Universidad de La Habana
Facultad de Matemática y Computación

Mitigación de sesgos con ensembles y optimización multiobjetivo

Autor:
Jorge Mederos Alvarado

Tutores:

Juan Pablo Consuegra Ayala
Alejandro Piad Morfis

Trabajo de Diploma
presentado en opción al título de
Licenciado en (Matemática o Ciencia de la Computación)

Fecha

github.com/jmederosalvarado/thesis

Dedicación

Agradecimientos

Agradecimientos

Opinión del tutor

Opiniones de los tutores

Resumen

Resumen en español

Abstract

Resumen en inglés

Índice general

Índice de figuras

Ejemplos de código

Introducción

En la actualidad los algoritmos de aprendizaje automático están siendo aplicados en disimilies areas de la vida humana. Es comun encontrarlos aplicados en sistemas de recomendacion de compras, aplicaciones de citas, solicitudes de prestamos, contratacion personal y muchas otras areas. A raiz de ello, ha surgido un creciete interes en estudiar las potencialidades y limitaciones de los modelos de aprendizaje automatico, asi como las posibles implicaciones de confiar ciegamente en sus predicciones.

En particular, su incorporacion a tareas de toma de decisiones de alto riesgo ha dirigido la atencion de muchos investigadores hacia una nueva interrogante: ¿estaran siendo "justos"los algoritmos de aprendizaje automatico al tomar sus decisiones?

En este escenario, ha ganado popularidad el desarrollo de tecnicas para detectar y mitigar los sesgos en colecciones de datos y algoritmos de aprendizaje automatico. Tales herramientas son cruciales para desarrollar sistemas de toma de decisiones mas justos. Los estudios orientados hacia la equidad en algoritmos de aprendizaje automatico se enfocan principalmente en desarrollar tecnicas que consideren tanto la precision como la equidad de los modelos.

Motivación

Un modelo de aprendizaje de maquina se entrena con el objetivo de optimizar una unica metrica, en la mayoria de los casos la precision. Esto significa que los modelos aprenden muy bien los patrones que se presentan en los datos de entrenamiento, incluyendo aquellos patrones que representan sesgos y prejuicios que estan desafortunadamente presente en la sociedad y por ende en los datos recopilados, en algunos casos incluso amplifican estos patrones negativos. Son varias la tecnicas que se han explorado para resolver este problema, algunas se enfocan en un preprocesamiento de los datos para eliminar aquellos elementos que puedan inducir un sesgo en el modelo, otras realizan variacinoes en el metodo de entrenamiento con el mismo objetivo. Sin embargo permanece relativamente poco explorado el uso de tecnicas de optimizacion multiobjetivo que permitan al modelo optimizar hasta encontrar un buen balance entre cuan justo es y cuan preciso.

Otra tecnica que ha demostrado ser de gran utilidad en la prevencion de los sesgos en los modelos de aprendizaje de maquina es la construccion de ensamblados de multiples modelos que maximizan la varianza entre si, por lo que se minimiza el sesgo del ensamblado final.

Problematica

A pesar de que existe AutoGOAL, una biblioteca de AutoML, que permite obtener modelos para resolver problemas arbitrarios utilizando entre otras tecnicas aprendizaje de maquina. No existe una biblioteca o herramienta que permita resolver de principio a fin un problema de clasificacion utilizando aprendizaje de maquina y donde exita alguna garantia de que el modelo aprendido sea justo.

Objetivo general

Proponer una herramienta que permita resolver problemas de clasificacion utilizando aprendizaje de maquina y que permita garantizar que el modelo aprendido sea justo.

Objetivos especifico

- Encontrar modelos que maximicen la varianza para minimizar el sesgo.
- Metodos basados en metaheuristicas para optimizar los modelos utilizando simultaneamente metricas de equidad y precision.
- Explorar adicion de optimizacion multiobjetivo a AutoGOAL para que el modelo aprendido sea justo.
- Metodos basados en la combinacion de diferentes metricas en una sola, para poder aprovechar los multiples metodos de optimizacion que existen.

Capítulo 1

Estado del Arte

1.1. Equidad

No existe en estos momentos una unica definicion ampliamnte aceptada de lo que es *equidad*, sino que diferentes definiciones codifican diferentes caracteristicas que se muestran utiles en diferentes contextos. Incluso, algunas de las definiciones mas comunes presentan conflictos entre si.

A continuacion se presentan algunas de las definiciones mas utilizadas. Sea Y la clasificacion (binaria) correcta, S el atributo protegido, y Y^p la clasificacion obtenida del modelo. Las definiciones mas comunes pueden ser agrupadas en tres categorias: (i) considerando el resultado obtenido dada la clasificacion correcta; (ii) considerando la clasificacion correcta dada la clasificacion obtenida; (iii) considerando solamente el resultado obtenido. Los siguientes son ejemplos de estos tipos de metricas.

- Equal Opportunity (EO) requiere que la proporcion de verdaderos positivos sea la misma en todos los subgrupos: $P(Y^p = 1|Y = 1, S = 0) = P(Y^p = 1|Y = 1, S = 1)$
- Equalized Odds (EOdd) requiere igual proporcion de falsos positivos, ademas de EO
- Statistical Parity (SP) require que las predicciones positivas no se vean afectadas por el valor del atributo protegido, sin tomar en cuenta la clasificacion correcta: $P(Y^p = 1|S = 0) = P(Y^p = 1|S = 1)$

Los tradoff inherentes a cada nocion de fairness han sido estudiados extensamente Dwork y col. 2012; Friedler y col. 2016; Kleinberg 2018. Escoger la definicion correcta para un problema determinado es dificil, y en la practica no puede ser delegado a un agente automatico. En su lugar, una decision humana es preferida para asegurar una decision informada.

1.2. Mitigacion de sesgos

Las tecnicas de mitigacion de sesges pueden ser divididas fundamentalmente en pre-procesamiento, in-procesamiento, post-procesamiento.

Los metodos del primer grupo logran equidad modificando la representacion de los datos, i.e., pre-procesando los datos, y luago adoptan una solucion de machine learning estandard Calmon y col. 2017; Kamiran y Calders 2011; Zemel y col. 2013. Por ejemplo aprender una representacion a partir de resolver un problema de optimzacion con dos objetivos, codificar la informacion preservando la mayor cantidad de informacion posible y ofuscar al mismo tiempo la pertenencia al conjunto de atributos protegidos Zemel y col. 2013. Los metodos de pre-procesamiento son agnosticos al modelo, pero sus hiperparametros, asi como los del modelos de aprendizaje de maquina seleccionado todavia necesitan ser ajustados para mejor rendimiento.

El segundo grupo de la familia consiste de en metodos de in-procesamiento que se aseguran que se cumplan ciertas restricciones de equidad durante el entrenamiento (e.g. Donini y col. 2018; Zafar, Valera, Gomez-Rodriguez y col. 2019; Zafar, Valera, Rogriguez y col. 2017), sin embargo solo es aplicable a una cierta clase de modelos. Por ejemplo el algoritmo aplicado en Donini y col. 2018 solo puede ser aplicado a *kernel machines* (tales como *maquinas de soporte vectorial*), y solo a una unica definicion de equidad (i.e., EO). Aunque metodos de in-procesamiento pueden brindar muy buenos resultados para la clase del modelo que estan diseñados, frecuentemente son dificiles, o a veces imposible de extender para nuevas clases de modelos. Estos metodos tambien pueden introducir nuevos hiper-parametros que podrian requerir conocimiento especificos del dominio y experimentacion.

Las tecnicas de post-procesamiento operan ajustando el umbral de decision de modelos pre-entrenados para eventualmente lograr resultados mas justos respecto a una metrica de equidad dada. El principal problema es que post-procecesar el umbral de decision es inherentemente suboptimo y puede llevar a peores tradeoffs de eficacia y equidad. Lo que es mas, estas tecnicas no son utilizables si la informacion sensible no esta disponible en el momento de realizar las predicciones, lo cual a su vez puede llevar a problemas legales por la utilizacion de informacion sensible para realizar predicciones MacCarthy 2018.

1.3. Metodos de ensamblado

Los metodos de ensamblado estan diseñados para intentar resolver el problema de *low bias/high variance* que muestran la mayoria de los modelos de aprendizaje de maquina, haciendolos apropieados para modelos de clasificacion mas robustos Polikar 2006. Un modelo de ensamblado esta diseñado de muchos modelos con *low bias* cuyas predicciones son combinadas para producir una prediccion final. Se asume fundamen-

talemente que la combinacion de varias predicciones de bajo nivel producira una salida con baja varianza mientras mantiene un low bias. Tener un conjunto diverso de modelos de bajo nivel es una caracteristica fundamental para lograr esto Polikar 2006. Sin embargo, esto requiere, que clasificadores individuales cometan errores en diferentes instancias. La intuicion es que si cada clasificador comete diferentes errores, entonces una combinacion estrategica de estos clasificadores puede reducir el error total. Luego, es necesario lograr que cada clasificador sea lo mas unico posible, particularmente con respecto a ejemplos erroneamente clasificados.

La naturaleza de multiples hipotesis de estos modelos asegura que, si son ajustados lo suficiente, tendran resultados mejores que cualquiera de los modelos individuales en el caso general. Esto les permite tambien estimar el grado de confianza o calidad de las predicciones que producen. Las tecnicas clasicas de ensamblado incluyen *voting* and *weighted voting* Dietterich 2000, boosting Schapire 1990, y bagging Breiman 1996.

En un problema de clasificacion, *voting* produce como salidao la etiqueta que tiene la mayoria de los votos, tratando cada prediccion de los modelos ensamblados como un voto. *Weighted-voting* funciona de forma similar a *voting*, pero cada modelo del ensamblado es asignado un *peso* que indica la importancia de su voto. *Boosting* ejecuta un proceso iterativo donde los modelos son entrenados secuencialmente, cada uno tratando de mejorar su rendimineto en los ejemplos entrenantes donde los modelos anteriores tuvieron peor rendimiento. Durante este proceso, cada sub-modelo es tambien asignado un peso que marca la importancia de su prediccion. *Bagging* entrena cada sub-modelo en una seleccion diferente (con reemplazo) de los ejemplos entrenantes originales. De esta forma, el modelo con una alta varianza deberia producir modelos entrenados con alta diversidad. Alternativamente, *feature bagging* funciona de forma similar, seleccionando un subconjunto de los features en lugar de los ejemplos entrenantes, causando que features correlacionados sean analizados de forma separada en algunos sub-modelos.

Las capacidad de los metodos de ensamblado para construir modelos mas robustos los ha hecho apropiados para multiples aplicaciones. El dominio de la salud es uno de los ejemplos donde estos metodos han sido aplicados con gran exito. Por ejemplo, una aplicacion hibrida de metodos de ensamblado con redes neuronales en un entorno de aprendizaje for reforzamiento es presentada para al prediccion de infeccion de COVID-19 con gran precision W. Jin y col. 2022. De forma similar, un metodo de toma de decision multi-criterio basado en ensemblados es propuesto para la deteccion de COVID-19 a partir del sonido de la tos en pacientes Chowdhury y col. 2022. Existen ejemplos tambien no relacionados a la medicina, por ejemplo, en Livieris y col. 2020 se emplearon estrategias de ensamblado como *ensemble-averaging*, *bagging* y *stacking* con metodologias avanzadas de aprendizaje profundo para predecir los precios, a nivel de hora, de criptomonedas como *Ethereum*, *Bitcoin* y *Ripple*.

Una simple pero poderosa tecnica en el contexto de los metodos de ensamblado

es la llamada *snapshot ensemble* Huang y col. 2017. Esta tecnica genera multiples clasificadores base, entrenando una sola red neuronal mientras la hace converger de forma rapida y repetida a multiples optimos locales y salvando en cada uno de dichos puntos los parametros del modelo. Todas las redes neuronales son entonces ensambladas para producir el clasificador final. Estos *snapshot ensemble* son mas robustos y precisos que las redes individuales dada su naturaleza de ensamblado, a ningun costo adicional de entrenamiento.

1.4. Metodos de Aprendizaje de Maquina Automatico

Aprendizaje de Maquina Automatico (Auto-ML) es el proceso de automatizar la solucino de problemas del mundo real a traves de tecnicas de aprendizaje de maquina. El proceso involucra eliminar la necesidad de humanos expertos en aprendizaje de maquina para seleccionar apropiadamente las caracteristicas, flujos, paradigmas, algoritmos y sus hiperparametros para resolver un problema Dimitrakakis y col. 2019. Las prinicpales ventajas de las tecnologias de AutoML incluyen: (1) reducir el tiempo empleado en resolver problemas bien estudiados; y, (2) eliminar la necesidad de conocimiento experto. Adicionalmente, estas tecnologias tienden a generar soluciones mas simples que a menudo tienen mejor desempeño que soluciones diseñadas por humanos.

Multiples tecnologias han sido propuestas para resolver el problema de Auto-ML. Auto-Weka Thornton y col. 2013 fue una de las primeras soluciones presentadas. Esta está basada en el software Weka Witten y col. 2009, un software construido a partir de varias herramientas de visualizacion y algoritmos para analisis de datos y modelacion predictiva. *Auto-Weka* resuelve el problema de *Auto-ML* como un problema CASH segun definido a continuacion.

Definición 1.1 *Sea* $A = \{A^{(1)}, \ldots, A^{(R)}\}$ *un conjunto de algoritmos, y sea* $\Lambda^{(j)}$ *el dominio de los hiperparametros del algoritmo* $A^{(j)}$*. Sea,* $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ *el conjunto de entrenamiento, el cual es dividido en* K *cross-validation folds de la forma* $\{D^{(1)}_{valid}, \ldots, D^{(K)}_{valid}\}$ *y* $\{D^{(1)}_{train}, \ldots, D^{(K)}_{train}\}$ *tal que* $D^{(i)}_{train} = D \setminus D^{(i)}_{valid}$ *para todo* $i = 1, \ldots, K$*. Finalmente, denotese* $L(A^{(j)}_{\lambda}, D^{(i)}_{train}, D^{(i)}_{valid})$ *la perdida del algoritmo* $A^{(j)}$ *en* $D^{(i)}_{valid}$ *con hiperparametros* λ*. Entonces, el problema de Seleccion de Algoritmo y Optimizacion de Hiperparametros Combinado (CASH) es encontrar la configuracion conjunta de algoritmo e hiperparametros que minimiza la perdida:*

$$A^{\star}, \lambda_{\star} \in argmin_{A^{(j)}, \lambda \in \Lambda^{(j)}} \frac{1}{K} \sum_{i=1}^{K} L(A^{(j)}_{\lambda}, D^{(i)}_{train}, D^{(i)}_{valid}) \tag{1.1}$$

Otros sistemas populares de Auto-ML son Auto-Sklearn Feurer y col. 2015 y Auto-Keras H. Jin y col. 2018. Estos sistemas estan basados en las bibliotecas de aprendizaje de maquina, *Scikit-Learn* Pedregosa y col. 2011 y *Keras Keras* s.f. respectivamente. Ambos sistemas proveen una interfaz para encontrar la mejor arquitectura de aprendizaje de maquina para resolver un problema. Una diferencia fundamental entre ellos es la forma en que sus espacios de busqueda son definidos. Mientras Auto-SkLearn explora espacio de busqueda condicional, i.e., un espacio con algunos hiperparametros condicionados a otros, Auto-Keras realiza una Busqueda de Arqutectura Neuronal (NAS Elsken y col. 2018), la cual implica explorar espacios jerarquicos de complejidad arbitraria.

AutoGOAL Estévez Velarde y col. 2020; Estévez-Velarde y col. 2020 es una de las mas recientes contribucinoes al campo del Auto-ML. AutoGOAL es un sistema que utiliza tecnicas heterogeneas para resolver el problema CASH. La escencia de AutoGOAL radica en su espacio personalizable de pipelines y su pool de algoritmos de busqueda, que son usados para encontrar la mejor configuracion para resolver un problema. Cada pipeline esta definido como un conjunto de algoritmos interconectados que traducen una entrada predefinida a su salida correspondiente. El espacio de pipelines comprende no solo el conjunto de algoritmos, sino tambien sus hiperparametros.

Multiples fuentes de algoritmos estan incluidos en el espacio de AutoGOAL, tales como *Scikit-Learn*Pedregosa y col. 2011, *NLTK*Loper y Bird 2002, *KerasKeras* s.f., y *Pytorch*Paszke y col. 2019. Sin embargo, AutoGOAL carece de la habilidad de combinar multiples end-to-end pipelines para generar una solucion. Esta limitacion puede ser superada con la utilizacion de ensamblados.

1.5. Optimizacion Multiobjetivo

1.6. Discusion

Capítulo 2

Propuesta

2.1. Descripcion general

El sistema toma como entrada un dataset $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ y una funcion de perdida L y una o varias metricas de equidad $F_1, F_2, \ldots, F_n$. El objetivo del sistema es producir un modelo de clasificacion que es a la vez efectivo segun L y justo segun $F_1, F_2, \ldots, F_n$. El sistema consiste en dos fases fundamentales. La primera es responsable de generar una coleccionde modelos, cada uno llamado modelo base. Esta coleccion es construida optimizando una funcion de perdida L_1, mientras se asegura *diversidad* a lo largo de toda la poblacion. La segunda fase es responsable de producir un modelo de clasificacion. Este modelo es generado ensamblando la coleccion de modelos base de forma tal que optimice su efectividad segun L_2, a la vez que es lo mas *justo* posible segun $F_1, F_2, \ldots, F_n$. En el contexto de esta tesis se asume $L = L_1 = L_2$. Las secciones 2.2 y 2.4 abordan con mas detalles la primera y segunda fase, respectivamente.

2.2. Generacion de modelos base

En esta fase al sistema se le da la tarea de generar N modelos para ajustar D de acuerdo a la perdida L.

La Definicion 1.1 se modifica para buscar una coleccion de modelos en lugar de un solo modelo, sujeto a una metrica de diversidad δ. Esto es, se desea encontrar una coleccion de modelos base (modelos que optimicen la efectividad en el dataset D de acuerdo a la funcion de perdida L) mientras garantiza algunas diferencias entre sus hipotesis utilizando la metrica δ. Asegurar diversidad en la coleccion de modelos base es importante porque los metodos de ensemble no son capaces de mojorar su rendimiento si todos los modelos base tienen exactamente la misma hipotesis, i.e.,

todos realizan las mismas predicciones.

El procedimiento aplicado para generar la coleccion de modelos base esta resumido por la funcion *GenerateBaseModels* 2.1. El espacio de algoritmos y hiperparametros es explorado utilizando una estrategia de busqueda pre-seleccionada. Todo esto es capturado por la funcion **explore**. Luego de evaluar las arquitecturas generadas y estimar la diversidad entre los modelos actualmente seleccionados y la nueva generacion de modelos, la coleccion de modelos base es actualizada para ajustarse a su capacidad N. Todo esto es capturado por la funcion **reselect**.

(2.1)

Para explorar inteligentemente el espacio de algoritmos y hiperparametros, i.e., para resolver el problema *CASH* modificado, se utiliza la implementacion de *Probabilistic Grammatical Evolution Search* presente en AutoGOAL. AutoGOAL se refiere a los modelos que construye como pipelines, dado que cada uno de ellos esta formado por algoritmos interconectados. La busqueda comienza con una estrategia de muestreo aleatorio, pero segun evalua mas pipelines, modifica el modelo de muestreo probabilista para que pipelines similares a los mejores encontrados hasta el momento, sean generados con mayor frecuencia. El espacio de algoritmos y hiperparametros utilizados es el utilizado por defecto en AutoGOAL, el cual incluye varios algoritmos clasicos de aprendizaje de maquina presentes en las diferentes bibliotecas utilizadas por AutoGOAL.

Para reseleccionar la coleccion de modelos base, i.e. la coleccion de pipelines de AutoGOAL, un enfoque greedy es utilizado y estudiado. La funcion **reselect** 2.2 resume la estrategia propuesta. El algoritmo siempre incluye el modelo que mejor se desempeña de acuerdo a L en la seleccion. Cada iteracion siguiente añade el modelo no todavia seleccionado, que maximiza la diversidad respecto a todos los modelos anteriormente seleccionados. El enfoque greedy no garantiza que la coleccion final logre la mejor posible diversidad respecto a δ. La precision tampoco es tomada en cuenta, excepto para seleccionar el modelo de mejor desempeño.

(2.2)

Tres metodos *reselect* de referencia fueron implementaods para realizar comparaciones entre las metricas de diversidad *disagreement* y *double-fault*, las cuales son presentadas en la seccion 2.3.

- Shuffle: La coleccion de modelso base es construida barajeando aleatoriamente la selecciona ctual de modelos base y los nuevos encontrados. Los primeros N modelos luego de barajear son seleccionados para la siguiente generacion.

- Arbitrary: La coleccion de modelos base es construida de la misma forma que con la estrategia *shuffle*, pero el modelo de mejor rendimiento siempre es incluido en la coleccion de modelos base seleccionados.
- Best: La coleccion de modelos base es construida a partir de seleccionar los modelos de mejor rendimiento entre los previamente seleccionados y los recien encontrados.

A continuacion, la seccion 2.3 provee algunos detalles acerca de las metricas de diversidad estudiadas en este trabajo.

2.3. Metricas de diversidad

Dos metricas fueron implementadas para estimar la diversidad de una coleccion dada de modelos base. Ambas de ellas precomputan una matriz de clasificaciones incorrectas, la cual es usada entonces para computar una metrica que aporta informacion sobre la diversidad entre los modelos base dos a dos. La matriz de clasificaciones incorrectas se construye de la siguiente manera.

$$M_{i,j} = \begin{cases} 1 & \text{si el modelo j correctamente clasifica el ejemplo i } (D_valid) \\ -1 & \text{otherwise} \end{cases} \tag{2.3}$$

Las siguientes metricas son computadas entre pares de modelos base para estimar cuand diferentes son sus hipotesis, y por tanto la diversidad de la collecion incluyendo a ambos a la vez.

Disagreement. Esta mide la frecuencia con la cual uno de los mdelos falla cuando el otro no lo hace, y viceversa. Mientras mas alto el valor de la metrica, mas diferentes son los modelos.

$$disagreement(m^{(a)}, m^{(b)}) = \frac{|\{M_{i,a} \neq Mi, b | s^{(i)} \in D^{(\star)}_{valid}\}|}{|D^{(\star)}_{valid}|} \tag{2.4}$$

Double Fault. Esta mide cuan a menudo ambos modelos fallan a la vez. Mientras mas alta esta medida mayor la diferencia entre ambos.

$$double - fault(m^{(a)}, m^{(b)}) = 1 - \frac{|\{M_{i,a} = Mi, b = -1 | s^{(i)} \in D^{(\star)}_{valid}\}|}{|D^{(\star)}_{valid}|} \tag{2.5}$$

2.4. Ensamblado inteligente de modelos justos

Capítulo 3

Detalles de Implementación y Experimentos

Conclusiones

Conclusiones

Recomendaciones

Recomendaciones

Bibliografía

Breiman, L. (1996). Bagging predictors. *Machine learning*, *24*(2), 123-140 (vid. pág. 5).

Calmon, F., Wei, D., Vinzamuri, B., Natesan Ramamurthy, K. & Varshney, K. R. (2017). Optimized Pre-Processing for Discrimination Prevention. En I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan & R. Garnett (Eds.), *Advances in Neural Information Processing Systems*. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/9a49a25d845a483fae4be7e341368e36-Paper.pdf. (Vid. pág. 4)

Chowdhury, N. K., Kabir, M. A., Rahman, M. M. & Islam, S. M. S. (2022). Machine learning for detecting COVID-19 from cough sounds: An ensemble-based MCDM method. *Computers in Biology and Medicine*, *145*, 105405. https://doi.org/https://doi.org/10.1016/j.compbiomed.2022.105405 (vid. pág. 5)

Dietterich, T. G. (2000). Ensemble Methods in Machine Learning. *Multiple Classifier Systems*, 1-15 (vid. pág. 5).

Dimitrakakis, C., Liu, Y., Parkes, D. C. & Radanovic, G. (2019). Bayesian Fairness. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*(01), 509-516. https://doi.org/10.1609/aaai.v33i01.3301509 (vid. pág. 6)

Donini, M., Oneto, L., Ben-David, S., Shawe-Taylor, J. S. & Pontil, M. (2018). Empirical risk minimization under fairness constraints. *Advances in Neural Information Processing Systems*, *31* (vid. pág. 4).

Dwork, C., Hardt, M., Pitassi, T., Reingold, O. & Zemel, R. (2012). Fairness through awareness. *Proceedings of the 3rd innovations in theoretical computer science conference*, 214-226 (vid. pág. 3).

Elsken, T., Metzen, J. H. & Hutter, F. (2018). Neural Architecture Search: A Survey. https://doi.org/10.48550/ARXIV.1808.05377 (vid. pág. 7)

Estévez Velarde, S., Gutiérrez, Y., Montoyo, A. & Almeida Cruz, Y. (2020). Automatic Discovery of Heterogeneous Machine Learning Pipelines: An Application to Natural Language Processing, 3558-3568. https://doi.org/10.18653/v1/2020.coling-main.317 (vid. pág. 7)

Estévez-Velarde, S., Gutiérrez, Y., Almeida-Cruz, Y. & Montoyo, A. (2020). General-purpose hierarchical optimisation of machine learning pipelines with gramma-

tical evolution. *Information Sciences*. https://doi.org/10.1016/j.ins.2020.07.035 (vid. pág. 7)

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M. & Hutter, F. (2015). Efficient and robust automated machine learning. *Advances in neural information processing systems*, *28* (vid. pág. 7).

Friedler, S. A., Scheidegger, C. & Venkatasubramanian, S. (2016). On the (im) possibility of fairness. *arXiv preprint arXiv:1609.07236* (vid. pág. 3).

Huang, G., Li, Y., Pleiss, G., Liu, Z., Hopcroft, J. E. & Weinberger, K. Q. (2017). Snapshot Ensembles: Train 1, get M for free. *CoRR*, *abs/1704.00109*. http://arxiv.org/abs/1704.00109 (vid. pág. 6)

Jin, H., Song, Q. & Hu, X. (2018). Efficient Neural Architecture Search with Network Morphism. *CoRR*, *abs/1806.10282*. http://arxiv.org/abs/1806.10282 (vid. pág. 7)

Jin, W., Dong, S., Yu, C. & Luo, Q. (2022). A data-driven hybrid ensemble AI model for COVID-19 infection forecast using multiple neural networks and reinforced learning. *Computers in Biology and Medicine*, *146*, 105560. https://doi.org/https://doi.org/10.1016/j.compbiomed.2022.105560 (vid. pág. 5)

Kamiran, F. & Calders, T. (2011). Data preprocessing techniques for classification without discrimination. *Knowledge and Information Systems*, *33*, 1-33 (vid. pág. 4).

Keras. (s.f.). https://github.com/fchollet/keras. (Vid. pág. 7)

Kleinberg, J. (2018). Inherent trade-offs in algorithmic fairness. *Abstracts of the 2018 ACM International Conference on Measurement and Modeling of Computer Systems*, 40-40 (vid. pág. 3).

Livieris, I. E., Pintelas, E., Stavroyiannis, S. & Pintelas, P. (2020). Ensemble deep learning models for forecasting cryptocurrency time-series. *Algorithms*, *13*(5), 121 (vid. pág. 5).

Loper, E. & Bird, S. (2002). NLTK: The Natural Language Toolkit. *CoRR*, *cs.CL/0205028*. https://arxiv.org/abs/cs/0205028 (vid. pág. 7)

MacCarthy, M. (2018). Standards of Fairness for Disparate Impact Assessment of Big Data Algorithms. *Other Information Systems & eBusiness eJournal* (vid. pág. 4).

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L. y col. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, *32* (vid. pág. 7).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. y col. (2011). Scikit-learn: Machine learning in Python. *the Journal of machine Learning research*, *12*, 2825-2830 (vid. pág. 7).

Polikar, R. (2006). Ensemble based systems in decision making. *IEEE Circuits and systems magazine*, *6*(3), 21-45 (vid. págs. 4, 5).

Schapire, R. E. (1990). The strength of weak learnability. *Machine learning*, *5*(2), 197-227 (vid. pág. 5).

Thornton, C., Hutter, F., Hoos, H. H. & Leyton-Brown, K. (2013). Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms. *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 847-855. https://doi.org/10.1145/2487575.2487629 (vid. pág. 6)

Witten, I., Hall, M., Frank, E., Holmes, G., Pfahringer, B. & Reutemann, P. (2009). The WEKA data mining software: An update. *SIGKDD Explorations*, *11*, 10-18. https://doi.org/10.1145/1656274.1656278 (vid. pág. 6)

Zafar, M. B., Valera, I., Gomez-Rodriguez, M. & Gummadi, K. P. (2019). Fairness Constraints: A Flexible Approach for Fair Classification. *Journal of Machine Learning Research*, *20*(75), 1-42. http://jmlr.org/papers/v20/18-262.html (vid. pág. 4)

Zafar, M. B., Valera, I., Rogriguez, M. G. & Gummadi, K. P. (2017). Fairness constraints: Mechanisms for fair classification. *Artificial intelligence and statistics*, 962-970 (vid. pág. 4).

Zemel, R., Wu, Y., Swersky, K., Pitassi, T. & Dwork, C. (2013). Learning fair representations. *International conference on machine learning*, 325-333 (vid. pág. 4).