Universidad de La Habana
Facultad de Matemática y Computación

Resolución Automática de Problemas Arbitrarios de Clasificación de Forma Justa

Autor:
Jorge Mederos Alvarado

Tutores:

Juan Pablo Consuegra Ayala
Alejandro Piad Morfis

Trabajo de Diploma
presentado en opción al título de
Licenciado en Ciencia de la Computación

Noviembre 2022

github.com/jmederosalvarado/thesis

Dedicación

Agradecimientos

Agradecimientos

Opinión del tutor

En la actualidad, los algoritmos de aprendizaje automático están siendo aplicados en disímiles áreas de la vida humana. En particular, su incorporación a tareas de toma de decisiones de alto riesgo ha dirigido la atención de muchos investigadores hacia una nueva interrogante: ¿estarán siendo "justos" los algoritmos de aprendizaje automático al tomar sus decisiones? El concepto de justicia o equidad se interpreta en este contexto como la ausencia de cualquier prejuicio o favoritismo hacia un individuo o grupo basado en sus características inherentes o adquiridas. El peligro fundamental de ignorar la interrogante planteada anteriormente radica en que los métodos de aprendizaje automático podrían no solo reflejar los sesgos presentes en nuestra sociedad, sino que también podrían amplificarlos. Resolver problemas de forma justa debería convertirse en un estándar en todos los contextos en los que es aplicable. En este marco se desarrolla la tesis de licenciatura de Jorge Mederos Alvarado, con quien pude trabajar este último año en el diseño y validación de un sistema para la resolución de problemas de clasificación de forma justa.

La propuesta de Jorge consiste en un enfoque que combina técnicas de AutoML, métodos de ensemble y optimización multiobjetivo, resultando en un sistema que permite resolver problemas de clasificación arbitrarios mientras se garantiza control tanto sobre la precisión del clasificador final como sobre su equidad. A diferencia de otros enfoques existentes, la propuesta de esta investigación se centra en proporcionar una interfaz única para solucionar problemas de clasificación arbitrarios. El sistema utiliza una representación basada en atributos protegidos y expone un conjunto de métricas de equidad. Internamente se realiza una búsqueda de la mejor configuración para ensamblar un grupo de arquitecturas de aprendizaje automático. Ambas características posibilitan obtener una solución agnóstica al problema a resolver. Su aplicabilidad a problemas de procesamiento de lenguaje natural potencia la relevancia del trabajo. El resultado final es una propuesta teórica, respaldada por un prototipo computacional, que demuestra que el estudiante posee las habilidades necesarias para aplicar en la práctica sus conocimientos.

Durante el desarrollo de esta investigación, Jorge ha tenido que asimilar por su cuenta conocimientos de diversas áreas, como optimización e inteligencia artificial. Además, ha tenido que estudiar en profundidad un campo de investigación tan nove-

doso y variado como es el análisis de sesgos en algoritmos de aprendizaje automático. El proceso de investigación e implementación desarrollado por Jorge queda recogido en un documento de tesis que avala además sus habilidades para llevar a buen término una investigación científica con la formalidad que el campo requiere. El estudiante ha demostrado así no solo dominio técnico del área, sino además capacidad de organización. Todo esto lo han realizado a la par de las actividades docentes, como estudiante de pregrado y como alumno ayudante de la asignatura Programación, donde ha sabido asumir con éxito todas las responsabilidades y retos.

Conozco a Jorge desde que cursaba el primer año de la carrera, donde le impartí clases de la asignatura Programación. Habría sido fácil predecir desde aquel entonces que terminaría supervisando su tesis, no necesariamente porque él hubiera querido, sino porque yo no habría permitido que fuese de otra forma. Querer llevar sus conocimientos al límite es una característica que no se puede dejar pasar desapercibida en un estudiante, y desde el minuto uno estaba claro que Jorge la poseía. El querer aprender más, entender mejor y encontrar una forma de aplicarlo luego, forma parte de la naturaleza de Jorge. Desde aquel entonces he podido compartir con él como parte del colectivo de la asignatura, trabajando en proyectos del grupo de investigación, y participando en eventos a nivel de universidad, nacional, e incluso internacional. Son pocos los estudiantes que pueden afirmar haber tenido una trayectoria tan rica a lo largo de sus años de pregrado como la de Jorge. Por supuesto, no todo es color de rosa, pues si hay otra cosa que describe a Jorge es el estar probando cada dos por tres alguna tecnología nueva; lo cual es perfecto mientras quede solo de su lado, pero para sorpresa de nadie, siempre terminaba arrastrado también yo a probar sus nuevos juguetes; prueba de ello es que hoy en día me encuentro atrapado usando Windows de nuevo por su culpa, y lo peor es que solo, porque Jorge ya cambió nuevamente. Habiendo dicho esto, me siento muy complacido de haber podido trabajar con él. Espero en los próximos años podamos seguir compartiendo y brindándole mi apoyo en las nuevas experiencias que se avecinan, con las que estoy seguro crecerá aún más. Tengo plena confianza en que ha de cosechar las recompensas por todo el empeño que ha puesto en sus estudios y en la investigación, y que ejercerá como un excelente profesional.

MSc. Juan Pablo Consuegra Ayala
Facultad de Matemática y Computación
Universidad de La Habana

Resumen

En los últimos años ha habido un incremento notable en la aplicación de técnicas de aprendizaje automático en la solución de numerosos problemas. En particular, estas técnicas han sido empleadas para remplazar o asistir a los humanos en escenarios de toma de decisión como los sistemas de admisión de individuos a ciertos puestos de trabajo, los sistemas que asisten en la evaluación del riesgo de reincidencia criminal, entre otros. El empleo de estos sistemas en escenarios tan críticos y con tanta influencia sobre el futuro de los individuos sobre los cuales toma las decisiones, ha despertado preocupaciones acerca de la imparcialidad y la justeza de estos sistemas. Varios estudios han sido realizados con el objetivo de estudiar los posibles sesgos que los diferentes sistemas de aprendizaje podrían presentar, y, en efecto, se ha observado que varios de estos sistemas que son actualmente utilizados no son justos al decidir sobre determinados grupos de individuos. Con el propósito de mitigar los sesgos de estos sistemas han surgido diferentes métodos, muchos de ellos muy efectivos. Sin embargo la mayoría de estos métodos presenta una seria de limitaciones, entre las que resalta el solo ser aplicables a determinadas clases de modelos o problemas.

En este trabajo se propone un sistema para la solución de problemas de clasificación arbitrarios de forma justa. Este sistema es agnóstico a los modelos de clasificación en cuestión y al método de entrenamiento del mismo. Nuestro sistema esta dividido en dos fases, una primera fase se encarga de generar una colección de modelos base diversos entre sí y una segunda fase encargada de ensamblar los modelos base obtenidos con el propósito de optimizar múltiples métricas simultáneamente, por ejemplo una métrica de precisión y varias métricas de equidad. Se realizan experimentos con diferentes métricas para lograr diversidad entre los clasificadores base para arribar a conclusiones acerca de las ventajas y limitaciones de cada una. Por último se compara nuestro enfoque con múltiples métodos propuestos en la literatura para la solución del problema de mitigación de sesgos y se observa que este, además de ser mas versátil, es sumamente competitivo. Una implementación del sistema propuesto en este trabajo se encuentra disponible a la comunidad en una biblioteca de *Python* llamada *BFair*.

Abstract

In recent years there has been a notable increase in the application of machine learning techniques to the solution of numerous problems. Particularly, these techniques have been employed to replace or assist humans in decision making scenarios such as applicant acceptance for certain jobs and criminal recidivism systems. The employment of these systems in such critical scenarios with so much influence over the future of the individuals have raised concerns about the impartiality and fairness of these systems. Several studies have been conducted with the goal of studying the possible bias that the different machine learning systems could present, and indeed, it has been observed that several of these systems are not fair when judging certain subgroups of the population. With the purpose of mitigating the bias in these systems, different methods have come up and many of them have proved to be very effective. However, most of these methods present several limitations, particularly relevant, the majority these methods are only applicable to certain classes of models and problems.

This work proposes a system for the solution of arbitrary classification problems in a fair manner. This system is agnostic to both the classification model and the method used to train it. Our approach is separated in two phases, a first phase takes care of generating a collection of base models that are diverse among them and a second phase is in charge of assembling the base models obtained, with the purpose of optimizing multiple metrics simultaneously, for instance a performance metric and several fairness metrics. Experiments are performed using different metrics to achieve diversity among the base classifiers in order to come to conclusions regarding advantages and disadvantages of each one of them. Lastly, our approach is compared to multiple methods proposed in the literature for the solution of the bias mitigation problem, and we observe that, on top of being more versatile, it is extremely competitive. An implementation of the proposed system is available to the community in a *Python* library called *BFair*.

Índice general

Introducción

En los últimos años, ha habido un creciente interés en aplicar técnicas de aprendizaje automático para resolver diferentes tipos de tareas. Análisis de emociones en textos, etiquetado de imágenes y traducción automática son algunos ejemplos de estas tareas [5, 24, 49, 59]. En general, los problemas de clasificación son uno de los problemas más comunes para los cuales los algoritmos de aprendizaje automático tienden a producir buenos resultados.

Con el uso cada vez más frecuente de los métodos de aprendizaje automático en dominios tales como *prestamos financieros*, *contratación*, *justicia criminal*, y *admisiones en las universidades*, ha habido una mayor preocupación por el potencial de estas técnicas de accidentalmente codificar sesgos sociales y resultar en una discriminación sistemática [3, 6, 7, 10, 11]. Un clasificador que solamente es ajustado para maximizar la precisión de las predicciones puede injustamente predecir un alto riesgo en el crédito para algunos subgrupos de la población aplicando a un préstamo. Un ejemplo de tratamiento imparcial a grupos de la población puede ser encontrado en el *software COMPAS* (*Correctional Offender Management Profiling for Alternative Sanctions*), utilizado por las cortes en los Estados Unidos para determinar los riesgos de un individuo de reincidir en un crimen. Los resultados que produce este sistema son utilizados para motivar decisiones respecto a si los defendidos deben ser liberados, en diferentes etapas del proceso de justicia. Problemáticamente, este software falsamente etiquetaba defendidos de raza no blanca con un mayor riesgo que los defendidos blancos [3].

Cuantificar y mitigar los sesgos durante las diferentes etapas del ciclo de vida de los modelos de aprendizaje automático ha sido objeto de estudio de numerosas investigaciones [6]. Específicamente, un número de métodos han sido propuestos recientemente para incrementar la equidad en los resultados producidos por estos modelos. La clave del diseño detrás de todos estos métodos es maximizar la precisión de las predicciones, sujeto a restricciones de equidad sobre los resultados. Sin embargo, como consecuencia de mantener la tratabilidad computacional de estas técnicas, estos métodos sufren de una o más de las siguientes desventajas. La técnica de mitigación es (i) específica a la clase del modelo utilizado (p.e. solamente modelos lineales), (ii) limitada a un conjunto específico de definiciones de equidad, (iii) limitado a un único atributo protegido

binario, (iv) requiere acceso a información sensible en el momento de las predicciones, o (v) resulta en un clasificador *randomizado* que puede generar diferentes predicciones para la misma entrada en diferentes momentos. Estas desventajas limitan la habilidad de los profesionales de poner en funcionamiento modelos justos de aprendizaje automático para problemas de la práctica con objetivos y restricciones arbitrarias. Por ejemplo, como resultado de la limitación (i), cualquier ensemble o solución de aprendizaje híbrido combinando diferentes clases de modelos y/o conocimiento del dominio son descartadas. Una consecuencia de (iv) es que se necesita información acerca de atributos sensibles en el momento de realizar las predicciones, lo cual puede traer problemas con las leyes que protegen la privacidad de los usuarios. Finalmente, (v) puede resultar en el mismo modelo, aceptando y denegando préstamos para el mismo individuo en diferentes ocasiones de forma arbitraria.

Recientemente, avances en *AutoML* (del inglés *Automatic Machine Learning*) han permitido el desarrollo de bibliotecas y herramientas efectivas para encontrar la mejor combinación de algoritmos e hiperparametros para resolver un problema. Múltiples tecnologías han sido propuestas para resolver el problema de *AutoML*, tales como *AutoWeka* [58] o *AutoKeras* [35]. Estas herramientas son alternativas para reducir el tiempo empleado por investigadores resolviendo problemas bien estudiados. Incluso si las herramientas de *AutoML* tienden a consumir más tiempo y recursos que bibliotecas estándar de aprendizaje automático, no tener que razonar acerca de cual arquitectura podría ser la más apropiada para el problema en cuestión vale el esfuerzo de desarrollarlas. Además, al ser aplicables a un rango amplio de problemas, aprender a utilizar estas herramientas puede ser más sencillo que aprender a utilizar diferentes bibliotecas independientes de aprendizaje automático. Se podría decir entonces que uno de los objetivos de las técnicas de *AutoML* es la *democratización del Aprendizaje Automático*.

En este contexto se encuentra también *AutoGOAL* [27] como un buen ejemplo de estas bibliotecas de *AutoML*. *AutoGOAL* es una propuesta reciente que utiliza técnicas que le permite explorar un espacio de búsqueda heterogéneo de modelos e hiperparámetros. A diferencia de otras tecnologías de *AutoML* existentes, *AutoGOAL* puede de forma automática construir flujos que le permitan combinar técnicas y algoritmos de diferentes bibliotecas, incluyendo clasificadores lineales, redes neuronales y herramientas de procesamiento del lenguaje. A pesar de que *AutoGOAL* es capaz de combinar algoritmos de diferentes categorías para construir flujos, no tiene la habilidad de combinar múltiples flujos para generar una solución. Adicionalmente, su algoritmo de búsqueda le permite solamente optimizar una función objetivo y no permite atacar problemas que en su naturaleza son de múltiples objetivos.

Las técnicas de *AutoML* y en particular la estrategia que propone *AutoGOAL* resulta sumamente útil para superar las limitación que tienen muchos de los enfoques de mitigación de sesgos al solo ser aplicables a determinadas clases de modelos

y métricas. La idea es utilizar la capacidad *AutoGOAL* de explorar un espacio de modelos heterogéneo y componer diferentes técnicas de aprendizaje automático, para poder encontrar una solución que cumpla con determinadas restricciones de equidad. Sin embargo, se mantiene la interrogante de como lograr mantener determinadas restricciones de equidad a la vez que el método se mantiene agnóstico al modelo de aprendizaje. Para ello resultaría muy útil poder utilizar *AutoGOAL* en entornos multiobjetivo, donde se optimizan simultáneamente métricas de equidad y de efectividad del modelo, esto es precisamente lo que permite lograr una de las propuestas de este trabajo. Adicionalmente, se propone una solución al problema de *AutoGOAL* de no poder combinar diferentes flujos de extremo a extremo en un flujo compuesto. Esto se logra a partir de la utilización de métodos de ensemble, los cuales permiten la combinación de las predicciones de diferentes modelos base para lograr modelos finales más robustos [54].

Objetivo general

El objetivo de este trabajo es proponer y estudiar un sistema que permita la solución de problemas de clasificación arbitrarios de forma justa. Este sistema debe ser agnóstico al modelo de solución y el proceso de optimización del mismo, así como a las métricas de equidad y efectividad que se deseen utilizar. De esta forma contribuir a la democratización de las técnicas de solución de problemas de clasificación de forma justa.

Objetivos específicos

- Realizar un estudio de la literatura en los temas relacionados a la mitigación de sesgos y otros temas relevantes a la solución propuesta.
- Concebir un método que permita atacar el problema de encontrar clasificadores justos y efectivos para problemas de clasificación arbitrarios de forma agnóstica a los modelos utilizados.
- Implementar un prototipo computacional del método propuesto.
- Diseñar un marco experimental donde evaluar los resultados del método de solución diseñado.
- Comparar los resultados del sistema propuesto con otros métodos propuestos en la literatura.
- Analizar en profundidad los resultados obtenidos y arribar a conclusiones.

Contribuciones

- Se propone un sistema, cuya implementación queda disponible a la comunidad, que permite resolver problemas de clasificación arbitrarios con control sobre la equidad.
- Este sistema permite al usuario especificar diferentes funciones de equidad a optimizar.
- Adicionalmente, trabaja sobre colecciones de datos heterogéneas.
- El sistema es agnóstico al modelo y no requiere conocimiento específico del dominio del problema para utilizarlo satisfactoriamente.

Organización del trabajo

El resto de este trabajo esta organizado de la siguiente forma. El capítulo 1 realiza una revisión de la literatura y el estado del arte en los tópicos relacionados con el problema a resolver. Luego el capítulo 2 describe detalladamente el método de solución propuesto. El capítulo 3 describe la experimentación realizada, los resultados obtenidos y un análisis en profundidad de los mismos. Finalmente se arriban a conclusiones y se discuten las líneas de investigación futuras.

Capítulo 1

Estado del Arte

Los métodos de aprendizaje automático se han vuelto cada vez más populares en los últimos años, tanto en la diversidad como en la importancia de sus aplicaciones [16]. El aprendizaje automático es utilizado en una variedad de aplicaciones críticas de toma de decisiones. Una de las mayores preocupaciones de esta situación es el hecho de que estos algoritmos pudieran estar tomando decisiones de forma sesgada y por tanto afectando a determinados grupos de individuos, lo cual se mostró era lo que estaba sucediendo [3].

Los sesgos que muestran los algoritmos de aprendizaje automático provienen de diversas fuentes. Una de las más comunes son las colecciones de datos, de las cuales los modelos de aprendizaje automático implícitamente aprenden a expresar tales sesgos. Los sesgos contenidos en las colecciones de datos pueden ser a veces el resultado de errores durante su construcción, pero es usual que tengan su origen en procesos históricos. Cuando estos sesgos son utilizados para realizar una predicción, puede dar lugar a decisiones injustas acerca de los individuos.

Los sesgos pueden incluso provenir de fuentes más difíciles de detectar para el usuario. Por ejemplo, la popularización de representaciones semánticas autogeneradas, tales como *word embeddings* pueden contribuir a la propagación de sesgos contenidos en datos históricos [8]. Luego, incluso cuando se trabaja con una colección de datos que no contiene explícitamente nada relacionado con género o raza, por ejemplo, al utilizar *word embeddings* clásicos para procesar datos en forma de texto, pueden obtenerse decisiones sesgadas a causa de sesgos que, como se ha probado, contienen los *word embeddings*.

Es posible que los modelos de aprendizaje automático produzcan decisiones injustas incluso asumiendo que sus datos de entrenamiento no tenían sesgos. Es decir, el modelo de aprendizaje automático ha construido una hipótesis que no generaliza completamente los datos no vistos durante el entrenamiento o incluso los propios ejemplos entrenantes. Luego, cuantificar los sesgos que un modelo de aprendizaje automático

tiene respecto a un conjunto de datos es una tarea usual. Se hace imprescindible proveer una definición que permita capturar el concepto de equidad y analizar los sesgos que presenta un modelo de forma objetiva.

1.1. Definiciones de Equidad

Múltiples definiciones han sido propuestas para capturar diferentes criterios de equidad. No existe en estos momentos una única definición ampliamente aceptada de lo que es *equidad*, sino que diferentes definiciones codifican diferentes características que se muestran útiles en diferentes contextos. Incluso, algunas de las definiciones más comunes presentan conflictos entre sí. A continuación se presentan algunas de las definiciones más utilizadas.

- **Statistical Parity (SP)**: Un clasificador binario $\hat{Y}$ satisface *statistical parity* si $P(\hat{Y}=1|P=1) = P(\hat{Y}=1|P=0)$. Esto es, la probabilidad de un resultados positivo debería ser la misma sin importar si el individuo pertenece a un grupo protegido [60].
- **Equal Opportunity (EO)**: Un clasificador binario $\hat{Y}$ satisface *equal opportunity* con respecto a P y Y si $P(\hat{Y}=1|Y=1,P=1) = P(\hat{Y}=1|Y=1,P=0)$. Esto significa que la probabilidad de que a una persona en la clase positiva le sea asignada un resultado positivo debería ser igual para miembros tanto de grupos protegidos como no protegidos [60].
- **Equalized Odds (EOdd)**: Un clasificador binario $\hat{Y}$ satisface *equalized odds* con respecto al atributo protegido P y predicción Y, si $P(\hat{Y}=1|Y=y,P=1) = P(\hat{Y}=1|Y=y,P=0)$, es decir, $\hat{Y}$ y P son independientemente condicionales a Y. Esto significa que la probabilidad de que a una persona en la clase positiva le sea asignada correctamente una predicción positiva y la probabilidad de que a una persona en la clase negativa le sea incorrectamente asignada una predicción positiva debería ser la misma para miembros de grupos protegidos y no protegidos [60].

Las concesiones inherentes a utilizar cada noción de equidad han sido estudiados extensamente [25, 31, 41]. Escoger la definición correcta para un problema determinado es difícil, y en la práctica no puede ser delegado a un agente automático. En su lugar, una decisión humana es preferida para asegurar una decisión informada.

Es posible transformar las definiciones de equidad presentadas anteriormente para obtener métricas que de cuantifiquen el sesgo de un clasificador. Por ejemplo, *Differential Statistical Parity* (`DSP`) puede ser obtenida utilizando la definición de *Statistical*

Parity para obtener una medida de la diferencia entre la cantidad de resultados positivos entre los diferentes grupos protegidos. Luego, estas métricas pueden ser utilizadas como funciones de pérdida a emplear en algoritmos de optimización que permitirán mitigar los sesgos de los modelos.

1.2. Mitigación de sesgos

Las técnicas de mitigación de sesgos pueden ser divididas fundamentalmente en técnicas de pre-procesamiento, post-procesamiento y técnicas durante el procesamiento. Adicionalmente un conjunto de técnicas llamadas meta-algoritmos han surgido recientemente, presentando muy buenos resultados.

Las técnicas de pre-procesamiento logran equidad modificando la representación de los datos, es decir, pre-procesando los datos y luego adoptando una solución de aprendizaje automático estándar [12, 38, 64]. Un ejemplo de esto es: aprender una representación a partir de resolver un problema de optimización con dos objetivos, codificar la información preservando la mayor cantidad de información posible y ofuscar al mismo tiempo la pertenencia al conjunto de atributos protegidos [64]. Una ventaja de los métodos de pre-procesamiento es que son agnósticos al modelo. Sin embargo, sus hiperparámetros, así como los del modelos de aprendizaje automático seleccionado, todavía necesitan ser ajustados para mejor rendimiento.

Los métodos aplicados durante el procesamiento aseguran que se cumplan ciertas restricciones de equidad durante el entrenamiento [23, 63, 62], sin embargo, esto los hace aplicable solo a una cierta clase de modelos. Por ejemplo, el algoritmo propuesto por *Donini et al.* [23] solo puede ser aplicado a *kernel machines* (tales como *maquinas de soporte vectorial*), y con la limitación adicional de poder trabajar exclusivamente con una única definición de equidad (*Equal Opportunity*). Aunque las técnicas de mitigación durante el procesamiento pueden brindar muy buenos resultados para la clase del modelo que están diseñados, frecuentemente son difíciles, o a veces imposible, de extender para nuevas clases de modelos. Estos métodos también pueden introducir nuevos hiperparámetros que podrían requerir conocimiento específicos del dominio y experimentación.

Las técnicas de post-procesamiento operan ajustando el umbral de decisión de modelos pre-entrenados para eventualmente lograr resultados más justos respecto a una métrica de equidad dada. El principal problema es que post-procesar el umbral de decisión es inherentemente subóptimo y puede llevar a peores balances de eficacia y equidad. Adicionalmente, estas técnicas no son utilizables si la información sensible no esta disponible en el momento de realizar las predicciones El conocimiento de información sensible de los individuos reales por parte del sistema en el momento de realizar las predicciones puede llevar a problemas legales [46].

Finalmente, resultan relevantes los llamados meta-algoritmos, una clase de métodos recientemente propuesta para tareas de clasificación justa. Estos reducen la tarea de clasificación justa a una secuencia de problemas de clasificación con costo asociado a sus errores de predicción [2, 1, 39]. Las soluciones a estos problemas suelen producir un clasificador *randomizado*. Contrario a los métodos que funcionan durante el procesamiento, los meta-algoritmos no dependen del tipo de los modelos que se utilizan en el clasificador, sino en la capacidad de los mismos para ser reentrenados repetidamente. En el contexto de algoritmos de Minimización del Riesgo Empírico (*ERM* por sus siglas en inglés), estos métodos son agnósticos al modelo de aprendizaje automático, pero aun necesitan implementaciones específicas basadas en la definición de equidad seleccionada y necesitan producir un ensemble de modelos. Limitaciones similares caracterizan un número de enfoques que utilizan optimización [14, 21] o inferencia bayesiana [39, 57], sus implementaciones tienen que estar diseñadas específicamente para ciertas definiciones de equidad.

Varios algoritmos han sido propuestos en la literatura, que se acogen a uno de los enfoques anteriormente descritos o combinan varios de ellos. A continuación se exploran algunos de estos métodos que resultan sumamente relevantes para el presente trabajo.

***Fair Bayesian Optimization* [53]** Motivado por la versatilidad del ajuste de hiperparámetros, propone un enfoque basado en *Optimizacion Bayesiana* (*BO*) general restringida. La *Optimizacion Bayesiana* es una metodología bien establecida para optimizar funciones de caja negra costosas de evaluar. La técnica propuesta por *FBO* optimiza los hiperparámetros de una función de caja negra de forma agnóstica al modelo, mientras se satisfacen restricciones de equidad. *FBO* se basa fundamentalmente en la hipótesis de que ajustar los hiperparámetros de un modelo es suficiente para encontrar un balance adecuado entre eficacia y equidad.

***SMOTE* [13]** Propone un método de preprocesamiento para colecciones de datos desbalanceadas. El enfoque de este método se basa fundamentalmente en el sobre-muestreo de los datos pertenecientes a las clases menos representadas en la colección. Utiliza un algoritmo que genera nuevos ejemplos a partir de modificaciones en los ejemplos pertenecientes a dicha clase existente en los datos. En muchas ocasiones los sesgos en los modelos vienen dado precisamente por un desbalance en la cantidad de individuos de los grupos protegidos que pertenecen a una clase determinada respecto a otras, por lo que técnicas como *SMOTE* permiten mitigar este problema.

***Fair Empirical Risk Minimization* (*FERM*) [22].** Introduce una generalización de las diferentes nociones de equidad, las cuales restringen el *riesgo condicional* de un clasificador de acuerdo a una función de pérdida predeterminada y

un parámetro de aproximación. Este intenta resolver el problema de minimizar el riesgo esperado para un conjunto predeterminado de funciones, utilizando una modificación de *Empirical Risk Minimization* a la cual se refieren como *Fair Empirical Risk Minimization*. Además, se proponen ejemplos concretos de algoritmos como *Máquinas de Soporte Vectorial* mejorados para satisfacer estas restricciones de equidad. Finalmente se muestra como para el caso lineal de la restricción de equidad modificada, este enfoque se reduce a un paso de preprocesamiento sobre la colección de datos.

***Zafar et al.* [63].** El trabajo realizado por los autores introduce una métrica para la equidad del umbral de decisión, la *covarianza del umbral de decisión*. Utilizando esta métrica como base, se definen dos problemas a resolver, la optimización de la equidad con restricciones sobre la precisión, y la optimización de la precisión con restricciones sobre la equidad. Los autores realizan modificaciones a *Regresión Logística* y *Maquinas de Soporte Vectorial* para utilizar estas restricciones y resolver estos problemas propuestos.

***Mitigación de Sesgos mediante Aprendizaje Adversarial* [65].** Replantea el problema de mitigación de sesgos como un problema de aprendizaje profundo adversarial [32]. Esto significa que la tarea se reduce a encontrar un modelo predictor que resuelva el problema de clasificación a la vez que intenta que un adversario no pueda inferir el valor de los atributos protegidos a partir de la predicción.

1.3. Métodos de ensemble

Los métodos de ensemble están diseñados para intentar resolver el problema de bajo *bias* / alta varianza que muestran la mayoría de los modelos de aprendizaje automático, haciéndolos apropiados para generar modelos de clasificación más robustos [54]. Un modelo de ensemble está diseñado a partir de muchos modelos con bajo *bias* cuyas predicciones son combinadas para producir una predicción final. Se asume fundamentalmente que la combinación de varias predicciones de bajo nivel producirá una salida con baja varianza mientras mantiene un bajo *bias*. Tener un conjunto diverso de modelos de bajo nivel es una característica fundamental para lograr esto [54]. Sin embargo, esto requiere que clasificadores individuales cometan errores en diferentes instancias. La intuición es que si cada clasificador comete errores diferentes, entonces una combinación estratégica de estos clasificadores puede reducir el error total. Luego, es necesario lograr que cada clasificador sea lo más único posible, particularmente con respecto a ejemplos erróneamente clasificados.

La naturaleza de múltiples hipótesis de los métodos de ensemble asegura que, si

son ajustados lo suficiente, tendrán resultados mejores que cualquiera de los modelos individuales en el caso general. Esto les permite también estimar el grado de confianza o calidad de las predicciones que producen. Las técnicas clásicas de ensemble incluyen *voting* y *weighted voting* [20], *boosting* [55], y *bagging* [9].

En un problema de clasificación, *voting* produce como salida la etiqueta que tiene la mayoría de los votos, tratando cada predicción de los modelos ensamblados como un voto. *Weighted-voting* funciona de forma similar a *voting*, pero a cada modelo del ensemble le 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 rendimiento en los ejemplos entrenantes donde los modelos anteriores tuvieron peor rendimiento. Durante este proceso, a cada submodelo le es también asignado un peso que marca la importancia de su predicción. *Bagging* entrena cada submodelo en una selección diferente (con reemplazo) de los ejemplos entrenantes originales. Alternativamente, *feature bagging* funciona de forma similar, seleccionando un subconjunto de características en lugar de los ejemplos entrenantes, causando que características correlacionadas sean analizadas de forma separada en algunos submodelos.

Las capacidad de los métodos de ensemble para construir modelos más robustos los ha hecho apropiados para múltiples aplicaciones. El dominio de la salud es uno de los ejemplos donde estos métodos han sido aplicados con gran éxito. Por ejemplo, una aplicación híbrida de métodos de ensemble con redes neuronales en un entorno de aprendizaje por reforzamiento ha sido presentada para la predicción de infección de COVID-19 con gran precisión [36]. De forma similar, un método de toma de decisión multicriterio basado en ensembles ha sido propuesto para la detección de COVID-19 a partir del sonido de la tos en pacientes [17]. Existen ejemplos también no relacionados a la medicina, por ejemplo, *Livieris et al.* [44] emplea estrategias de ensemble como *ensemble-averaging*, *bagging* y *stacking* con metodologías avanzadas de aprendizaje profundo para predecir los precios, a nivel de hora, de criptomonedas como *Ethereum*, *Bitcoin* y *Ripple*.

Una simple pero poderosa técnica en el contexto de los métodos de ensemble es la llamada *snapshot ensemble* [34]. Esta técnica genera múltiples clasificadores base, entrenando una sola red neuronal mientras la hace converger de forma rápida y repetida a múltiples óptimos locales, y salvando, en cada uno de dichos puntos, los parámetros del modelo. Todas las redes neuronales son luego ensambladas para producir el clasificador final. Estos *snapshot ensemble* son más robustos y precisos que las redes individuales dada su naturaleza de ensemble, sin ningún costo adicional de entrenamiento.

Como se puede observar, los métodos de ensemble son muy poderosos. Estos son capaces de construir sistemas que a partir de las predicciones de un conjunto de modelos (quizás no lo suficientemente expresivos) generalizan bien y obtienen resultados

satisfactorios. Sin embargo, es necesario para la construcción de estos ensembles se debe partir de un conjunto de modelos base previamente obtenidos, que cumplan determinadas características. Luego, resulta sumamente interesante explorar enfoques que permitan obtener estos modelos base de forma automática para un problema determinado, de forma tal que no sea necesario manualmente tomar decisiones acerca de el tipo de modelos a utilizar, los hiperparámetros de los mismos, etc.

1.4. Métodos de AutoML

AutoML (del inglés *Automated Machine Learning*) es el proceso de automatizar la solución de problemas del mundo real a través de técnicas de aprendizaje automático. El proceso intenta eliminar la necesidad de humanos expertos en aprendizaje automático para seleccionar apropiadamente las características, flujos, paradigmas, algoritmos y sus hiperparámetros para resolver un problema [21]. Las principales ventajas de las tecnologías de *AutoML* incluyen: (1) reducir el tiempo empleado en resolver problemas bien estudiados; y, (2) eliminar la necesidad de conocimiento experto. Adicionalmente, estas tecnologías tienden a generar soluciones más simples que a menudo tienen mejor desempeño que soluciones diseñadas por humanos.

Múltiples tecnologías han sido propuestas para resolver el problema de *AutoML*. *AutoWeka* [58] fue una de las primeras soluciones presentadas. Esta solución está basada en el software *Weka* [61], un software construido a partir de varias herramientas de visualización y algoritmos para análisis de datos y modelación predictiva. *AutoWeka* resuelve el problema de *AutoML* como un problema *CASH* según definido a continuación.

Definición 1.1 *Sea* $A = \{A^{(1)}, \ldots, A^{(R)}\}$ *un conjunto de algoritmos, y sea* $\Lambda^{(j)}$ *el dominio de los hiperparámetros 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, denótese* $L(A^{(j)}_{\lambda}, D^{(i)}_{train}, D^{(i)}_{valid})$ *la pérdida del algoritmo* $A^{(j)}$ *en* $D^{(i)}_{valid}$ *con hiperparámetros* λ*. Entonces, el problema de Selección de Algoritmo y Optimización de Hiperparámetros Combinado (CASH) es encontrar la configuración conjunta de algoritmo e hiperparámetros que minimiza la pérdida:*

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

Otros sistemas populares de *AutoML* son *AutoSklearn* [30] y *AutoKeras* [35]. Estos sistemas están basados en las bibliotecas de aprendizaje automático *ScikitLearn* [52]

y *Keras* [40], respectivamente. Ambos sistemas proveen una interfaz para encontrar la mejor arquitectura de aprendizaje automático para resolver un problema. Una diferencia fundamental entre ellos es la forma en que sus espacios de búsqueda son definidos. Mientras AutoSkLearn explora espacio de búsqueda condicional, es decir, un espacio con algunos hiperparámetros condicionados a otros, AutoKeras realiza una *Búsqueda de Arquitectura Neuronal (NAS)* [26], la cual implica explorar espacios jerárquicos de complejidad arbitraria.

AutoGOAL [27, 29] es una de las más recientes contribuciones al campo del AutoML. AutoGOAL es un sistema que utiliza técnicas heterogéneas para resolver el problema CASH. AutoGOAL se refiere a los modelos que construye como flujos, dado que cada uno de ellos esta formado por algoritmos interconectados. La esencia de AutoGOAL radica en su espacio personalizable de flujos y su conjunto de algoritmos de búsqueda, que son usados para encontrar la mejor configuración para resolver un problema. Cada flujo está definido como un conjunto de algoritmos interconectados que traducen una entrada predefinida a su salida correspondiente. El espacio de flujos comprende no solo el conjunto de algoritmos, sino también sus hiperparámetros.

Múltiples fuentes de algoritmos están incluidos en el espacio de AutoGOAL, tales como *ScikitLearn* [52], *NLTK* [45], *Keras* [40], y *Pytorch* [51]. Sin embargo, AutoGOAL carece de la habilidad de combinar múltiples flujos de extremo a extremo para generar una solución. Esta limitación puede ser superada con la utilización de ensembles.

El proceso fundamental de optimización utilizado en AutoGOAL en estos momentos esta basado en una técnica de optimización con Evolución Gramatical para gramáticas probabilistas libres del contexto [47]. El proceso consiste de un ciclo de generación y evaluación, utilizando una gramática G apropiada para describir el problema de aprendizaje de máquina que se intenta resolver. En cada iteración un conjunto de N flujos es generado a partir de tomar muestras de la gramática G de acuerdo a las probabilidades θ asignadas a cada producción. Estas probabilidades son inicializadas con una distribución uniforme θ_0 para todas las producciones. Cada flujo es evaluado (lo cual consiste simplemente en entrenamiento y ejecución), y los de mejor desempeño son utilizados para modificar θ, con el objetivo de maximizar la probabilidad de que estos sean generados. Este proceso se ilustra en el algoritmo

A pesar de la gran variedad de algoritmos que AutoGOAL tiene a su disposición y los satisfactorios resultados que obtiene para una gran variedad de problemas, es una limitación significativa el hecho de que solo es posible optimizar una función objetivo. La capacidad de AutoGOAL para encontrar modelos que optimicen múltiples funciones objetivos simultáneamente posibilitaría la solución de un amplio espectro de problemas que son inherentemente multi-objetivo.

Algoritmo 1.1: PGE

```
Input:
    N ← tamaño de la población
    n ← número de individuos seleccionados en cada iteración
    α ← factor de aprendizaje
    G ← gramática que describe los flujos de aprendizaje automático a explorar
    θ_0 ← probabilidades iniciales (uniforme)
    f ← función de fitness (entrenamiento y evaluación de flujos)
resultado ← none
para cada iteración i hacer
    P_i ← generar población utilizando gramática G, con probabilidades θ_{i-1}
    para cada solución S ∈ P_i hacer
        f(S) ← calcular fitness de S (evaluar el flujo)
    fin
    resultado ← argmax_{S∈P∪{resultado}}{f(S)}
    P*_i ← seleccionar los n mejores individuos de P_i
    θ*_i ← calcular la distribución marginal de P*
    θ_i ← αθ*_i + (1 − α)θ_{i-1}
fin
devolver resultado
```

1.5. Optimización Multiobjetivo

Los métodos de optimización multiobjetivo son aquellos que exploran el espacio de búsqueda optimizando simultáneamente diferentes funciones.

Definición 1.2 (Optimización Multiobjetivo) *Dadas m funciones objetivo $f_1 : X \to \mathbf{R}, \ldots, f_m : X \to \mathbf{R}$ las cuales traducen el espacio X en $\mathbf{R}$, un problema de optimización multiobjetivo esta dado es expresado de la siguiente forma:*

$$minimizar f_1(x), \ldots, minimizar f_m(x), x \in X \tag{1.2}$$

Al trabajar con múltiples funciones objetivos es necesario encontrar formas de comparar dos soluciones en el espacio de soluciones factibles. El concepto de *Pareto dominación* juega un papel fundamental en el ámbito de la optimización multiobjetivo, dado que permite comparar objetivamente dos vectores de forma precisa, sin requerir información adicional de preferencia.

Definición 1.3 (Pareto Dominación) *Dados dos vectores en el espacio objetivo, dígase* $y^{(1)}, y^{(2)} \in \mathbf{R}^m$ *, entonces el punto* $y^{(1)}$ *se dice que* ***pareto-domina*** *a* $y^{(2)}$ *si y solo si:*

$$\forall_{i\in\{1,\ldots,m\}} : y_i^{(1)} \leq y_i^{(2)} y \exists_{j\in\{1,\ldots,m\}} : y_j^{(1)} \leq y_j^{(2)} \tag{1.3}$$

Definición 1.4 (Frente pareto) *Todos aquellos vectores* x *del espacio objetivo tal que no exista otro vector* z *en el espacio objetivo que* ***pareto-domine*** *a* x*.*

Tradicionalmente los problemas de optimización multiobjetivo han sido atacados utilizando técnicas de escalarización [48]. Estas técnicas consisten en de alguna forma combinar todas las funciones objetivos en una sola o reescribirlas como restricciones. Varias técnicas existen en este contexto. *Linear Weighting* es una de estas, en la cual se construye una nueva función objetivo a partir de la combinación lineal de las funciones objetivo del problema original, esto es $min \sum w_i f_i(x), x \in X$. Este enfoque tiene un problema fundamental y es que si el *frente pareto* no es convexo, no es posible encontrar soluciones en esta zona, no importa los pesos w_i que se utilicen. Otra forma de escalarización es ϵ-*constrain*, esta técnica selecciona una función como la *principal* y las demás se establecen como restricciones al conjunto de soluciones factibles, exigiendo que sean menores que un ϵ. *Random Scalarization* [50] propone una estrategia basada en escalarizaciones aleatorias de las funciones objetivo. En lugar de optimizar una única escalarización de las funciones objetivos, este enfoque itera sobre un conjunto de escalarizaciones durante un proceso de *Optimización Bayesiana*. Finalmente, *ParEGO* [42] es una extensión de *EGO* [37] para la utilización en escenarios multiobjetivos. Utiliza un enfoque que combina las diferentes funciones objetivo en una sola a partir de una escalarización utilizando un vector de pesos parametrizado, utilizando diferentes parámetros para dicho vector de pesos el *Frente Pareto* es construído de forma gradual.

Existen métodos numéricos que intentan resolver el problema haciendo cumplir las condiciones de *Karush-Kuhn-Tucker* [43]. La idea va de encontrar al menos una solución del sistema de ecuaciones que se produce al tratar el problema de *KKT*. Es posible utilizar métodos de continuación y homotopía para obtener todas las soluciones [33, 56]. Estos métodos requiere que las soluciones satisfagan condiciones de convexidad local y diferenciabilidad.

Los algoritmos genéticos utilizan paradigmas basados en procesos evolutivos naturales, como *selección natural*, *mutación* y *recombinación* para mover una población (conjunto de vectores de decisión) a soluciones óptimas o casi óptimas [4]. Los algoritmos genéticos multiobjetivo generalizan esta idea y son diseñados para en cada iteración acercarse más al frente pareto. En este contexto destaca *NSGA-II* [18], el cual se explica en mayor detalle a continuación.

NSGA-II

NSGA-II es un algoritmo sumamente sencillo, pero aun así ha mostrado ser muy efectivo en la resolución de problemas de optimización multiobjetivo. El algoritmo consiste básicamente de un ciclo generacional que se divide en dos partes. En la primera parte, la población pasa por un proceso de variación. En la segunda parte, un proceso de selección toma lugar, el cual resulta en la población de la nueva generación. Este proceso se repite hasta que se cumple algún criterio de convergencia o se excede una cantidad de computo predefinida.

En la parte de la variación, λ nuevos individuos son generados. Para cada uno de ellos dos padres son seleccionados de la población actual P_t. Para escoger estos, se utiliza una selección de torneo binario, es decir se escogen aleatoriamente dos individuos de la población y se selecciona el mejor de acuerdo a su *orden* en la población. Los padres son entonces recombinados utilizando un operador de combinación, el individuo resultante es luego mutado utilizando un operador de mutación. De esta forma es creado un nuevo conjunto Q_t de individuos, los cuales son añadidos junto a la población actual al conjunto de individuos a considerar para la siguiente generación.

La segunda fase, fase de selección, los μ mejores individuos son seleccionados del conjunto $P_t \cup Q_t$ utilizando un mecanismo de ordenación multiobjetivo, de esta forma la población de la nueva generación P_{t+1} es formada. El mecanismo de selección de *NSGA-II* es el ingrediente fundamental que lo distingue del resto de los algoritmos genéticos que son utilizados para resolver problemas de optimización de un único objetivo. Este consiste de dos niveles. Primero se realiza un ***non-dominated sorting***. Este depende únicamente del *pareto-orden* entre los individuos. Finalmente los individuos que comparten el mismo *pareto-orden* son ordenados de acuerdo al ***crowding-distance***, la cual es una medida de la diversidad.

Non-dominated sorting

Sea $\mathrm{ND}(P)$ el conjunto de soluciones no dominadas 1.3 en una población P. *Non-dominated sorting* particiona la población en subconjuntos, basado en la *pareto dominación* 1.3 como especifica la siguiente recurrencia.

$$R_1 = \mathrm{ND}(P) \tag{1.4}$$

$$R_{k+1} = \mathrm{ND}(P \setminus \cup_{i=1}^{k} R_i) \quad k = 1, 2, \ldots \tag{1.5}$$

Como en cada paso de la recurrencia al menos una solución es eliminada de la población, el número máximo de *capas* es $|P|$. El orden de una solución esta dado por el subíndice k del R_k en el cual queda dicha solución.

Crownding distance

Si más de una solución queda en el mismo subconjunto de la población R_k luego de realizar la ordenación anterior, se procede a ordenar las soluciones dentro de dichos subconjunto a partir de su *crowding distance*. Esta es calculada para una solución x como la suma de las contribuciones c_i a la i- ésima función objetivo:

$$l_i(x) = \text{máx}\{f_i(y)|y \in R \setminus \{x\} \wedge_i (y) \leq f_i(x)\} \cup \{-\infty\} \tag{1.6}$$

$$u_i(x) = \text{mín}\{f_i(y)|y \in R \setminus \{x\} \wedge f_i(y) \geq f_i(x)\} \cup \{+\infty\} \tag{1.7}$$

$$c_i(x) = u_i - l_i, \quad i = 1, \ldots, m \tag{1.8}$$

$$c(x) = \frac{1}{m} \sum_{i=1}^{m} c_i(x), x \in R \tag{1.9}$$

Intuitivamente mientras más *espacio* exista alrededor de una solución, mayor será el *crowding distance* de la misma. Por tanto, aquellas soluciones con elevado *crowding distance* son preferidas a aquellas con baja distancia, con el propósito de mantener la diversidad en la población.

1.6. Discusión

Una de las mayores limitaciones de los métodos de mitigación de sesgos estudiados anteriormente es que son específicos al tipo de modelo que utilizan. Además muchos de ellos solamente pueden utilizar una única métrica predeterminada de equidad. Estas limitaciones hacen que estos métodos no puedan ser ampliamente adoptados a lo largo del campo del aprendizaje automático, entre otras cosas porque requiere de expertos en el tema para su aplicación satisfactoria.

El método propuesto en este trabajo explora la utilización de AutoGOAL como vía para obtener un conjunto de modelos *diversos* base que puedan ser posteriormente ensamblados. El proceso de construir el ensemble a partir de estos modelos base se realiza mediante una modificación del algoritmo de optimización de AutoGOAL, la cual permite explorar el espacio de búsqueda optimizando simultáneamente varias métricas, como por ejemplo la precisión y una o varias métricas de equidad. Para la modificación de AutoGOAL este trabajo se inspira en los métodos de selección propuestos por *NSGA-II*, estos son, *Non-dominated Sorting* y *Crowding Distance*.

El enfoque que se presenta en este trabajo no presenta las limitaciones expuestas anteriormente. Este se apoya en técnicas de *AutoML* para mantenerse agnóstico al problema, modelo de aprendizaje y su método de entrenamiento. Además, acepta simultáneamente diferentes métricas de equidad a partir de la utilización de técnicas de optimización multiobjetivo en el proceso de optimización de *AutoGOAL*.

Capítulo 2

Propuesta

En este capítulo se presenta un sistema que permite la resolución automática de problemas de clasificación arbitrarios. Este sistema tiene entre sus objetivos fundamentales producir clasificadores que sean justos respecto a una o varias métricas de equidad, a la vez que minimizan una función de perdida determinada. Para ello se propone un enfoque dividido en dos fases, que utiliza una combinación de técnicas de *AutoML*, métodos de ensemble y optimización multiobjetivo. La sección 2.1 provee una descripción general del sistema. Las secciones 2.2 y 2.3 detalla el funcionamiento de la primera y segunda fase del sistema respectivamente.

2.1. Descripción general

El sistema toma como entrada una colección de datos $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, una función de pérdida $\mathcal{L}$ y una o varias métricas de equidad $F_1, F_2, \ldots, F_n$. El objetivo del sistema es producir un modelo de clasificación que es a la vez efectivo según L y justo según $F_1, F_2, \ldots, F_n$. El sistema consiste en dos fases fundamentales. La primera es responsable de generar una colección de modelos, cada uno llamado modelo base. Esta colección es construida optimizando la función de perdida $\mathcal{L}$, mientras se asegura *diversidad* a lo largo de toda la población. La segunda fase es responsable de producir un conjunto de modelos de clasificación. Estos modelos son generados ensamblando la colección de modelos base de forma tal que optimice su efectividad según L, a la vez que es lo más *justo* posible según $F_1, F_2, \ldots, F_n$. Las secciones 2.2 y 2.3 abordan con más detalles la primera y segunda fase, respectivamente.

Párrafo con la imagen del overview, y que describe la imagen.

2.2. Fase 1: Generación de modelos base

En esta fase, al sistema se le da la tarea de generar N modelos para ajustar D de acuerdo a la pérdida $\mathcal{L}$.

La Definición 1.1 se modifica para buscar una colección de modelos en lugar de un solo modelo, sujeto a una métrica de diversidad $\mathcal{D}$. Esto es, se desea encontrar una colección de modelos base (modelos que optimicen la efectividad en el conjunto de datos D de acuerdo a la función de pérdida $\mathcal{L}$) mientras garantiza algunas diferencias entre sus hipótesis utilizando la métrica $\mathcal{D}$. Asegurar diversidad en la colección de modelos base es importante porque los métodos de ensemble no son capaces de mejorar su rendimiento si todos los modelos base tienen exactamente la misma hipótesis, es decir, si todos realizan las mismas predicciones.

El procedimiento aplicado para generar la colección de modelos base esta resumido por la función GenerateBaseModels. El espacio de algoritmos e hiperparámetros es explorado utilizando una estrategia de búsqueda preseleccionada. Todo esto es capturado por la función **explore**. Luego de (1) evaluar las arquitecturas generadas y (2) estimar la diversidad entre los modelos actualmente seleccionados y la nueva generación de modelos, la colección de modelos base es actualizada para ajustarse a su capacidad N. Todo esto es capturado por la función reselect.

Función GenerateBaseModels$(N, D, A, \Lambda, \mathcal{L}, \mathcal{D})$

```
set base_models ← ∅
para generation ∈ explore(A, Λ) hacer
    scores ← ∅
    para A_λ^(j) ∈ generation hacer
        scores^(j) ← (1/K) Σ_{i=1}^{K} L(A_λ^(j), D_train^(i), D_valid^(i))
    fin
    diversity ← D(base_models ⌢ generation, D)
    base_models ← reselect(base_models ⌢ generation, scores, diversity, N)
fin
devolver base_models
```

Para explorar inteligentemente el espacio de algoritmos e hiperparámetros, es decir, para resolver el problema *CASH* modificado, se utiliza la implementación de *Probabilistic Grammatical Evolution Search* (algoritmo 1.1) presente en AutoGOAL. La búsqueda comienza con una estrategia de muestreo aleatorio, pero según evalúa más flujos, modifica el modelo de muestreo probabilístico para que flujos similares a los mejores encontrados hasta el momento, sean generados con mayor frecuencia. El espacio de algoritmos e hiperparámetros empleado es el utilizado por defecto en AutoGOAL, el cual incluye varios algoritmos clásicos de aprendizaje automático presentes en las diferentes bibliotecas utilizadas por AutoGOAL.

Para reseleccionar la colección de modelos base, o sea, la colección de flujos de AutoGOAL, un enfoque goloso es utilizado. La función reselect resume la estrategia propuesta. El algoritmo siempre incluye el modelo que mejor se desempeña de acuerdo a $\mathcal{L}$ en la selección. Cada iteración siguiente añade el modelo, todavía no seleccionado, que maximiza la diversidad respecto a todos los modelos anteriormente seleccionados. El enfoque goloso no garantiza que la colección final logre la mejor posible diversidad respecto a $\mathcal{D}$. La precisión tampoco es tomada en cuenta, excepto para seleccionar el modelo de mejor desempeño.

Función reselect(M, $scores$, $diversity$, N)

inicializar $R \leftarrow \emptyset$
inicializar $R^{(0)} \leftarrow \underset{m^{(j)} \in M}{argmin}\ scores^{(j)}$
para $r \leftarrow 1$ **a** N **hacer**
  $R^{(r)} \leftarrow \underset{(m^{(j)} \in M \setminus R)}{argmax} \sum_{(m^{(i)} \in R)} diversity^{(i,j)}$
fin
devolver $R^{(0)} \cdots R^{(N)}$

Utilizando el enfoque goloso presentado anteriormente, tres implementaciones de referencia del método reselect son descritas a continuación. Estas servirán para establecer comparaciones con implementaciones en las cuales se utiliza el método goloso presentado anteriormente, variando la métrica de equidad del mismo para utilizar *disagreement* y *double-fault*, las cuales serán presentadas en la sección 2.2.1.

Shuffle: La colección de modelos base es construida barajando aleatoriamente la selección actual de modelos base y los nuevos encontrados. Los primeros N modelos luego de barajar son seleccionados para la siguiente generación.

Arbitrary: La colección 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 colección de modelos base seleccionados.

Best: La colección de modelos base es construida a partir de seleccionar los modelos de mejor rendimiento entre los previamente seleccionados y los recién encontrados.

A continuación, la sección 2.2.1 provee algunos detalles acerca de las métricas de diversidad estudiadas en este trabajo.

2.2.1. Métricas de diversidad

Dos métricas fueron implementadas para estimar la diversidad de una colección dada de modelos base. Ambas de ellas precomputan una matriz de clasificaciones incorrectas, la cual es utilizada entonces para computar una métrica que aporta información sobre la diversidad entre los modelos base dos a dos. La matriz de clasificaciones incorrectas se construye de la siguiente manera.

$$M_{i,j} = \begin{cases} 1 & \text{si el modelo j correctamente clasifica el ejemplo i } (D_{valid}) \\ -1 & \text{en otro caso} \end{cases} \tag{2.1}$$

Las siguientes métricas son computadas entre pares de modelos base para estimar cuan diferentes son sus hipótesis, y por tanto la diversidad de la colección incluyendo a ambos a la vez.

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

$$\texttt{disagreement}(m^{(a)}, m^{(b)}) = \frac{|\{M_{i,a} \neq Mi, b | s^{(i)} \in D^{(\star)}_{valid}\}|}{|D^{(\star)}_{valid}|} \tag{2.2}$$

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

$$\texttt{double-fault}(m^{(a)}, m^{(b)}) = 1 - \frac{|\{M_{i,a} = Mi, b = -1 | s^{(i)} \in D^{(\star)}_{valid}\}|}{|D^{(\star)}_{valid}|} \tag{2.3}$$

La diversidad de los modelos base conocemos que es un factor fundamental en la capacidad de generalización y en general el rendimiento de los ensembles. Los métodos de ensemble, como se verá a continuación son una parte clave de nuestro sistema y de los resultados que este logra. Por tanto, el estudio de la capacidad de estas métricas para producir conjuntos de clasificadores base diversos es de suma importancia. La influencia de estas métricas en los resultados de nuestro sistema será estudiada en detalle en la sección 3.2.1.

2.3. Fase 2: Ensamblado de modelos justos

En esta fase el sistema tiene la tarea de combinar las predicciones de N modelos para ajustar D de acuerdo tanto a la pérdida $\mathcal{L}$ como a las métricas de equidad $F_1, \ldots, F_n$.

El sistema una vez más resuelve un problema de *CASH* como se presenta en la definición 1.1. En lugar de trabajar directamente en $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, esta vez el sistema trabaja sobre $D^e = \{(y_1^{(*)}, y_1), \ldots, (y_n^{(*)}, y_n)\}$, donde $y_i^{(*)} = [y_i^{(0)}, \ldots, y_i^{(n)}]$ y cada $y_i^{(j)}$ es la salida de un modelo base j para el ejemplo i, $i \in [1 \ldots n]$, $j \in [1 \ldots N]$. En otras palabras, al sistema se le pide encontrar las mejores combinaciones de algoritmos y sus hiperparámetros para ensamblar las salidas de los modelos base.

2.3.1. Espacio de búsqueda

Con el propósito de encontrar el ensemble que optimiza las funciones objetivos en cuestión, el sistema utiliza técnicas de AutoML para explorar un espacio de posibles soluciones. Este espacio de búsqueda esta conformado por las diferentes maneras de formar un ensemble a partir de los modelos base. Más específicamente, el espacio de búsqueda se forma a partir de tomar decisiones sobre una serie de hiperparámetros. Estos hiperparámetros son: (1) la cantidad de modelos base que serán escogidos para formar el ensemble, (2) el subconjunto de modelos base a partir de la cantidad escogida anteriormente y (3) el tipo de ensemble que será utilizado. Los diferentes tipos de ensemble se describen con más detalle a continuación.

Voting Classifiers. Asigna la etiqueta más común entre las predichas por los modelos base. En caso de empate, se selecciona la etiqueta producida por el modelo más preciso entre los modelos base.

Overfitted Voting Classifiers. Asigna a cada combinación de salida de los modelos base la etiqueta que asegura el mejor desempeño en D^e_{train}. En el momento de predicción, si una combinación no antes vista es encontrada, este selecciona la etiqueta predicha por el modelo base más preciso (ignorando si esta fue la etiqueta más votada).

ML Voting Classifiers. Ajusta un modelo de aprendizaje automático sobre D^e_{train} para optimizar $\mathcal{L}$. La arquitectura del modelo de aprendizaje automático es tomado del conjunto de algoritmos disponibles por defecto a AutoGOAL.

2.3.2. Encontrando modelos justos y efectivos

Probabilistic Grammatical Evolution (1.1) es un algoritmo que a pesar de ser sumamente sencillo, permite explorar espacios de búsqueda muy variados. Adicionalmente,

su utilización en AutoGOAL ha mostrado que puede brindar excelentes resultados en la tarea de encontrar modelos capaces de aprendizaje automático. Sin embargo una de las limitaciones fundamentales de este algoritmo es la estrategia de selección de los individuos más aptos dentro población, dado que solamente considera una métrica como objetivo de la optimización. Nuestra propuesta realiza una modificación en este proceso de selección, el cual define la forma en que se explora el espacio, para acomodar más de una métrica en el mismo. Esto nos permitirá explorar el espacio de posibles ensembles e hiperparámetros de los mismos, mientras simultáneamente se dirige la búsqueda en direcciones que optimicen tanto métricas de equidad como de efectividad.

El funcionamiento de nuestro algoritmo consiste fundamentalmente de tres procesos siendo ejecutados continuamente en un ciclo. Primeramente, de forma similar a *PGE*, un conjunto de N flujos es generado a partir de tomar muestras de una gramática G de acuerdo a las probabilidades θ asignadas a cada producción. Seguido de esto, cada flujo generado es evaluado en cada una de las funciones objetivos a optimizar y se procede a un proceso de selección en el cual se determinan los mejores candidatos en la población para pasar a la siguiente generación. Este proceso de selección es el que guía las direcciones en las que se explora el espacio de búsqueda, por tanto necesita tener en cuenta la evaluación de cada una de las funciones objetivo. Dicho proceso se realiza utilizando *Non-dominated Sorting* (sección 1.5) y *Crowding Distance* (sección 1.5). De la misma forma en que se utiliza en *NSGA-II*, los N individuos de la población son seleccionados según el orden que se obtiene de aplicar *Non-dominated Sorting* y empleando *Crowding Distance* como métrica para desambiguar entre individuos con el mismo *pareto-orden*. Además, se mantiene en todo momento durante la ejecución del algoritmo la mejor aproximación del *Frente Pareto* encontrada hasta ese instante. Para esto actualizamos al final de cada iteración este conjunto aproximación con las soluciones ***no*** *pareto-dominadas* de la unión de aquellas que ya pertenecían al conjunto y las nuevas soluciones encontradas. Finalmente, como parte de la ejecución tradicional de *PGE*, se actualiza el modelo probabilístico de la gramática a partir de las soluciones que pasan el proceso de selección. La propuesta anteriormente descrita se resume en el algoritmo 2.1.

Esta propuesta tiene su motivación en tratar de aprovechar las ventajas que brindan individualmente *Probabilistic Grammatical Evolution* y *NSGA-II*. *PGE* es extremadamente versátil respecto al tipo de espacio que permite explorar gracias a permite explorar cualquier espacio que pueda ser capturado por una gramática. Al mismo tiempo *PGE* es muy flexible y no impone restricciones algunas sobre la forma en que se realiza el proceso de selección de los individuos que continúan de una generación a otra. *NSGA-II* por otro lado ha mostrado ser muy efectivo en converger a buenas aproximaciones del *Frente Pareto* para problemas de optimización multiobjetivo. Se considera entonces que una combinación de estos algoritmos puede ser

Algoritmo 2.1: NSPGE($D, G, \mathcal{F}$)

set $\sigma \leftarrow$ **uniform-init**(G)
set models $\leftarrow \emptyset$
para generation $\in$ **sample**(G, σ) **hacer**
    set scores $\leftarrow \emptyset$
    para $\mathcal{L}^{(i)} \in \mathcal{F}$ **hacer**
        para $A_{\lambda}^{(j)} \in$ generation **hacer**
            scores$^{(i,j)} \leftarrow \mathcal{L}^{(i)}(A_{\lambda}^{(j)},\ D_{train}^{(i)},\ D_{valid}^{(i)})$
        fin
    fin
    set indices $\leftarrow$ **non-dominated-sort**(generation, scores)
    set updates $\leftarrow$ **select-best-parameters**(generation, scores, indices)

    $\sigma \leftarrow$ **update-probabilities**(σ, updates)
    models $\leftarrow$ **select**(generation, indices)
fin
devolver models

capaz de explorar el espacio de búsqueda de la tarea en cuestión, de forma tal que la estrategia de exploración induzca a cada vez mejores aproximaciones del *Frente Pareto*, a partir de utilizar el método de selección de *NSGA-II*.

Una de las ventajas de nuestra propuesta es que mantiene en todo momento un conjunto de soluciones que aproximan el *Frente Pareto*. Esto significa que al terminar la ejecución, el sistema no da una única solución al usuario. Dar una única solución es problemático debido a que la naturaleza de estos problemas de optimización multi-objetivo implica que existe un conjunto de soluciones entre las cuales no hay manera general de determinar un orden. Cada una de dichas soluciones representa un balance diferente entre las diferentes métricas a optimizar. Nuestro sistema le provee al usuario un conjunto de soluciones óptimas entre sí, y el usuario puede posteriormente seleccionar la solución que mejor considere de acuerdo al problema en cuestión que se está resolviendo y las restricciones del mismo. Por ejemplo, en determinados casos puede ser tolerable ceder en la equidad del modelo con el objetivo de ganar precisión, mientras que otros escenarios más sensibles puede que la restricción sobre la equidad del modelo sea primordial y no pueda ser comprometida. Luego, el usuario tiene a su disposición una serie de soluciones que representan diferentes balances entre los objetivos, cada una útil en escenarios con diferentes características, todo esto luego de una única ejecución de extremo a extremo de nuestro sistema. Adicionalmente, el sistema presentado, permite establecer restricciones sobre el conjunto de soluciones

factibles. Por ejemplo, es posible especificar que determinada métrica de equidad no supere un valor X determinado, y nuestro sistema se encargará de desechar de forma inmediata durante el proceso de entrenamiento soluciones que no cumplan estas restricciones.

Capítulo 3

Análisis Experimental

En este capítulo se evalúa la capacidad de nuestro sistema de optimización para resolver problemas de clasificación y lograr buenos resultados según métricas de precisión y equidad. La experimentación realizada consiste de dos etapas. Inicialmente se analiza la capacidad de la primera fase del sistema para obtener un conjunto de modelos base lo suficientemente diverso como para que ensamblar sus predicciones resulte en un modelo de mayor precisión que los modelos base. Finalmente, se estudia si el algoritmo propuesto permite encontrar formas de ensamblar los modelos base resultantes de la primera etapa de manera tal que se obtengan valores satisfactorios tanto de precisión como en las métricas de equidad.

3.1. Marco Experimental

Las dos etapas en las que queda divida nuestra experimentación se describen a continuación.

La **primera etapa experimental** tiene como objetivo estudiar la capacidad del sistema de producir un conjunto de modelos base que, al ser ensamblado, pueda generalizar y obtener mejores resultados que dichos modelos base. Adicionalmente, se desea estudiar que influencia tienen las distintas métricas de diversidad utilizadas en esta capacidad del sistema.

Para estimar el mejor rendimiento obtenible a partir de ensamblar el conjunto de modelos base encontrado, dos medidas son propuestas a continuación. Estas medidas estiman el rendimiento que logran modelos de ensemble artificiales, estos son modelos de ensemble que conocen los resultados correctos a priori, y por tanto son modelos que no tienen utilidad real fuera de ser utilizados con propósitos de comparación. Llamaremos a dichos ensembles: *Oráculo Optimista* y *Oráculo Sobreajustado*.

Oráculo Optimista. Este modelo oráculo devuelve la etiqueta correcta si al menos

uno de los modelos base predijo dicha etiqueta. La única forma de que este modelo falle es si ninguno de los modelos base fue capaz de sugerir la etiqueta correcta.

$$O^{(i)}_{optimista}(M) = \begin{cases} y_i & \text{si } y_i \in \{\, y_i^{(j)} \mid m^{(j)} \in M \,\} \\ y_i^{(0)} & \text{en otro caso} \end{cases}$$

Los resultados alcanzados por este ensemble reflejan el grado de cobertura de los modelos base sobre la colección de evaluación, esto es, cuantos ejemplos son correctamente predichos por al menos uno de los clasificadores base.

Oráculo Sobre-ajustado. Este modelo oráculo computa todas las combinaciones de salidas de los modelos base y asigna a cada combinación la etiqueta más frecuente encontrada en el conjunto correspondiente de etiquetas correctas. Este modelo falla cuando la misma combinación de modelos base tiene que producir diferentes etiquetas para que todas las posibles entradas sean clasificadas correctamente.

$$O^{(i)}_{sobreajustado}(M) = \textbf{max\_count}\left(\left\{ y_k \,\middle|\, (x_k, y_k) \in D, \underset{m^{(j)} \in M}{\forall} y_k^{(j)} = y_i^{(j)} \right\}\right)$$

La función **max_count** devuelve el elemento más frecuente de la colección (en caso de un empate, siempre devuelve el primero que encuentra).

Los resultados que obtiene este ensemble, proveen una cota superior del mejor rendimiento que puede ser obtenido utilizando un conjunto de reglas **consistente** para ensamblar la colección de modelos base.

En esta primera etapa experimental se desea validar que nuestro sistema es capaz de generar modelos base en la primera fase que al ser ensamblados en la segunda fase dan lugar a un ensemble que obtiene mejores resultados sobre el conjunto de evaluación que cualquiera de los modelos base de los que se compone. Para ello, los resultados obtenidos por el ensemble producido por nuestro sistema son comparados con los obtenidos por el mejor de los modelos base, y analizados respecto a los resultados obtenidos por el *Oráculo Sobre-ajustado* descrito anteriormente.

Además, en esta etapa experimental se desea también comprobar la influencia de las diferentes métricas de diversidad utilizadas en la capacidad del sistema para producir modelos base que generalicen luego de ser ensamblados. Con este propósito, se realiza un análisis de que tan bien los modelos base cubren el espacio de entrada de nuestro problema. Para este análisis resulta de suma utilidad el concepto de *Oráculo*

Optimista, el cual nos darán información acerca de la influencia de las distintas métricas de diversidad en la distribución de los distintos modelos base sobre el espacio de entrada de nuestros datos. Los resultados obtenidos por nuestro sistema son entonces comparados con aquellos del *Oráculo Optimista* para cada una de las métricas de diversidad utilizadas.

Finalmente, se analiza también en esta etapa el comportamiento del sistema bajo diferentes combinaciones de otros hiperparámetros como la cantidad de modelos base. La mejor configuración de dichos hiperparámetros, incluyendo la métrica de diversidad que mejores resultados proporciona al sistema, son utilizados en la segunda etapa experimental.

La **segunda etapa experimental** consiste en el estudio de la capacidad del sistema para lograr producir modelos de ensemble que son precisos de acuerdo a una función de pérdida determinada y justos según una o varias métricas de equidad. Con este fin se realiza una comparación entre los resultados obtenidos por nuestro sistema y los obtenidos por varios otros sistemas que resultan relevantes en la literatura para la solución de este problema.

En el contexto de mitigación de sesgos, múltiples métodos *ad-hoc* o dependientes del modelo han sido propuestos. Estos métodos tratan de forzar equidad durante el entrenamiento y guían el proceso de optimización de los parámetros para que los modelos resultantes sean efectivos y justos respecto a una métrica de equidad predeterminada. Uno de los objetivos de esta etapa experimental es precisamente comprobar si nuestro sistema es lo suficientemente poderoso como para tener mejor rendimiento que estos sistemas que influencian directamente el entrenamiento de los modelos con objetivos de mejorar los resultados de equidad obtenidos.

Alternativamente existen estrategias en la literatura, agnósticas al modelo, para lograr resultados justos. Algunas realizan un preprocesamiento sobre la colección de datos para luego poder ajustar un modelos a los mismos. Otras, como *Fair Bayesian Optimization*, realizan un ajuste de los hiperparámetros del modelo deseado. Estas propuestas, al igual que la nuestra, son muy versátiles, pues no imponen restricciones sobre el tipo de modelo a utilizar. Por tanto, es de interés el estudio del rendimiento de nuestro sistema frente a estos enfoques para lograr modelos eficaces y justos.

Además, una de las ventajas de nuestro sistema es que permite optimizar simultáneamente varias métricas de equidad. Por lo que se contrastan los resultados de nuestro método en este escenario con los de *Fair Bayesian Optimization*, el cual por sus características también permite múltiples restricciones de equidad.

Finalmente se desea estudiar la capacidad de nuestro sistema de encontrar diferentes balances de equidad y precisión, no solo cumplir con una restricción de equidad. Para ello nuestro sistema es comparado con otros métodos de optimización multiobjetivo relevantes en la literatura y que han mostrado buenos resultados en la solución de este problema.

3.1.1. Escenarios de Evaluación

La primera etapa experimental evalúa el sistema en la tarea *HAHA 2019*(*Humor Analysis based on Human Annotation*), con marco en el evento *IberLEF 2019* [15]. El proceso de evaluación consiste primero de la ejecución de la primera fase de nuestro sistema, esto es, la exploración del espacio de modelos y obtención del conjunto de modelos base a ser ensamblados. Posteriormente la segunda fase de nuestro sistema es ejecutada, utilizando como única función objetivo la función de pérdida que fue utilizada en la obtención de los modelos base. La función objetivo a optimizar en ambas fases del sistema es la precisión del sistema. Como parte de esta etapa se realizan múltiples ejecuciones con diferentes configuraciones de hiperparámetros. La configuración de hiperparámetros para la cual se observen los mejores resultados en esta etapa, sera la utilizada en la segunda etapa experimental, como se describe a continuación.

La segunda etapa experimental consiste en la ejecución de extremo a extremo de nuestro sistema, utilizando la colección de datos *Adult* [24] como escenario de evaluación. En esta etapa se utiliza la mejor configuración de hiperparámetros acorde a los resultados de la primera fase, es relevante destacar que entre estos hiperparámetros preseleccionados en la etapa anterior se encuentran la medida de diversificación a utilizar en la selección de modelos base y la cantidad máxima de estos. Ambas fases optimizan la precisión del modelo, en particular la segunda fase incorpora al proceso de optimización métricas de equidad, dígase *Statistical Parity* y *Equalized Opportunity*, como funciones objetivo adicionales a optimizar simultáneamente con la precisión.

3.1.2. Corpus de Evaluación

Dos conjuntos de datos son utilizados para la evaluación del sistema en los experimentos realizados.

HAHA 2019. Colección de datos utilizada en la tarea *HAHA 2019* (*Humor Analysis based on Human Annotation*), con marco en *IberLEF 2019* [15]. El corpus contiene 30 000 tweets en Español clasificados manualmente, de los cuales 24 000 son para entrenamiento y 6 000 para evaluación. Cada uno de estos tweets es clasificado en *gracioso* o *no-gracioso*.

La colección de datos es anotada a partir de asignar una clasificación de *gracioso* o *no-gracioso* a cada tweet, en caso de ser *gracioso* se da una puntuación de $[1, 5]$ de cuan *gracioso* es dicho tweet.

Tabla 3.1: Composición de la colección de datos de la tarea *HAHA 2019* en términos de cantidad de votos para cada clase.

	Entrenamiento	Evaluación	Total
Tweets	24 000	6 000	30 000
Graciosos	9 253	2 342	11 595
No graciosos	14 757	3 658	18 405
Puntuación promedio	2.04	2.03	2.04
Total de Votos	59 440	13 605	73 045
Votos 1	19 063	4 818	23 881
Votos 2	14 713	3 777	18 490
Votos 3	10 206	2 649	12 855
Votos 4	4 493	1 122	5 615
Votos 5	1 305	275	1 580

Adult. La colección de datos *Adult* [24] presenta información extraída del censo de 1994 en los Estados Unidos por Barry Becker. Los datos contienen detalles personales de los individuos, tales como nivel de educación, horas de trabajo a la semana, raza, sexo, etc. El objetivo es predecir si el individuo ganará un salario mayor a `$50K` al año. Hay un total de 48 842 filas de datos, y de estas, 3 620 contienen casillas con valores desconocidos, dejando 45 222 filas completas. Existen dos clases en las cuales clasificar a los individuos dependiendo de su salario anual, estas son, `>50K` o $\leq$`50K`. Las clases están desbalanceadas, con una tendencia hacia la etiqueta `<50K`, la cual representa aproximadamente el 75 % de los ejemplos.

3.1.3. Configuración Experimental

En todos los experimentos, el sistema fue configurado para permitir que cada fase ejecutara a lo sumo 10 000 iteraciones o por una hora. Los parámetros de búsqueda de AutoGOAL fueron los siguientes:

- `popsize=50`
- `selection=10`
- `cross_validation_steps=3`
- `validation_split=0.3`

Debido a limitaciones de infraestructura, los algoritmos de aprendizaje profundo fueron excluidos del conjunto de algoritmos disponibles para AutoGOAL. Por ejemplo, flujos basados en *Keras* y *BERT* fueron excluidos y fundamentalmente flujos basados en *ScikitLearn* fueron utilizados.

No incluir los algoritmos de aprendizaje profundo en la configuración experimental puede tener un impacto negativo en el rendimiento máximo que puede ser alcanzado por el sistema. Es decir, la precisión del sistema no puede ser comparada directamente con otras soluciones reportadas, en términos de magnitud, pues aquellas soluciones que utilizan técnicas de aprendizaje profundo tienen una ventaja inherente en problemas donde modelos más simples no son tan competitivos. Sin embargo, la ausencia de estos algoritmos no deberían afectar la capacidad del sistema de mejorar el rendimiento respecto a los modelos base encontrados en la primera fase. Por tanto, si el sistema es capaz de mejorar el rendimiento de los modelos base, entonces el rendimiento del sistema se espera que mejore aún más una vez que las arquitecturas de aprendizaje profundo sean compatibles. Esto se espera que ocurra no solo porque los modelos base ahora tendrían mejor rendimiento, sino también porque arquitecturas de ensemble más poderosas serían añadidas al mismo tiempo sin esfuerzo alguno. AutoGOAL ya ha probado lograr resultados competitivos cuando es configurado correctamente [28]. Además, es importante destacar que, en el caso en que se optimiza buscando un buen balance entre la precisión y las métricas de equidad, por ejemplo en la segunda fase de nuestro sistema, no necesariamente modelos más poderosos (como los de aprendizaje profundo) implican mejores resultados en dicho balance. Por tanto, la comparativa con otros métodos de mitigación no debería verse afectada significativamente.

Biblioteca

El sistema propuesto en este trabajo es parte de la biblioteca en desarrollo `BFair`[1] La biblioteca tiene como objetivo atacar los problemas de sesgos que emergen de entrenar modelos de *Aprendizaje Automático* que en datos que muestran sesgos de los humanos.

Hardware

Los experimentos fueron ejecutados en un equipo con las siguientes propiedades: CPU Intel Core i9-9900K (-MT-MCP-) con velocidad máxima de 3651/5000 MHz, cache de 16384KB y RAM de 64GB.

[1] https://github.com/bfair-ml/bfair

3.2. Primera Etapa Experimental

A continuación, la sección 3.2.1 muestra los resultados de los obtenidos a partir de realizar los experimentos de la forma descrita en la sección 3.1 para la tarea *HAHA 2019*. Luego la sección 3.2.2 realiza un análisis en profundidad de dichos resultados y arriba a conclusiones a partir de los mismos.

3.2.1. Resultados

Las tablas 3.2, 3.3, 3.4, 3.5 resumen los resultados obtenidos por el sistema en la tarea *HAHA 2019*, para configuraciones con número máximo de clasificadores base siendo 5, 20, 50 y 100 respectivamente. El sistema fue configurado para utilizar en ambas fases una función de pérdida basada en F_1, la cual fue propuesta como métrica de puntuación en la descripción de la tarea. Cinco estrategias de control de población fueron evaluadas con el objetivo de establecer comparaciones, de las cuales dos son las métricas de diversidad discutidas en la sección 2.2.1, y las tres restantes son las implementaciones de referencia del método reselect presentados en la sección 2.2. Cada tabla muestra la métrica F_1 alcanzada por: (i) los oráculos optimista y sobreajustados (sección 3.1); (ii) el modelo obtenido a partir de ensamblar los modelos base (sección 2.3); y, (iii) el mejor modelo base encontrado (sección 2.2). Además, el tipo de algoritmo de ensemble utilizado por el mejor de los modelos de ensemble encontrados es adicionado al final de cada tabla.

Tabla 3.2: HAHA 2019. Máximo número de modelos base es 5. Cada columna muestra el resultado obtenido utilizando la estrategia de selección de modelos base correspondiente. A^* y E^* representan el modelo base de mejor rendimiento y la mejor configuración de ensemble encontrada, respectivamente. Tipos de ensemble: *voting*, *overfit*, y *learning*, representan a *Voting Classifier*, *Overfitted Voting Classifier*, y *ML Voting Classifier*, respectivamente.

	shuffle	arbitrary	best	disagreement	double-fault
oráculo optimista (F_1)	0.996	0.917	0.893	1.000	0.970
oráculo sobreajustado (F_1)	0.715	0.711	0.744	0.748	0.754
A^* (F_1, entrenamiento)	0.989	0.973	0.945	0.846	0.876
A^* (F_1, evaluación)	0.690	0.711	0.722	0.749	0.757
E^* (F_1, entrenamiento)	0.913	0.961	0.917	0.846	0.941
E^* (F_1, evaluación)	0.731	0.726	0.755	0.749	0.761
tipo de ensemble	voting	learning	learning	voting	voting

Tabla 3.3: HAHA 2019. Máximo número de modelos base es 20. Cada columna muestra el resultado obtenido utilizando la estrategia de selección de modelos base correspondiente. A^* y E^* representan el modelo base de mejor rendimiento y la mejor configuración de ensemble encontrada, respectivamente. Tipos de ensemble: *voting*, *overfit*, y *learning*, representan a *Voting Classifier*, *Overfitted Voting Classifier*, y *ML Voting Classifier*, respectivamente.

	shuffle	arbitrary	best	disagreement	double fault
oráculo optimista (F_1)	1.000	1.000	0.936	1.000	0.998
oráculo sobreajustado (F_1)	0.729	0.771	0.831	0.735	0.889
A^* (F_1, entrenamiento)	0.876	0.901	0.855	0.875	0.856
A^* (F_1, evaluación)	0.705	0.721	0.759	0.732	0.755
E^* (F_1, entrenamiento)	0.870	0.930	0.883	0.875	0.942
E^* (F_1, evaluación)	0.719	0.740	0.765	0.732	0.767
tipo de ensemble	learning	overfit	voting	learning	voting

Tabla 3.4: HAHA 2019. Máximo número de modelos base es 50. Cada columna muestra el resultado obtenido utilizando la estrategia de selección de modelos base correspondiente. A^* y E^* representan el modelo base de mejor rendimiento y la mejor configuración de ensemble encontrada, respectivamente. Tipos de ensemble: *voting*, *overfit*, y *learning*, representan a *Voting Classifier*, *Overfitted Voting Classifier*, y *ML Voting Classifier*, respectivamente.

	shuffle	arbitrary	best	disagreement	double fault
oráculo optimista (F_1)	1.000	1.000	0.978	1.000	1.000
oráculo sobreajustado (F_1)	0.953	0.950	0.927	0.996	0.969
A^* (F_1, entrenamiento)	1.000	0.928	0.877	0.857	0.913
A^* (F_1, evaluación)	0.639	0.720	0.720	0.750	0.749
E^* (F_1, entrenamiento)	1.000	0.924	0.921	1.000	0.923
E^* (F_1, evaluación)	0.741	0.736	0.731	0.747	0.756
tipo de ensemble	overfit	learning	voting	overfit	voting

Tabla 3.5: HAHA 2019. Máximo número de modelos base es 100. Cada columna muestra el resultado obtenido utilizando la estrategia de selección de modelos base correspondiente. A^* y E^* representan el modelo base de mejor rendimiento y la mejor configuración de ensemble encontrada, respectivamente. Tipos de ensemble: *voting*, *overfit*, y *learning*, representan a *Voting Classifier*, *Overfitted Voting Classifier*, y *ML Voting Classifier*, respectivamente.

	shuffle	arbitrary	best	disagreement	double fault
oráculo optimista (F_1)	1.000	1.000	0.997	1.000	1.000
oráculo sobreajustado (F_1)	0.988	0.992	0.984	0.998	0.988
A^* (F_1, entrenamiento)	0.998	0.856	0.853	0.902	0.920
A^* (F_1, evaluación)	0.683	0.748	0.759	0.745	0.750
E^* (F_1, entrenamiento)	1.000	1.000	1.000	0.970	0.940
E^* (F_1, evaluación)	0.756	0.745	0.761	0.728	0.743
tipo de ensemble	overfit	overfit	overfit	learning	voting

Como puede observarse, **los mejores resultados fueron obtenidos cuando el máximo número de clasificadores base esta entre 20 y 50 y la estrategia de selección de modelos base es *double-fault*.** A continuación, la sección 3.2.2 profundiza en los resultados presentados en esta sección 3.2.1.

3.2.2. Discusión

Algunos patrones interesantes pueden ser observados a partir de analizar el rendimiento de los ensembles oráculo para diferentes estrategias de reselección de clasificadores base. La tabla 3.6 resume el rendimiento de los oráculos optimista y sobreajustado en el conjunto de evaluación. Algunos de estos patrones son presentados a continuación:

- La métrica *disagreement* asegura la máxima cobertura del conjunto de entrenamiento sin importar el número de clasificadores base seleccionado. Esto tiene sentido dado que la medida de *disagreement* premia la reselección de modelos que tienen predicciones con conflictos entre sí. Observando el rendimiento del oráculo sobreajustado, se puede notar que a pesar de que provee un cubrimiento máximo, no hay un conjunto de reglas **consistente** que pueda ser aplicado para explotar dicho cubrimiento.

- La métrica *double-fault* provee el rendimiento más consistente para ambos el oráculo optimista y sobreajustado, en especial cuando el número de clasificado-

res es bajo. Esto sugiere que la estrategia de diversificación basada en *double-fault* puede proveer el mejor rendimiento en la subsecuente fase de ensamblado, dado que obtiene una mayor puntuación cuando se utiliza un conjunto de reglas consistente.

- El cubrimiento mostrado por los ensembles oráculo se incrementa significativamente según se incrementa el número de clasificadores base, independientemente de la estrategia de reselección. Esto tiene sentido dado que un mayor conjunto de votantes incrementa la probabilidad de encontrar uno que correctamente clasifique los ejemplos de mayor conflicto si hay cierta diversidad entre los votantes. Esta idea es apoyada por el hecho de que la estrategia que reselecciona de forma golosa solo los modelos de mejor rendimiento es la que logra el menor cubrimiento (de acuerdo al ensemble optimista).

Tabla 3.6: Resumen de las métricas en los oráculos para cada estrategia de selección de modelos base según el número máximo de clasificadores base se incrementa.

Oráculo Optimista (F_1)					
n-clasificadores	shuffle	arbitrary	best	disagreement	double fault
5	0.996	0.917	0.893	1.000	0.970
20	1.000	1.000	0.936	1.000	0.998
50	1.000	1.000	0.978	1.000	1.000
100	1.000	1.000	0.997	1.000	1.000
Oráculo Sobreajustado (F_1)					
n-clasificadores	shuffle	arbitrary	best	disagreement	double fault
5	0.715	0.711	0.744	0.748	0.754
20	0.729	0.771	0.831	0.735	0.889
50	0.953	0.950	0.927	0.996	0.969
100	0.988	0.992	0.984	0.998	0.988

La tabla 3.7 resume las diferencias entre el rendimiento logrado por el mejor ensemble encontrado E^* y el mejor modelo base A^*, en ambas la colección de entrenamiento y de evaluación. Es notable que los modelos de ensemble son capaces de sobrepasar a sus modelos base en la colección de entrenamiento, incluso cuando algunas estrategias requieren un mayor número de modelos base para lograr esto. La estrategia de *double-fault* es capaz de mejorar el rendimiento incluso con un número bajo de clasificadores base. Esto tiene sentido dado que esta estrategia penaliza al

sistema por construir una colección en la cual todos los modelos base fallen en los mismos ejemplos. Por tanto, estos resulta en una colección de modelos base, en la cual cada modelo arregla los fallos del otro. Este comportamiento es diferente al que muestra la estrategia de *disagreement*, esta premia a los modelos base por tener diferentes predicciones independientemente de si estas eran correctas o incorrectas. Esto puede llevar a situaciones en las cuales es difícil para el ensemble decidir en cual de los modelos base confiar, especialmente si el número de modelos base es muy bajo.

Aún más importante, la tabla 3.7 también muestra que la optimización de ensemble es capaz de producir ensembles que generalizan mejor que sus modelos base, es decir, los ensembles tienen mejor rendimiento en el conjunto de evaluación. Esto es particularmente cierto cuando el número de clasificadores base no es muy grande. Según incrementa el número de clasificadores base, también lo hace el número de diferentes combinaciones, y por tanto, el ensemble es más susceptible a sobre-ajustar el conjunto de entrenamiento.

Tabla 3.7: Resumen de la diferencia en rendimiento entre el mejor modelo base encontrado (A^*) y el ensemble correspondiente (E^*), para cada estrategia de selección de modelo base según incrementa el número de clasificadores base.

$(E^* - A^*)$ en entrenamiento					
n-clasificadores	shuffle	arbitrary	best	disagreement	double fault
5	-0.076	-0.012	-0.027	0.000	0.065
20	-0.006	0.030	0.029	0.000	0.086
50	0.000	-0.004	0.044	0.143	0.010
100	0.002	0.144	0.147	0.069	0.020
$(E^* - A^*)$ en evaluación					
n-clasificadores	shuffle	arbitrary	best	disagreement	double fault
5	0.040	0.015	0.033	0.000	0.004
20	0.014	0.018	0.005	0.000	0.012
50	0.102	0.015	0.012	-0.003	0.006
100	0.073	-0.003	0.002	-0.017	-0.007

Algunas otras ideas que resaltan de la tabla 3.7 se resumen a continuación:

- Las métricas de diversidad, dígase *disagreement* y *double-fault*, tienen un mayor impacto en el rendimiento del ensemble en el conjunto de entrenamiento cuando el número de clasificadores base es bajo. Esto tiene sentido dado que mientras menor cantidad de modelos en la colección de modelos base, más difícil será

encontrar una *buena* selección de forma arbitraria. Sin embargo, las medidas de diversidad no parecen tener un impacto en las capacidades de generalización del ensemble.

- A la estrategia de *disagreement* le resulta difícil mejorar el rendimiento, tanto en el conjunto de entrenamiento como en el de evaluación. Esto es particularmente cierto cuando el número de clasificadores base es bajo. Esta estrategia no penaliza a los modelos por realizar predicciones erróneas y en su lugar los premia si sus predicciones son diferentes a la del resto de los clasificadores. Luego, es usual que el ensemble simplemente decida confiar en la predicción de solo uno de los clasificadores base, y por tanto el ensemble produce los mismos resultados que su mejor modelo base. Esto muestra que lograr el mejor cubrimiento de acuerdo al oráculo optimista, como lo logra la medida de *disagreement*, no es necesariamente útil.

La tabla 3.8 resume el rendimiento obtenido por el ensemble E^* según el número de clasificadores base y las estrategias de reselección cambian. La estrategia de *double-fault* siempre logra los mejores resultados, excepto en el ultimo escenario, en el cual el número de clasificadores es el más alto. En ese caso, aunque el ensemble obtuvo un mejor rendimiento que sus modelos base en la colección de entrenamiento, el número alto de votantes puede haber afectado sus capacidades de generalización. dado que el espacio de posibles combinaciones de votantes es mayor.

Tabla 3.8: Resumen del rendimiento obtenido por el ensemble (E^*) en el conjunto de entrenamiento y evaluación, para cada estrategia de selección de modelos base según el número máximo de clasificadores incrementa.

E^* en entrenamiento					
n-clasificadores	shuffle	arbitrary	best	disagreement	double fault
5	0.913	**0.961**	0.917	0.846	0.941
20	0.870	0.930	0.883	0.875	**0.942**
50	**1.000**	0.924	0.921	**1.000**	0.923
100	**1.000**	**1.000**	**1.000**	0.970	0.940
E^* @ evaluación					
n-clasificadores	shuffle	arbitrary	best	disagreement	double fault
5	0.731	0.726	0.755	0.749	**0.761**
20	0.719	0.740	0.765	0.732	**0.767**
50	0.741	0.736	0.731	0.747	**0.756**
100	0.756	0.745	**0.761**	0.728	0.743

Como es de esperarse, la ausencia de algoritmos de aprendizaje profundo en el conjunto de métodos disponibles tuvo un impacto negativo en el mayor rendimiento alcanzado. La arquitectura obtenida con mayor rendimiento logra alcanzar 0,767 F_1, mientras que la mejor solución reportada por AutoGOAL en la tarea *HAHA 2019* es 0,789 [28], la cual utiliza algoritmos de aprendizaje profundo - una estrategia de pre-procesamiento utilizando BERT [19] y una red neuronal con 2 nodos recurrentes (una capa *BiLSTM* y una *LSTM*) seguidas de dos capas densas distribuidas en tiempo -. En comparación, la mejor arquitectura obtenida en estos experimentos consiste de un *Voting Classifier* que ensambla un conjunto de simples modelos de *ScikitLearn* - una estrategia de pre-procesamiento utilizando tokenización (*BlanklineTokenizer*, *TreebankWordTokenizer*, etc.) y algoritmos de vectorización (*TfIdfVectorizer*, *CountVectorizer*, etc.) seguidos de una capa de clasificación (*Perceptron*, *LinearSVC*, *NearestCentroid*, *MultinomialNB*, etc.) -. A pesar de utilizar una arquitectura mucho más simple, resultados competitivos son alcanzados.

Adicionalmente, como se muestra en la tabla 3.7, la optimización utilizando ensembles es capaz de producir ensembles que generalizan mejor que sus modelos base. Luego, se espera que el rendimiento de los modelos producidos por nuestro sistema mejore una vez que algoritmos más poderosos estén disponibles.

Las tablas 3.9 y 3.10 resumen los diferentes tipos de ensemble que fueron encontrados como óptimos en cada experimento. La técnica de ensemble "ML Voting

Classifier" no parece ser utilizada por ninguna estrategia de reselección de forma consistente. La técnica de ensemble "Voting Classifier" es utilizada consistentemente en las estrategias basadas en *double-fault*, y también parece ser utilizada frecuentemente cuando el número de clasificadores base es bajo. Además, la técnica de ensemble "Overfitted Voting Classifier" parece ser utilizada más cuando el número de clasificadores es bajo.

Tabla 3.9: Resume la estrategia de ensemble utilizada por la mejor configuración de ensemble encontrada (E^*) para cada estrategia de selección de modelos base según el número máximo de clasificadores base incrementa. Tipos de ensemble: *voting*, *overfit*, y *learning*, representan *"Voting Classifier"*, *"Overfitted Voting Classifier"*, y *"ML Voting Classifier"*, respectivamente.

n-clasificadores	shuffle	arbitrary	best	disagreement	double fault
5	voting	learning	learning	voting	voting
20	learning	overfit	voting	learning	voting
50	overfit	learning	voting	overfit	voting
100	overfit	overfit	overfit	learning	voting

Tabla 3.10: Frecuencia de uso de cada técnica de ensemble. Las columnas *voting*, *overfit*, y *learning*, representan *"Voting Classifier"*, *"Overfitted Voting Classifier"*, y *"ML Voting Classifier"*, respectivamente.

n-clasificadores	voting	overfitted	learning
5	3	0	2
20	2	1	2
50	2	2	1
100	1	3	1

Para concluir, se muestra que el sistema brinda los resultados más prometedores cuando es configurado para realizar la búsqueda sobre una colección de 20 o 50 modelos base y utilizando la estrategia de selección de modelos base *double-fault*.

3.3. Segunda Etapa Experimental

A continuación, la sección 3.3.1 muestra los resultados de los obtenidos a partir de realizar los experimentos de la forma descrita en la sección 3.1 utilizando la colección

de datos *Adult*. Luego la sección 3.3.2 realiza un análisis en profundidad de dichos resultados y arriba a conclusiones a partir de los mismos.

3.3.1. Resultados

La tabla 3.11 muestra una comparativa de nuestro enfoque con los métodos agnósticos (últimas tres filas), y los no agnósticos al modelo (primeras tres filas). La tabla 3.12 muestra una comparativa con *FBO* optimizando simultáneamente *Statistical Parity*, *Equal Oportunity* y *Equalized Odds*. La tabla 3.13 contrasta los resultados obtenidos por nuestro sistema con los obtenidos por otros métodos de optimización multiobjetivo aplicados al problema de mitigación de sesgos. Por último la figura 3.1 permite visualizar la aproximación del *Frente Pareto* encontrado por nuestro sistema.

Tabla 3.11: Evaluación en *Adult*, con restricción sobre *statistical parity* como métrica de equidad. El máximo número de clasificadores utilizado es 20. La estrategia de selección de modelos base es *double-fault*. *Zafar* se refiere al método propuesto por *Zafar et al.* [63]. *Error* se refiere a la pérdida del modelo en el conjunto de validación.

Método	Error	Statistical Parity
FERM	$0{,}164 \pm 0{,}010$	$\leq 0{,}1$
Zafar	$0{,}187 \pm 0{,}001$	$\leq 0{,}1$
Adversarial debiasing	$0{,}237 \pm 0{,}001$	$\leq 0{,}1$
FERM Preprocesamiento	$0{,}228 \pm 0{,}013$	$\leq 0{,}1$
SMOTE	$0{,}178 \pm 0{,}005$	$\leq 0{,}1$
FairBO	$0{,}175 \pm 0{,}007$	$\leq 0{,}1$
BFair (20)	0,170	0,087

Tabla 3.12: Evaluación en *Adult*, con restricción en las tres medidas de equidad. El máximo número de clasificadores utilizado es 20. La estrategia de selección de modelos base es *double-fault*. *RF*, *NN*, *LL* se refieren a *Random Forests*, *Redes Neuronales* y *Linear Learner* respectivamente.

método	precisión	statistical parity	equal opportunity	equalized odds
FBO (RF)	$\approx 0{,}785$	$\leq 0{,}05$	$\leq 0{,}05$	$\leq 0{,}05$
FBO (XGBoost)	$\approx 0{,}835$	$\leq 0{,}05$	$\leq 0{,}05$	$\leq 0{,}05$
FBO (NN)	$\approx 0{,}775$	$\leq 0{,}05$	$\leq 0{,}05$	$\leq 0{,}05$
FBO (LL)	$\approx 0{,}810$	$\leq 0{,}05$	$\leq 0{,}05$	$\leq 0{,}05$
BFair (20)	0.817	0.049	0.014	0.007

Tabla 3.13: Evaluación en *Adult*, con restricción sobre *statistical parity* como métrica de equidad. El máximo número de clasificadores utilizado aparece entre paréntesis. La estrategia de selección de modelos base es *double-fault*. *Dragonfly* se refiere a la implementación en la biblioteca del mismo nombre, del método propuesto por *Paria et al.* [50].

Método	Precisión	statistical parity
ParEGO	$\approx 0{,}764$	$\approx 0{,}0175$
Dragonfly	$\approx 0{,}763$	$\approx 0{,}0188$
BFair (20)	0,813	0,0487
BFair (20)	0,803	0,0332
BFair (50)	0,800	0,0269
BFair (50)	0,777	0,0072
BFair (50)	0,777	0,0072
BFair (20)	0,764	0,0000

A continuación, la sección 3.3.2 profundiza en los resultados presentados en esta sección 3.3.1.

3.3.2. Discusión

Los resultados observados en la tabla 3.11 muestran que nuestro sistema es sumamente competitivo y en la mayoría de los casos superior que el resto de los métodos de mitigación de sesgos con los cuales es comparado. El enfoque propuesto obtiene mejores resultados que todos los métodos agnósticos al modelo con los que fue comparado,

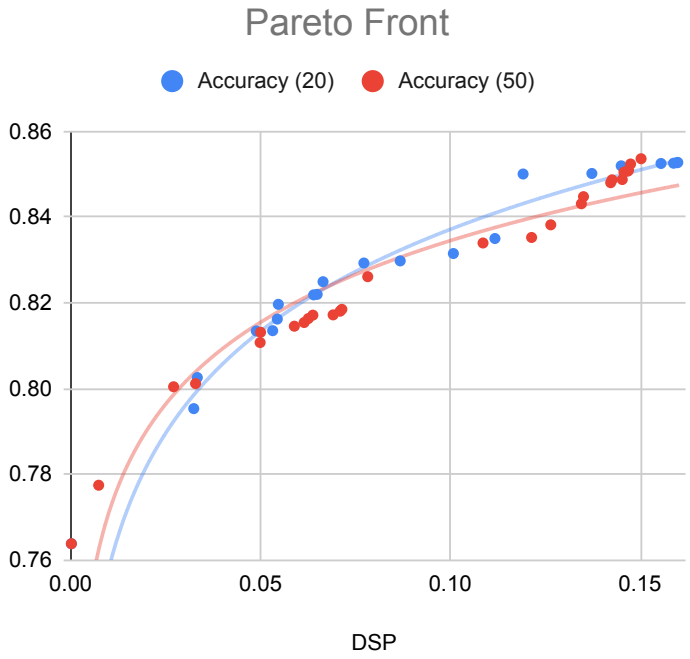

Figura 3.1: Aproximación del *Frente Pareto* encontrada por el sistema.

estos son, *FERM Preprocesamiento*, *SMOTE* y *FairBO*. Comparado con métodos que modifican directamente el proceso de optimización de los modelos para ajustarlos a los objetivos de equidad, nuestro sistema también muestra resultados satisfactorios, siendo solamente superado por *FERM*, y aun en este caso se mantiene muy competitivo. Esto ratifica el potencial de nuestro enfoque multiobjetivo para encontrar modelos que sean efectivos a la vez que son justos de acuerdo a la métrica de equidad seleccionada. Particularmente interesante es el hecho de que se obtienen resultados competitivos, incluso superiores a las estrategias que son *ad-hoc* y directamente modifican el método de optimización del modelo en cuestión. Solo el caso de *FERM* obtiene resultados mejores a los nuestros como se esperaba inicialmente dado que las restricciones de equidad son mantenidas durante el proceso de optimización. Sin embargo la diferencia es modesta y nuestro sistema logra ser muy competitivo mientras que es mucho más versátil al mantenerse agnóstico al modelo y método de entrenamiento. Adicionalmente es importante destacar que *Adversarial debiasing* (tercera fila) utiliza métodos de aprendizaje profundo, dígase *Redes Neuronales Adversariales*, mientras que nuestro sistema no incluye este tipo de modelos que sabemos son más poderosos en muchos escenarios.

La tabla 3.12 compara nuestra propuesta con *Fair Bayesian Optimization*. *FBO* consiste en un proceso de ajuste de hiperparámetros mediante optimización bayesiana para lograr los resultados de equidad requeridos. La flexibilidad de este método le permite hacer cumplir restricciones sobre diferentes métricas de equidad simultáneamente, por lo que nuestro sistema es comparado en este escenario con dicha propuesta. Como se puede observar nuestro sistema obtiene mejores resultados que la alternativa en la mayoría de los casos. Incluso para configuraciones de *FBO* basadas en redes neuronales, nuestro sistema fue capaz, una vez más de obtener significativamente mejores resultados aun cuando nuestras configuraciones no utilizan algoritmos de aprendizaje profundo. Debido a la forma en que *FBO* funciona, requiere un umbral para cada métrica y la única garantía que provee es que dichas métricas de equidad se mantendrán dentro de esta región factible luego de ajustar los hiperparámetros. En cambio nuestro enfoque no necesita que el usuario indique el umbral admisible para las métricas de equidad y es capaz de explorar los diferentes balances entre las diferentes métricas (tanto de equidad como de precisión) y permitir al usuario seleccionar el que considere conveniente. Adicionalmente puede observarse como a pesar de que nuestro sistema obtiene resultados muy cercanos al umbral para *statistical parity*, este a la vez obtiene considerablemente mejores resultados en el resto de las métricas de equidad. Esto quiere decir que nuestro sistema es capaz de mostrar resultados sumamente satisfactorios utilizando no solo una, pero varias métricas de equidad simultáneamente. Una vez más, la configuración que obtiene mejores resultados que los nuestros no supera nuestro modelo significativamente, además, se conoce que el algoritmo utilizado por este modelo (*XGBoost*) no forma parte del conjunto de algoritmos disponibles a AutoGOAL en nuestra configuración, lo que sugiere que la comparativa podría haber sido similar al resto de los casos de haber tenido acceso al mismo.

La tabla 3.13 muestra una comparación con otros métodos de optimización multiobjetivo que han sido aplicados en la literatura al problema de mitigación de sesgos. Una vez más nuestro sistema muestra ser competitivo en este ámbito, encontrando diferentes balances entre precisión y equidad. En todos los casos nuestro sistema supera los sistemas alternativos. Además se puede observar como nuestro sistema es capaz de encontrar soluciones que a pesar de que ceden de forma mínima sus valores de equidad son capaces de lograr significativamente mejores valores de precisión. Esto habla entre otras cosas de la habilidad de nuestro enfoque para cubrir diferentes regiones del *Frente Pareto*. La figura 3.1 permite verificar visualmente que en efecto nuestro modelo encuentra una aproximación del *Frente Pareto* que contiene soluciones con diferentes balances de precisión y equidad.

De forma general, como se ha podido observar, el sistema muestra un elevada capacidad para encontrar balances satisfactorios entre las métricas de equidad deseadas y la pérdida de los modelos. Todo esto mientras se mantiene extremadamente versátil, siendo agnóstico al modelo que se utilice, a las métricas que se desean optimizar,

incluso a la naturaleza del problema que se desea resolver.

Conclusiones

Este trabajo presenta un sistema de optimización de dos fases para resolver problemas de clasificación arbitrarios de forma justa. Los principales objetivos de este sistema son (1) sacar provecho de todas las soluciones que fueron producidas mientras se resuelve un problema de *AutoML*, y (2) utilizar estos modelos para construir una solución que además de efectiva sea justa en la toma de decisiones. La evaluación en los conjuntos de datos *Adult* y de la tarea *HAHA 2019* probaron que utilizando la estrategia de diversificación `double-fault` y el método de optimización aquí propuesto, se pueden construir modelos que tengan un buen rendimiento a la vez que mantienen un alto nivel de equidad. Luego, nuestro trabajo confirma la hipótesis de que una solución efectiva y justa puede ser construida a partir de ensamblar de forma inteligente un subconjunto de los modelos generados mientras se resuelve el problema de *AutoML*.

Dos métricas de diversidad tomadas de la literatura fueron estudiadas: la métrica de `disagreement` y `double-fault`. Para cuantificar la calidad de los resultados obtenidos a partir de utilizar estas métricas, dos métricas adicionales, basadas en el concepto de *ensembles oráculo*, son presentadas en este trabajo. Los resultados permiten observar como el proceso de selección de modelos base influencia el rendimiento de las técnicas de ensemble. La métrica de *disagreement* asegura el máximo cubrimiento de los datos, sin embargo, las colecciones construidas utilizando esta estrategia no necesariamente proveen salidas consistentes. La métrica de *double-fault* brinda los mejores resultados en general.

Una modificación al algoritmo de búsqueda de *AutoGOAL* fue propuesta para aceptar múltiples funciones objetivo. La capacidad de este para optimizar métricas de equidad y precisión simultáneamente fue evaluada en el conjunto de datos *Adult* y los resultados comparados con otros métodos de la literatura. Como era de esperarse, nuestra propuesta mostró ser sumamente competitiva y obtener resultados satisfactorios. En particular este sistema mostró tener la capacidad no solo de encontrar resultados que cumplan con determinadas restricciones de equidad, sino de encontrar diferentes balances entre equidad y precisión.

Los resultados de este trabajo son resumidos a continuación:

- Al utilizar técnicas de *AutoML* combinadas con métodos de ensemble y algoritmos multiobjetivo se logra un sistema que de forma agnóstica al modelo logra resolver problemas de clasificación arbitrarios de manera justa.
- Se mostró la influencia de las diferentes métricas de diversidad en la formación de modelos base que den lugar a ensembles más robustos.
- Al utilizar un método multiobjetivo para ensamblar un conjunto clasificadores base se logra encontrar modelos **justos** con alto rendimiento.
- Este método permite además trabajar con múltiples métricas de equidad simultáneamente.

Recomendaciones

Este trabajo está orientado a la resolución de problemas de clasificación arbitrarios de forma justa y automatizando la mayor parte del proceso. Sin embargo, una de las limitaciones del sistema propuesto es que requiere que el usuario manualmente indique los atributos protegidos en los datos. Por lo que se propone integrar el sistema aquí propuesto con propuestas actualmente en desarrollo para la anotación automática de los atributos protegidos de un conjunto de datos.

Actualmente la primera fase de nuestro sistema utiliza una estrategia golosa para computar la diversidad entre los clasificadores y de esta forma seleccionar los modelos base que serán ensamblados. Este método sin embargo puede ser subóptimo, se recomienda el estudio de otras estrategias, como métodos de *clustering* como alternativa. La experimentación realizada en este trabajo no estudió el comportamiento del sistema utilizando métodos de aprendizaje profundo. Se sugiere la exploración en futuros trabajos de como estas técnicas podrían afectar el rendimiento del sistema. Finalmente la experimentación en conjuntos de datos con características diferentes a los aquí empleados es imprescindible para corroborar el comportamiento del sistema en problemas de mayor escala.

Bibliografía

[1] Alekh Agarwal, Miroslav Dudík y Zhiwei Steven Wu. «Fair regression: Quantitative definitions and reduction-based algorithms». En: *International Conference on Machine Learning*. PMLR. 2019, págs. 120-129 (vid. pág. 8).

[2] Alekh Agarwal y col. «A reductions approach to fair classification». En: *International Conference on Machine Learning*. PMLR. 2018, págs. 60-69 (vid. pág. 8).

[3] J. Angwin y col. *Machine bias. propublica*. 2016. URL: `https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing` (vid. págs. 1, 5).

[4] Thomas Back. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996 (vid. pág. 14).

[5] Dzmitry Bahdanau, Kyunghyun Cho y Yoshua Bengio. «Neural machine translation by jointly learning to align and translate». En: *arXiv preprint arXiv:1409.0473* (2014) (vid. pág. 1).

[6] Solon Barocas, Moritz Hardt y Arvind Narayanan. *Fairness and Machine Learning: Limitations and Opportunities*. `http://www.fairmlbook.org`. fairmlbook.org, 2019 (vid. pág. 1).

[7] Tolga Bolukbasi y col. «Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings». En: *Advances in Neural Information Processing Systems*. Ed. por D. Lee y col. Vol. 29. Curran Associates, Inc., 2016. URL: `https://proceedings.neurips.cc/paper/2016/file/a486cd07e4ac3d270571622f4f316ec5-Paper.pdf` (vid. pág. 1).

[8] Tolga Bolukbasi y col. «Man is to computer programmer as woman is to homemaker? debiasing word embeddings». En: *Advances in neural information processing systems* 29 (2016) (vid. pág. 5).

[9] Leo Breiman. «Bagging predictors». En: *Machine learning* 24.2 (1996), págs. 123-140 (vid. pág. 10).

[10] Joy Buolamwini y Timnit Gebru. «Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification». En: *FAT*. 2018 (vid. pág. 1).

[11] Aylin Caliskan, Joanna J Bryson y Arvind Narayanan. «Semantics derived automatically from language corpora contain human-like biases». En: *Science* 356.6334 (2017), págs. 183-186 (vid. pág. 1).

[12] Flavio Calmon y col. «Optimized Pre-Processing for Discrimination Prevention». En: *Advances in Neural Information Processing Systems*. Ed. por I. Guyon y col. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/9a49a25d845a483fae4be7e341368e36-Paper.pdf (vid. pág. 7).

[13] Nitesh V Chawla y col. «SMOTE: synthetic minority over-sampling technique». En: *Journal of artificial intelligence research* 16 (2002), págs. 321-357 (vid. pág. 8).

[14] Silvia Chiappa y William S Isaac. «A causal Bayesian networks viewpoint on fairness». En: *IFIP International Summer School on Privacy and Identity Management*. Springer. 2018, págs. 3-20 (vid. pág. 8).

[15] Luis Chiruzzo y col. «Overview of HAHA at IberLEF 2019: Humor analysis based on human annotation.» En: *IberLEF@ SEPLN*. 2019, págs. 132-144 (vid. pág. 28).

[16] Alexandra Chouldechova y Aaron Roth. «The frontiers of fairness in machine learning». En: *arXiv preprint arXiv:1810.08810* (2018) (vid. pág. 5).

[17] Nihad Karim Chowdhury y col. «Machine learning for detecting COVID-19 from cough sounds: An ensemble-based MCDM method». En: *Computers in Biology and Medicine* 145 (2022), pág. 105405. ISSN: 0010-4825. DOI: https://doi.org/10.1016/j.compbiomed.2022.105405. URL: https://www.sciencedirect.com/science/article/pii/S0010482522001974 (vid. pág. 10).

[18] K. Deb y col. «A fast and elitist multiobjective genetic algorithm: NSGA-II». En: *IEEE Transactions on Evolutionary Computation* 6.2 (abr. de 2002), págs. 182-197. ISSN: 1941-0026. DOI: 10.1109/4235.996017 (vid. pág. 14).

[19] Jacob Devlin y col. «Bert: Pre-training of deep bidirectional transformers for language understanding». En: *arXiv preprint arXiv:1810.04805* (2018) (vid. pág. 37).

[20] Thomas G. Dietterich. «Ensemble Methods in Machine Learning». En: *Multiple Classifier Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, págs. 1-15. ISBN: 978-3-540-45014-6 (vid. pág. 10).

[21] Christos Dimitrakakis y col. «Bayesian Fairness». En: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01 (jul. de 2019), págs. 509-516. DOI: 10.1609/aaai.v33i01.3301509. URL: https://ojs.aaai.org/index.php/AAAI/article/view/3824 (vid. págs. 8, 11).

[22] Michele Donini y col. «Empirical Risk Minimization Under Fairness Constraints». En: *Advances in Neural Information Processing Systems*. Ed. por S. Bengio y col. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper/2018/file/83cdcec08fbf90370fcf53bdd56604ff-Paper.pdf (vid. pág. 8).

[23] Michele Donini y col. «Empirical risk minimization under fairness constraints». En: *Advances in Neural Information Processing Systems* 31 (2018) (vid. pág. 7).

[24] Dheeru Dua y Casey Graff. *UCI Machine Learning Repository*. 2017. URL: http://archive.ics.uci.edu/ml (vid. págs. 1, 28, 29).

[25] Cynthia Dwork y col. «Fairness through awareness». En: *Proceedings of the 3rd innovations in theoretical computer science conference*. 2012, págs. 214-226 (vid. pág. 6).

[26] Thomas Elsken, Jan Hendrik Metzen y Frank Hutter. «Neural Architecture Search: A Survey». En: (2018). DOI: 10.48550/ARXIV.1808.05377. URL: https://arxiv.org/abs/1808.05377 (vid. pág. 12).

[27] Suilan Estévez Velarde y col. «Automatic Discovery of Heterogeneous Machine Learning Pipelines: An Application to Natural Language Processing». En: ene. de 2020, págs. 3558-3568. DOI: 10.18653/v1/2020.coling-main.317 (vid. págs. 2, 12).

[28] Suilan Estevez-Velarde y col. «Automatic Discovery of Heterogeneous Machine Learning Pipelines: An Application to Natural Language Processing». En: *Proceedings of the 28th International Conference on Computational Linguistics*. 2020, págs. 3558-3568 (vid. págs. 30, 37).

[29] Suilan Estévez-Velarde y col. «General-purpose hierarchical optimisation of machine learning pipelines with grammatical evolution». En: *Information Sciences* (2020). DOI: 10.1016/j.ins.2020.07.035 (vid. pág. 12).

[30] Matthias Feurer y col. «Efficient and robust automated machine learning». En: *Advances in neural information processing systems* 28 (2015) (vid. pág. 11).

[31] Sorelle A Friedler, Carlos Scheidegger y Suresh Venkatasubramanian. «On the (im) possibility of fairness». En: *arXiv preprint arXiv:1609.07236* (2016) (vid. pág. 6).

[32] Ian Goodfellow y col. «Generative Adversarial Nets». En: *Advances in Neural Information Processing Systems*. Ed. por Z. Ghahramani y col. Vol. 27. Curran Associates, Inc., 2014. URL: https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf (vid. pág. 9).

[33] Claus Hillermeier y col. *Nonlinear multiobjective optimization: a generalized homotopy approach*. Vol. 135. Springer Science & Business Media, 2001 (vid. pág. 14).

[34] Gao Huang y col. «Snapshot Ensembles: Train 1, get M for free». En: *CoRR* abs/1704.00109 (2017). arXiv: 1704.00109. URL: http://arxiv.org/abs/1704.00109 (vid. pág. 10).

[35] Haifeng Jin, Qingquan Song y Xia Hu. «Efficient Neural Architecture Search with Network Morphism». En: *CoRR* abs/1806.10282 (2018). arXiv: 1806.10282. URL: http://arxiv.org/abs/1806.10282 (vid. págs. 2, 11).

[36] Weiqiu Jin y col. «A data-driven hybrid ensemble AI model for COVID-19 infection forecast using multiple neural networks and reinforced learning». En: *Computers in Biology and Medicine* 146 (2022), pág. 105560. ISSN: 0010-4825. DOI: https://doi.org/10.1016/j.compbiomed.2022.105560. URL: https://www.sciencedirect.com/science/article/pii/S0010482522003523 (vid. pág. 10).

[37] Donald R. Jones, Matthias Schonlau y William J. Welch. «Efficient Global Optimization of Expensive Black-Box Functions». En: *Journal of Global Optimization* 13.4 (dic. de 1998), págs. 455-492. ISSN: 1573-2916. DOI: 10.1023/A:1008306431147. URL: https://doi.org/10.1023/A:1008306431147 (vid. pág. 14).

[38] Faisal Kamiran y Toon Calders. «Data preprocessing techniques for classification without discrimination». En: *Knowledge and Information Systems* 33 (2011), págs. 1-33 (vid. pág. 7).

[39] Michael Kearns y col. «Preventing fairness gerrymandering: Auditing and learning for subgroup fairness». En: *International Conference on Machine Learning*. PMLR. 2018, págs. 2564-2572 (vid. pág. 8).

[40] *Keras*. URL: https://github.com/fchollet/keras (vid. pág. 12).

[41] Jon Kleinberg. «Inherent trade-offs in algorithmic fairness». En: *Abstracts of the 2018 ACM International Conference on Measurement and Modeling of Computer Systems*. 2018, págs. 40-40 (vid. pág. 6).

[42] J. Knowles. «ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems». En: *IEEE Transactions on Evolutionary Computation* 10.1 (2006), págs. 50-66. DOI: 10.1109/TEVC.2005.851274 (vid. pág. 14).

[43] Harold W Kuhn y Albert W Tucker. «Nonlinear programming». En: *Traces and emergence of nonlinear programming*. Springer, 2014, págs. 247-258 (vid. pág. 14).

[44] Ioannis E Livieris y col. «Ensemble deep learning models for forecasting cryptocurrency time-series». En: *Algorithms* 13.5 (2020), pág. 121 (vid. pág. 10).

[45] Edward Loper y Steven Bird. «NLTK: The Natural Language Toolkit». En: *CoRR* cs.CL/0205028 (2002). URL: https://arxiv.org/abs/cs/0205028 (vid. pág. 12).

[46] Mark MacCarthy. «Standards of Fairness for Disparate Impact Assessment of Big Data Algorithms». En: *Other Information Systems & eBusiness eJournal* (2018) (vid. pág. 7).

[47] Jessica Mégane, Nuno Lourenço y Penousal Machado. «Probabilistic grammatical evolution». En: *European Conference on Genetic Programming (Part of EvoStar)*. Springer. 2021, págs. 198-213 (vid. pág. 12).

[48] Kaisa Miettinen. *Nonlinear multiobjective optimization*. Vol. 12. Springer Science & Business Media, 2012 (vid. pág. 14).

[49] Volodymyr Mnih. *Machine learning for aerial image labeling*. University of Toronto (Canada), 2013 (vid. pág. 1).

[50] Biswajit Paria, Kirthevasan Kandasamy y Barnab ás Póczos. «A flexible framework for multi-objective bayesian optimization using random scalarizations». En: *Uncertainty in Artificial Intelligence*. PMLR. 2020, págs. 766-776 (vid. págs. 14, 40).

[51] Adam Paszke y col. «Pytorch: An imperative style, high-performance deep learning library». En: *Advances in neural information processing systems* 32 (2019) (vid. pág. 12).

[52] Fabian Pedregosa y col. «Scikit-learn: Machine learning in Python». En: *the Journal of machine Learning research* 12 (2011), págs. 2825-2830 (vid. págs. 11, 12).

[53] Valerio Perrone y col. «Fair Bayesian Optimization». En: *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*. AIES '21. Virtual Event, USA: Association for Computing Machinery, 2021, págs. 854-863. ISBN: 9781450384735. DOI: 10.1145/3461702.3462629. URL: https://doi.org/10.1145/3461702.3462629 (vid. pág. 8).

[54] Robi Polikar. «Ensemble based systems in decision making». En: *IEEE Circuits and systems magazine* 6.3 (2006), págs. 21-45 (vid. págs. 3, 9).

[55] Robert E Schapire. «The strength of weak learnability». En: *Machine learning* 5.2 (1990), págs. 197-227 (vid. pág. 10).

[56] Oliver Schütze, Alessandro Dell'Aere y Michael Dellnitz. «On Continuation Methods for the Numerical Treatment of Multi-Objective Optimization Problems». En: *Practical Approaches to Multi-Objective Optimization*. Ed. por Jürgen Branke y col. Vol. 4461. Dagstuhl Seminar Proceedings (DagSemProc). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2005, págs. 1-15. DOI: 10.4230/DagSemProc.04461.16. URL: https://drops.dagstuhl.de/opus/volltexte/2005/349 (vid. pág. 14).

[57] Philip S Thomas y col. «Preventing undesirable behavior of intelligent machines». En: *Science* 366.6468 (2019), págs. 999-1004 (vid. pág. 8).

[58] Chris Thornton y col. «Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms». En: *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '13. Chicago, Illinois, USA: Association for Computing Machinery, 2013, págs. 847-855. ISBN: 9781450321747. DOI: 10.1145/2487575.2487629. URL: https://doi.org/10.1145/2487575.2487629 (vid. págs. 2, 11).

[59] Yavuz Ünal y col. «Comparison of Current Convolutional Neural Network Architectures for Classification of Damaged and Undamaged Cars». En: *Advances in Deep Learning, Artificial Intelligence and Robotics*. Ed. por Luigi Troiano y col. Cham: Springer International Publishing, 2022, págs. 141-149. ISBN: 978-3-030-85365-5 (vid. pág. 1).

[60] Sahil Verma y Julia Rubin. «Fairness definitions explained». En: *2018 ieee/acm international workshop on software fairness (fairware)*. IEEE. 2018, págs. 1-7 (vid. pág. 6).

[61] Ian Witten y col. «The WEKA data mining software: An update». En: *SIGKDD Explorations* 11 (nov. de 2009), págs. 10-18. DOI: 10.1145/1656274.1656278 (vid. pág. 11).

[62] Muhammad Bilal Zafar y col. «Fairness Constraints: A Flexible Approach for Fair Classification». En: *Journal of Machine Learning Research* 20.75 (2019), págs. 1-42. URL: http://jmlr.org/papers/v20/18-262.html (vid. pág. 7).

[63] Muhammad Bilal Zafar y col. «Fairness constraints: Mechanisms for fair classification». En: *Artificial intelligence and statistics*. PMLR. 2017, págs. 962-970 (vid. págs. 7, 9, 39).

[64] Rich Zemel y col. «Learning fair representations». En: *International conference on machine learning*. PMLR. 2013, págs. 325-333 (vid. pág. 7).

[65] Brian Hu Zhang, Blake Lemoine y Margaret Mitchell. «Mitigating unwanted biases with adversarial learning». En: *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*. 2018, págs. 335-340 (vid. pág. 9).