas a la resolución de problemas de optimización. Traballo Fin de Mestrado, Universidade de Santiago de Compostela.

[26] Schapire, R. E. (1990). The Strength of Weak Learnability. Machine Learning, vol. 5, no. 2, pp. 197-227.

[27] Sherali, H. D. and Tuncbilek, C. H. (1991). A Global Optimization Algorithm for Polynomial Programming Problems Using a Reformulation-Linearization Technique. *Journal of Global Optimization 2: 101-112*.

[28] Tawarmalani, M. and Sahinidis, N. V. (2002). Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications. Noncovex Optimization and Its Applications, vol. 65, Kluwer Academic Publishers, 2002.

[29] Zeng, M. and Hongwei, F. (2017). Applications of polynomial optimization in financial risk investment. IOP Conference Series: Materials Science and Engineering, vol. 242, 012076.