



# Detección de incendios con Aprendizaje Automático aplicado a IoT

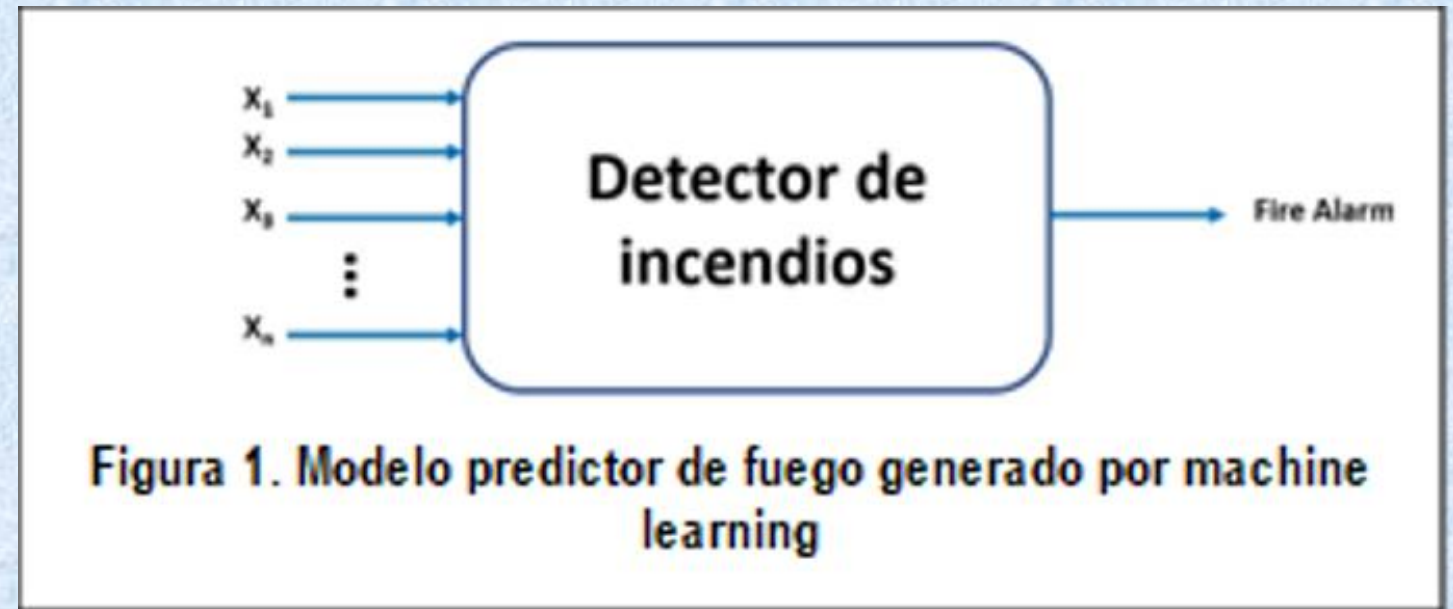
Carlos Binker, Hugo Tantignone, Guillermo Buranits,  
Eliseo Zurdo, Lautaro Lasorsa, Maximiliano Frattini.

UNLaM

laulasorsa@unlam.edu.ar

# Problemática a resolver

- Un detector de incendios recolecta datos del ambiente y debe decidir si hay o no un incendio.
- Teniendo un set de datos ya clasificado, proponemos resolver este problema utilizando distintos modelos de aprendizaje automático.





# Problemática a resolver

- Elegimos el modelo que mejor precisión muestra en los datos de validación cruzada.
- También comparamos la robustez de los modelos introduciendo artificialmente errores en los datos de entrada.

	Temperature[C]	Humidity[%]	TVOC[ppb]	eCO2[ppm]	Raw H2	Raw Ethanol	Pressure[hPa]	PM1.0	PM2.5	NC0.5	NC1.0	NC2.5	Fire Alarm
0	20.000	57.36	0	400	12306	18520	939.735	0.00	0.00	0.00	0.000	0.000	0
1	20.015	56.67	0	400	12345	18651	939.744	0.00	0.00	0.00	0.000	0.000	0
2	20.029	55.96	0	400	12374	18764	939.738	0.00	0.00	0.00	0.000	0.000	0
3	20.044	55.28	0	400	12390	18849	939.736	0.00	0.00	0.00	0.000	0.000	0
4	20.059	54.69	0	400	12403	18921	939.744	0.00	0.00	0.00	0.000	0.000	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
62625	18.438	15.79	625	400	13723	20569	936.670	0.63	0.65	4.32	0.673	0.015	0
62626	18.653	15.87	612	400	13731	20568	936.678	0.61	0.63	4.18	0.652	0.015	0
62627	18.867	15.84	627	400	13725	20582	936.687	0.57	0.60	3.95	0.617	0.014	0
62628	19.083	16.04	638	400	13712	20566	936.680	0.57	0.59	3.92	0.611	0.014	0
62629	19.299	16.52	643	400	13696	20543	936.676	0.57	0.59	3.90	0.607	0.014	0

62630 rows × 13 columns

**Figura 2. Vista del dataset a emplear en nuestro modelo predictor de fuego**

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	Y
0	20.000	57.36	0	400	12306	18520	939.735	0.00	0.00	0.00	0.000	0.000	0
1	20.015	56.67	0	400	12345	18651	939.744	0.00	0.00	0.00	0.000	0.000	0
2	20.029	55.96	0	400	12374	18764	939.738	0.00	0.00	0.00	0.000	0.000	0
3	20.044	55.28	0	400	12390	18849	939.736	0.00	0.00	0.00	0.000	0.000	0
4	20.059	54.69	0	400	12403	18921	939.744	0.00	0.00	0.00	0.000	0.000	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
62625	18.438	15.79	625	400	13723	20569	936.670	0.63	0.65	4.32	0.673	0.015	0
62626	18.653	15.87	612	400	13731	20588	936.678	0.61	0.63	4.18	0.652	0.015	0
62627	18.867	15.84	627	400	13725	20582	936.687	0.57	0.60	3.95	0.617	0.014	0
62628	19.083	16.04	638	400	13712	20566	936.680	0.57	0.59	3.92	0.611	0.014	0
62629	19.299	16.52	643	400	13696	20543	936.676	0.57	0.59	3.90	0.607	0.014	0

62630 rows × 13 columns

**Figura 4. Mapeo del dataset con nuevos nombres en columnas de las variables independientes**

# Formalización del problema

Se busca una función  $f(x)$  que dados los valores  $x$  de las variables predictoras estime el valor de  $Y$ , y que cumpla:

$$f(x) = \left\{ \begin{array}{l} 1 \text{ si } P(Y = 1|X = x) \geq P(Y = 0|X = x) \\ 0 \text{ si } P(Y = 0|X = x) \geq P(Y = 1|X = x) \end{array} \right\}$$

Todos los modelos salvo el Random Forest se basan en estimar  $P(Y = 1|X=x)$

# Regresión Logística

Este modelo se basa en estimar:

$$P(Y = 1|X = x) = \textit{sigmoide}(\theta' * x + \beta)$$

Donde las sigmoides son una familia de funciones que cumplen ser acotadas, diferenciables, tener derivada positiva en cada punto y tener un único punto de inflexión (raíz de la segunda derivada).

En este caso los parámetros entrenables son  $\theta$  y  $\beta$

Una función particular de esta familia que es muy usada es:

$$L(x) = \frac{1}{1 + e^{-x}}$$



# Redes Neuronales

Consiste en tener múltiples capas de *neuronas* que cada una se comporta en si misma como una regresión logística.

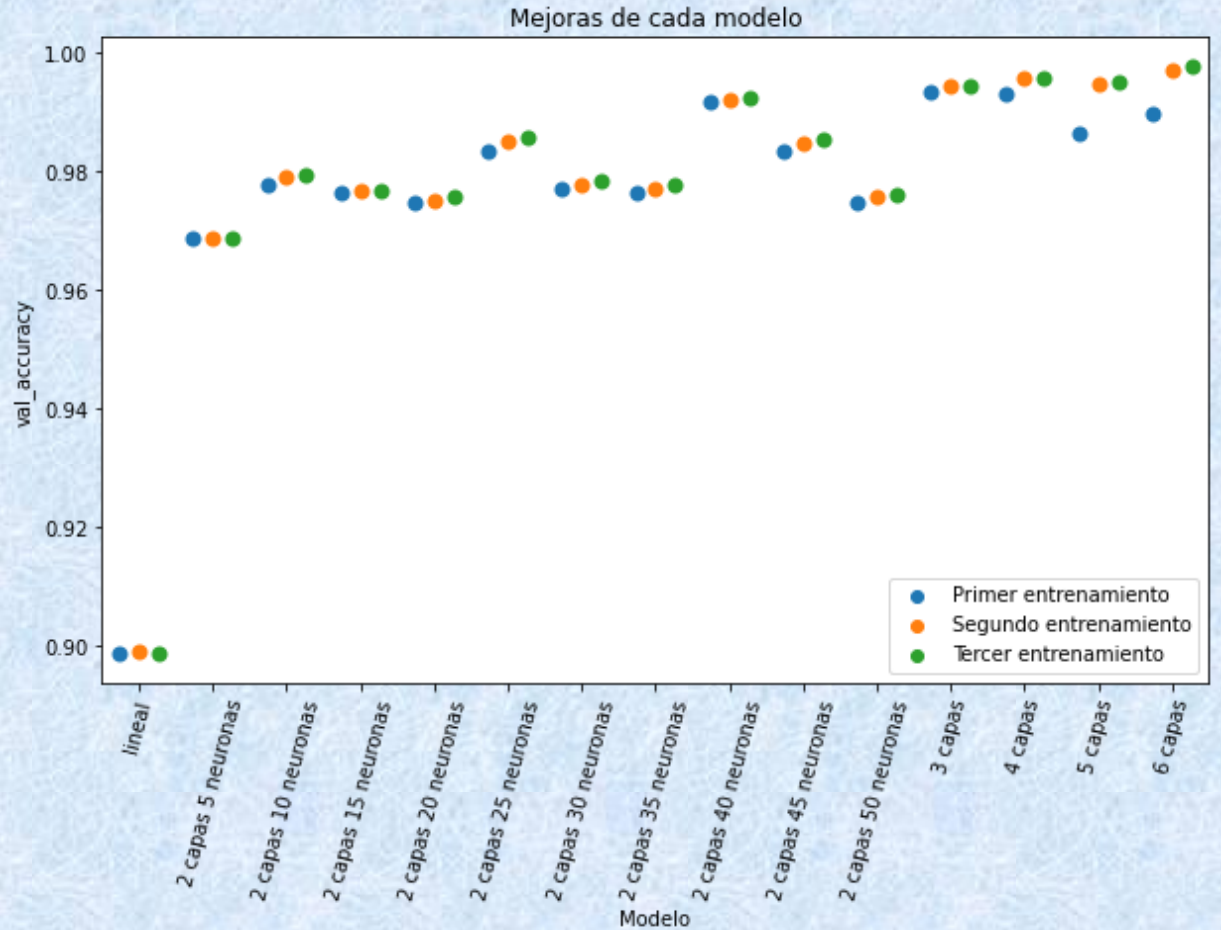
La primera capa son los datos de entrada, luego se tienen N capas intermedias de A neuronas cada una, donde la entrada de cada neurona son las salidas de la capa anterior, y finalmente una neurona de salida cuyo valor se toma como la estimación de  $P(Y=1|X=x)$ .

Como la *función de activación* de las neuronas es no lineal, permite capturar patrones no lineales en los datos.

En este caso A y N son hiper parametros a entrenar en validación cruzada

# Entrenamiento

Los modelos de Regresión Logística y Redes Neuronales se entrenan con el método Adam basado en descenso de gradiente. Para esto se dividen los datos en lotes, lo que permite converger más rápido pero aumenta ligeramente el error. Se probó entrenar a los modelos en varias etapas con tamaños de lote 32, 1024 y luego todo el dataset como un solo lote.



# Árboles de decisión y Bosques aleatorios

- Un árbol de decisión se construye de forma recursiva dividiendo los datos en cada paso según la variable que permite una división lo más equilibrada posible hasta llegar a las hojas donde todos los datos de entrenamiento son del mismo tipo
- Un bosque aleatorio crea muchos árboles de decisión con subconjuntos aleatorios de las filas y columnas del dataset y decide lo que deciden la mayoría de estos árboles aleatorios.
- No realiza una estimación de  $P(Y=1|X=x)$



# Árboles de decisión y Bosques aleatorios

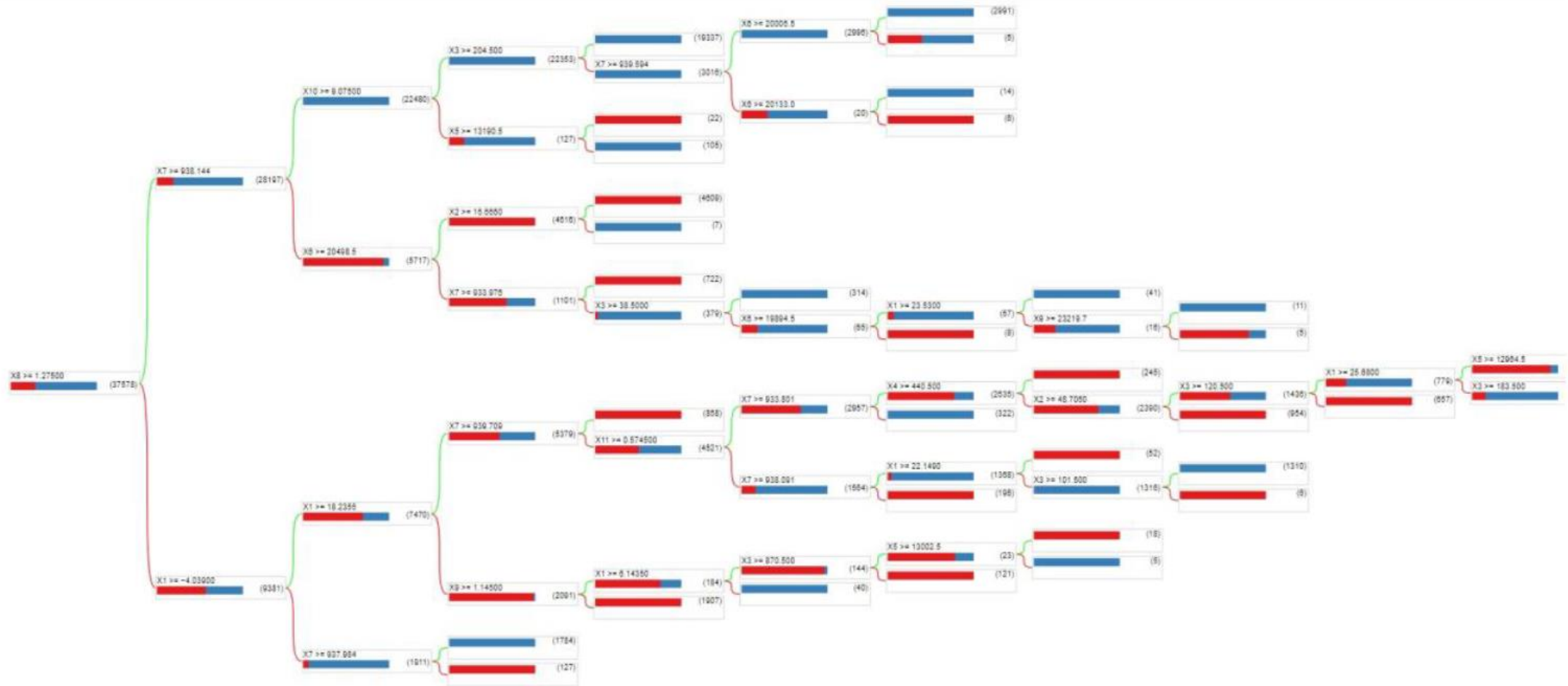


Figura 11. Modelo Random Forest con precisión del 99,992 %

# Mejor modelo en datos de validación cruzada

```
El mejor modelo es : Random Forest
38/38 [=====] - 1s 14ms/step - loss: 0.0000e+00 - accuracy: 0.9999
13/13 [=====] - 0s 14ms/step - loss: 0.0000e+00 - accuracy: 0.9999
13/13 [=====] - 0s 13ms/step - loss: 0.0000e+00 - accuracy: 0.9998
Tiene:
Train Accuracy:      [0.0, 0.9999201893806458]
Cross Validation Accuracy: [0.0, 0.9999201893806458]
Test Accuracy:      [0.0, 0.9997605085372925]
```

**Figura 12. Obtención de la precisión para los datos de entrenamiento, validación y prueba para Random Forest**

# Incorporación de errores

La segunda parte del trabajo se centro el evaluar la robustes de distintos modelos frente a la incorporación de errores en los modelos, esto se hizo de 2 formas:


- Deshabilitando un sensor, es decir eliminando una variable del dataset, y entrenando y evaluando modelos que no lo usen.
- Introduciendo errores aleatorios en una de las columnas de test y evaluando el modelo entrenado con los datos de entrenamiento correctos. Esto simula que un sensor funciona mal sin que esta situación sea detectada.



# Eliminación de un campo

Para esto se entrenaron únicamente modelos de Bosques Aleatorios por ser mucho más rápidos de entrenar y haber sido el modelo con mejor desempeño en la etapa anterior.

Como se ve, ningún sensor por si mismo reduce significativamente la performance del modelo al ser deshabilitado.



Sensor anulado	Precisión Comp.
-	0.999787
Temperature[C]	0.999787
Humidity[%]	0.999734
TVOC[ppb]	0.999042
eCO2[ppm]	0.999734
Raw H2	0.999734
Raw Ethanol	0.999734
Pressure[hPa]	0.999681
PM1.0	0.999734
PM2.5	0.999734
NC0.5	0.999734
NC1.0	0.999734
NC2.5	0.999734

Figura 14. Precisión obtenida para los datos de prueba considerando la anulación abrupta de cada sensor

# Introducción de errores

Para este proceso se entrenaron 4 modelos

1. Regresión logística simple
2. Red neuronal con 6 capas intermedias de 25 neuronas cada una
3. Red neuronal como la anterior pero agregando 3 capas de dropout, que en el entrenamiento desactivan aleatoriamente neuronas para evitar el sobre ajuste y queremos ver si también aportan robustez
4. Bosque aleatorio



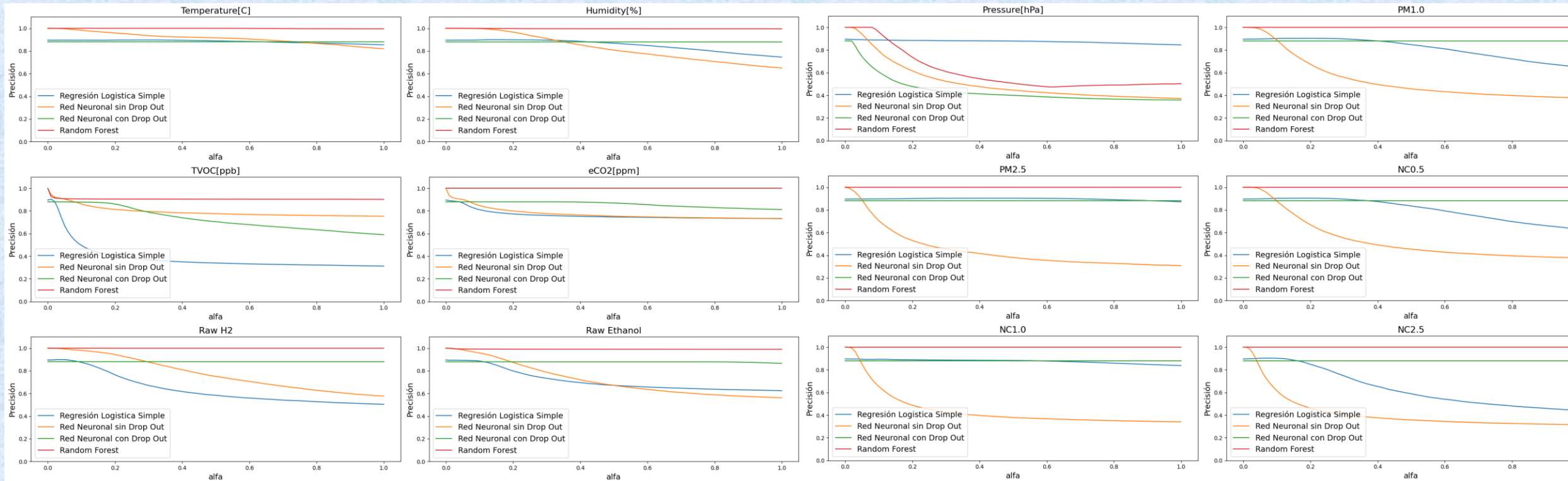
# Introducción de errores

Los errores se introducen de la siguiente manera:

- Se introducen de a una columna a la vez, cada una de forma independiente.
- Dada una columna  $X$ , siendo  $m$  el mínimo valor de  $X$  y  $M$  el máximo valor de  $X$ , se considera  $U$  un vector donde cada posición tiene distribución Uniforme  $[m, M]$
- Dado un valor  $0 \leq \alpha \leq 1$ , se considera  $O\alpha = (1 - \alpha) * X + \alpha * U$
- Se considera la performance del modelo para una grilla de valores de  $\alpha$  entre 0 y 1. Notar que el vector  $U$  es fijo para todas las evaluaciones de todos los modelos.



# Introducción de errores



Como se ve, para todas las variables excepto Presión, el Random Forest es mejor y más robusto que los demás modelos

# Conclusiones

Tanto las Redes Neuronales como los Bosques Aleatorios pueden alcanzar niveles de precisión cercanos al 100% en los datos de validación y de prueba, mientras que la Regresión Logística y las redes con dropout solo alcanzan el 90%.

Sin embargo, el Bosque Aleatorio no solo ha sido levemente mejor sino que además es mucho más rápido de entrenar y ligero de almacenar, y es también más robusto, viendo su performance mucho menos perjudicada por la introducción de errores en los datos de entrada.

11° CONGRESO  
**CoNaIISI** 2023



# Gracias por ver!

San Miguel de Tucumán, Tucumán, Argentina – 2 y 3 de Noviembre 2023