

Machine learning y deep learning

Rolando Gonzales Martinez, PhD

Fellow postdoctoral Marie Skłodowska-Curie

Universidad de Groningen
(Países Bajos)

Investigador (researcher)
Iniciativa de Pobreza y Desarrollo Humano de la Universidad de Oxford (UK)

Contenido del curso

(2) Introducción a Machine Learning

- Definición y tipos de aprendizaje: supervisado, no supervisado y por refuerzo.
- Herramientas y lenguajes de programación más utilizados.
- Laboratorio: prácticas en Python y R

Machine learning

Dado un espacio de entrada X (espacio de características) y un espacio de salida Y , machine learning es un problema de optimización en el que el objetivo es encontrar una función $f: X \rightarrow Y$ que predice la salida $y \in Y$ dada una entrada $x \in X$, siendo f^* óptima en términos de **generalización y regularización**:

$$\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\} \quad x_i \in \mathcal{X} \quad y_i \in \mathcal{Y}$$

$$f^* = \arg \min_{f \in \mathcal{F}} \mathbb{E}_{(x,y) \sim P} [L(f(x), y)]$$

$$f^* = \arg \min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n L(f(x_i), y_i)$$

$$f^* = \arg \min_{f \in \mathcal{F}} \left[\frac{1}{n} \sum_{i=1}^n L(f(x_i), y_i) + \lambda R(f) \right]$$

Machine learning

Dado un conjunto de funciones (modelos), M_1, M_2, \dots, M_k , es necesario considerar medidas de performance para variables continuas y variables discretas al momento de decidir entre la mejor función (modelo):

$$M_1, M_2, \dots, M_k$$

$$\hat{\beta}_{\text{train},i} = \arg \max_{\beta} \log \mathcal{L}(\beta; X_{\text{train},i}, y_{\text{train}})$$

$$\hat{y}_{\text{test},i} = g^{-1}(X_{\text{test},i} \hat{\beta}_{\text{train},i})$$

$$\text{MSE}_i = \frac{1}{n_{\text{test}}} \sum_{j=1}^{n_{\text{test}}} (y_{\text{test},j} - \hat{y}_{\text{test},ij})^2$$

$$R_i^2 = 1 - \frac{\sum_{j=1}^{n_{\text{test}}} (y_{\text{test},j} - \hat{y}_{\text{test},ij})^2}{\sum_{j=1}^{n_{\text{test}}} (y_{\text{test},j} - \bar{y}_{\text{test}})^2}$$

$$\text{AUC-ROC}_i = \text{AUC}(\text{Curva ROC}_i)$$

$$M_{\text{best}} = \arg \min_i \text{MSE}_i$$

$$M_{\text{best}} = \arg \max_i R_i^2$$

$$M_{\text{best}} = \arg \max_i \text{AUC-ROC}_i$$

Principales tipos de aprendizaje

- **Supervisado:** Modelo ML entrenado con datos etiquetados.
- **Semi-supervisado:** Modelo ML entrenado con una combinación de datos etiquetados y no etiquetados.
- **No supervisado:** Modelo ML entrenado con datos no etiquetados.

$$f^* = \arg \min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n L(f(x_i), y_i)$$
$$f^* = \arg \min_{f \in \mathcal{F}} \left[\frac{1}{|L|} \sum_{(x_i, y_i) \in \mathcal{D}_L} L(f(x_i), y_i) + \lambda R(f, \mathcal{D}_U) \right]$$
$$f^* = \arg \min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n L(f(x_i))$$

Otros tipos de aprendizaje

- **Reforzamiento:** Modelo ML se optimiza en base a un agente a y un conjunto de estados s , $\pi(a|s)$
- **Transferencia:** Modelo ML entrenado se re-utiliza y adapta a una nueva tarea
- **Meta-aprendizaje:** El modelo ML se adapta a nuevas tareas T en base a pocos ejemplos de entrenamiento

$$\pi^* = \arg \max_{\pi} \mathbb{E} \left[\sum_{t=0}^T \gamma^t r(s_t, a_t) \right]$$

$$f_T^* = \arg \min_{f \in \mathcal{F}_T} \frac{1}{m} \sum_{j=1}^m L(f(x_j), y_j)$$

$$f^* = \arg \min_{f \in \mathcal{F}} \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})} \left[\min_{\theta_i} L_{\mathcal{T}_i}(f(\theta_i)) \right]$$

LÍNEA DEL TIEMPO DE MACHINE LEARNING

- Se publica el **Teorema de Bayes**.

$$P[A_n/B] = \frac{P[B/A_n] \cdot P[A_n]}{\sum P[B/A_i] \cdot P[A_i]}$$

1764

1842

1847

1936

1952

1959

1985

2006

2012

2016

2018

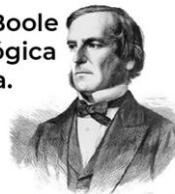
- Ada Lovelace sienta las bases del primer algoritmo.



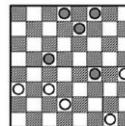
- Alan Turing propone una máquina que pueda aprender.



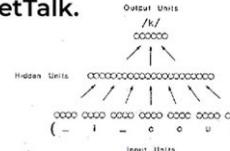
- George Boole crea la Lógica Booleana.



- Arthur Samuel crea los primeros programas para computadora en IBM.



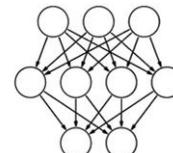
- Sejnowski y Rosenberg crean la red neuronal, **NetTalk**.



- Google crea una red neuronal no supervisada.



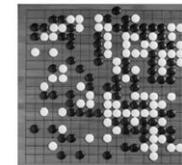
- Se crea MADALINE, la primera red neuronal artificial.



- Geoffrey Hinton inventa el término "Deep Learning".



- AlphaGo vence al primer jugador humano.



- Se crea AlphaFold 1, tecnología capaz de predecir estructuras de proteínas.

Algoritmos y modelos

Modelo: Representación simplificada de un proceso, sistema o fenómeno del mundo real. Es una construcción matemática o estadística.

$$\text{salario} = f(\text{educación}, \text{edad}, \text{experiencia})$$

$$\text{salario} = f(\text{educación}, \text{edad}, \text{experiencia}; \Theta)$$

$$\Theta = [\theta_{\text{educ}}, \theta_{\text{edad}}, \theta_{\text{expe}}]$$

$$\text{salario} = f(\text{educación}, \text{edad}, \text{experiencia}; \theta_{\text{educ}}, \theta_{\text{edad}}, \theta_{\text{expe}})$$

Asumiendo que $f(\cdot)$ es lineal:

$$\text{salario} = \theta_0 + \theta_{\text{educ}} \text{educación} + \theta_{\text{edad}} \text{edad} + \theta_{\text{expe}} \text{experiencia}$$

Añadiendo un término i.i.d. $\varepsilon \sim \mathcal{N}(\mu, \sigma^2)$:

$$\text{salario} = \theta_0 + \theta_{\text{educ}} \text{educación} + \theta_{\text{edad}} \text{edad} + \theta_{\text{expe}} \text{experiencia} + \varepsilon$$

Algoritmos y modelos

Algoritmo: Serie de pasos o instrucciones lógicas que se utilizan para realizar una tarea específica.

Output: **salario**

Input: edad, experiencia, educación

if edad > 6

 for (edad) t = 1:n

 educación(t) = $f(\text{estudios})$

 while (educación & edad) > aceptable

 for (años) e = 1:m

 experiencia = e

 end

salario(t) = $f(\text{educación}, \text{edad}, \text{experiencia}; \Theta)$

 end

 end

end

Algoritmos y modelos

En el contexto de machine learning:

- **Modelo:** representación matemática/estadística que se ajusta a los datos y que se utiliza para hacer predicciones o tomar decisiones.
- **Algoritmo:** instrucciones que guían cómo se ajustan los parámetros de un modelo ML durante el proceso de entrenamiento.
- En el proceso de ML, primero se selecciona un algoritmo apropiado para el tipo de problema que se está abordando (supervisado, no supervisado, etc.). Luego, este algoritmo se utiliza para entrenar un modelo específico, ajustando sus parámetros para que el modelo pueda generalizar la tarea a datos nuevos sin sobreajuste.

Algoritmos y modelos

$$\text{salario}_i = \theta_0 + \theta_{educ} \text{educación}_i + \theta_{edad} \text{edad}_i + \theta_{expe} \text{experiencia}_i + \varepsilon_i$$

Algoritmo de estimación:

Input: datos de salarios, educación, edad, experiencia

Output: estimadores de $\theta_{educ}, \theta_{edad}, \theta_{expe}$

1. Representación matricial:

$$\Theta = [\theta_{educ}, \theta_{edad}, \theta_{expe}]$$

$$X = [\text{educación}_i, \text{edad}_i, \text{experiencia}_i]$$

$$y = [\text{salario}_1, \text{salario}_2, \dots, \text{salario}_n]$$

$$y = X \Theta + \varepsilon$$

$$y - X \Theta = \varepsilon$$

2. Operaciones matriciales: cálculo de matrices transpuestas e inversas, X' , $(X'X)^{-1}$

3. Estimadores MCO:

$$\underset{\substack{\Theta \in \mathbb{R} \\ i=1,2,\dots,n}}{\operatorname{argmin}} \varepsilon: \widehat{\Theta} = (X'X)^{-1} X'y$$

4. Estimadores MCO: $\widehat{\theta}_{educ}, \widehat{\theta}_{edad}, \widehat{\theta}_{expe}$

Particiones muestrales en machine learning

- **Partición en muestra train y test:** Método más básico y común. Se divide los datos en dos partes: un conjunto de entrenamiento para entrenar el modelo (estimar los parámetros del modelo) y un conjunto de prueba para evaluar el rendimiento del modelo en datos no vistos.
- **Hold-Out con Validación:** similar a la partición en entrenamiento y prueba, pero incluye un tercer conjunto de validación:
 - **Entrenamiento (Train):** Para ajustar los parámetros del modelo.
 - **Validación (Validation):** Para ajustar hiperparámetros y evitar sobreajuste.
 - **Prueba (Test):** Para la evaluación final del modelo.

Particiones muestrales en machine learning

Partición de datos **en 4 grupos**:

Entrenamiento (train): $i = 1, \dots, m$

persona	salario	educación	edad	experiencia
$i = 1$	2640	24	40	10
\vdots	\vdots	\vdots	\vdots	\vdots
$i = 8$	6400	30	57	12
\vdots	\vdots	\vdots	\vdots	\vdots
$i = m$	3555	29	32	9

y train **X entrenamiento (train)**

Evaluación (test): $i = m, m+1, \dots, n$

persona	salario	educación	edad	experiencia
$i = m$	5656	27	38	5
$i = m+1$	3322	24	37	10
\vdots	\vdots	\vdots	\vdots	\vdots
$i = n$	4800	32	40	13

y test **X test**

El conjunto de entrenamiento se utiliza para entrenar el modelo (estimar los parámetros del modelo)

El conjunto de prueba se emplea para evaluar el rendimiento del modelo en datos no vistos

Particiones muestrales en machine learning: La validación cruzada k-fold (k-fold cross-validation)

- En muestras pequeñas, $k = n$ (partición LOO)
- Valores convencionales: $k = 5, k = 10, k = 20$. Depende del tamaño de los datos y el costo computacional

$$\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$$

$$\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_k$$

$\mathcal{D}_i \subset \mathcal{D}$ para todo i y $\mathcal{D}_i \cap \mathcal{D}_j = \emptyset$ para $i \neq j$.

Para cada $i = 1, 2, \dots, k$:

$$\mathcal{D}_{-i} = \mathcal{D} \setminus \mathcal{D}_i = \bigcup_{j \neq i} \mathcal{D}_j.$$

$h(\cdot; \theta_i)$ usando \mathcal{D}_{-i}

$$E_i = \frac{1}{|\mathcal{D}_i|} \sum_{(x_j, y_j) \in \mathcal{D}_i} \ell(h(x_j; \theta_i), y_j)$$

$$E_{\text{CV}} = \frac{1}{k} \sum_{i=1}^k E_i$$

Particiones muestrales en machine learning

Partición de datos:

Entrenamiento (train): $i = 1, \dots, m$

$$\text{salario}_i = \theta_0 + \theta_{\text{educ}} \text{educación}_i + \theta_{\text{edad}} \text{edad}_i + \theta_{\text{expe}} \text{experiencia}_i + \varepsilon_i$$

$$\Theta = [\theta_{\text{educ}}, \theta_{\text{edad}}, \theta_{\text{expe}}]$$

$$X_{\text{train}} = [\text{educación}_i, \text{edad}_i, \text{experiencia}_i]$$

$$y_{\text{train}} = [\text{salario}_1, \text{salario}_2, \dots, \text{salario}_m]$$

$$y_{\text{train}} = X \Theta + \varepsilon$$

$$\boxed{\begin{aligned} & \underset{\substack{\Theta \in \mathbb{R} \\ i=1,2,\dots,m}}{\operatorname{argmin}} \varepsilon: \hat{\Theta} = (X_{\text{train}}' X_{\text{train}})^{-1} X_{\text{train}}' y_{\text{train}} \end{aligned}}$$

El conjunto de entrenamiento se utiliza para entrenar el modelo (estimar los parámetros del modelo)

Evaluación (test): $i = m, m+1, \dots, n$

$$y_{\text{pronostico}} = X_{\text{test}} \hat{\Theta}$$

Error cuadrático medio (MSE):

$$MSE = \frac{1}{m} \sum_{i=m}^n (y_{\text{test},i} - y_{\text{pronostico},i})^2$$

El conjunto de prueba se emplea para evaluar el rendimiento del modelo en datos no vistos

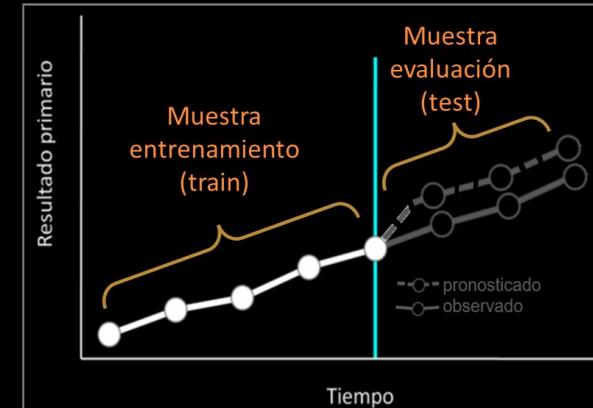
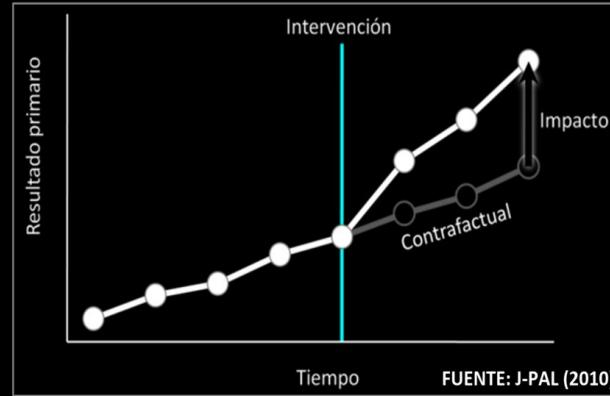
Paralelismos en Econometría/Estadística y ML

Econometría y Estadística	Machine learning
Estimación	Aprendizaje, entrenamiento
Parámetros	Pesos
Variables explicativas (X)	Features/inputs
Variable dependiente (y)	Target/output
Estimación en muestra completa* (muestras limitadas)	Partición de datos (train/test) (big data)
Propósito: inferencia, predicción	Propósito: predicción
Maximización del ajuste-en-muestra	Minimización del error de predicción

(*) En general, pero no siempre, e.g. rolling estimation

Paralelismos en Econometría/Estadística y ML

- **Evaluación de impacto:** se compara la situación con un escenario contrafactual después de la intervención. El impacto es la diferencia entre lo observado y el contrafactual después de la intervención
- **Machine learning:** se partitiona los datos en entrenamiento y evaluación, y la muestra de entrenamiento se utiliza para realizar un pronóstico que se compara con la muestra test. Se busca que los pronósticos sean los más cercanos a los datos observados en la muestra test



Machine learning y deep learning: matrices de confusión

Rolando Gonzales Martinez, PhD

Fellow postdoctoral Marie Skłodowska-Curie

Universidad de Groningen
(Países Bajos)

Investigador (researcher)
Iniciativa de Pobreza y Desarrollo Humano de la Universidad de Oxford (UK)

Matriz de confusión

En problemas de clasificación, métricas basadas en la **matriz de confusión** pueden utilizarse para comparar y seleccionar modelos de machine learning:

	Predictión Positiva	Predictión Negativa
Clase Positiva	Verdadero Positivo (TP)	Falso Negativo (FN)
Clase Negativa	Falso Positivo (FP)	Verdadero Negativo (TN)

Métricas basadas en la Matriz de confusión

- La exactitud se define como la proporción de predicciones correctas (tanto positivas como negativas) sobre el total de predicciones realizadas.

$$\text{Exactitud (Accuracy)} = \frac{\text{Número de predicciones correctas}}{\text{Número total de predicciones}}$$

$$\text{Exactitud (Accuracy)} = \frac{TP+TN}{TP+TN+FP+FN}$$

En caso de datos desbalanceados otras medidas como el score F1 y la sensibilidad (recall) son más apropiadas

Métricas basadas en la Matriz de confusión

- **Precisión (Positive Predictive Value):** Proporción de ejemplos correctamente clasificados como positivos entre todos los clasificados como positivos.
- **Sensibilidad (Recall, True Positive Rate, TPR):** Proporción de ejemplos positivos correctamente identificados entre todos los ejemplos que son realmente positivos.
- **Especificidad (Specificity o True Negative Rate, TNR):** Proporción de ejemplos negativos correctamente identificados entre todos los ejemplos que son realmente negativos.

$$\text{Precisión} = \frac{TP}{TP + FP}$$

$$\text{Sensibilidad} = \frac{TP}{TP + FN}$$

$$\text{Especificidad} = \frac{TN}{TN + FP}$$

Métricas basadas en la Matriz de confusión

- **Valor Predictivo Negativo:** Proporción de negativos entre todos los ejemplos que fueron clasificados como negativos.
- **F1-Score:** Media armónica entre la precisión y la sensibilidad
- **Tasa de Falsos Positivos:** Proporción de ejemplos negativos que fueron incorrectamente clasificados como positivos.

$$\text{Valor Predictivo Negativo} = \frac{TN}{TN + FN}$$

$$\text{F1-Score} = \frac{2 \times \text{Precisión} \times \text{Sensibilidad}}{\text{Precisión} + \text{Sensibilidad}}$$

$$\text{Tasa de Falsos Positivos} = \frac{FP}{FP + TN}$$

Métricas basadas en la Matriz de confusión

Exactitud balanceada, prevalencia (porcentaje de casos positivos), prevalencia de detección (casos predichos como positivos), tasa de detección (casos correctamente predichos como positivos):

$$\text{Exactitud Balanceada} = \frac{\text{Sensibilidad} + \text{Especificidad}}{2}$$

$$\text{Prevalencia} = \frac{TP + FN}{TP + TN + FP + FN}$$

$$\text{Prevalencia de Detección} = \frac{TP + FP}{TP + TN + FP + FN}$$

$$\text{Tasa de Detección} = \frac{TP}{TP + TN + FP + FN}$$

Otras métricas basadas en la matriz de confusión: Kappa de Cohen

- Kappa de Cohen es una métrica que se utiliza para evaluar la concordancia entre dos clasificadores.
- Es una medida estadística que compara la precisión observada (P_o) de un modelo con la precisión esperada (P_e) si las predicciones fueran completamente aleatorias

$$\text{Kappa} = \frac{P_o - P_e}{1 - P_e} \quad P_o = \frac{TP + TN}{\text{Total de casos}}$$

Probabilidad marginal para la clase positiva:

$$P_{\text{real, positiva}} = \frac{TP + FN}{\text{Total}} \quad P_{\text{predicha, positiva}} = \frac{TP + FP}{\text{Total}}$$

Probabilidad marginal para la clase negativa:

$$P_{\text{real, negativa}} = \frac{TN + FP}{\text{Total}} \quad P_{\text{predicha, negativa}} = \frac{TN + FN}{\text{Total}}$$

$$P_e = (P_{\text{real, positiva}} \times P_{\text{predicha, positiva}}) \\ + (P_{\text{real, negativa}} \times P_{\text{predicha, negativa}})$$

Otras métricas basadas en la matriz de confusión: Kappa de Cohen

Interpretación:

- **Kappa = 1:** Indica un acuerdo perfecto entre el modelo y las etiquetas verdaderas.
- **Kappa = 0:** Indica que el acuerdo es igual al que se esperaría por azar (es decir, no hay un acuerdo más allá de lo que el azar proporciona).
- **Kappa < 0:** Indica un acuerdo peor que el azar, lo que sugiere que el modelo está haciendo un trabajo pobre en clasificar correctamente.
- **Kappa entre 0.6 y 0.8:** Se considera un acuerdo sustancial.
- **Kappa entre 0.8 y 1:** Se considera un acuerdo casi perfecto.

Otras métricas basadas en la matriz de confusión: Test de McNemar

- Se puede utilizar para comparar (1) las predicciones de un clasificador contra los datos reales, o (2) las predicciones de dos clasificadores.

	Observado Positivo (1)	Observado Negativo (0)
Predicción Positiva (1)	n_{11}	n_{10}
Predicción Negativa (0)	n_{01}	n_{00}

$$M = \frac{(n_{10} - n_{01})^2}{n_{10} + n_{01}} \sim \chi^2(df = 1)$$

Otras métricas basadas en la matriz de confusión:

Test de McNemar

- Se puede utilizar para comparar (1) las predicciones de un clasificador contra los datos reales, o (2) las predicciones de dos clasificadores.
- La hipótesis nula es que no hay diferencia entre las proporciones de discordancia (i.e. los elementos fuera de la diagonal son iguales), por lo que no rechazar la hipótesis nula:
 - En el caso de (1) sugiere un buen ajuste del modelo.
 - En el caso de (2) indica que no hay diferencias entre clasificadores. El rechazo de la nula implica que uno de los clasificadores es mejor que el otro clasificador.

Machine learning y deep learning: Lenguajes de programación más utilizados en machine learning y deep learning

Rolando Gonzales Martinez, PhD

Fellow postdoctoral Marie
Skłodowska-Curie

Universidad de Groningen
(Países Bajos)

Investigador (researcher)
Iniciativa de Pobreza y Desarrollo
Humano de la Universidad de
Oxford (UK)

R

<https://www.r-project.org/>

Lenguaje y entorno para computación desarrollado por Robert Gentleman y Ross Ihaka (Universidad de Auckland) en 1993.



- Similar al lenguaje y entorno S que fue desarrollado en Bell Laboratories (anteriormente AT&T, ahora Lucent Technologies) por John Chambers y colegas a finales de los 70's.
- R está disponible como Software Libre bajo los términos de la Licencia Pública General GNU de la Free Software Foundation en forma de código fuente.
- Compila y se ejecuta en una amplia variedad de plataformas UNIX y sistemas similares (Linux), Windows y MacOS.

Python

- Lenguaje de programación.
- Python fue concebido a fines de la década de 1980 por Guido van Rossum en Centrum Wiskunde & Informatica (CWI) en los Países Bajos.
- 2000: Python 2.0
- 2018: Guido van Rossum, el "dictador benévolο de por vida" de Python, responsable de la toma de decisiones del proyecto, deja el proyecto
- 2022: Python 3.x

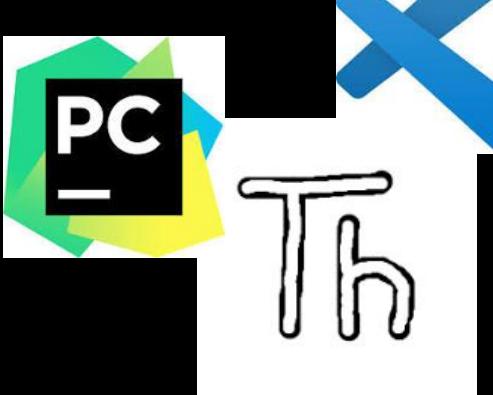
Wikipedia, Google, Yahoo!, CERN, NASA, Facebook, Amazon, Instagram, Spotify



Python

IDE (Integrated Development Environments):

- PyCharm
- Jupyter
- Spyder
- Python IDLE
- Thonny
- ...



Spyder (Python 3.9)

File Edit Search Source Run Debug Consoles Projects Tools View Help

Z:\UpWork\UpWork2023\Heolpython\Hao.py

tempy X Hao.py X IRT_Andrew.py X

Nam Type Size Value

Console I/A X

Python 3.9.13 (main, Aug 25 2022, 23:51:50) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.
IPython 7.31.1 -- An enhanced Interactive Python.

In [1]:

```
# -*- coding: utf-8 -*-
# Created on Mon Jul 24 07:22:02 2023
# Author: Rolando
"""

from IPython import get_ipython
get_ipython().magic('reset -sf')

import pandas as pd

import os
os.getcwd()
db_in = 'Z:\UpWork\UpWork2023\Hao'
os.chdir(db_in)

# Loading data
du = pd.read_csv('5_esm.csv', sep=';')
du.head()

do = pd.read_csv('DisplayOn.csv', header = None)
do.head()
column_names = ['id', 'flog'] # Replace with your desired column names
do.columns = column_names

da = pd.read_csv('AppUsage.csv', header = None)
da.head()
column_names = ['id', 'app', 'text'] # Replace with your desired column names
da.columns = column_names
da['text'] = da['text'].str.lower()

ds = [facebook] + da['text'].str.extract(r'(facebook)')
ds[facebook] = ds[facebook].str.extract(r'(https://www\.facebook\.com/[^/]+)')
ds[Instagram] = da['text'].str.extract(r'(Instagram)')
ds[Twitter] = da['text'].str.extract(r'(Twitter)')
ds[Snaphat] = da['text'].str.extract(r'(snapchat)')
ds[TikTok] = da['text'].str.extract(r'(tiktok)')
ds[Youtube] = da['text'].str.extract(r'(youtube)')
```

Python y Spyder



The screenshot shows the Spyder Python IDE interface. The top menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, and Help. The main window has tabs for temp.py, Hao.py, and IRT_Andrew.py. The code editor displays Python code for a Stan model, including imports for numpy, pandas, cmdstanpy, and various plotting libraries like matplotlib, plotnine, and plotly. It also loads a 'caries.csv' dataset and defines a 'model' dictionary. The Variable Explorer on the right shows variables like db_in (str), dentist_data (DataFrame), I (int), J (int), and model (dict). The bottom IPython Console shows command-line output related to CmdStan installations and configuration.

```
# -*- coding: utf-8 -*-
# Created on Fri Jul 21 18:30:39 2023
@author: Rolando
from IPython import get_ipython
get_ipython().magic('reset -sf')

db_in = 'Z:\UpWork\UpWork2023\Andrew_Bayes'
import os
os.getcwd()
os.chdir(db_in)

# data handling libs
import numpy as np
import pandas as pd

# stan modeling lib
import cmdstanpy
cmdstanpy.install_cmdstan()
cmdstanpy.install_cmdstan(compiler=True) # only valid on Windows
from cmdstanpy import CmdStanModel

# plotting libs
import matplotlib.pyplot as plt
import plotnine as p9
import plotly.express as px
import plotly.graph_objects as go
import plotly.io as pioy

## Loading data
dentist_data = pd.read_csv("caries.csv", skiprows=3)
dentist_data.head()

## Defining parameters of the MCMC and functions
# dictionary to store different models
model = {'sbc':{}, 'ppc':{}, 'cv':{}}
```

Name	Type	Size	Value
db_in	str	33	Z:\UpWork\UpWork2023\Andrew_Bayes
dentist_data	DataFrame	(19345, 3)	Column names: coder, item, response
I	int	1	3869
J	int	1	5
model	dict	3	{'sbc':{}, 'ppc':{}, 'cv':{}}

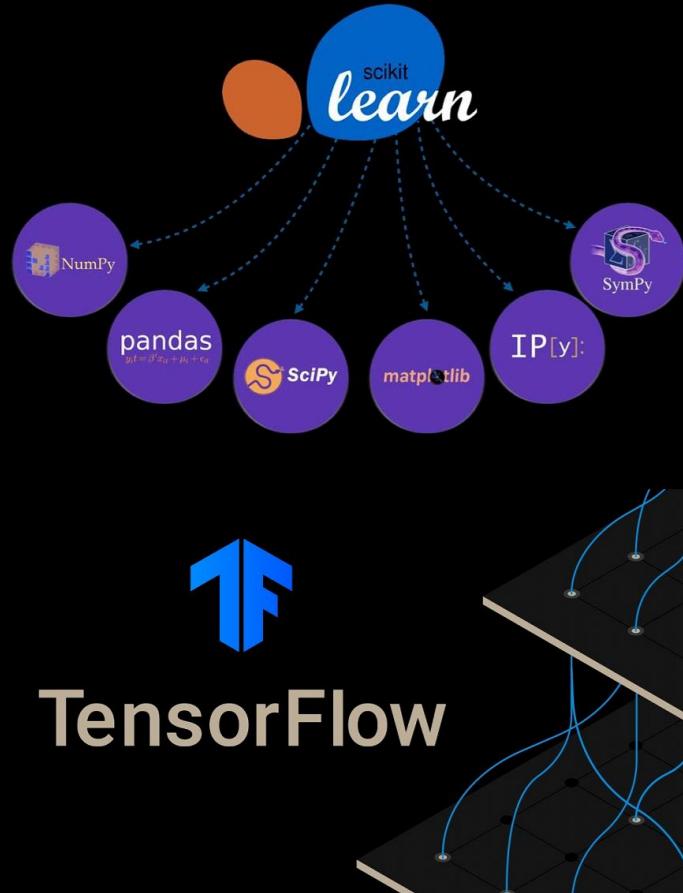
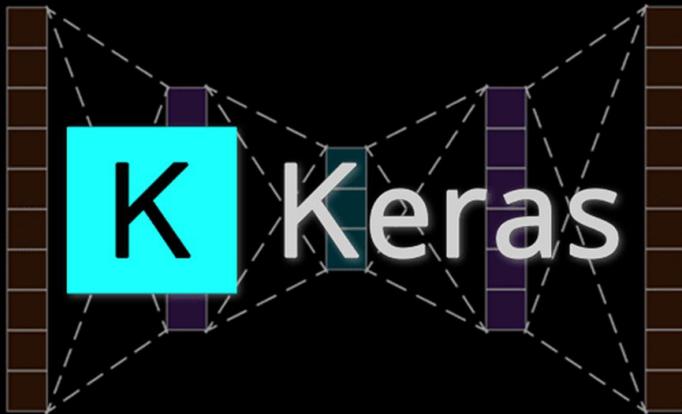
```
... num_draws = NUM_POSTERIOR_SAMPLES//THINNING_GAP*CHAINS
... def run_stan(model, data, show_console=True):
...     # generic function to run stan model given a set of config parameters
...
...     fit = model.sample(data = data,
...                         chains = CHAINS,
...                         iter_warmup = WARMUP_SAMPLES,
...                         iter_sampling = NUM_POSTERIOR_SAMPLES,
...                         thin = THINNING_GAP,
...                         show_progress = False,
...                         show_console = show_console)
...
...     return fit
...
... # SBC config variables
... NUM_SIMS = 100
Installing CmdStan version: 2.32.2
Install directory: C:\Users\Rolando\.cmdstan
CmdStan version 2.32.2 already installed
23:25:19 - cmdstanpy - INFO - Add C++ toolchain to $PATH: C:\Users\Rolando\.cmdstan\RTools40
Installing CmdStan version: 2.32.2
IPython Console History
```

LSP Python: ready conda base (Python 3.9.13) Line 24, Col 1 UTF-8 CRLF RW Mem 59%

Python: librerias

Librerias/paquetes especializados:

- Numpy
- Pandas
- Matplotlib
- Scikit-learn
- Tensorflow
- Keras



Python: Scikit-learn

<https://scikit-learn.org/stable/>

- Clasificación
- Regresión
- Agrupamiento (clustering)
- Reducción de la dimensionalidad
- Selección de modelos
- Pre-procesamiento de datos (preparación de datos)

The screenshot shows the official scikit-learn website. At the top, there's a navigation bar with links for Install, User Guide, API, Examples, Community, and More. Below the header, there's a main title "scikit-learn Machine Learning in Python". A sidebar on the right lists the following benefits:

- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

The main content area is divided into several sections, each with a title, a brief description, and a "Examples" section featuring a small image:

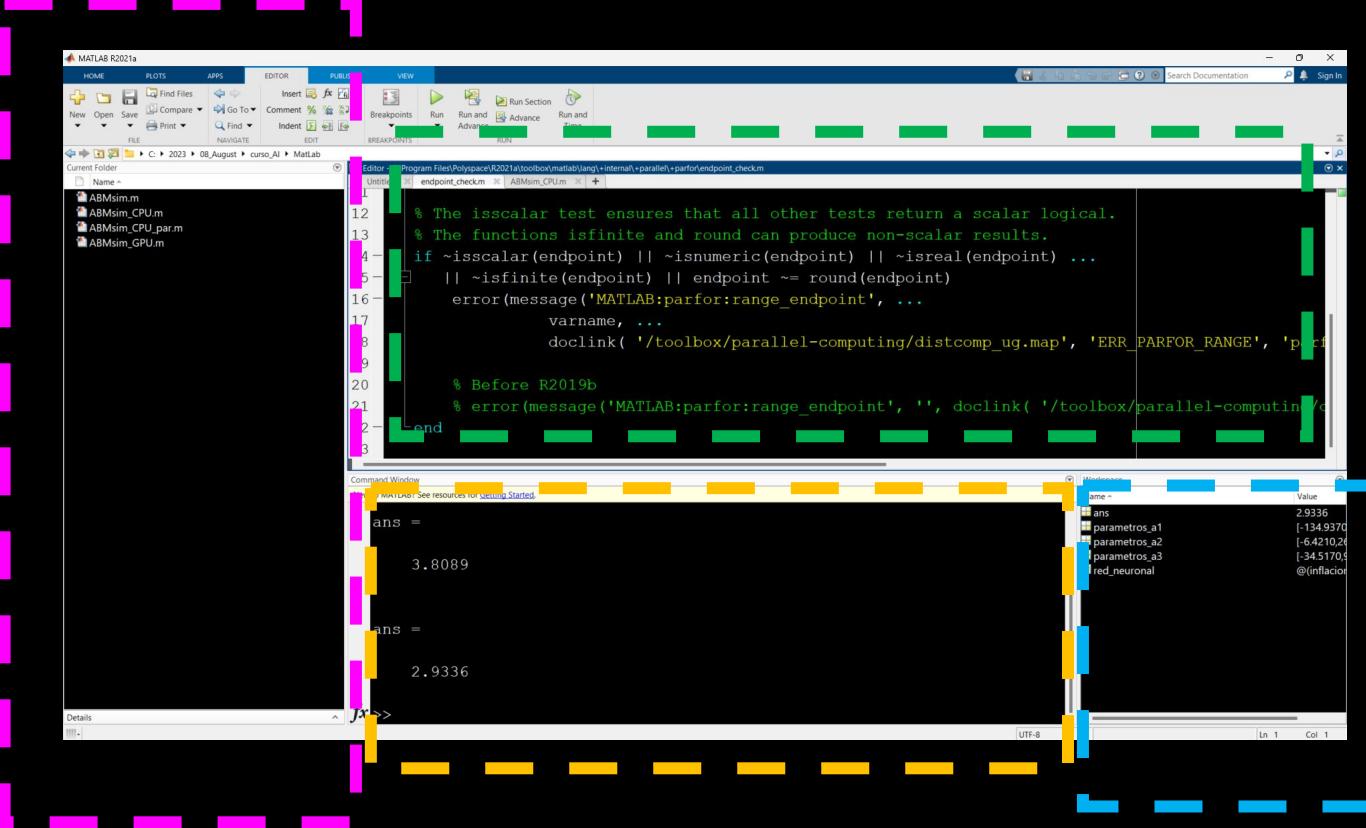
- Classification**: Identifying which category an object belongs to. Applications: Spam detection, image recognition. Algorithms: Gradient boosting, nearest neighbors, random forest, logistic regression, and more... Examples: A grid of 9 plots showing 2D classification boundaries.
- Regression**: Predicting a continuous-valued attribute associated with an object. Applications: Drug response, Stock prices. Algorithms: Gradient boosting, nearest neighbors, random forest, ridge, and more... Examples: A line plot titled "Boosted Decision Tree Regression" showing data points and two fitted curves.
- Clustering**: Automatic grouping of similar objects into sets. Applications: Customer segmentation, Grouping experiment outcomes. Algorithms: k-Means, HDBSCAN, hierarchical clustering, and more... Examples: A scatter plot titled "K-means clustering on the digits dataset" where data points are colored by cluster assignment.
- Dimensionality reduction**: Reducing the number of random variables to consider. Applications: Visualization, Increased efficiency. Algorithms: PCA, feature selection, non-negative matrix factorization, and more... Examples: A 3D scatter plot of the Iris dataset labeled "Virginica", "Versicolor", and "Setosa".
- Model selection**: Comparing, validating and choosing parameters and models. Applications: Improved accuracy via parameter tuning. Algorithms: grid search, cross validation, metrics, and more... Examples: A line plot showing the relationship between a parameter and model performance.
- Preprocessing**: Feature extraction and normalization. Applications: Transforming input data such as text for use with machine learning algorithms. Algorithms: preprocessing, feature extraction, and more... Examples: A grid of 9 plots showing various data transformations like scaling and normalization.

MatLab

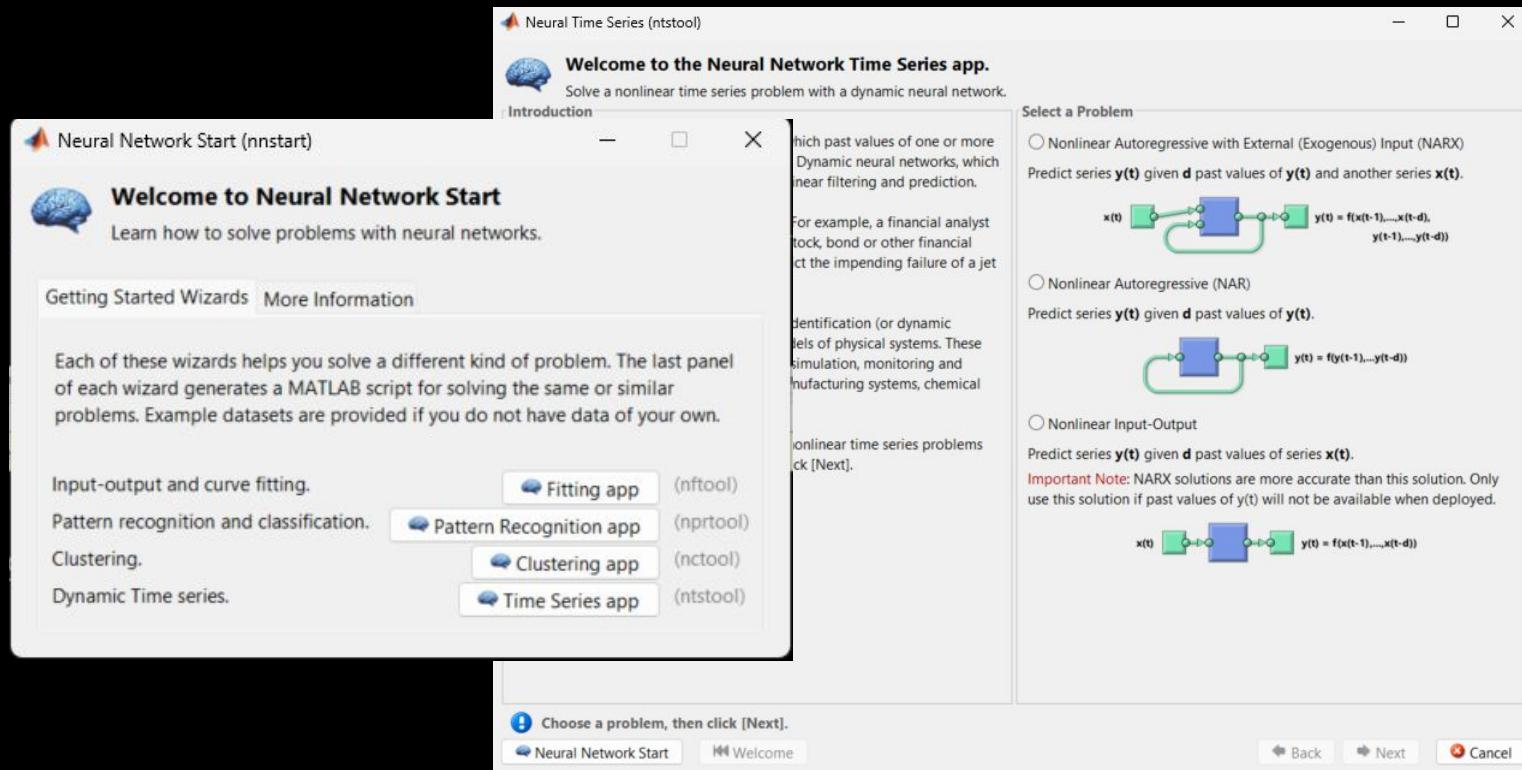
- MATLAB (abreviatura de "Matrix Laboratory") es un entorno de programación y un lenguaje de programación
- Cálculo numérico, análisis de datos, visualización y la simulación.
- Desarrollado por MathWorks
- Ampliamente utilizado en campos como la ingeniería, las ciencias de la computación, las matemáticas aplicadas y otras disciplinas científicas y técnicas.



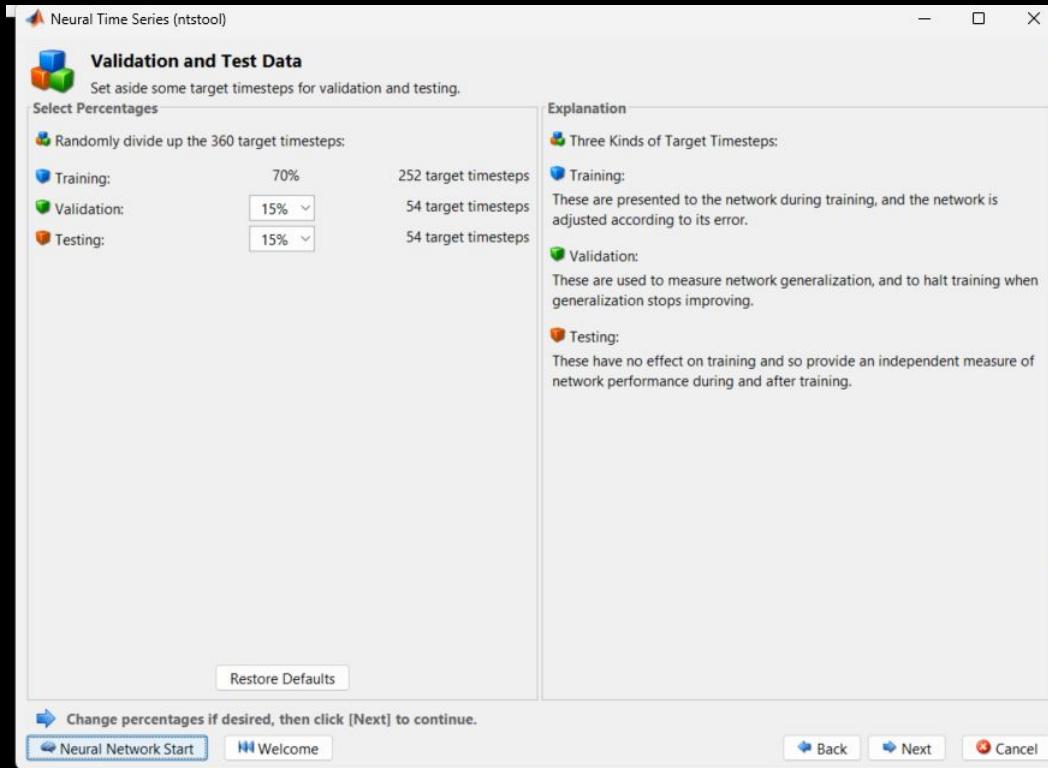
MatLab



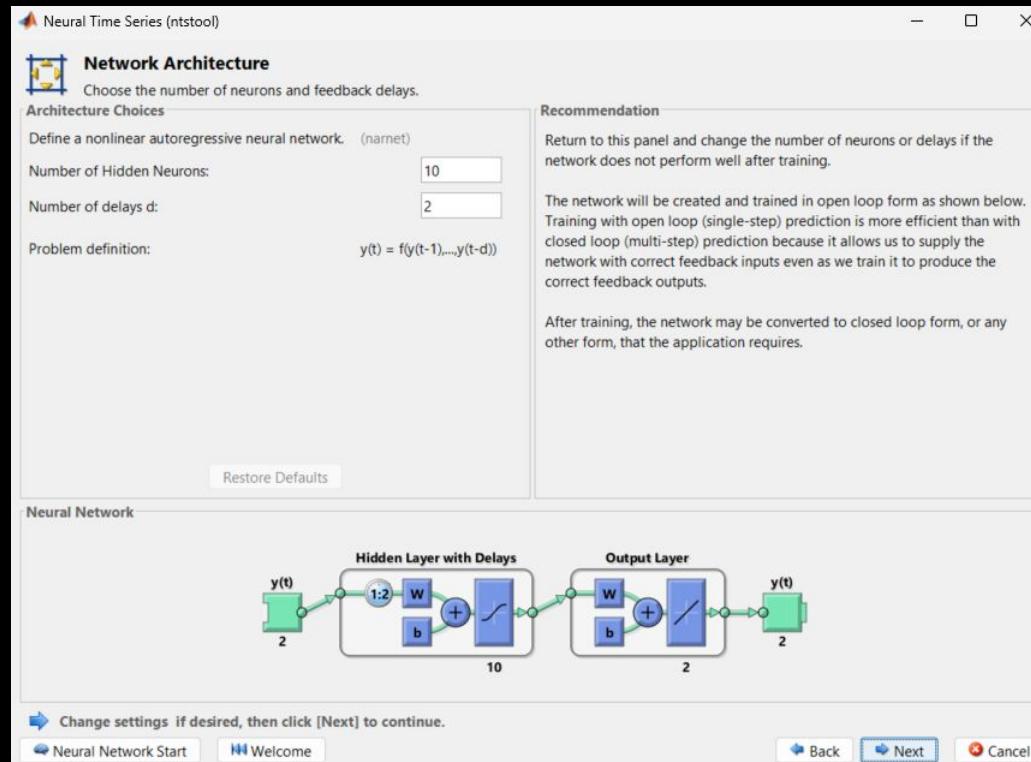
MatLab y redes neuronales artificiales



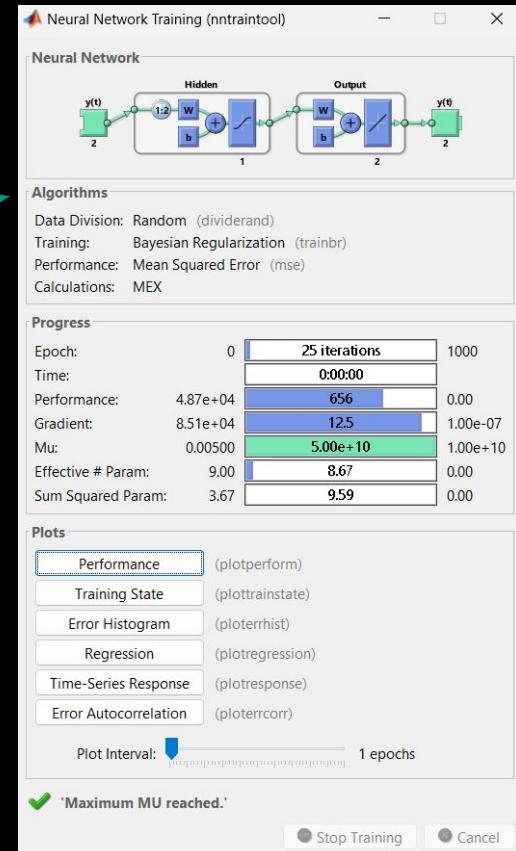
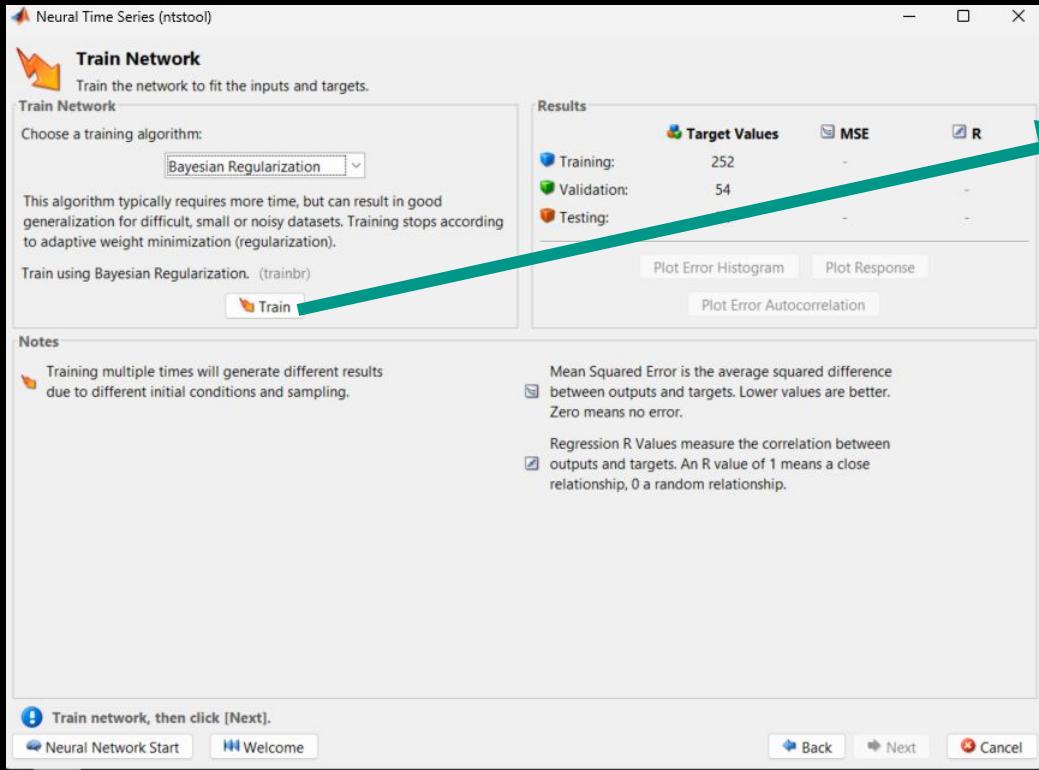
Matlab y Redes Neuronales Artificiales



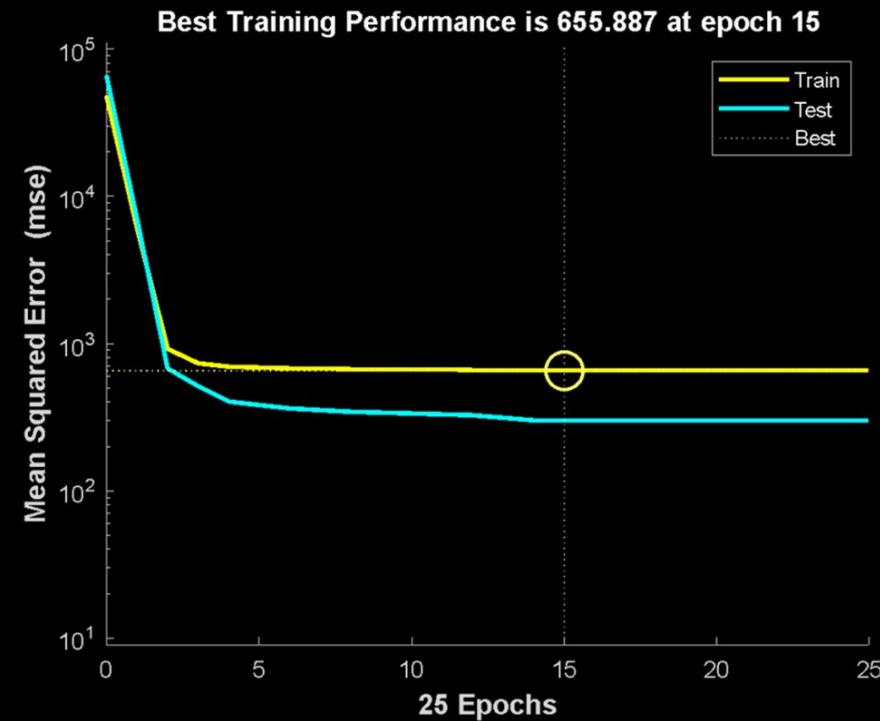
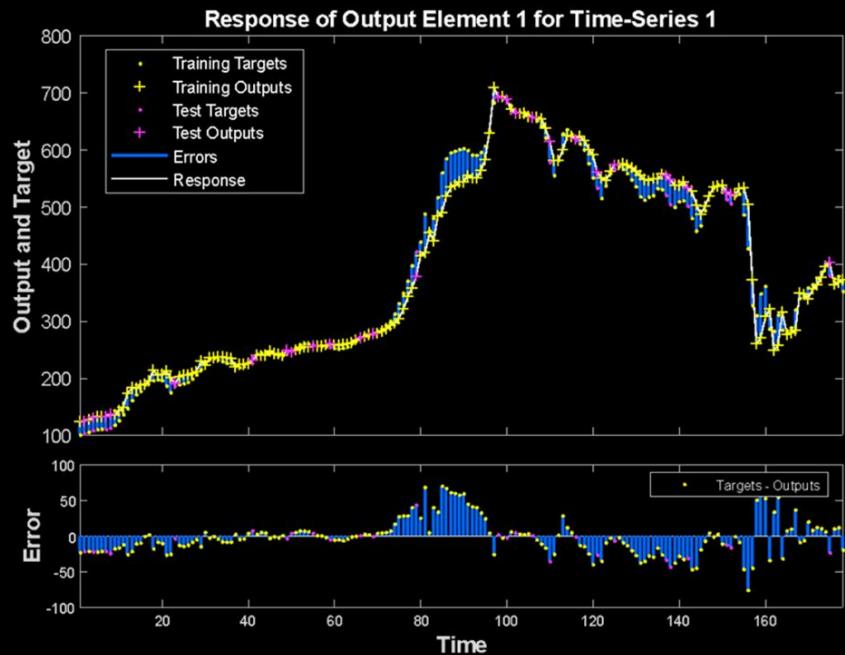
Matlab y Redes Neuronales Artificiales



Matlab y Redes Neuronales Artificiales



Matlab y Redes Neuronales Artificiales



Big data

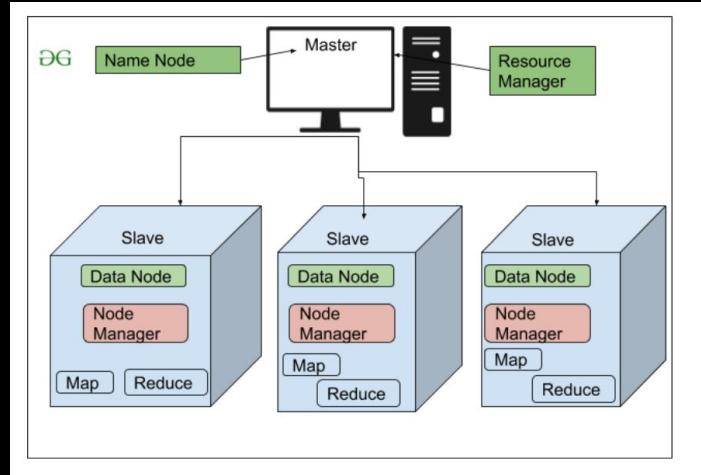
3 V's:

- Volumen: Terabytes, petabytes o incluso exabytes que no pueden ser gestionados o procesados mediante sistemas convencionales (sensores, redes sociales, registros de transacciones)
- Velocidad: se genera a una velocidad increíblemente rápida, en tiempo real o en intervalos muy cortos.
- Variedad: Los datos masivos pueden presentar una amplia variedad de formatos y tipos, como texto, imágenes, audio, vídeo, datos geoespaciales, datos estructurados y no estructurados.



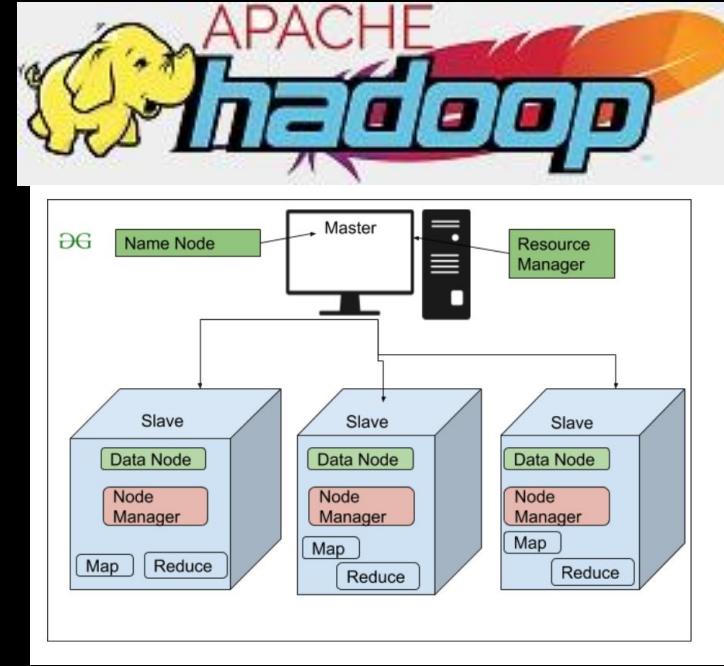
Infraestructura

- Apache Hadoop: sistema diseñado para almacenar grandes cantidades de datos en clústeres de servidores. Divide los archivos en bloques y los distribuye a través de los nodos del clúster.
- Estructura maestro-esclavo: El NameNode actúa como el maestro y es responsable de mantener el metadata del sistema de archivos, incluidos los nombres de los archivos, las ubicaciones de los bloques y los permisos. Los DataNodes son los nodos esclavos que almacenan los bloques de datos reales.



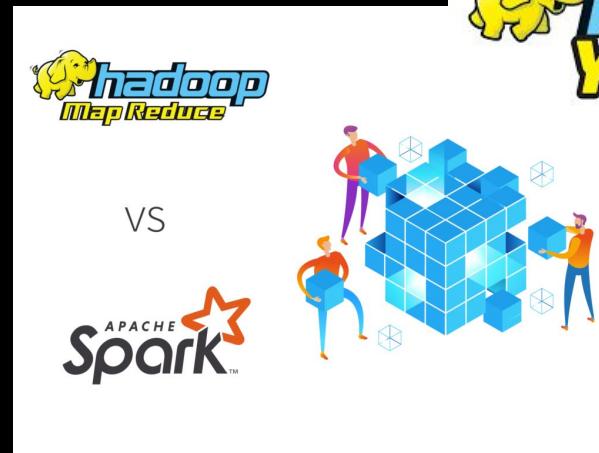
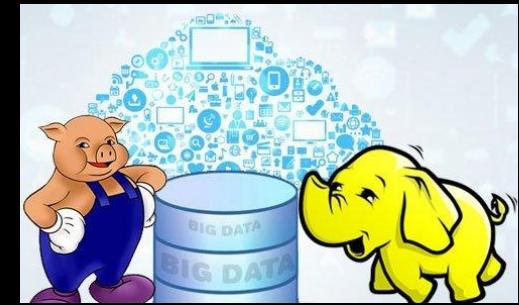
Infraestructura

- MapReduce: procesamiento de datos que permite dividir tareas complejas en pasos de "map" y "reduce".
- La fase de "map" implica procesar y filtrar los datos, y la fase de "reduce" se encarga de combinar y resumir los resultados del paso de "map". Esto permite el procesamiento paralelo y distribuido de tareas de análisis de datos en el clúster.



Ecosistema Hadoop

- Hive: Almacenamiento de datos y consulta que permite realizar consultas tipo SQL en datos de almacenamiento distribuido (HDFS).
- Pig: Un lenguaje para realizar transformaciones y análisis de datos en Hadoop.
- Spark: Un sistema de procesamiento de datos en memoria que es más rápido que MapReduce para ciertos tipos de tareas.
- YARN: Un administrador de recursos que permite ejecutar varias aplicaciones en el mismo clúster de manera eficiente.



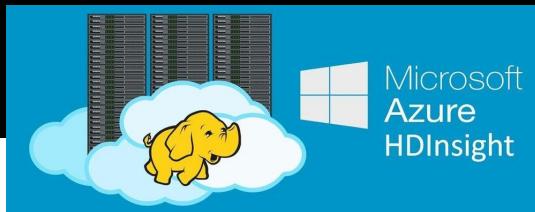
Opciones a Apache Hadoop

- **Apache Flink:** Admite el procesamiento por lotes. Está diseñado para el procesamiento de baja latencia y alto rendimiento de datos en tiempo real. Las capacidades de procesamiento en tiempo de evento y cálculos con estado hacen que sea adecuado para análisis en tiempo real y aplicaciones que requieren un procesamiento de eventos complejos.
- **Amazon EMR (Elastic MapReduce):** EMR es un servicio gestionado proporcionado por Amazon Web Services (AWS) que simplifica la implementación y administración de marcos de datos grandes, incluidos Hadoop, Spark, Flink y otros. Es una alternativa basada en la nube que te permite crear clústeres y centrarte en el procesamiento de datos sin lidiar con la gestión de la infraestructura.
- **Cloudera:** Cloudera proporciona una plataforma que incluye Hadoop y otras herramientas de datos grandes de código abierto, junto con capacidades de administración y supervisión. Su objetivo es simplificar la implementación y administración de tecnologías de datos grandes y ofrece soluciones tanto en las instalaciones como en la nube.



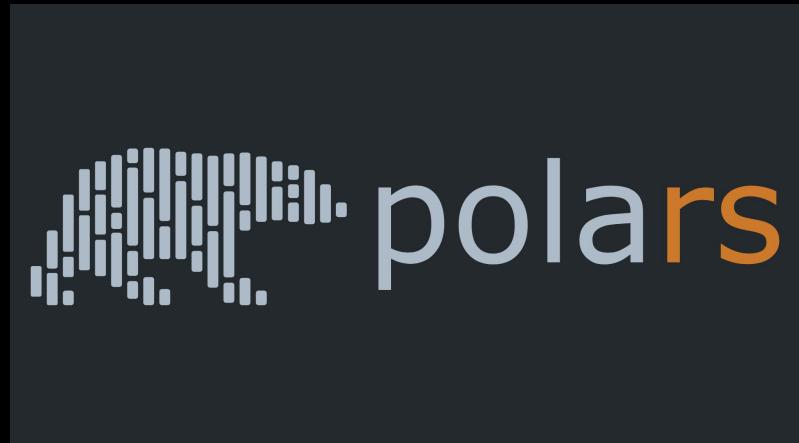
Opciones a Apache Hadoop

- **MapR:** MapR ofrece su propia distribución de Hadoop con mejoras adicionales para un mejor rendimiento y confiabilidad. También proporciona capacidades para el procesamiento de datos en tiempo real, el procesamiento de transmisiones y la integración de datos.
- **Google Cloud Dataproc:** Similar a Amazon EMR, Google Cloud Dataproc es un servicio gestionado para ejecutar Apache Hadoop, Spark y otros marcos de datos grandes en Google Cloud Platform.
- **Microsoft Azure HDInsight:** Este es un servicio basado en la nube de Microsoft que ofrece Hadoop, Spark y otras herramientas de datos grandes como un servicio gestionado en la plataforma Azure.
- **IBM BigInsights:** IBM ofrece su propia distribución de Hadoop junto con herramientas adicionales para gobernanza de datos, integración y análisis.



Opciones más accesibles

- Computación paralela (CPU)
- GPU, TPU
- Computación en la nube (CPU, GPU)
- Uso de librerías como Polars



Machine learning y deep learning: Aplicaciones

Rolando Gonzales Martinez, PhD

Fellow postdoctoral Marie Skłodowska-Curie

Universidad de Groningen
(Países Bajos)

Investigador (researcher)
Iniciativa de Pobreza y Desarrollo Humano de la Universidad de Oxford (UK)

Ejemplos de aplicaciones

- Bosques aleatorios se emplean para identificar qué programas sociales promueven un mejor desempeño financiero y una mejor situación social.
- Las predicciones de los árboles de decisión se combinan en una predicción final de los rendimientos financieros.
- El resultado combinado es el “bosque” aleatorio.

INTERNATIONAL JOURNAL OF SUSTAINABLE DEVELOPMENT & WORLD ECOLOGY
2020, VOL. 27, NO. 5, 389–395
<https://doi.org/10.1080/13504509.2019.1706059>

 Taylor & Francis
Taylor & Francis Group

 OPEN ACCESS 

**Which social program supports sustainable grass-root finance?
Machine-learning evidence**

R. Gonzales Martinez 

Handelshøyskolen, University of Agder, Kristiansand, Norway

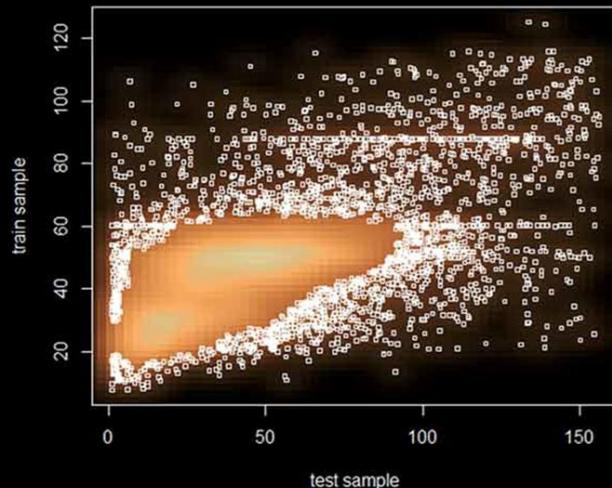
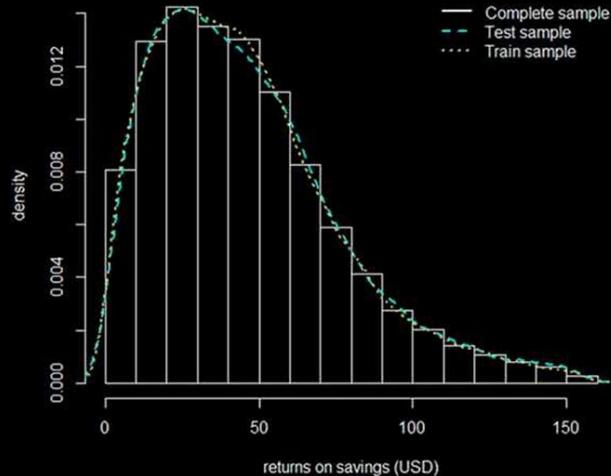
ABSTRACT
Resources for development are used efficiently when social programs help to promote at the same time the sustainability of grass-root financial associations at the bottom of the pyramid. This study applies machine-learning to a worldwide database of grass-root associations in order to identify which social programs are good predictors of financial returns in the groups. The results indicate that education, income-generating activities and health programs are the most frequent programs provided by development agencies. Business training is not the most frequent intervention applied to grass-root associations, but it is in fact the most important social program to encourage financial sustainability, particularly after a development agency stops working with a group and leaves the community. Theoretical and practical implications of the findings are discussed.

ARTICLE HISTORY
Received 7 November 2019
Accepted 13 December 2019

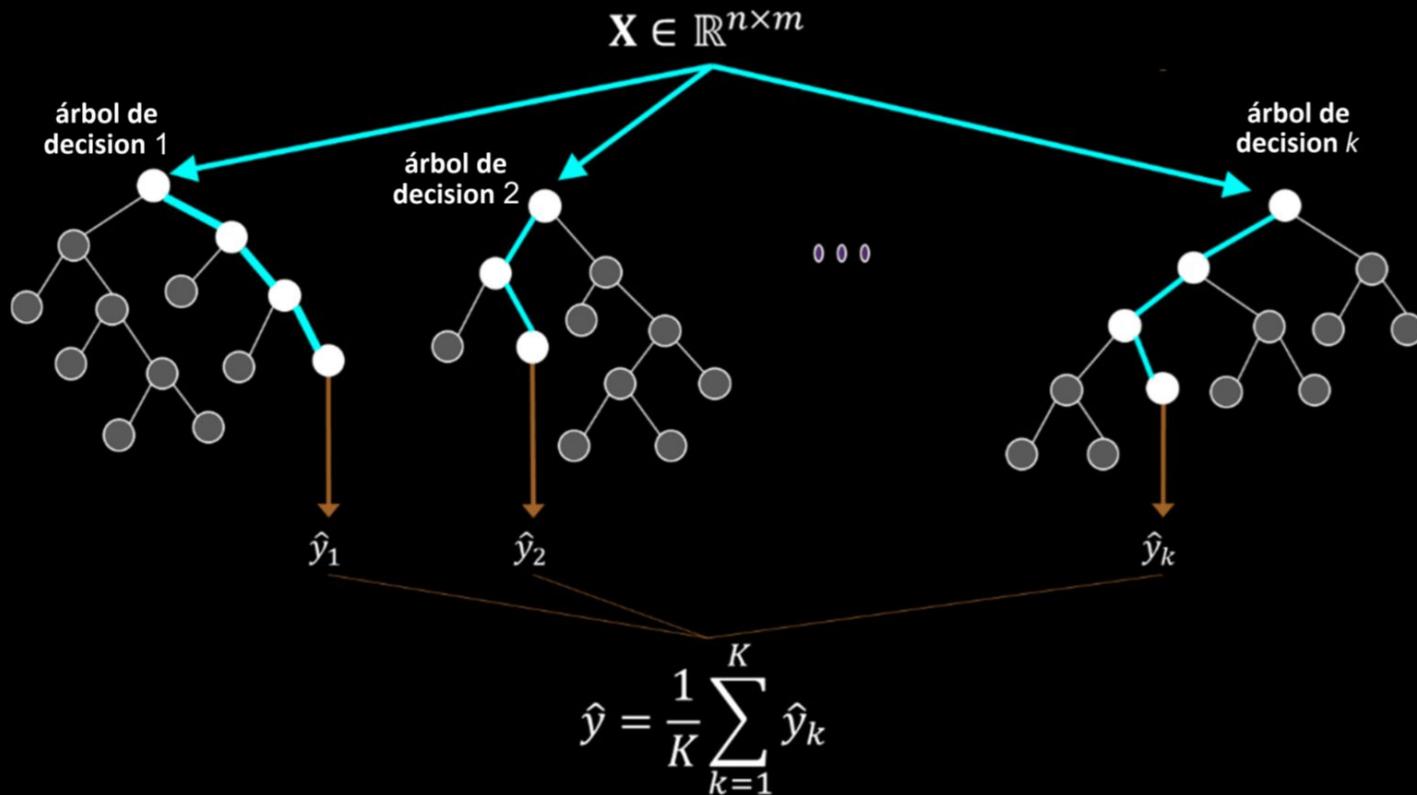
KEYWORDS
Grass-root finance; health and business interventions; sustainable development; machine-learning

Ejemplos de aplicaciones

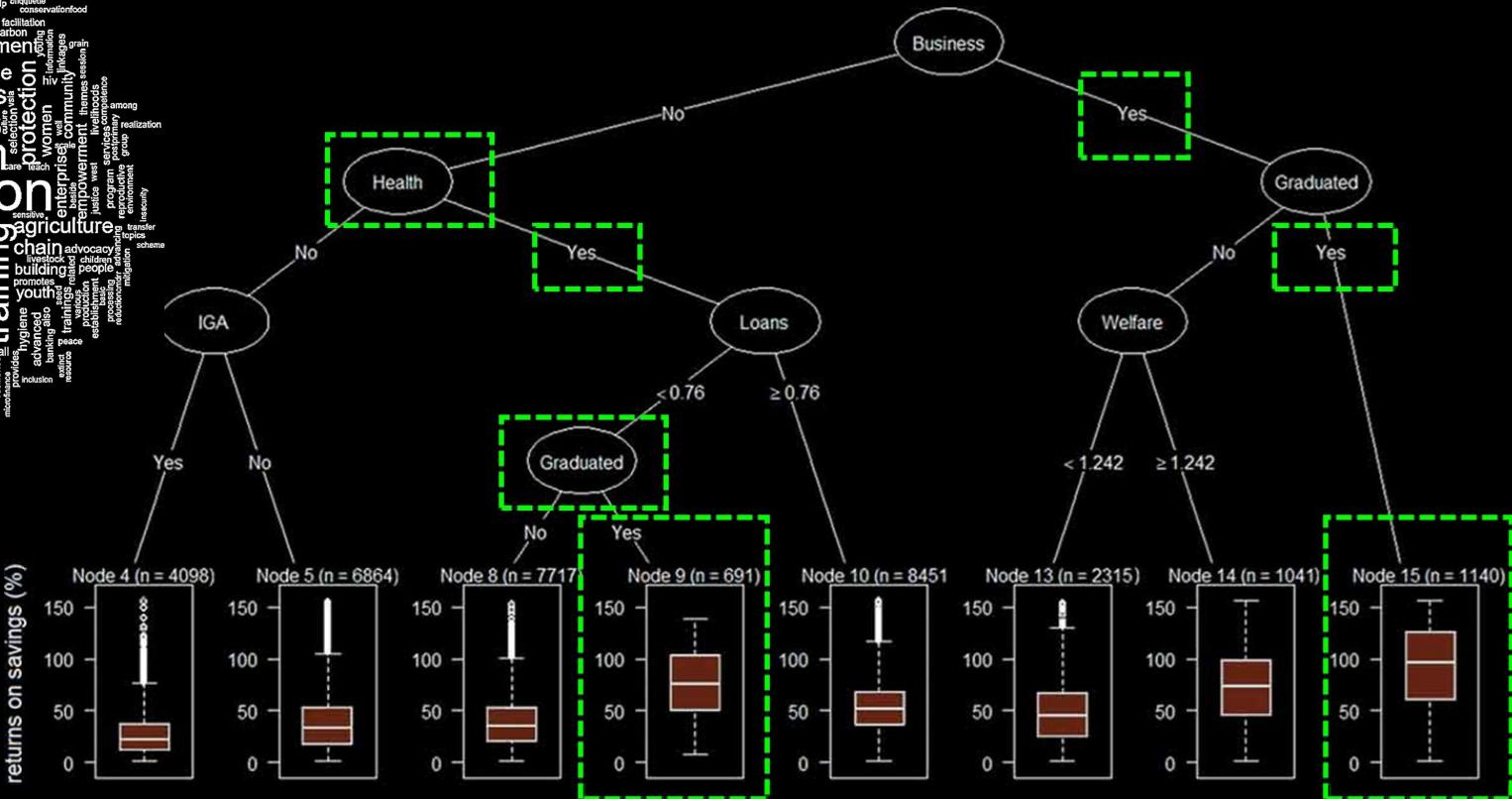
- Validación cruzada: la base de datos se dividió aleatoriamente en una muestra de entrenamiento y una muestra de validación
- 65% de los datos en el conjunto de entrenamiento.
- Se evaluó el error cuadrático medio (RMSE) y error de predicción absoluto medio (MAPE).



Ejemplos de aplicaciones

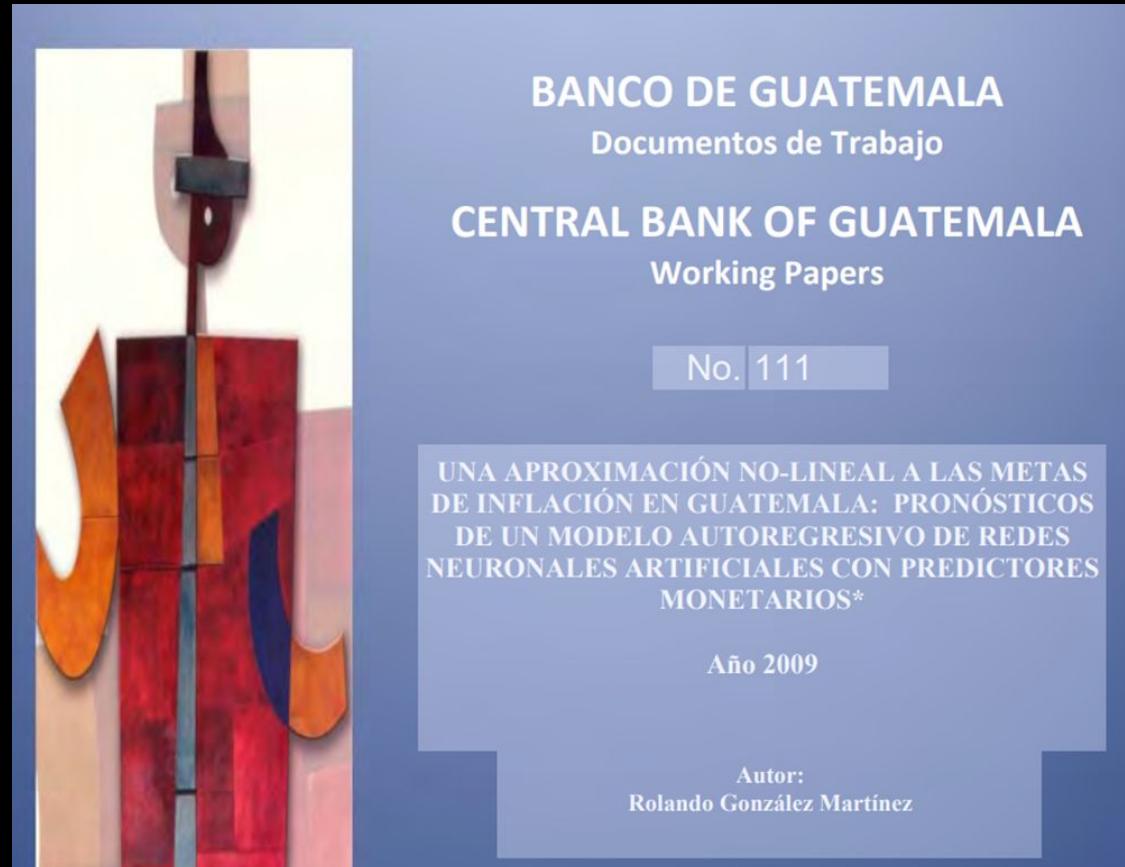


Ejemplos de aplicaciones



Ejemplos de aplicaciones

- Pronóstico de la inflación con redes neuronales artificiales
- Pronóstico condicionado a la evolución de los agregados monetarios y la tasa de interés de referencia



Ejemplos de aplicaciones

- Los bancos centrales tienen que cumplir la meta de inflación con cualquier instrumento
- Pronósticos precisos son necesarios y si la meta se cumple los bancos centrales aumentan su reputación ante los agentes económicos
- Modelos no-lineales pueden ser más precisos que modelos lineales



BANCO DE GUATEMALA

Documentos de Trabajo

CENTRAL BANK OF GUATEMALA

Working Papers

No. 111

UNA APROXIMACIÓN NO-LINEAL A LAS METAS
DE INFLACIÓN EN GUATEMALA: PRONÓSTICOS
DE UN MODELO AUTOREGRESIVO DE REDES
NEURONALES ARTIFICIALES CON PREDICTORES
MONETARIOS*

Año 2009

Autor:
Rolando González Martínez

Ejemplos de aplicaciones

Para ajustar y evaluar el poder predictivo del modelo ARNAX se dividió la serie de tiempo de la inflación en tres partes (véase también el gráfico 1):

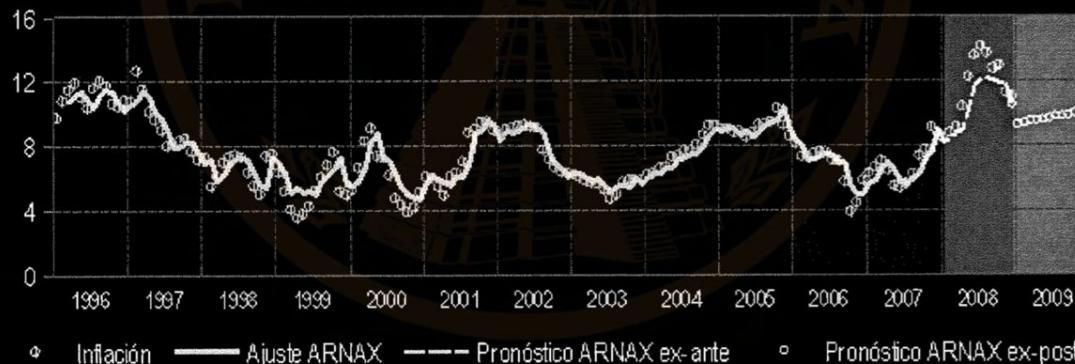
Entrenamiento:	enero 1996 a diciembre 2007
Pronóstico expost:	enero 2008 a noviembre 2008
Pronóstico exante:	diciembre 2008 a diciembre 2009



Ejemplos de aplicaciones

Tabla 1. Estimación de los parámetros del modelo ARNAX y ajuste en el segmento de entrenamiento y validación

Capa de Salida Paráme- etros	Estimado- res	Capa oculta nodo I Paráme- etros	Estimado- res	Capa oculta nodo II Paráme- etros	Estimado- res
ω_0	5.107	$\theta_{1,1}$	-5.179	$\theta_{2,1}$	0.345
ω_1	7.384	$\theta_{1,2}$	0.595	$\theta_{2,2}$	-0.324
ω_2	-1.231	$\theta_{1,3}$	-0.007	$\theta_{2,3}$	-0.086
		$\theta_{1,4}$	0.047	$\theta_{2,4}$	0.544
Raíz de la suma de cuadrados de los errores					
Segmento de entrenamiento			5.7198		
Segmento de validación			2.4917		



Evaluación de los Pronósticos (expost)

Raíz del error cuadrático medio	3.5121	1.3182
Error promedio absoluto	2.9163	1.1426
Coeficiente de Theil	0.1724	0.0588

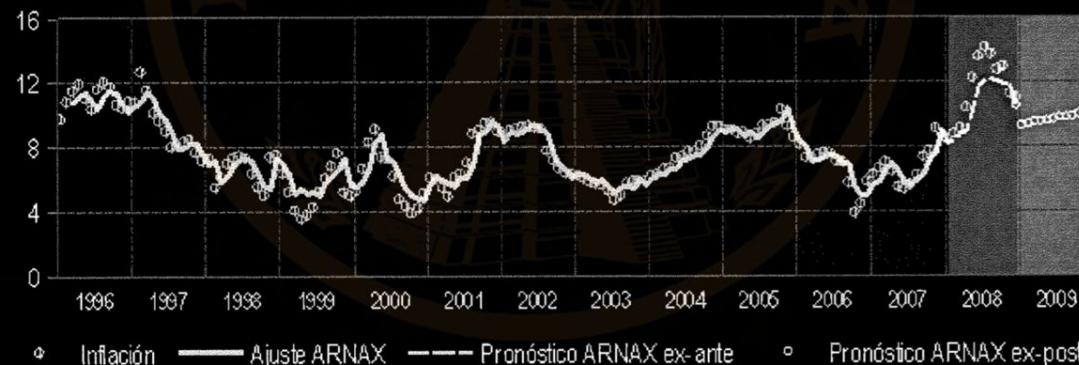
Estadígrafo Diebold- Mariano para comparar la exactitud del pronóstico

Estadígrafo	2.713602	Valores críticos:
p-value	0.003327	nivel del 1% 2.32
		nivel del 5% 1.64

Ejemplos de aplicaciones

Tabla 1. Estimación de los parámetros del modelo ARNAX y ajuste en el segmento de entrenamiento y validación

Capa de Salida Paráme- etros	Estimado- res	Capa oculta nodo I Paráme- etros	Estimado- res	Capa oculta nodo II Paráme- etros	Estimado- res
ω_0	5.107	$\theta_{1,1}$	-5.179	$\theta_{2,1}$	0.345
ω_1	7.384	$\theta_{1,2}$	0.595	$\theta_{2,2}$	-0.324
ω_2	-1.231	$\theta_{1,3}$	-0.007	$\theta_{2,3}$	-0.086
		$\theta_{1,4}$	0.047	$\theta_{2,4}$	0.544
Raíz de la suma de cuadrados de los errores					
Segmento de entrenamiento			5.7198		
Segmento de validación			2.4917		



Evaluación de los Pronósticos (expost)

Raíz del error cuadrático medio	3.5121	1.3182
Error promedio absoluto	2.9163	1.1426
Coeficiente de Theil	0.1724	0.0588

Estadígrafo Diebold- Mariano para comparar la exactitud del pronóstico

Estadígrafo	2.713602	Valores críticos:
p-value	0.003327	nivel del 1% 2.32
		nivel del 5% 1.64

Otras métricas basadas en la matriz de confusión:

Test de McNemar

- Se puede utilizar para comparar (1) las predicciones de un clasificador contra los datos reales, o (2) las predicciones de dos clasificadores.
- La hipótesis nula es que no hay diferencia entre las proporciones de discordancia (i.e. los elementos fuera de la diagonal son iguales), por lo que no rechazar la hipótesis nula:
 - En el caso de (1) sugiere un buen ajuste del modelo.
 - En el caso de (2) indica que no hay diferencias entre clasificadores. El rechazo de la nula implica que uno de los clasificadores es mejor que el otro clasificador.

Ejemplos de aplicaciones

- **Problema:** en datos ruidosos, el ruido reduce la exactitud y precisión de los KPIs.
- **Objetivo:** metodología de dos pasos para calcular KPIs probabilísticos en conjuntos de datos ruidosos:
 - a. Optimización de enjambre para reducir el ruido
 - b. ML bayesiano para estimar KPI en datos con ruido reducido (relevance vector machines).

The screenshot shows a ScienceDirect article page. At the top right, it says "Journals & Books". The main title is "Systems and Soft Computing" (Volume 4, December 2022, 200036). To the right is a thumbnail of the journal cover. The article title is "How good is good? Bayesian machine-learning estimation of probabilistic benchmarks in noisy datasets and an application to nanofinance+". Below the title, the author is listed as "Rolando Gonzales Martinez¹✉". There are links for "Add to Mendeley", "Share", and "Cite". The URL is "https://doi.org/10.1016/j.sasc.2022.200036". Other options include "Get rights and content" and "open access". On the left side of the page, there is a sidebar with links for "Outline", "Abstract", "Keywords", "1. Introduction", "2. Multivariate probabilistic benchmarks", "3. Estimation of multivariate probabilistic be...", "4. Empirical application: probabilistic bench...", "5. Conclusion", "Declaration of Competing Interest", "References", and "Show full outline". At the bottom left, it says "Figures (4)".

Ejemplos de aplicaciones

Box 1. Pseudo-code of the swarm algorithm

Data: $\{y_1, y_2, \dots, y_j\} \ni y$

Result: ψ, ψ^\perp

initialization;

$\delta, M, \theta_0, p_0, p_0^\perp, \zeta, \zeta^* ;$

while $m \in \mathbb{Z}_+$ **do**

$$w_\delta = \delta \frac{|p|}{\|p\|}, \quad w_\delta^\perp = \delta \frac{|p^\perp|}{\|p^\perp\|};$$

for $m \leftarrow 1$ **do**

random exploration of hyperbola parameters;

$$p_m = p_{m-1} + w_\delta \epsilon, \quad p_m^\perp = p_{m-1}^\perp + w_\delta^\perp \epsilon, \quad \epsilon \sim (0, 1);$$

hyperbolic undersampling;

$$y_h = f_h(p_m, y), \quad y_h^\perp = f_h^\perp(p_m^\perp, y), \quad \{y_h, y_h^\perp\} \ni y_h;$$

copula dependence estimated with filtered m -samples;

$$\hat{\theta}_m = C_\theta(y_h);$$

end

$$\hat{\theta}^* = \max \{\hat{\theta}_l\}_{l=1}^m \text{ (optimal dependence);}$$

$$p^* = p(\hat{\theta}^*), \quad p^{*\perp} = p^\perp(\hat{\theta}^*) \text{ (optimal hyperbola parameters);}$$

$$\text{cohesion} = \frac{1}{2} (\|p_m - p_m^*\| + \|p_m^\perp - p_m^{\perp*}\|);$$

$$\text{separation} = \frac{1}{2} (\|p_m - \bar{p}_m^*\| + \|p_m^\perp - \bar{p}_m^{\perp*}\|);$$

if $\hat{\theta}^* > \hat{\theta}^{m-1}$ **then**

$$\hat{\theta}_m = \hat{\theta}^*;$$

$$p_m = p^*, \quad p_m^\perp = p^{\perp*};$$

alignment;

$$\delta_m = \delta_{m-1}(\zeta^*);$$

$$m = M - 1;$$

else

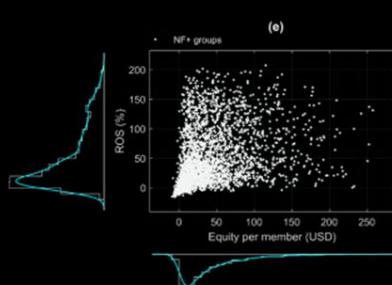
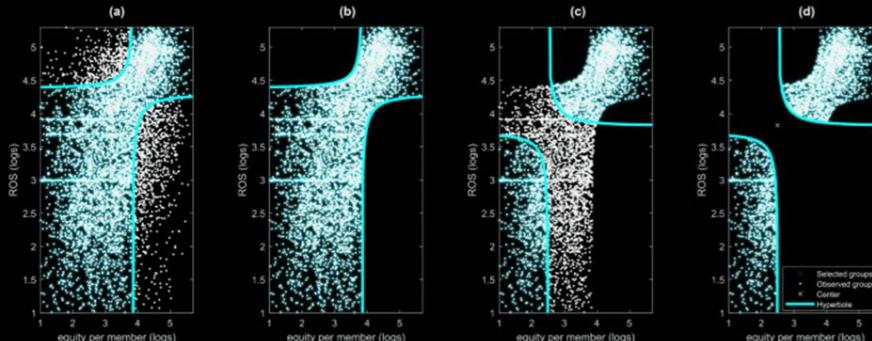
$$\delta_m = \delta_{m-1}(\zeta);$$

end

end

$$p(\mathbf{w}|\mathbf{t}, \boldsymbol{\alpha}, \sigma^2) = \frac{p(\mathbf{t}|\mathbf{w}, \sigma^2)p(\mathbf{w}|\boldsymbol{\alpha})}{p(\mathbf{t}|\boldsymbol{\alpha}, \sigma^2)},$$

$$= (2\pi)^{-(k+1)/2} \left| \boldsymbol{\Sigma} \right|^{-1/2} \exp \left\{ -\frac{1}{2} (\mathbf{w} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{w} - \boldsymbol{\mu}) \right\}$$



Ejemplos de aplicaciones

