Bosques RFW para regresión

Álvar Arnaiz-González, José F. Diez-Pastor, César García-Osorio, and Juan J. Rodríguez

Universidad de Burgos, España
{alvarag,jfdpastor,cgosorio,jjrodriguez}@ubu.es

Resumen Los ensembles son métodos de aprendizaje que basan su funcionamiento en la combinación de diferentes modelos base. La diversidad en los ensembles es un aspecto fundamental que condiciona su funcionamiento. Random Feature Weigths ($\mathcal{RFW}$) fue propuesto como un método de construcción de ensembles de árboles de clasificación en el que la diversidad es introducida en cada árbol mediante un peso aleatorio asociado a cada atributo, estos pesos varían de un árbol a otro dentro del ensemble. En este artículo se adapta la idea de $\mathcal{RFW}$ a árboles de regresión y se hace una comparación contra otros métodos de construcción de ensembles de regresión: Bagging, Random Forest, Iterated Bagging, Random Subspaces y AdaBoost.R2 obteniendo unos resultados competitivos.

Keywords: ensembles, regresión, random feature weights, bagging

1 Introducción

Los ensembles son estrategias de aprendizaje que basan su funcionamiento en construir varios modelos y combinar sus resultados para realizar la predicción [17]. Cada uno de los modelos que componen el ensemble se denominan modelos base. La idea que subyace tras estos métodos es que la combinación de diversos clasificadores o regresores es capaz de mejorar la precisión que se obtendría al utilizar cada uno de ellos de forma independiente, sobre todo en modelos inestables[1] como los árboles de decisión o las redes neuronales [14]. Un aspecto importante que se busca en los ensembles es la diversidad, puesto que la combinación de múltiples modelos muy similares no aporta beneficios frente a utilizar únicamente el método base. De acuerdo a la fórmula de descomposición del error en ensembles [11], se puede mejorar la precisión del mismo de dos maneras: mejorando la precisión de los regresores base o aumentando su diversidad. Tradicionalmente, la diversidad se ha conseguido de dos maneras: utilizando algoritmos de aprendizaje distintos en cada modelo (ensembles heterogéneos) o utilizando un mismo modelo pero entrenando cada uno con un conjunto de datos distinto

[1] Se denominan inestables a aquellos métodos que pueden producir modelos muy diferentes con pequeños cambios en el conjunto de entrenamiento o en los parámetros del método.

(ensembles homogéneos) [18]. Dentro de los ensembles homogéneos existen principalmente dos enfoques: modificar el conjunto de entrenamiento para cada uno de los modelos base como hacen Bagging [2], Random Subspaces [12]..., o inyectar aleatoriedad en el modelo como hacen Random Forest [3], GRASP Forest [7] o Random Feature Weights [16] entre otros.

Otro aspecto a tener en cuenta en el manejo de ensembles, es el tiempo de entrenamiento y de test así como la memoria requerida. Ambos dependen directamente del modelo base que se esté utilizando, este es el motivo por el cual no suelen construirse ensembles de métodos pesados o lentos.

Uno de los métodos homogéneos de construcción de ensembles más conocido es Bagging (abreviatura de *bootstrap aggregating*) [2]. Pese a su simplicidad, diversos estudios han demostrado que en clasificación normalmente obtiene modelos mas precisos que los clasificadores que los componen [21]. Su funcionamiento se basa en la construcción de modelos sobre subconjuntos aleatorios del conjunto de datos original y su posterior combinación. Los subconjuntos son aleatoriamente generados con reemplazamiento por lo que una determinada instancia puede estar repetida en algunos subconjuntos o no estar presente.

Iterated Bagging [4] construye ensembles de Bagging de manera iterativa, de ahí su nombre. En la primera iteración se construye un ensemble de Bagging (sin ningún cambio). El siguiente ensemble se entrena sobre los residuos, calculados como la diferencia entre el valor predicho por el ensemble inicial y el valor real. En lo sucesivo, cada ensemble se entrena sobre los residuos de la predicción combinada de los ensembles previos. El error de las predicciones no se calcula sobre el conjunto original, sino con una estimación *out-of-bag*, es decir, la predicción se realiza utilizando solamente aquellos árboles que no hayan sido entrenados con la instancia correspondiente.

Random Subspaces [12], al contrario que Bagging, utiliza todo el conjunto de entrenamiento para construir cada miembro pero selecciona aleatoriamente un subconjunto de atributos. Variando el porcentaje de atributos a seleccionar se consigue una mayor o menor diversidad, aunque este porcentaje también afecta a su precisión.

Random Forest [3] combina Bagging y la selección aleatoria de atributos de Random Subspaces. En cada nodo del árbol la división se hace teniendo únicamente en cuenta un subconjunto aleatorio de atributos.

Otra popular familia de ensembles son los basados en Boosting, en este artículo se se ha utilizado AdaBoost.R2 [8] una adaptación de AdaBoost (abreviatura de *Adaptive Boosting*) [10] para regresión. En AdaBoost.R2 el ensemble se construye de manera iterativa, ajustando los pesos de cada instancia en función de la precisión del regresor base construido. Los pesos de aquellas instancias que han sido incorrectamente clasificadas aumentan, mientras que los pesos de las instancias correctamente etiquetadas disminuyen. El peso de cada instancia es actualizado por medio de una función de pérdida aplicada sobre el error y el peso previo de la misma. En el artículo original [8] se definen tres tipos de funciones de pérdida: lineal, cuadrática y exponencial. Si el modelo se construye con *re-weighting*, lo cual suele ser la opción por defecto, la información de los pesos pasa

al regresor base que la utiliza para formular su hipótesis. Si por el contrario se construye con *resampling*, aquellas instancias más difíciles son seleccionadas con mayor probabilidad para entrenar el siguiente modelo base del ensemble [20].

Otro método de construcción de ensembles más reciente es Random Feature Weights ($\mathcal{RFW}$) [16]. Originalmente propuesto para árboles de clasificación basa su funcionamiento en utilizar un peso asociado a cada atributo de modo que la elección de atributos en la construcción del árbol tiene un sesgo que depende de este peso. Los pesos son distintos para cada árbol del ensemble y esto es lo que aporta la diversidad del mismo. En este artículo se investiga el desempeño de este método cuando los árboles que se combinan son árboles de regresión.

El artículo se estructura del siguiente modo: en la Sección 2 se describe el método Random Feature Weights y su adaptación a regresión. La Sección 3 muestra el estudio experimental comparándolo contra otros métodos de construcción de ensembles. Por último, en la Sección 4, se detallan las conclusiones y aspectos relevantes del trabajo.

2 Random feature weights

Los árboles de decisión son métodos comúnmente utilizados en tareas de clasificación y regresión. La predicción se realiza sobre el modelo (árbol) construido, el cual está formado por nodos y ramas. La estructura comienza con un nodo raíz, los nodos interiores se corresponden con una variable de entrada, de tal modo que las instancias se derivan a cada uno de sus nodos hijos dependiendo del valor que éstas tengan en dicha variable. Los nodos que no tienen hijos se denominan hojas y realizan una predicción utilizando la información de la variable a predecir de las instancias que acaban en dicho nodo. La decisión de qué variable utilizar para dividir un nodo, se realiza empleando una función de evaluación (f). Diversas funciones de evaluación pueden ser utilizadas: coeficiente de Gini [15], ganancia de información (utilizado en C4.5 [19])... El método de construcción divide el nodo tratando de maximizar dicha función, ya que ello implica que ha seleccionado un buen atributo para la división.

El método Random Feature Weights ($\mathcal{RFW}$) [16] modifica la función de evaluación f multiplicándola por un peso aleatorio w_i para cada atributo. De este modo, cada nodo del árbol tiene un mismo conjunto de pesos aleatorios $\mathbf{w}$, pero cada árbol del ensemble tiene un conjunto distinto de pesos lo que incrementa la diversidad. Se puede considerar una generalización del método Random Subspaces en el que los pesos pueden valer cualquier valor en el intervalo $[0, 1]$ y no solo los dos extremos.

En el artículo se presenta la adaptación del método $\mathcal{RFW}$ para regresión $\mathcal{RFW}$Reg (ver Algoritmo 1). La función de evaluación viene determinada por el algoritmo de construcción de árboles a utilizar, el tamaño del ensemble determina el número de regresores base que se construirán y el exponente determina la magnitud del cambio aleatorio que se introduce en la construcción, es decir, a mayor exponente más diversos serán cada uno de los árboles del ensemble.

Algoritmo 1: Método de construcción de ensembles de árboles de regresión $\mathcal{RFW}$Reg

Input: Conjunto de entrenamiento D_T, $f(a, D)$ función de evaluación, L tamaño del ensemble, p exponente
Output: Ensemble de árboles de regresión

```
for l = 1...L do
    foreach atributo a_i ∈ D_T do
        u ← valor aleatorio en {0, 1}
        w_i ← u^p
    Entrenar un árbol de regresión utilizando D_T y la función de
    evaluación f_W(a_i, D) = w_i f(a_i, D)
    Añadir el árbol de regresión al ensemble
return Ensemble construido
```

En el artículo se ha adaptado el algoritmo REPTree [9] con el pesado aleatorio de atributos de $\mathcal{RFW}$. El Algoritmo 2 muestra la función recursiva que genera el árbol de regresión donde:

- Nodo es una estructura compuesta por:
  - hijo[0] e hijo[1]: sendas referencias a estructuras Nodo con los descendientes del nodo actual. Solo utilizado en los nodos no hoja.
  - punto.division: valor del umbral por el que se dividen las instancias en el nodo. Solo utilizado en los nodos no hoja.
  - etiqueta: valor a predecir por el nodo hoja.
- modelo es una estructura compuesta por:
  - varGanancia: valor de la ganancia (entropía) de la división.
  - puntoDivisión: valor del atributo por el que dividir el atributo
- REPTreeSplit es una función que devuelve una estructura de tipo modelo con la información de la división para un atributo determinado.

3 Configuración experimental

La experimentación se ha realizado en Weka [22] modificando el algoritmo de construcción de árboles REPTree [9] para adaptarlo al pesado aleatorio de $\mathcal{RFW}$ como se ha explicado en la sección previa. Los ensembles se han construido utilizando estos árboles con Bagging.

El algoritmo propuesto se ha enfrentado a un variado número de métodos de construcción de ensembles que se listan a continuación. Todos ellos han sido configurados con un tamaño máximo 100 y se ha utilizado como modelo base el regresor REPTree con poda y sin ella, excepto Random Forest que internamente utiliza Random Trees. Solo se han modificado los parámetros indicados a continuación, para el resto se han mantenido los valores por defecto.

Algoritmo 2: Función Entrenar$\mathcal{RFW}$Reg

Input: Conjunto de entrenamiento D_T, conjunto atributos Atributos, conjunto de pesos aleatorios $\mathbf{u}$, exponente p
Output: Árbol de regresión

```
if Atributos esta vacío o |D_T| < mín. permitido por rama then
    Nodo.etiqueta ← media de los valores de los ejemplos de D_T
    return Nodo
else
    foreach atributo a_j ∈ Atributos do
        modelo[j] ← REPTreeSplit (D_T, a_j)
        modelo[j].varGanancia ← modelo[j].varGanancia · u_j^p
    g ← arg máx(modelo.varGanancia)
    Nodo.puntoDivisión ← modelo[g].puntoDivisión
    D_l ← {x ∈ D_T | x_{i,j_g} ≤ Nodo.puntoDivisión}
    D_r ← {x ∈ D_T | x_{i,j_g} > Nodo.puntoDivisión}
    Nodo.hijo[0] ← EntrenarRFWReg (D_l, Atributos, u, p)
    Nodo.hijo[1] ← EntrenarRFWReg (D_r, Atributos, u, p)
return Nodo
```

- Bagging.
- AdaBoost.R2. Ejecutado con *reweighting* (W) y *resampling* (S). Se ha variado la función de pérdida: lineal, cuadrática y exponencial.
- Random Subspaces. Dos configuraciones: utilizando el 50 % y el 75 % de los atributos del conjunto original.
- Random Forest.
- Iterated Bagging. Dos configuraciones: 10×10 y 5×20.

Se ha realizado validación cruzada con 10 grupos (*folds*) sobre los 61 conjuntos de datos utilizados en [6] y disponibles en el repositorio UCI [1] y la colección de Luis Torgo[2] (ver Tabla 1). La aleatoriedad de las particiones y métodos afecta a los resultados. No se ha repetido la validación cruzada ya que aumentaría significativamente el tiempo de la experimentación. El uso de un número elevado de conjuntos de datos reduce los efectos de la aleatoriedad.

Para evaluar los resultados se ha utilizado el error cuadrático medio (root mean squared error o RMSE) y el error cuadrático relativo (root relative squared error o RRSE), calculado como el error cuadrático medio dividido entre el error de la predicción de un regresor que devuelve la media y multiplicándolo por cien. Valores cercanos o superiores a cien no son deseables ya que indican un comportamiento peor que un regresor que base su predicción en la media. Por consiguiente, cuanto más cercano a cero se encuentre el RRSE más preciso será el regresor.

[2] `http://www.dcc.fc.up.pt/~ltorgo/Regression/DataSets.html`

Tabla 1. Conjuntos de datos utilizados en la experimentación. Se detalla su nombre, número de atributos desglosados en numéricos (N) y discretos (D) y número de instancias.

Dataset	**#Atrib.**		**#Inst.**	**Dataset**	**#Atrib.**		**#Inst.**
	N	**D**			**N**	**D**	
detroit	13	0	13	hungarian	6	7	294
longley	6	0	16	cholesterol	6	7	303
gascons	4	0	27	cleveland	6	7	303
schlvote	4	1	38	auto-mpg	4	3	398
bolts	7	0	40	pbc	10	8	418
diabetes-num	2	0	43	housing	12	1	506
vineyard	3	0	52	meta	19	2	528
elusage	1	1	55	sensory	0	11	576
pollution	15	0	60	strike	5	1	625
mbagrade	1	1	61	stock	9	0	950
sleep	7	0	62	quake	3	0	2 178
pyrimidines	27	0	74	abalone	7	1	4 177
auto93.names	16	6	93	delta-ailerons	5	0	7 129
baskball	4	0	96	cpu-act	21	0	8 192
cloud	4	2	108	cpu-small	12	0	8 192
fruitfly	2	2	125	kin8nm	8	0	8 192
echo-months	6	3	130	puma32H	32	0	8 192
veteran	3	4	137	puma8NH	8	0	8 192
fishcatch	5	2	158	bank32nh	32	0	8 193
autoPrice	15	0	159	bank8FM	8	0	8 193
servo	0	4	167	delta-elevators	6	0	9 517
triazines	60	0	186	ailerons	40	0	13 750
lowbwt	2	7	189	pole	48	0	15 000
wisconsin	32	0	194	elevators	18	6	16 599
pharynx	1	10	195	cal-housing	8	0	20 640
pw-linear	10	0	200	house-16H	16	0	22 784
auto-horse	17	8	205	house-8L	8	0	22 784
cpu	6	1	209	2dplanes	10	0	40 768
machine-cpu	6	0	209	friedman	10	0	40 768
bodyfat	14	0	256	mv	7	3	40 768
breast-tumor	1	8	286				

Sobre los resultados del RMSE y del RRSE, de los 61 conjuntos de datos, se ha calculado el ranking medio [5] del siguiente modo: se ordenan los resultados de tal manera que al mejor se le asigna el valor uno, al siguiente el dos y así sucesivamente. En caso de empate se suman los valores y se divide entre el número de métodos que habían empatado. Una vez calculado el ranking para cada conjunto de datos, se hace la media para cada método. El mejor ranking es aquel cercano a uno.

En la tabla 2 se muestra el ranking medio y el resultado del procedimiento de Hochberg [13] calculado sobre el RRSE (a) y el RMSE (b). La línea discontinua indica a partir de qué método la diferencia con el primero es significativa al 95 % de confianza según Hochberg. Se observa como el $\mathcal{RFW}$Reg sin poda es el mejor método en ambas tablas, aunque su diferencia con respecto a los siguientes 6 métodos (utilizando RRSE, Tabla 2 (a)) o 7 métodos (utilizando RMSE, Tabla 2 (b)) no es significativa al 95 % de confianza.

En el artículo donde se presentó el $\mathcal{RFW}$ para clasificación se evaluó el comportamiento del método para los exponentes 1, 2, 3 y 4. En regresión el efecto del exponente no es conocido, por lo que se ha variado desde 0.25 hasta 6 en pasos de 0.25. En la Figura 1 se muestra el ranking medio para el $\mathcal{RFW}$Reg con (P) y sin poda (U). En la gráfica se observa que el método que mejores resultados ofrece es el ensemble de árboles sin poda con un exponente de 4 mientras que para árboles con poda el mejor exponente es 5.

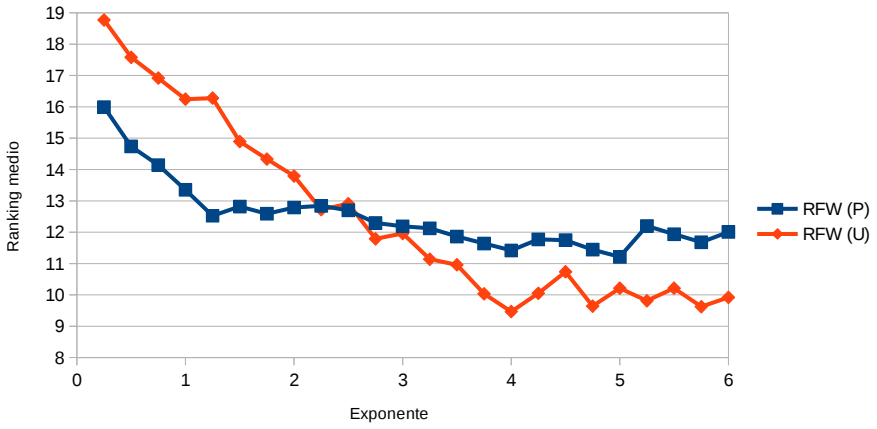

Figura 1. Ranking medio del $\mathcal{RFW}$Reg a medida que varía el exponente con REPTree como regresor base con (P) y sin poda (U).

Tabla 2. Ranking medio sobre el RRSE (a) y RMSE (b) de los ensembles. La (U) indica que el regresor base del ensemble no tiene poda y la (P) que ha sido podado. La línea discontinua indica a partir de que punto las diferencias respecto al primer método son significativas al 95 % de confianza.

(a)

	Algoritmo	**Ranking**
1	$\mathcal{RFW}$Reg E 4 (U)	7.4098
2	IteratedBagging 5×20 (P)	8.4262
3	RandomForest	9.2787
4	$\mathcal{RFW}$Reg E 5 (P)	9.4262
5	Bagging (U)	10.4508
6	AdaBoost.R2 S. Exp. (P)	10.0574
7	AdaBoost.R2 S. Lin. (P)	10.2951
8	IteratedBagging 5×20 (U)	11.2787
9	Bagging (P)	11.4262
10	IteratedBagging 10×10 (P)	12.1639
11	AdaBoost.R2 S. Sq. (P)	12.6721
12	AdaBoost.R2 S. Exp. (U)	13.1066
13	AdaBoost.R2 S. Lin. (U)	13.6721
14	RandomSubspaces 75 % (P)	13.6967
15	AdaBoost.R2 S. Sq. (U)	14.3033
16	IteratedBagging 10×10 (U)	14.7131
17	AdaBoost.R2 W. Lin. (P)	14.7951
18	AdaBoost.R2 W. Sq. (U)	14.9344
19	RandomSubspaces 50 % (U)	15.1148
20	RandomSubspaces 50 % (P)	15.6066
21	AdaBoost.R2 W. Lin. (U)	15.7131
22	AdaBoost.R2 W. Exp. (P)	16.0902
23	RandomSubspaces 75 %(U)	16.6967
24	AdaBoost.R2 W. Exp. (U)	16.8279
25	AdaBoost.R2 W. Sq. (P)	16.8443

(b)

	Algoritmo	**Ranking**
1	$\mathcal{RFW}$Reg E 4 (U)	8.2459
2	IteratedBagging 5×20 (P)	9.4016
3	RandomForest	9.5000
4	AdaBoost.R2 S Exp. (P)	10.5246
5	Bagging (U)	10.8197
6	$\mathcal{RFW}$Reg E 5 (P)	11.0574
7	AdaBoost.R2 S Lin. (P)	11.2295
8	IteratedBagging 5×20 (U)	11.3279
9	Bagging (P)	12.4262
10	AdaBoost.R2 S Exp. (U)	12.7213
11	IteratedBagging 10×10 (P)	12.8361
12	AdaBoost.R2 S Sq. (P)	13.3033
13	AdaBoost.R2 S Lin. (U)	13.3525
14	RandomSubspaces 75 % (P)	13.8770
15	IteratedBagging 10×10 (U)	13.9180
16	AdaBoost.R2 S Sq. (U)	13.9836
17	AdaBoost.R2 W Lin. (P)	14.0492
18	RandomSubspaces 50 % (U)	14.2131
19	AdaBoost.R2 W Sq. (U)	14.3115
20	AdaBoost.R2 W Lin. (U)	14.9180
21	AdaBoost.R2 W Exp. (P)	15.3197
22	RandomSubspaces 75 % (U)	15.5246
23	RandomSubspaces 50 % (P)	15.7131
24	AdaBoost.R2 W Exp. (U)	16.0328
25	AdaBoost.R2 W Sq. (P)	16.3934

4 Conclusiones

En el artículo se presenta la adaptación del método de construcción de ensembles $\mathcal{RFW}$ para regresión, el cual en clasificación había demostrado muy buenos resultados. En este artículo se comprueba que, para regresión, el $\mathcal{RFW}$ obtiene también unos resultados competitivos cuando se compara contra: Bagging, Iterated Bagging, AdaBoost.R2, Random Forest y Random Subspaces. En $\mathcal{RFW}$Reg la diversidad se obtiene mediante la modificación aleatoria de la función de evaluación, lo que combinado con Bagging hace que sea relativamente inmediato de implementar y adaptar a cualquier método de construcción de árboles existente.

La aleatoriedad que se consigue en $\mathcal{RFW}$ se controla mediante un exponente, en este artículo se ha estudiado también cuál es la influencia de dicho exponente en problemas de regresión.

Al igual que Random Subspaces, se puede utilizar con métodos que no son árboles, como trabajo futuro sería interesante investigar la posibilidad de utilizar la idea del incremento de diversidad a través del peso aleatorio en los atributos en otros modelos no necesariamente basados en la construcción de árboles.

Agradecimientos

Este trabajo ha sido financiado por el Ministerio de Economía y Competitividad, proyecto TIN 2011-24046.

Referencias

1. Arthur Asuncion and David Newman. UCI machine learning repository, 2007.
2. Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
3. Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
4. Leo Breiman. Using iterated bagging to debias regressions. *Machine Learning*, 45(3):261–277, 2001.
5. Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30, December 2006.
6. José F Diez-Pastor, César Garcıa-Osorio, and Juan J Rodríguez. GRASP forest for regression: Grasp metaheuristic applied to the construction of ensembles of regression trees. In *14º Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA'11)*, 2011.
7. José F. Diez-Pastor, César García-Osorio, Juan J. Rodríguez, and Andrés Bustillo. Grasp forest: A new ensemble method for trees. In Carlo Sansone, Josef Kittler, and Fabio Roli, editors, *Multiple Classifier Systems*, volume 6713 of *Lecture Notes in Computer Science*, pages 66–75. Springer Berlin Heidelberg, 2011.
8. Harris Drucker. Improving regressors using boosting techniques. In *Proceedings of the Fourteenth International Conference on Machine Learning (ICML 1997), Nashville, Tennessee, USA, July 8-12, 1997*, pages 107–115, 1997.
9. Tapio Elomaa and Matti Kääriäinen. An analysis of reduced error pruning. *CoRR*, abs/1106.0668, 2011.
10. Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Machine Learning, Proceedings of the Thirteenth International Conference (ICML '96), Bari, Italy, July 3-6, 1996*, pages 148–156, 1996.

11. Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural computation*, 4(1):1–58, 1992.
12. Tin Kam Ho. The random subspace method for constructing decision forests. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(8):832–844, Aug 1998.
13. Yosef Hochberg. A sharper bonferroni procedure for multiple tests of significance. *Biometrika*, 75(4):800–802, 1988.
14. Ludmila I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, 2004.
15. Wei-Yin Loh. Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):14–23, 2011.
16. Jesús Maudes, Juan J. Rodríguez, César García-Osorio, and Nicolás García-Pedrajas. Random feature weights for decision tree ensemble construction. *Information Fusion*, 13(1):20 – 30, 2012.
17. João Mendes-Moreira, Carlos Soares, Alípio Mário Jorge, and Jorge Freire De Sousa. Ensemble approaches for regression: A survey. *ACM Comput. Surv.*, 45(1):10:1–10:40, December 2012.
18. Niall Rooney, David W. Patterson, Sarab S. Anand, and Alexey Tsymbal. Random subspacing for regression ensembles. In *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, Miami Beach, Florida, USA*, pages 532–537, 2004.
19. Steven L. Salzberg. C4.5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993. *Machine Learning*, 16(3):235–240, 1994.
20. Chris Seiffert, Taghi M Khoshgoftaar, Jason Van Hulse, and Amri Napolitano. Resampling or reweighting: A comparison of boosting implementations. In *Tools with Artificial Intelligence, 2008. ICTAI'08. 20th IEEE International Conference on*, volume 1, pages 445–451. IEEE, 2008.
21. D.P. Solomatine and D.L. Shrestha. Adaboost.rt: a boosting algorithm for regression problems. In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, volume 2, pages 1163–1168 vol.2, July 2004.
22. Ian H. Witten, Eibe Frank, and Mark A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2011.