Facultad de Matemática,
Astronomía, Física y
Computación

Universidad
Nacional
de Córdoba

Aprendizaje Automático para la Selección de Grupos de Control en Evaluación de Impacto

[1]

Autor: Benjamín Bas Peralta.
Directores: Dr. Martín A. Domínguez, Mgter. David Giuliodori.
Facultad de Matemática, Astronomía, Física y Computación
Universidad Nacional de Córdoba

14 de marzo de 2025

Resumen

Hello, here is some text without a meaning...

Palabras Clave: Análisis de Redes Sociales, Aprendizaje Automático, Detección de influenciadores, Detección de comunidades, Modelos de predicción, Twitter

Agradecimientos

– TO DO –

Índice general

Capítulo 1

Introducción

Capítulo 2

Marco Teórico

2.1. Evaluación de Impacto

2.1.1. ¿Qué es una Evaluación de Impacto?

Los programas y políticas de desarrollo suelen estar diseñados para cambiar resultados, como aumentar los ingresos, mejorar el aprendizaje o reducir las enfermedades [5]. Saber si estos cambios se logran o no es una pregunta crucial para las políticas públicas [5], y aquí es donde entra en juego lo que se conoce como **evaluación de impacto**.

Una evaluación de impacto mide los cambios en el bienestar de los individuos que se pueden atribuir a un proyecto, un programa o una política específicos [5]. Su sello distintivo radica en que pueden proporcionar **evidencia** robusta y creíble sobre si un programa concreto ha alcanzado o está alcanzando sus resultados deseados [5].

Este tipo de evaluaciones pone un fuerte énfasis en los resultados y busca responder una pregunta específica de causa y efecto: ¿cuál es el impacto (o efecto causal) de un programa en un resultado de interés? [5]. Es decir, se focalizan en los cambios directamente atribuibles al tratamiento [5] sobre un conjunto de **variables de resultado** o **de interés** en un conjunto de individuos [1]. Estas variables son aquellas sobre las cuales se espera que el programa tenga un efecto en los beneficiarios [1]. Por ejemplo, podrían ser la estatura y peso de los individuos, o la cantidad de empleados o de ventas en una empresa.

Por lo tanto, el objetivo final de la evaluación de impacto consiste en establecer la diferencia entre la variable de resultado del individuo participante en el programa en presencia del programa, y la variable de resultado de ese individuo en ausencia del programa [1], lo que se conoce como **efecto del tratamiento**. Sin embargo, es evidente que la respuesta a "¿qué habría pasado con los beneficiarios en ausencia del programa?" se refiere a una situación que no es observable. Este resultado hipotético se denomina **contrafactual** y es lo que se debe estimar en cualquier evaluación de impacto.

Algunas de las principales razones por las que se debería promover el uso de estas evaluaciones como herramientas de gestión provienen del hecho que permiten mejorar la

rendición de cuentas, la inversión de recursos públicos o la efectividad de una política, obtener financiamiento, como así también probar modalidades de programas alternativos o innovaciones de diseño [5] y revelar la realidad de muchas políticas públicas para de esta forma contribuir a la fiscalización mediática [1].

Un ejemplo claro de por qué son necesarias las evaluaciones de impacto es el que describe Howard White con respecto al Programa Integrado de Nutrición en Bangladesh (PINB) [14]. Este programa identificaba, mediante mediciones en campo, a los niños desnutridos y los asignaba a un tratamiento que incluía alimentación suplementaria a los menores y educación nutricional a las madres [1]. Inicialmente, el programa fue considerado como un éxito ya que los datos de monitoreo mostraban caídas importantes en los niveles de desnutrición. El Banco Mundial decidió, con base en esta evidencia y previo a cualquier tipo de evaluación, aumentar los recursos destinados al programa. Sin embargo, las primeras evaluaciones de impacto, realizadas por el Grupo Independiente de Evaluación del mismo Banco Mundial y por la ONG inglesa *Save the Children*, mostraron que la mejoría de los indicadores de los beneficiarios era similar o inferior a la de otros niños con características comparables que no hacían parte del programa [1]. Estos resultados reflejaron que las percepciones de los administradores del programa y de las entidades financiadoras eran erradas, y sugirieron algunos correctivos al programa [1].

2.1.2. Estimación del Tratamiento

El marco teórico estándar para formalizar el problema de la evaluación de impacto se basa en el modelo de resultado *potencial* o modelo de Roy-Rubin [11].

Formalmente, definimos dos elementos para cada individuo $i = 1, ..., N$, donde N denota la población total:

- Por un lado, el indicador de tratamiento D_i, tal que $D_i = 1$ implica que el individuo i participó del tratamiento, y $D_i = 0$ en caso contrario.
- Por otro lado, las variables de resultado las definimos como $Y_i(D_i) = Y_i|D_i$ - se lee como "el valor de Y_i *dado* D_i". De esta forma, $Y_i(1)$ es la variable de resultado si el individuo i es tratado, e $Y_i(0)$ es la variable de resultado si el individuo i no es tratado.

Con esto, el **efecto del tratamiento** para un individuo i se puede escribir como:

$$\tau_i = Y_i(1) - Y_i(0) = (Y_i|D_i = 1) - (Y_i|D_i = 0) \tag{2.1}$$

Según esta fórmula, el impacto causal (τ) de un programa (D) en una variable resultado (Y) para un individuo i es la diferencia entre la variable con el programa ($Y_i(1)$) y la misma variable sin el programa ($Y_i(0)$).

De nuevo, el problema fundamental de la evaluación de impacto es que se intenta medir una variable en un mismo momento del tiempo para la misma unidad de observación pero

en dos realidad diferentes [5]. Sin embargo, claramente solo se da uno de los dos resultados potenciales $Y_i(1)$ o $Y_i(0)$, pero no ambos. Es decir, en los datos queda solamente registrado $Y_i(1)$ si $D_i = 1$ y $Y_i(0)$ si $D_i = 0$; no se dispone de $Y_i(1)$ si el individuo no fue tratado ($D_i = 0$), ni tampoco de $Y_i(0)$ si el individuo fue tratado ($D_i = 1$). De esta manera, el **resultado observado** de Y_i se puede expresar como:

$$Y_i = D_i Y_i(1) + (1 - D_i)Y_i(0) = \begin{cases} Y_i(1) \text{ si } D_i = 1 \\ Y_i(0) \text{ si } D_i = 0 \end{cases} \tag{2.2}$$

Si nos concentramos en las unidades tratadas, en la ecuación 2.1, el término $Y_i(0) = (Y_i|D_i = 0)$ representa el contrafactual, es decir *cuál habría sido el resultado si la unidad no hubiera participado en el programa.* Al ser imposible observar directamente el contrafactual, es necesario **estimarlo**. La forma más directa de solucionar este problema sería hallando un "clon perfecto" para cada uno de los individuos participantes del programa, lo cual resulta bastante difícil. Por lo tanto, el primer paso para lograr esta estimación consiste en **desplazarse desde el nivel individual al nivel del grupo** [5], concentrando el análisis en el impacto o efecto *promedio* (y no individual).

En primera instancia, se puede estimar el *impacto promedio del programa* ***sobre la población*** (o ATE):

$$ATE = \mathbb{E}\left[Y_i(1) - Y_i(0)\right] \tag{2.3}$$

El ATE se interpreta como el cambio promedio en la variable de resultado cuando un individuo escogido al azar pasa aleatoriamente de ser participante a ser no participante [1]. Esta medida es relevante en el caso de la evaluación de un programa universal.

Sin embargo, en la mayoría de los casos, el tratamiento solo está disponible para un subconjunto de la población, por lo cual resulta más preciso utilizar un estimador que únicamente promedie el efecto sobre la población elegible [1]. De esta forma, definimos el *impacto promedio del programa* ***sobre los tratados*** (o ATT), que es en general el parámetro de mayor interés en una evaluación de impacto, y representa el efecto esperado del tratamiento en el subconjunto de individuos que fueron efectivamente tratados:

$$ATT = \mathbb{E}\left[Y_i(1) - Y_i(0)|D_i = 1\right] = \mathbb{E}\left[Y_i(1)|D_i = 1\right] - \mathbb{E}\left[Y_i(0)|D_i = 1\right] \tag{2.4}$$

Es decir, el ATT es la diferencia entre la media de la variable de resultado en el grupo de los participantes y la media que hubieran obtenido los participantes si el programa no hubiera existido [1].

2.1.3. El Grupo de Control como Estimador del Contrafactual

En la ecuación 2.4, el término $\mathbb{E}\left[Y_i(1)|D_i = 1\right]$ es un resultado observable mientras que $\mathbb{E}\left[Y_i(0)|D_i = 1\right]$ es el promedio contrafactual que debemos aproximar. Para ello, se construye lo que se conoce como **grupo de control**, formado por individuos que no

participan del programa pero que, idealmente, son estadísticamente similares [5] al **grupo de tratamiento**, compuesto por quienes sí recibieron el programa.

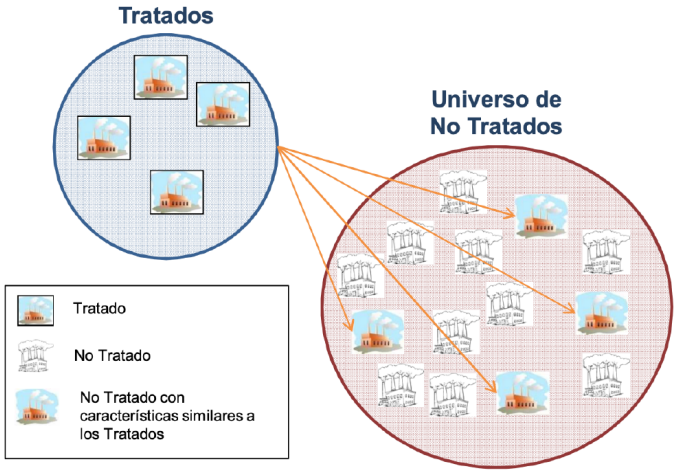

Figura 2.1:

Por lo tanto, en la práctica, el reto de una evaluación de impacto es definir un *buen* grupo de control, es decir uno que sea estadísticamente idéntico al de tratamiento, en promedio, en ausencia del programa [5]. Si se lograra que los dos grupos sean idénticos, con la única excepción que un grupo participa del programa y el otro no, entonces sería posible estar seguros que cualquier diferencia en los resultados tiene que deberse al programa [5].

Para que un grupo de comparación sea válido, debe satisfacer lo siguiente [5]:

- Las características *promedio* del grupo de tratamiento y del grupo de comparación deben ser idénticas en ausencia del programa. Cabe resaltar que no es necesario que las unidades individuales en el grupo de tratamiento tengan clones perfectos en el grupo de control.

- No debe ser afectado por el tratamiento de forma directa ni indirecta.

- El grupo de comparación debería reaccionar de la misma manera que el grupo de tratamiento si fuera objeto del programa.

Cuando el grupo de comparación no produce una estimación precisa del contrafactual, no se puede establecer el verdadero impacto del programa. A continuación, se presentan dos situaciones en las que el grupo de comparación seleccionado conduce a una estimación incorrecta del contrafactual.

Comparaciones antes-después

Este tipo de comparaciones intenta establecer el impacto de un programa a partir de un seguimiento de los cambios en los resultados en los participantes del programa a lo largo del tiempo. Consideran el contrafactual como el resultado para el grupo de tratamiento antes que comience la intervención, momento también conocido como **línea de base**. Esta comparación supone que si el programa no hubiera existido, el resultado para los participantes del programa habría sido igual a su situación antes del programa, lo cual en la mayoría de los tratamientos este supuesto no puede sostenerse [5].

Comparaciones de inscritos y no inscritos: Sesgo de autoselección

Una de las formas más directas de solucionar el problema del contrafactual podría ser simplemente utilizar el promedio de la variable de resultado entre los individuos no participantes del programa pero elegibles para participar. Es decir, podríamos utilizar a $\mathbb{E}[Y_i(0)|D_i = 0]$, que sí se puede observar, como una aproximación de $\mathbb{E}[Y_i(0)|D_i = 1]$ [1]. Sin embargo, esto podría generar estimaciones inexactas del efecto del programa, dado que los participantes y los no participantes generalmente son diferentes (en características observadas y no observadas), aún en ausencia del programa, y es justamente por tal motivo que unos *eligen* participar y otros no [1]. Este problema se conoce como **sesgo de autoselección**, ya que el grupo se *autoselecciona* para no participar de un programa. Más concretamente, este sesgo se produce cuando los motivos por los que un individuo participa en un programa, usualmente no observables y difíciles de medir, están correlacionados con los resultados [5], y por ende, es muy probable que la variable de resultado del grupo de tratamiento y del grupo de control sean diferentes, *aún si el programa no existiera* [1].

A modo de ejemplo, se puede pensar en un programa de nutrición infantil. Podría ocurrir que las madres de familias participantes del programa sean más proactivas respecto al desarrollo de sus hijos, por lo cual se preocuparon en lograr la participación en el programa. El problema de autoselección en este caso radica en que la motivación de las madres (que no se observa y es difícil de medir) afecta no solo la probabilidad de participar en el programa, sino también el estado nutricional de los niños ya que estas podrían vigilar mejor la dieta de sus hijos. De esta forma, la diferencia observada en el estado nutricional de los niños de los dos grupos podría deberse parcialmente a la diferencia en el nivel de compromiso de las madres, y no exclusivamente a que un grupo participa en el programa y el otro no [1].

Habiendo presentado este problema, volvamos a la fórmula del ATT (2.4), y notemos que la podemos reescribir de la siguiente manera:

$$ATT + \mathbb{E}[Y_i(0)|D_i = 1] = \mathbb{E}[Y_i(1)|D_i = 1] \tag{2.5}$$

Si ahora restamos $\mathbb{E}[Y_i(0)|D_i = 0]$ a ambos lados, obtenemos:

$$ATT + \mathbb{E}[Y_i(0)|D_i = 1] - \mathbb{E}[Y_i(0)|D_i = 0] = \mathbb{E}[Y_i(1)|D_i = 1] - \mathbb{E}[Y_i(0)|D_i = 0] \quad (2.6)$$

De aquí se deduce que utilizar $\mathbb{E}[Y_i(0)|D_i = 0]$ como aproximación del contrafactual $\mathbb{E}[Y_i(0)|D_i = 1]$ permite recuperar el ATT si y solo si:

$$\mathbb{E}[Y_i(0)|D_i = 1] - \mathbb{E}[Y_i(0)|D_i = 0] = 0 \quad (2.7)$$

Y es justamente este el gran desafío de la evaluación de impacto: determinar las condiciones o supuestos bajo los cuales se cumple 2.7, es decir que $\mathbb{E}[Y_i(0)|D_i = 0]$ sea una aproximación válida de $\mathbb{E}[Y_i(0)|D_i = 1]$ [1].

Es importante en este punto identificar claramente los diferentes resultados que hemos mencionado hasta el momento, los cuales se encuentran en la tabla 2.1. Teniendo en

Lo que se desea medir: el ATT.	$\mathbb{E}[Y_i(1) - Y_i(0)|D_i = 1]$
Lo que se observa: el promedio de la diferencia entre los resultados de los tratados y el grupo de control.	$\mathbb{E}[Y_i(1) - C_i(0)]$
El sesgo de selección: el promedio de la diferencia potencial entre lo que se observa y lo que se desea medir.	$\mathbb{E}[(Y_i(1) - Y_i(0)) - (Y_i(1) - C_i(0))] = \mathbb{E}[C_i(0) - Y_i(0)]$

Cuadro 2.1: Distinción entre las diferentes comparaciones presentes en una evaluación de impacto.

cuenta las fórmulas expuestas en esta tabla, podemos enunciar de otra forma lo que dijimos anteriormente: la calidad de una evaluación de impacto dependerá de los supuestos necesarios para asegurar que no hay sesgo de selección, es decir que $\mathbb{E}[C_i(0) - Y_i(0)] = 0$.

Las reglas de un programa para seleccionar a los participantes, también llamadas **reglas de asignación**, constituyen el parámetro clave para determinar el método de evaluación de impacto. A continuación, presentamos las distintas reglas y los métodos de evaluación que se utilizan en la práctica al trabajar con cada una de ellas.

2.1.4. Evaluaciones Experimentales o Aleatorias

En este tipo de evaluaciones, los beneficiarios del programa en cuestión son seleccionados al azar, es decir mediante un sorteo. La asignación aleatoria se considera la regla de oro de la evaluación de impacto ya que no solo proporciona a los administradores del programa una regla imparcial y transparente para asignar recursos escasos entre pobla-

ciones igualmente merecedoras de ellos, sino que también representa el método más sólido para evaluar el impacto de un programa [5].

La asignación aletoria trae dos consecuencias importantes: las unidades ya no pueden *elegir* si participar o no, y además todas tienen la misma probabilidad de ser seleccionadas para el programa. De esta forma, el resultado potencial de cada individuo es independiente de sus condiciones previas, ya que la asignación al tratamiento no está determinada por sus características, sino por el azar.

Como explicamos anteriormente, el grupo de comparación ideal sería lo más similar posible al grupo de tratamiento en todos los sentidos, excepto con respecto a su participación en el programa que se evalúa. Ahora bien, el proceso de asignación aleatoria producirá dos grupos que tienen una alta probabilidad de ser estadísticamente idénticos en todas sus características (observables y no observables), siempre que el número de unidades sea suficientemente grande [5]. Esto es así ya que en general, si la población de unidades elegibles es lo suficientemente grande, este mecanismo asegura que cualquier rasgo de la población se transfiera a ambos grupos. Dicho esto, lo que ocurre en estos casos es que todos aquellos individuos que no resultaron ser beneficiarios del programa conforman un grupo de control que resulta ser naturalmente un excelente estimador del contrafactual $\mathbb{E}[Y_i(0)|D_i = 1]$. Es decir, mediante este tipo de asignación, se asegura que:

$$\mathbb{E}[Y_i(0)|D_i = 1] = \mathbb{E}[Y_i(0)|D_i = 0] \tag{2.8}$$

o, en otros téminos:

$$\mathbb{E}[C_i(0) - Y_i(0)] = 0 \tag{2.9}$$

es decir, que no hay sesgo de selección presente.

La asignación aleatoria puede utilizarse como regla de asignación de un programa en dos escenarios específicos: cuando la población elegible es mayor que el número de plazas disponibles del programa, o cuando sea necesario ampliar un programa de manera progresiva hasta que cubra a toda la población elegible [5]. Además, la asignación aleatoria podría hacerse a nivel individual (por ejemplo, a nivel de hogares), o a nivel de conglomerado (por ejemplo, comunidades) [1].

Una vez que se haya asignado el tratamiento de manera aleatoria, gracias a 2.8, es bastante sencillo estimar el impacto del programa. Después de que el programa ha funcionado durante un tiempo, tendrán que medirse los resultados de las unidades de tratamiento y de comparación. El impacto del programa es sencillamente la diferencia entre el resultado promedio para el grupo de tratamiento y el resultado promedio para el grupo de comparación, es decir:

$$\mathbb{E}[Y_i(1)|D_i = 1] - \mathbb{E}[Y_i(0)|D_i = 0]$$

Resulta claro que las evaluaciones experimentales tienen varias ventajas: por diseño, resuelven el problema del sesgo de selección, y además los resultados obtenidos son transparentes, intuitivos y no es necesario utilizar herramientas econométricas sofisticadas para

alcanzarlos [1]. Esto hace que contribuyan a la transparencia de las políticas públicas y a la rendición de cuentas [1].

Sin embargo, este tipo de evaluaciones sufre varias limitaciones que hace que no sea tan común en la práctica. Por un lado, pueden ser costosas y de difícil implementación [1]. Esto se debe a que en general, el experimento aleatorio se hace específicamente para evaluar una intervención, por lo que es necesario destinar recursos para la prueba piloto, la recolección de información, los seguimientos, y a veces incluso la implementación del programa [1]. Otro problema que tienen, y probablemente el más importante, es que por pertenecer al grupo de control, se *excluye* a un segmento de la población, igualmente vulnerable y elegible, de los beneficios de la intervención [1]. Además, dada la imposibilidad de negar los beneficios a un grupo de control por largos períodos, con frecuencia es imposible estimar los impacto de largo lazo [1].

2.1.5. Evaluaciones Cuasi-experimentales

Las evaluaciones cuasi-experimentales son justamente aquellas que utilizan un contrafactual, pero se diferencian de las experimentales en el sentido de que no se basan en la asignación aleatoria de la intervención [5], debido a que realizarla no es posible o no es ético. Las técnicas cuasi-experimentales intentan *simular* la aleatorización para estimar el impacto causal de la intervención.

Puede que los métodos cuasi-experimentales sean más adecuados en algunos contextos operativos, pero requieren más supuestos con el fin de que el grupo de comparación provea una estimación válida del contrafactual, como veremos a continuación al profundizar en algunos de estos métodos.

Diferencias en Diferencias

Propensity Score Matching

2.2. Inteligencia Artificial

La definición más general acerca del campo de la Inteligencia Artificial (IA) establece que es el área de las Ciencias de la Computación que se enfoca tanto en entender como en crear sistemas que simulen comportamientos *inteligentes*. Si bien existen distintas perspectivas sobre qué significa que una computadora actúe de manera inteligente, la que más ha prevalecido a lo largo de los años es aquel que refiere con la capacidad de computar cómo actuar de la mejor manera posible en una determinada situación [12].

En sus comienzos, los métodos desarrollados en el área de la IA estaban principalmente **basados en conocimiento**, es decir reglas matemáticas formales que permitían a las computadoras llevar a cabo inferencias lógicas y de esta forma resolver problemas que eran intelectualmente difíciles para los humanos [6].

Sin embargo, determinar reglas que describan la complejidad y diversidad de la realidad no era una tarea fácil. De esta manera, con el objetivo de hacer a estos sistemas más flexibles y capaces de adaptarse y entender diferentes situaciones, se transicionó hacia un enfoque en el que estos pudieran obtener su propio conocimiento aprendiendo patrones directamente a partir de los datos en lugar de depender exclusivamente de reglas predefinidas. Este cambio de paradigma dio lugar a lo que hoy se conoce como **Aprendizaje Automático**, la subdisciplina de la IA que permite a los algoritmos mejorar su desempeño en una tarea específica automáticamente a partir de la experiencia. Diversos factores como la creciente disponibilidad de datos, el aumento en la capacidad computacional, y los avances en los algoritmos de optimización [6] han hecho que esta área sea la que mayor desarrollo e impacto ha tenido durante las últimas décadas.

Actualmente, la IA abarca una diversidad de tareas, que van desde lo general, como son las habilidades de aprendizaje, razonamiento, y percepción, entre otras; hasta lo específico, como probar teoremas matemáticos, diagnosticar enfermedades, manejar vehículos, mejorar procesos industriales o incluso diagnosticar enfermedades.

2.2.1. Aprendizaje Automático (*Machine Learning*)

El Aprendizaje Automático, más conocido por su nombre en inglés *Machine Learning* (ML), es un campo dentro de la IA cuyo objetivo es desarrollar técnicas que permitan que las computadoras *aprendan* automáticamente a partir de la *experiencia*, sin la necesidad de ser explícitamente programadas para hacerlo. Los algoritmos de ML son comúnmente llamados **modelos**.

En 1997, Tom Mitchell definió en su libro *Machine Learning* [9] el concepto de "aprender" de la siguiente manera: "Se dice que un programa de computadora aprende de la experiencia E con respecto a una clase de tareas T y una medida de desempeño P, si su desempeño en las tareas de T, medido por P, mejora con la experiencia E". Para tener un mejor entendimiento, nos enfocamos a continuación en cada uno de estos tres componentes.

Las tareas de ML se describen usualmente en términos de cómo el sistema debería procesar un *ejemplo* o *entrada*, entendiendo a esta como un conjunto de características (o en inglés, *features*) medidas cuantitativamente a partir de un cierto objeto o evento [6]. Dentro de las tareas que pueden ser resueltas por un sistema de ML se encuentran la clasificación, la regresión, la traducción, la detección de objetos en imágenes, la generación de nuevos datos, la detección de valores atípicos, entre muchas otras.

La experiencia hace referencia al tipo de información que el modelo "puede ver" durante su proceso de aprendizaje o *entrenamiento* [4]. A esta información se la conoce como "conjunto (de datos) de entrenamiento", y es un subconjunto de todos los datos con los que se cuentan (*dataset*). En base a la experiencia, los algoritmos de ML se clasifican en dos grandes categorías:

- **Algoritmos de Aprendizaje Supervisado**: experimentan un dataset que contiene pares de entrada-salida, esto es cada ejemplo contiene sus features pero también su "etiqueta", que vendría a ser la "respuesta correcta" para dicha entrada. El algoritmo aprende una función que asigna (*mapea*) entradas a salidas. Las tareas más comunes llevadas a cabo con este tipo de aprendizaje son las de regresión y clasificación.

- **Algoritmos de Aprendizaje No Supervisado**: ven un dataset que cuenta solamente con características de cada entrada pero sin etiquetas, e intentan aprender automáticamente patrones y propiedades útiles de la estructura de los datos. Algunas tareas que se llevan a cabo con este tipo de aprendizaje son *clustering*, reducción de dimensionalidad, y detección de anomalías.

También existen otros tipos de algoritmos, como los de **Aprendizaje Semi-supervisado**, en donde el dataset contiene algunos ejemplos etiquetados y otros sin etiquetas; y los de **Aprendizaje por Refuerzo**, en donde el algoritmo aprende la mejor estrategia para una situación a partir de la interacción con su entorno en forma de recompensas y castigos.

Por último, para medir el desempeño de estos modelos, se definen métricas cuantitativas que dependen de la tarea que se esté realizando y del objetivo que se intente lograr. Por ejemplo, en clasificación, una de las más comunes es la de exactitud - *accuracy* -, que es la proporción de ejemplos para los cuales el modelo predijo la salida correcta (dada por el dataset); aunque también existen otras como la precisión, sensibilidad, especificidad, y el puntaje F1. Por otro lado, en tareas de regresión, la métrica que se suele utilizar es la de Error Cuadrático Medio.

El objetivo fundamental del ML es que un algoritmo actúe correctamente antes nuevas entradas aún no vistas, es decir que **generalice** más allá de los ejemplos del conjunto de entrenamiento. Por lo tanto, para evaluar qué tan bien se comporta el modelo y poder comparar su desempeño con otros, su rendimiento se mide en otro subconjunto del dataset, distinto con el que aprendió, llamado "conjunto de test". Como buen práctica, este conjunto no debe ser mostrado al modelo en ningún momento durante su entrenamiento.

2.2.2. Redes Neuronales Artificiales - Aprendizaje Profundo

Las Redes Neuronales Artificiales (RNAs) son un modelo específico dentro del ML, cuya estructura y funcionamiento están inspirados en los del cerebro. Durante los últimos años, ha sido el área que más desarrollo e impacto ha tenido, principalmente gracias a su versatilidad, potencia y escalabilidad, cualidades que hacen que sean capaces de enfrentar problemas grandes y complejos [4].

Breve Contexto Histórico

Aunque su gran éxito ha sido reciente, lo cierto es que la idea de las RNAs data desde 1943, cuando Warren McCulloch y Walter Pitts presentaron en su artículo "A Logical Caclulus of Ideas Immanent of Nervous Activity" un modelo computacional simplificado utilizando lógica proposicional acerca de cómo funcionan las neuronas de cerebros animales en conjunto para llevar a cabo cómputos complejos [4].

Posteriormente, en 1958, Frank Rosenblatt [10] presentó una de las arquitecturas más simples de RNAs: el "Perceptrón", el cual sirve principalmente para resolver problemas de clasificación en donde los datos son linealmente separables. Su aporte más notorio es que definió un algoritmo para su entrenamiento.

Más tarde, se descubrió que los problemas que no podían ser resueltos por el Perceptrón, sí podían ser resueltos "apilando" múltiples perceptrones, lo cual llevó a la invención del "Perceptrón Multicapa" (PMC), también conocido como "Redes Neuronales de Propagación Directa" (del inglés *Feedforward Neural Networks*) [6].

Elementos de una RNA Feedforward

El elemento básico de cualquier red neuronal es la **neurona**, que es la que lleva a cabo el "cómputo". Una neurona está conectada con otras, de las cuales recibe información y a quienes también les envía. Estas conexiones representan las **sinapsis** y cada una tiene asociada un determinado número llamado **peso**, que simboliza la intensidad de la conexión. De esta manera, el trabajo que lleva a cabo una neurona es primero computar una suma pesada de sus entradas: $z = w_1x_1 + w_2x_2 + \ldots + w_nx_n = w^Tx$, donde x_i representa la entrada proveniente de la neurona i y w_i es el peso asociado a esta conexión. Y sobre este valor resultante, aplica una **función de activación** h, siendo como resultado final de la neurona $h(z) = h(w^Tx)$. Esto se puede ver gráficamente en la figura 2.2.

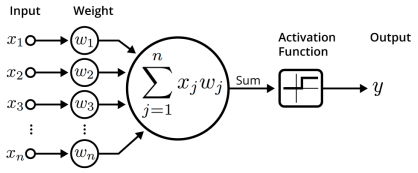

Figura 2.2: La neurona computa una suma pesada de sus entradas y aplica una función de activación sobre el resultado.

La idea detrás de la función de activación es el hecho de propagar la información proporcionalmente al estímulo provocado por las entradas en la neurona, basándose en la idea de "disparo" de una neurona biológica en el momento en que se supera el umbral de activación. La función de activación utilizada en el Perceptrón, uno de los primeros modelos de RNAs, fue la llamada *heavyside*, dada por heavyside$(z) = 0$ si $z < 0; 1$ si $z \geq 0$.

Durante los años, se han empleado diferentes funciones de activación, como la sigmoide, la tangente hiperbólica, y la ReLU, dada por $\text{ReLU}(z) = \max(0, z)$, siendo esta última la más común actualmente.

Dicho esto, una red neuronal está compuesta por diferentes capas de neuronas: la capa de entrada, que es por donde ingresan los datos, una o más capas ocultas o intermedias, en donde aparecen las funciones de activación, y finalmente la capa de salida. En una red *feedforward* particularmente, las neuronas de una capa están conectadas solamente con las neuronas de la capa siguiente, implicando de esta forma que la información fluye solamente en una dirección (de aquí la nomenclatura *feedforward*). Algo a notar también sobre estas redes es que todas las capas están totalmente conectadas entre sí, como puede verse en la figura 2.3.

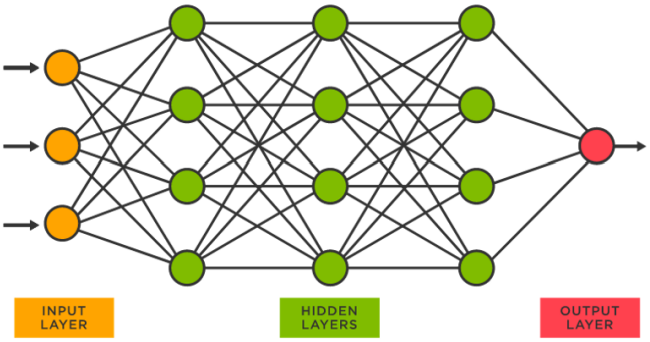

Figura 2.3: Esquema de una Red Neuronal Profunda, con tres capas ocultas. Las neuronas naranjas forman la capa de entrada, las neuronas verdes forman las tres capas ocultas, y las rojas la de salida.

Ahora bien, el objetivo de una red neuronal es **aproximar una función desconocida** g^*. Para un clasificador por ejemplo, $y = g^*(x)$ determina una categoría y para una entrada x. De esta forma, una red define una asignación $y = g(x; \theta)$ y aprende el valor de los **parámetros** θ que resultan en la mejor aproximación de la función "oculta" [6]. En este caso, los parámetros a aprender por la red serán los pesos sinápticos de todas las conexiones. Matemáticamente, los pesos entre una capa de neuronas y la siguiente se pueden pensar como una matriz w, en donde la entrada w_{ij} indica el peso sináptico de la conexión entre la neurona i de la capa "actual" y la neurona j de la capa "anterior". La pregunta que surge entonces es **cómo** hacer para que la red pueda aproximar esta función.

Entrenamiento de una RNA Feedforward

A grandes rasgos, el entrenamiento de una red consiste en una suma de **evaluación** y **optimización** [3]. Con evaluación, nos referimos al establecimiento de una medida que

determine qué tan lejos está el modelo de hallar una buena solución, para lo cual se suele definir una función a minimizar o maximizar llamada **función objetivo** o **criterio** [6]; y la optimización es el método usado para aproximarse a dicha solución, maximizando o minimizando la función objetivo según corresponda.

Cuando lo deseado es minimizar la función objetivo, esta pasa a ser llamada **función de costo**, **función de error** o **función de pérdida**. En problemas de aprendizaje supervisado, esta función compara la salida de la red para determinadas entradas - la predicción - en el estado actual de la red, dado por el valor de los pesos sinápticos en ese momento, con las salidas reales para dichas entradas, dadas por el conjunto de entrenamiento. En resumen, una vez que están fijos los datos del conjunto de entrenamiento, la función de costo es una función de los pesos de la red. Por lo tanto, la denotaremos con $f(w)$.

Dos de las funciones de costo más utilizadas son el Error Cuadrático Medio y la **Entropía Cruzada**. De estas dos funciones, nos concentraremos en la última ya que es la más apropiada para problemas de clasificación como el que tratamos. Esto se debe a que mide la diferencia entre la distribución de probabilidad predicha por el modelo y la distribución real de los datos. En este caso, la capa de salida tiene tantas neuronas como categorías haya, y la salida de cada una de ellas representa el puntaje que le da la red a la categoría correspondiente a cada neurona, donde un mayor puntaje indica una mayor probabilidad de pertenencia de la entrada particular a dicha clase. Para obtener una estimación de la distribución de probabilidad dada por estas salidas, se aplica la función *softmax* [4].

Una vez que se tiene esta función establecida y se calculan las pérdidas para los datos, el algoritmo por defecto que se utiliza para la optimización es el de **Descenso por el Gradiente** (*Gradient Descent*). Este se basa en modificar los parámetros (i.e. los pesos sinápticos) iterativamente con el objetivo de encontrar mínimos locales de una función, que en este caso será la de costo. Se basa en la idea que la dirección dada por el gradiente es la de mayor crecimiento, y por lo tanto su opuesta es la de menor. De esta forma, lo que se hace en cada paso es calcular el gradiente de la función de costo con respecto a los pesos, y luego actualizarlos siguiendo la siguiente regla:

$$w = w - \eta \, \nabla f(w),$$

donde w representa una matriz de pesos de una determinada capa, y la cantidad η es llamada la **tasa de aprendizaje**, que determina el tamaño de cada paso. Si η es muy pequeño, entonces el algoritmo tendrá que hacer muchos pasos para converger; pero si es demasiado grande, puede llegar a divergir.

Actualmente, se utilizan otros optimizadores más sofisticados y eficientes, pero que todos parten de la idea del Descenso por el Gradiente. Los que usaremos en los experimentos son el Descenso por el Gradiente Estocástico (*Stochastic Gradient Descent*, SGD) y Adam (*Adaptive Moment Estimation*), de los cuales se puede leer más en el Capítulo 8 de [6].

Ahora bien, recordando que la función f está fija en todas las entradas y salidas del conjunto de entrenamiento, hay algoritmos de optimización que utilizan todos estos datos para calcular el gradiente, otros que utilizan de a un ejemplo a la vez para calcular el gradiente y otros que están entre medio de los dos anteriores, es decir utilizan un subconjunto del conjunto de datos para calcular el gradiente (esto es lo que hace SGD).

Es necesario en este punto introducir el concepto de "**época**" y de "**lote**". Un lote es simplemente una subdivisión de tamaño fijo del conjunto de entrenamiento (o de test). Dicho esto, se llama época al proceso completo en el cual el modelo entrena utilizando **todos** los datos disponibles en el conjunto de entrenamiento una vez, ya sea recorriéndolos de a lotes o todos de una vez. Si el conjunto está efectivamente dividido en lotes, una época va a consistir de: **para cada lote**, calcular las predicciones en base a los pesos actuales, obtener los errores de cada muestra del lote, computar los gradientes en base a los ejemplos del lote y actualizar los pesos. Si no, se hacen los mismos pasos pero para todos los datos de una vez.

Al entrenar un modelo, existen ciertos ajustes que son utilizados para controlar su comportamiento llamados **hiperparámetros**. Los valores de los hiperparámetros no son aprendidos por el modelo [6], sino que se fijan de antemano y no se modifican a lo largo del entrenamiento. Algunos de estos son: la cantidad de neuronas en cada capa oculta, la función de activación utilizada en cada capa, el optimizador, la tasa de aprendizaje, el número de épocas, el tamaño de lote, entre otros. Todos estos pueden contribuir a que el modelo mejore su desempeño. Para probar distintos valores de hiperparámetros, que es lo que haremos a continuación, se utiliza un conjunto de datos extra además del de entrenamiento y el de test, llamado de **validación** que sirve justamente para encontrar los hiperparámetros óptimos.

Otras mejoras que se han implementado para mejorar la eficacia de los modelos, y que emplearemos en los experimentos, son las nombradas a continuación. Por un lado, tenemos la **detención temprana**, que consiste en frenar el entrenamiento en el último mejor "estado" del modelo cuando se observa que a partir de dicho punto el error sobre el conjunto de validación no mejora (notar que cuántas épocas se espera y qué se considera una mejora son nuevos hiperparámetros). Y por otro lado, el **dropout**, que es un algoritmo que implica suprimir un cierto número de neuronas de una capa de manera aleatoria [**apuntes-redes-neuronales**].

2.3. Redes Neuronales Convolucionales

2.4. Redes Neuronales Recurrentes

2.4.1. Long Short-Term Memory

2.5. Clasificación de Series de Tiempo

Una **serie de tiempo** es una secuencia **ordenada** de valores medidos sucesivamente en intervalos de tiempo igualmente espaciados [8]. Este tipo de datos está presente en distintos fenómenos, como lecturas meteorológicas, registros financieros, señales fisiológicas, y observaciones industriales [13].

Formalmente, una serie de tiempo **univariada** U es un conjunto ordenado de valores reales $U = (x_1, x_2, ..., x_N$, y su dimensión está dada por la cantidad N de valores [8].

Dicho esto, la **Clasificación de Series de Tiempo** (CST) consiste en: dado un conjunto de clases o categorías Y, y un conjunto de datos D formado por series de tiempo univariadas U_i donde a cada una le corresponde una etiqueta $y(U_i) \in Y$, el objetivo es encontrar una función f, a la que llamaremos "clasificador" o "modelo" de forma tal que $f(U) = y(U)$ [8]. Un buen modelo será uno que pueda capturar y generalizar el el patrón de las series de datos de forma tal que sea capaz de clasificar correctamente nuevos datos. El repositorio que se suele tomar como referencia para evaluar los diferentes modelos de CST es el de la Universidad de California Riverside, llamado *UCR Time Series Classification Archive* [2].

Durante las últimas décadas, se han propuestos diferentes métodos para afrontar este problema. Los primeros trabajaban directamente sobre las series de tiempo utilizando alguna medida de similitud predefinida que pueda capturar la similitud entre ellas, como por ejemplo la distancia euclideana o la deformación dinámica del tiempo (*Dynamic Time Warping*) [13].

Se han desarrollado también algoritmos basados en características o *features*, en donde cada serie temporal es convertida en un conjunto de features globales, que es posteriormente usando para definir la semejanza entre pares de series de tiempo [8]. Algunos de los más conocidos son ...

El gran problema de todas las estrategias mencionadas es que requieren un gran preprocesamiento de los datos y una ingeniería de atributos (*feature engineering*) [13], para lo cual generalmente se necesita tener un buen conocimiento de campo sobre las series sobre las que se está trabajando. Es por esto que recientemente, las redes neuronales profundas han comenzado a ser utilizadas para clasificar series temporales [13] ya que permiten trabajar de forma directa con los datos, y se encargan de detectar automáticamente cuáles son las características que distinguen a las series. Algunas arquitecturas utilizadas hasta el momento son el perceptrón multicapa [13], redes neuronales convolucionales [13], y

estas últimas combinadas con unidades LSTM unidireccionales [7] y bidireccionales [8].

Capítulo 3

Presentación del problema

Como explicamos anteriormente, el principal problema en la Evaluación de Impacto de tratamientos sin asignación aleatoria es el de la identificación de un grupo de control o comparación adecuado, que cumpla las características descritas previamente.

Cuando el ingreso al tratamiento por parte de los individuos se realiza en un único instante de tiempo, el grupo de control obtenido mediante la técnica de PSM conforma el mejor estimador del contrafactual. Sin embargo, cuando la entrada al programa ocurre de manera secuencial, los modelos logísticos no son capaces de gestionar esa probabilidad variable en el tiempo. Lo que se hace entonces es estimar la probabilidad por camada de ingreso al programa, es decir se agrupan individuos que hayan ingresado al tratamiento en temporalidades similares, y se buscan controles separadamente para cada una de estas camadas.

En este trabajo, exploramos una alternativa distinta para este último caso. Proponemos la utilización de redes neuronales, específicamente del tipo LSTM **y convolucionales????**, capaces de incorporar la dimensión temporal de los datos, para la detección automática de grupos de control, en el caso particular en que el programa en cuestión se implementa **secuencialmente** y existe una **alta dependencia temporal** entre una variable observada de los individuos y el momento de ingreso al tratamiento. El principal objetivo es evaluar el potencial y la efectividad de estas redes para capturar dependencias dinámicas y características temporales inherentes a los datos observados, para de esta forma mejorar el proceso de inferencia causal en la evaluación de impacto.

Capítulo 4

Marco Experimental

4.1. Conjuntos de Datos

A continuación, detallamos la metodología utilizada para llevar a cabo las simulaciones y medir el desempeño de las redes, y explicamos cómo este problema se termina enmarcando dentro de una clasificación de series de tiempo.

El primer paso para poder llevar a cabo los experimentos es contar con datos. Nuestro entorno - *framework* - de simulación está diseñado para generar **datos sintéticos** que imiten escenarios reales en donde la asignación al tratamiento es no aleatoria, escalonada y potencialmente dependiente de resultados pasados.

El proceso de generación de datos involucra la creación de una serie de tiempo univariada para cada individuo, que incorpora efectos fijos, componentes autoregresivos e impactos del tratamiento. Trabajamos bajo el supuesto que el comportamiento dinámico de la variable modelada para cada unidad es el que determina si esta ingresa al tratamiento o no, y es la misma sobre la que se espera que el tratamiento tenga un efecto.

Los parámetros para la generación de los datos son los siguientes:

- `n_sample`: cantidad de individuos en la simulación.
- `treated_pct`: porcentaje de individuos tratados.
- `control_pct`: porcentaje de individuos de control.
- `T`: cantidad de períodos observados de cada individuo.
- `first_tr_period`: primer período de tratamiento (o único, dependiendo de la cantidad de cohortes).
- `n_cohorts`: cantidad de cohortes.
- `phiT`: persistencia auto-regresiva para los tratados.
- `phiC`: persistencia auto-regresiva para los controles.

- `n_dep_periods`: cantidad de períodos de dependencia para la participación en el tratamiento.

El resultado de la simulación es un conjunto de datos de panel compuesto por series de tiempo univariadas de longitud `T` para los distintos tipos de individuos:

- **Individuos de tipo 1**: son aquellos que han recibido el tratamiento en algún período. Notar que en una situación real, son datos con los que contamos. Los llamaremos también "tratados".
- **Individuos de tipo 2**: son aquellos que en los datos sintéticos, sabemos que forman parte del grupo de control. Es importante notar que en un escenario real, no sabemos quiénes son estos individuos sino que son los que tratamos de identificar. Nos referiremos a ellos también como "controles".
- **Individuos de tipo 3**: son aquellas unidades que no han sido tratadas y que en los datos sintéticos, sabemos que no forman parte del grupo de control. A estos también los mencionaremos como "NiNi" (ni tratado ni control).

En el dataset, cada individuo tiene un identificador y su tipo; y aquellos de tipo 1 y 2 tienen como feature extra el período (desde `first_tr_period` hasta `first_tr_period + n_cohorts`) en el que ingresaron al programa. A continuación, se muestran algunos ejemplos, tomando `T = 6`, `first_tr_period = 3` y `n_cohorts = 2`:

En este punto cabe recordar cuál es nuestra meta: a partir de información sobre los individuos que fueron tratados, queremos identificar de entre los no tratados, quiénes son los que podrían formar parte del grupo de control.

También resulta muy importante tener en cuenta que en los datos generados sintéticamente, sabemos quiénes son los controles, pero en la realidad esto es justamente lo que queremos identificar.

Ahora bien, para entrenar y evaluar a nuestros modelos, tomamos en cuenta lo que ocurriría en un escenario real, en el que sabríamos solamente quiénes son los tratados y quiénes los no tratados. Por lo tanto, construimos el conjunto de entrenamiento con la totalidad de los individuos de tipo 1 con etiqueta 1 y algunos de tipo 3 con etiqueta 0, y en el conjunto de test colocamos a todos los de tipo 2, queriendo predecir en ellos un 1, y al resto de tipo 3, queriendo predecir en ellos un 0.

Más aún, como queremos identificar a los grupos de control de cada cohorte,

4.2. Arquitecturas de Redes Neuronales

4.3. Herramientas

4.3.1. Python

4.3.2. PyTorch

4.3.3. Pandas

4.3.4. Numpy

4.3.5. Jupyter

4.3.6. Optuna

4.3.7. MLflow

Capítulo 5

Resultados

Capítulo 6

Conclusiones y Trabajos Futuros

Bibliografía

[1] Raquel Bernal y Ximena Peña. *Guía práctica para la evaluación de impacto*. 1.ª ed. Universidad de los Andes, Colombia, 2011. ISBN: bernal2011. URL: http://www.jstor.org/stable/10.7440/j.ctt1b3t82z (visitado 10-02-2025).

[2] Hoang Anh Dau et al. *The UCR Time Series Classification Archive*. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/. Oct. de 2018.

[3] Pedro Domingos. "A few useful things to know about machine learning". En: *Commun. ACM* 55.10 (2012), págs. 78-87. ISSN: 0001-0782. DOI: 10.1145/2347736.2347755.

[4] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. 1st. O'Reilly Media, 2017. ISBN: 9781491962299.

[5] Paul Gertler et al. *La evaluación de impacto en la práctica: Segunda edición*. Banco Internacional para la Reconstrucción y el Desarrollo/Banco Mundial, 2016. DOI: https://doi.org/10.18235/0006529. URL: https://hdl.handle.net/10986/25030.

[6] Ian Goodfellow, Yoshua Bengio y Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[7] Fazle Karim et al. "LSTM Fully Convolutional Networks for Time Series Classification". En: *IEEE Access* 6 (2018), págs. 1662-1669. ISSN: 2169-3536. DOI: 10.1109/access.2017.2779939. URL: http://dx.doi.org/10.1109/ACCESS.2017.2779939.

[8] Riaz A. et al. Khan M. Wang H. "Bidirectional LSTM-RNN-based hybrid deep learning frameworks for univariate time series classification". En: *The Journal of Supercomputing* (2021), págs. 7021-7045. URL: https://doi.org/10.1007/s11227-020-03560-z.

[9] Tom M. Mitchell. *Machine Learning*. McGraw-hill, 1997. ISBN: 0070428077.

[10] Frank Rosenblatt. "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain". En: *Psychological Review* 65.6 (1958), págs. 386-408. DOI: 10.1037/h0042519.

[11] Donald Rubin. "ESTIMATING CAUSAL EFFECTS OF TREATMENTS IN EXPERIMENTAL AND OBSERVATIONAL STUDIES". En: *Journal of Education Psychology* 66.5 (1974), págs. 688-701. DOI: https://doi.org/10.1002/j.2333-8504.1972.tb00631.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/j.2333-8504.1972.tb00631.x.

[12] Stuart J. Russell y Peter Norvig. *Artificial Intelligence: A Modern Approach*. 4, Global Edition. Pearson Education, 2020. ISBN: 9781292401133.

[13] Zhiguang Wang, Weizhong Yan y Tim Oates. *Time Series Classification from Scratch with Deep Neural Networks: A Strong Baseline*. 2016. arXiv: 1611.06455 [cs.LG]. URL: https://arxiv.org/abs/1611.06455.

[14] Howard White. "Theory-based impact evaluation: principles and practice". En: *Journal of development effectiveness* 1.3 (2009), págs. 271-284.