# Evaluación #1:

# Entropía Mono-escla
# +
# Regresión Logística

Prof. Nibaldo Rodríguez Agurto

# OBJETIVOS

- General:
  - Implementar y evaluar el rendimiento de un modelo de clasificación binaria usando entropía mono-escala y regresión logística.
- Específicos:
  - Crear característica usando entropía dispersión y permutación.
  - Entrenar un modelo de regresión logística usando mGD.
  - Evaluar (test) un modelo de regresión logística.
  - Calcular métricas de rendimiento:
    - Matriz de Confusión.
    - F-scores de cada clase binaria .

# Etapa #1:
# Pre-Procesamiento de la Data

# Descripción de la Data

- Archivo de Datos para la Clase #1: **class1.csv**
- Archivo de Datos para la Clase #2: **class2.csv**

- Cada archivo contiene *N*-muestras de datos.
- Cada muestra corresponde a una vector de ***L-valores***
  - ***Data:=matriz(N,L)***

# Prep-procesamiento: ppr.py

- El propósito de este programa fuente es crear nuevas características desde la data original usando la entropía de Dispersión y Permutación.

- Escribir una función para crear características de entropía desde los archivos *class1.csv* y *class2.csv*.
  - Nombre de función:
    - ***get_features()***.

# gets_features()

- Crea nuevos archivos de característica usando la entropía.
  - **Opción #1:Dispersión**
    - Parámetros:
      - *d*: dimensión embbeding
      - ***tau:*** time de retardo
      - ***c***: número de clases
  - **Opción#2: Permutación**
    - Parámetros:
      - *d*: dimensión embbeding
      - ***tau:*** time de retardo
  - **Segmentar los archivos clases original**:
    - Tamaño de segmento: W-muestras ( filas del archivo) .

# gets_features()

- Crear nuevos archivos **csv**:
  - **Características Clase#1: dfeatues1.csv.**
    - W-columnas:
      - Cada columna representa una característica de entropía.
    - K-filas
      - Cala fila representa una nueva muestra de características.
  - **Características Clase#2: dfeatues2.csv.**
    - W-columnas:
      - Cada columna representa una característica de entropía.
    - K-filas
      - Cala fila representa una nueva muestra de características.
- Crear un nuevo archivo de datos cvs:
  - Nombre del archivo: ***dfeatures.csv***.
    - Es creados concatenado los dos archivos previos de características.
    - ***dfeatures.csv: concatena(dfeature1,dfeatures2)***
  - Tamaño del archivo:
    - 2K-filas.
    - W-columnas.

# gets_features()

- Crear un nuevo archivo de datos csv:
  - Nombre del archivo: ***dfeatures.csv***.
    - Es creado concatenado los dos archivos previos de características.
    - ***dfeatures.csv = concatena(dfeature1,dfeatures2)***
    - Tamaño del archivo:
      - 2K-filas.
      - W-columnas.
- Crear archivo de clases (etiquetas) para cada clase:
  - **Clase #1: Y(1:K/2)=1**
  - **Clase #2: Y(K/2+1:N)=0**
  - **Nombre archivo: label.csv**

# ppr.py

```
def main():
    conf_entropy()
    load_data()
    F1 = gets_features()
    F2 = gets_features()
    F  = np.concatenate((F1, F2), axis=0)
    save_data(F)
```

# Etapa #2:
# Algoritmo de Entrenamiento
# (trn.py)

# trn.py

- Cargar archivo de ***dfeatures.csv y label.csv:***
- Re-ordenar aleatoriamente las posiciones de matriz de características y del vector de Clases.
- Crear una función de para crear datos de training y test:
  - Datos de Training:
    - **dtrn.csv/dtrn_label.csv**
    - Número de muestras: ***L= round(N*p/100).***
    - *P: porcentaje de training* ***(60,80).***
  - Datos de Testing:
    - **dtst.csv/dtst_label.csv**
    - Número de muestras: ***K= N-L.***

# Entrenamiento: trn.py
# trn_logistic()

- Crear la función para el aprendizaje de pesos: ***trn_logistic()***
  - Cargar datos de configuración desde archivo: **conf_train.csv**.
  - Usar algoritmo ***Descenso del Gradiente con Momentum.***
- Crear archivo de pesos y costo del modelo de regresión:
  - Pesos: **pesos.csv.**
  - Vector de Costo: **costo.csv.**
    - Número de Filas : Max. Iteraciones
    - Numero de Columnas: 1.

# trn.py

```
def main():
    conf_train()
    load_data()
    train()
    save_w_cost(W,Cost, 'pesos.csv','costo.csv')
```

# Etapa #3:
# Evaluación de Rendimiento

# test.py

- Cargar data de test.
- Cargar los coeficientes desde archivo: **pesos.csv.**
- Generar los valores estimados usando Regresión Logit.
- Crear archivos de resultados:
  - **Matriz Confusión:**
    - **cmatriz.csv.: Matriz de (2,2)**
  - **F-scores:**
    - **fscores.csv: vector de (1,2).**

# test.py

```
def main():
        load_data()
        load_w()
        zv     = forward(xv,W)
        cm,Fsc = metricas(yv,zv)
        save_measure(cm,Fsc,'cmatrix.csv','Fscores.csv')
```

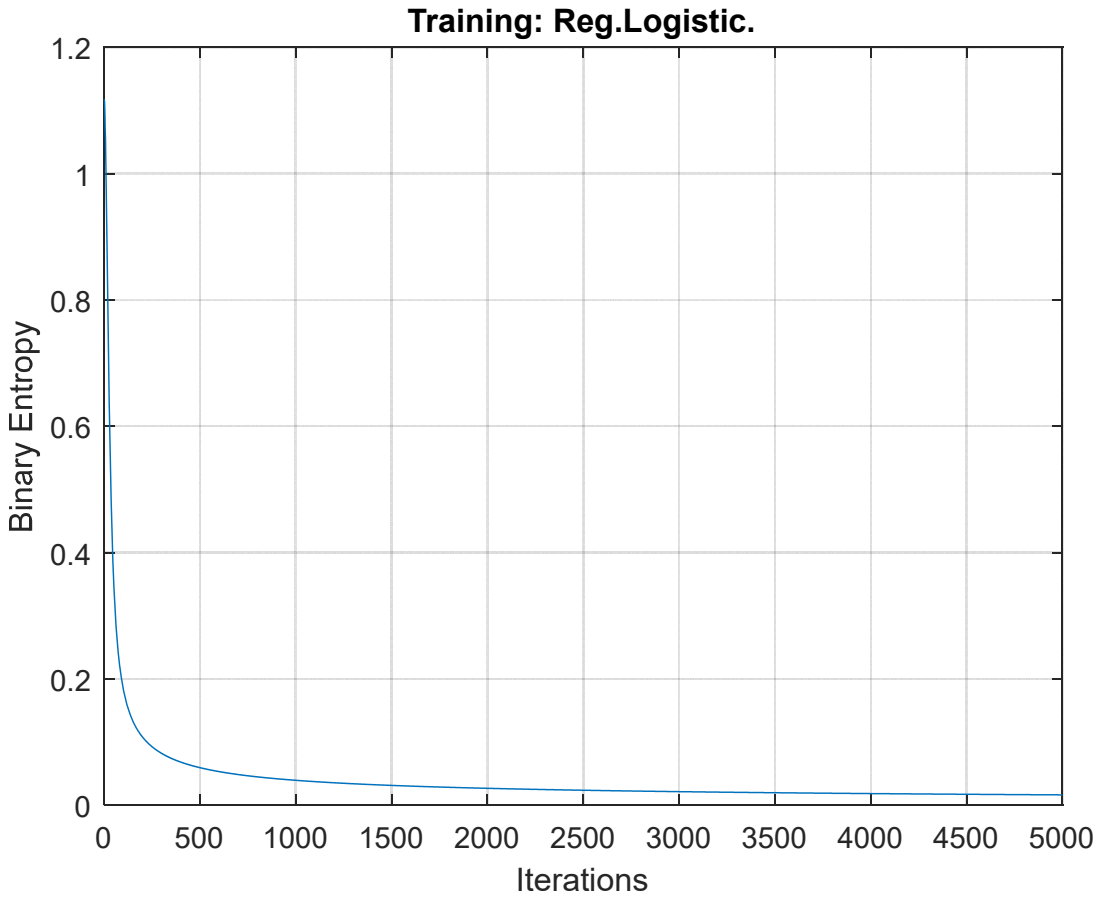
# conf_entropy.csv

- Línea 1: Tipo de entropía (***1-dispersión/2-permuta***).
- Línea 2: Dimensión embebida (***d***).
- Línea 3: Tiempo de retardo embebido (***tau***).
- Línea 4: Número de clase de Entropía Dispersión (***c***).
- Línea 5:Tamaño de Segmentación de los archivos clases

# conf_train.csv

- Línea 1: Máximo de Iteraciones.
- Línea 2: Tasa de aprendizaje (***mu***).
- Línea 3: Porcentaje de Training (***60<p<81***).

# Resultados del Modelo: Training



# Resultados del Modelo: Testing

## Matriz Confusión

| | | Target | |
|---|---|---|---|
| | | 1 | 0 |
| Out-put | 1 | 18 | 0 |
| | 0 | 0 | 22 |

**F-scores : 100% 100%**

# ENTREGA

- Viernes 13 de Septiembre 2025
  - Hora : 08:00 am.
  - Lugar : Aula Virtual del curso.

- **Lenguaje Programación:**
  - Python 3.12 window (anaconda)
    - numpy/panda/matplot

# OBSERVACIÓN:

- Si un estudiante no cumple los requerimientos funcionales y no-funcionales, entonces la escala de evaluación será entre 1.0 y 3.0.