UCP
UNIVERSIDAD DE LA CUENCA DEL PLATA

INGENIERIA EN SISTEMAS DE INFORMACION

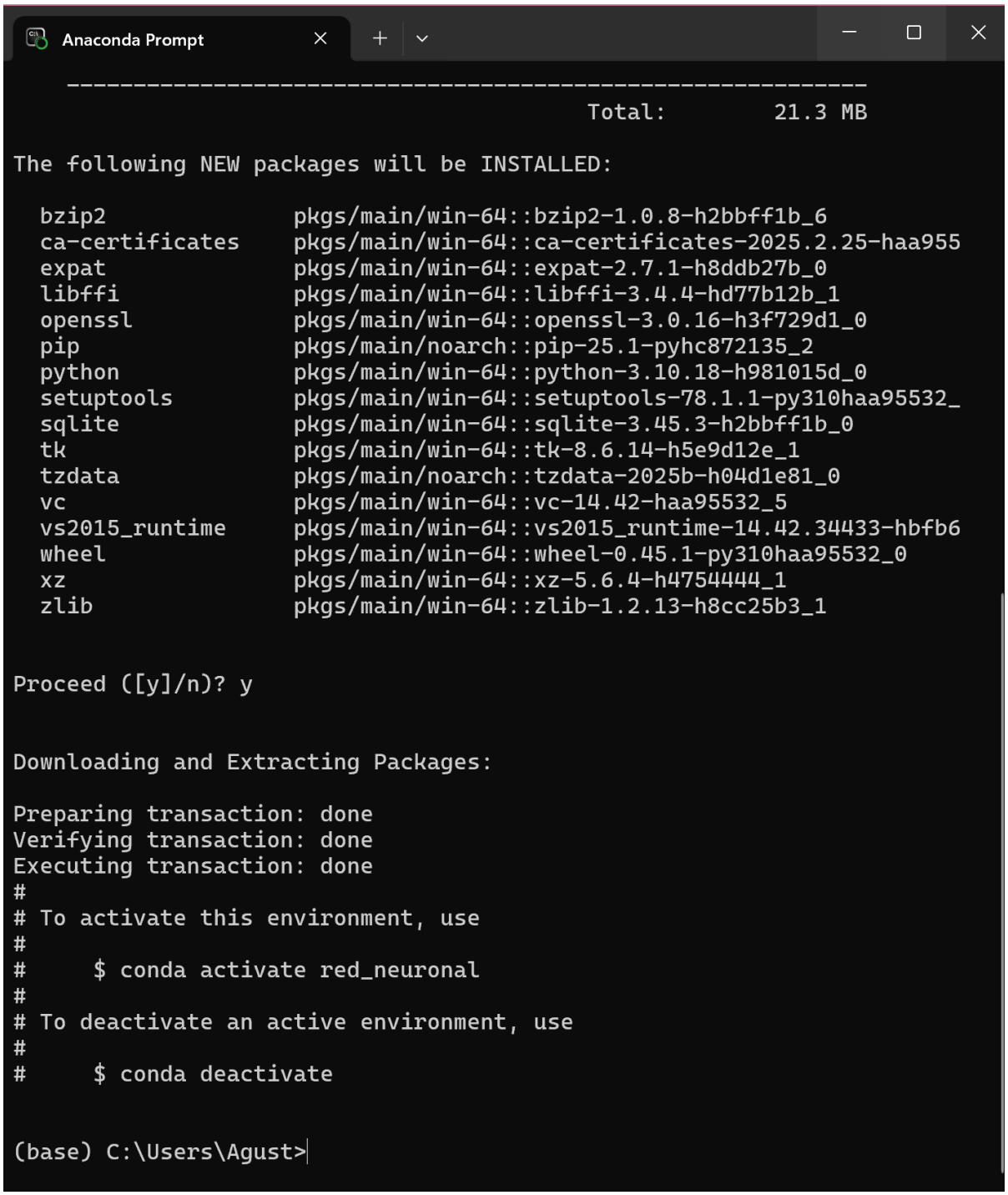
Trabajo Práctico Integrador
RESULTADOS
Sistemas Inteligentes

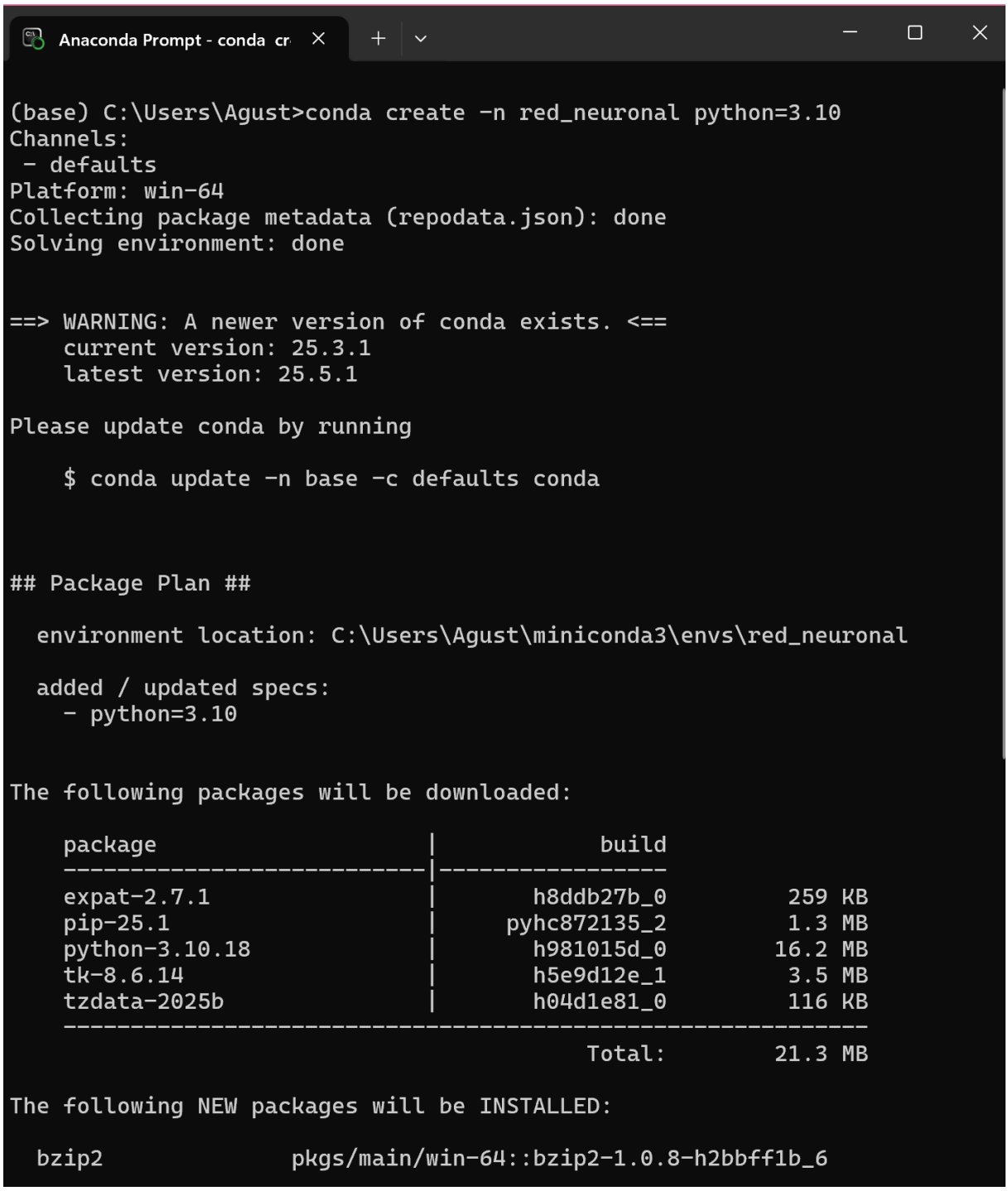
Autor: Cáceres Erika Agustina

Profesor: Lic. Del Rosario Gabriel Darío

Año: 2025

Primero crear el entorno e instalar las librerías correspondientes




Anaconda Prompt - conda cr
(base) C:\Users\Agust>conda create -n red_neuronal python=3.10
Channels:
 - defaults
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done
==> WARNING: A newer version of conda exists. <==
current version: 25.3.1
latest version: 25.5.1
Please update conda by running
$ conda update -n base -c defaults conda
Package Plan
environment location: C:\Users\Agust\miniconda3\envs\red_neuronal
added / updated specs:
- python=3.10
The following packages will be downloaded:
package | build
expat-2.7.1 | h8ddb27b_0 259 KB
pip-25.1 | pyhc872135_2 1.3 MB
python-3.10.18 | h981015d_0 16.2 MB
tk-8.6.14 | h5e9d12e_1 3.5 MB
tzdata-2025b | h04d1e81_0 116 KB
Total: 21.3 MB
The following NEW packages will be INSTALLED:
bzip2 pkgs/main/win-64::bzip2-1.0.8-h2bbff1b_6

```
#     $ conda deactivate


(base) C:\Users\Agust>conda activate red_neuronal

(red_neuronal) C:\Users\Agust>pip install tensorFlow pandas scikit-learn
 matplotlib seaborn
Collecting tensorFlow
  Downloading tensorflow-2.19.0-cp310-cp310-win_amd64.whl.metadata (4.1
kB)
Collecting pandas
  Downloading pandas-2.3.0-cp310-cp310-win_amd64.whl.metadata (19 kB)
Collecting scikit-learn
  Downloading scikit_learn-1.7.0-cp310-cp310-win_amd64.whl.metadata (14
kB)
Collecting matplotlib
  Downloading matplotlib-3.10.3-cp310-cp310-win_amd64.whl.metadata (11 k
B)
Collecting seaborn
  Using cached seaborn-0.13.2-py3-none-any.whl.metadata (5.4 kB)
Collecting absl-py>=1.0.0 (from tensorFlow)
  Downloading absl_py-2.3.0-py3-none-any.whl.metadata (2.4 kB)
Collecting astunparse>=1.6.0 (from tensorFlow)
  Downloading astunparse-1.6.3-py2.py3-none-any.whl.metadata (4.4 kB)
Collecting flatbuffers>=24.3.25 (from tensorFlow)
  Downloading flatbuffers-25.2.10-py2.py3-none-any.whl.metadata (875 byt
es)
Collecting gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 (from tensorFlow)
  Downloading gast-0.6.0-py3-none-any.whl.metadata (1.3 kB)
Collecting google-pasta>=0.1.1 (from tensorFlow)
  Downloading google_pasta-0.2.0-py3-none-any.whl.metadata (814 bytes)
Collecting libclang>=13.0.0 (from tensorFlow)
  Downloading libclang-18.1.1-py2.py3-none-win_amd64.whl.metadata (5.3 k
```

Preparación del entorno con Miniconda

Para desarrollar e implementar el modelo de red neuronal se utilizó **Miniconda**, una distribución liviana de Conda. Se creó un entorno virtual específico para mantener organizadas y sin conflicto las dependencias del proyecto.

Pasos realizados:

1. **Activar el entorno de trabajo:**

conda activate red_neuronal

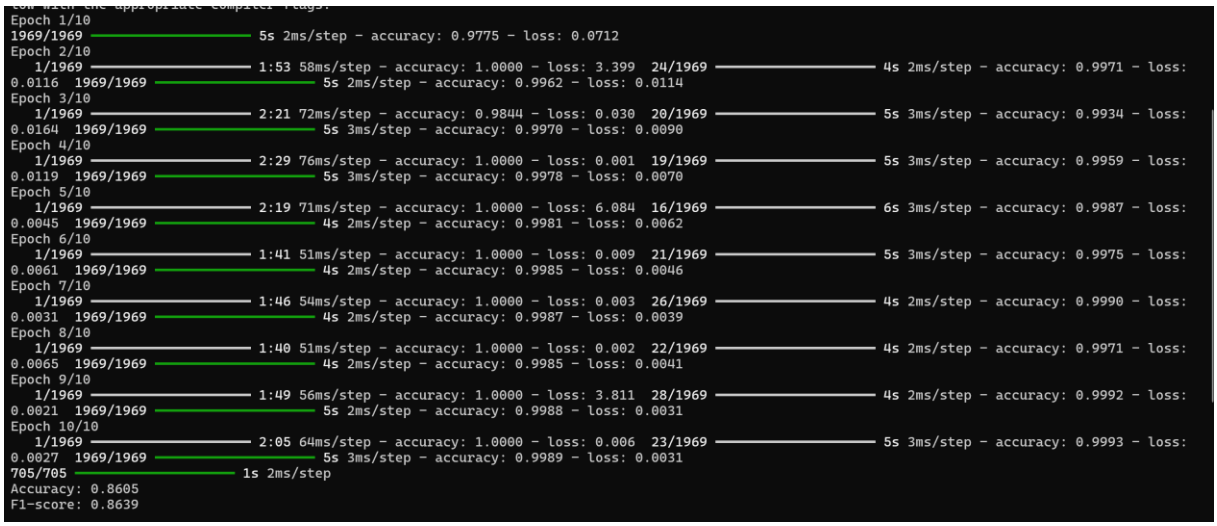
2. **Ubicarse en la carpeta del proyecto:**

cd "C:\Users\Agust\Documents\ING SISTEMAS DE INFO\IG SIST 5° AÑO\SISTEMAS INTELIGENTES\Integrador"

3. **Ejecutar el script Python:**

python si_intrusos.py

Este script ejecuta el preprocesamiento de datos, entrena el modelo y evalúa su rendimiento sobre un conjunto de prueba.



Modelo implementado

Modelo – MLP Simple (Perceptrón Multicapa)

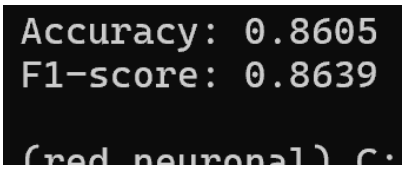
- **Arquitectura:**
  - Capa de entrada: coincide con la cantidad de atributos del dataset
  - Capa oculta 1: 64 neuronas – Activación: ReLU
  - Capa oculta 2: 32 neuronas – Activación: ReLU
  - Capa de salida: 1 neurona – Activación: Sigmoide (para clasificación binaria)
- **Compilación:**
  - Optimizador: Adam
  - Función de pérdida: Binary Crossentropy
  - Métrica: Accuracy

Este modelo fue entrenado con un conjunto de datos ya preprocesado que representa diferentes tipos de conexiones de red, clasificadas como normales o ataques.

Resultados obtenidos

Durante la ejecución, se entrenó el modelo por 10 épocas. El proceso se ejecutó correctamente y se observaron mejoras en cada iteración.

Resultados finales mostrados:



- **Precisión (Accuracy):** 86.05%
- **F1-score:** 86.39%

Estos valores reflejan un buen desempeño en la clasificación de eventos maliciosos y normales en la red.

Análisis e interpretación de resultados

- El modelo logró un **alto rendimiento**, superando el 86% en ambas métricas.
- El **F1-score elevado** indica que el modelo no solo es preciso, sino que también logra identificar correctamente los ataques sin dejar pasar muchos.
- A pesar de ser un modelo simple, **demostró buena capacidad de generalización**, por lo que constituye una base sólida para continuar iterando con mejoras.

Modelo – MLP con Dropout (Perceptrón Multicapa Regularizado)

- **Arquitectura:**
  - Capa de entrada: coincide con la cantidad de atributos del dataset
  - Capa oculta 1: 64 neuronas – Activación: ReLU
  - Dropout: tasa de desactivación del 20% (0.2)
  - Capa oculta 2: 32 neuronas – Activación: ReLU
  - Dropout: tasa de desactivación del 20% (0.2)
  - Capa de salida: 1 neurona – Activación: Sigmoide (para clasificación binaria)
- **Compilación:**
  - Optimizador: Adam
  - Función de pérdida: Binary Crossentropy
  - Métrica: Accuracy

Este modelo fue entrenado con un conjunto de datos ya preprocesado, representando conexiones de red clasificadas como normales o ataques, incorporando capas Dropout para mejorar la generalización y reducir el sobreajuste.

Resultados obtenidos

Durante la ejecución, se entrenó el modelo por 10 épocas. El proceso se ejecutó correctamente, y la precisión fue aumentando progresivamente.

Resultados finales mostrados:

- Precisión (Accuracy): 75.18%
- F1-score: 73.8%

Estos valores reflejan un desempeño aceptable, aunque inferior al modelo sin regularización. No se evidenció una mejora clara en generalización, aunque sí una base para futuros ajustes.

Análisis e interpretación de resultados

- El modelo alcanzó un rendimiento del 75.18% en accuracy y 73.8% en F1-score, algo menor al modelo sin regularización.
- Aunque el uso de Dropout busca evitar el sobreajuste, en este caso el modelo simple obtuvo mejores resultados en el conjunto de prueba, por lo que no se evidencia una mejora clara en generalización.
- Esto puede deberse a múltiples factores como: la necesidad de ajustar la tasa de Dropout, aumentar la cantidad de épocas, o afinar otros hiperparámetros.
- Aun así, el modelo con Dropout sirve como base para seguir experimentando con técnicas de regularización que busquen un mejor equilibrio entre ajuste y generalización.

```
(red_neuronal) C:\Users\Agust\Documents\ING SISTEMAS DE INFO\IG SIST 5º AÑO\SISTEMAS INTELIGENTES\Integrador>python si_intrusos_dropout.py
2025-06-13 11:51:30.568699: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results du
e to floating-point round-off errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2025-06-13 11:51:32.634209: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results du
e to floating-point round-off errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2025-06-13 11:51:38.175106: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructio
ns in performance-critical operations.
To enable the following instructions: SSE3 SSE4.1 SSE4.2 AVX AVX2 AVX512F AVX512_VNNI AVX512_BF16 AVX_VNNI FMA, in other operations, rebuild TensorF
low with the appropriate compiler flags.
Epoch 1/10
1969/1969 ━━━━━━━━━━━━━━━━━━━━ 8s 3ms/step - accuracy: 0.9455 - loss: 0.1552
Epoch 2/10
1969/1969 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9800 - loss: 0.0603
Epoch 3/10
1969/1969 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9821 - loss: 0.0506
Epoch 4/10
1969/1969 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9843 - loss: 0.0417
Epoch 5/10
1969/1969 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9850 - loss: 0.0403
Epoch 6/10
1969/1969 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9860 - loss: 0.0373
Epoch 7/10
1969/1969 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9864 - loss: 0.0365
Epoch 8/10
1969/1969 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9859 - loss: 0.0373
Epoch 9/10
1969/1969 ━━━━━━━━━━━━━━━━━━━━ 9s 3ms/step - accuracy: 0.9867 - loss: 0.0336
Epoch 10/10
1969/1969 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.9873 - loss: 0.0339
705/705 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step
Accuracy con Dropout: 0.7518
F1-score con Dropout: 0.738
```