UNIVERSIDAD NACIONAL DE SAN MARTÍN

Red neuronal

"Personalidad: Extrovertido o Introvertido"

Alumna: Julieta Sofía Gómez

Profesores: Josefina Bompensieri y Tomás Prudente

Matemática III

1° cuatrimestre 2025

Parte 1 – Análisis de la base de datos

Para este trabajo final, utilicé la base de datos "Personality" de Rakesh Kapilavayi para tener como objetivo el clasificar entre extrovertido o introvertido dependiendo de distintos factores. La base de datos posee 2900 entradas, sin embargo, en algunas columnas no presenta datos así que es útil para practicar qué realizar en estos casos.

Descripción columnas:

- "Time spent alone": Variable cuantitativa discreta, indica cuántas horas se estuvo solo.
- "Stage fear": Variable categórica, si se tiene pánico escénico.
- "Social event attendance": Variable cuantitativa discreta, indica cuán frecuente va a eventos sociables.
- "Going outside": Variable cuantitativa discreta, indica cuán frecuente sale de su casa.
- "Drained after socializing": Variable categórica, si se siente exhausto luego de socializar.
- "Friends circle size": Variable cuantitativa discreta, indica la cantidad de amigos cercanos.
- "Post frequency": Variable cuantitativa discreta, indica cuán frecuente sube algo a las redes sociales.
- "Personality": Variable categórica, si es Introvertido o Extrovertido.

Ya que esta base de datos posee algunas columnas con datos NaN en ciertas entradas, decidí probar dos opciones: eliminar estas filas o rellenarlas con la mediana de la columna.

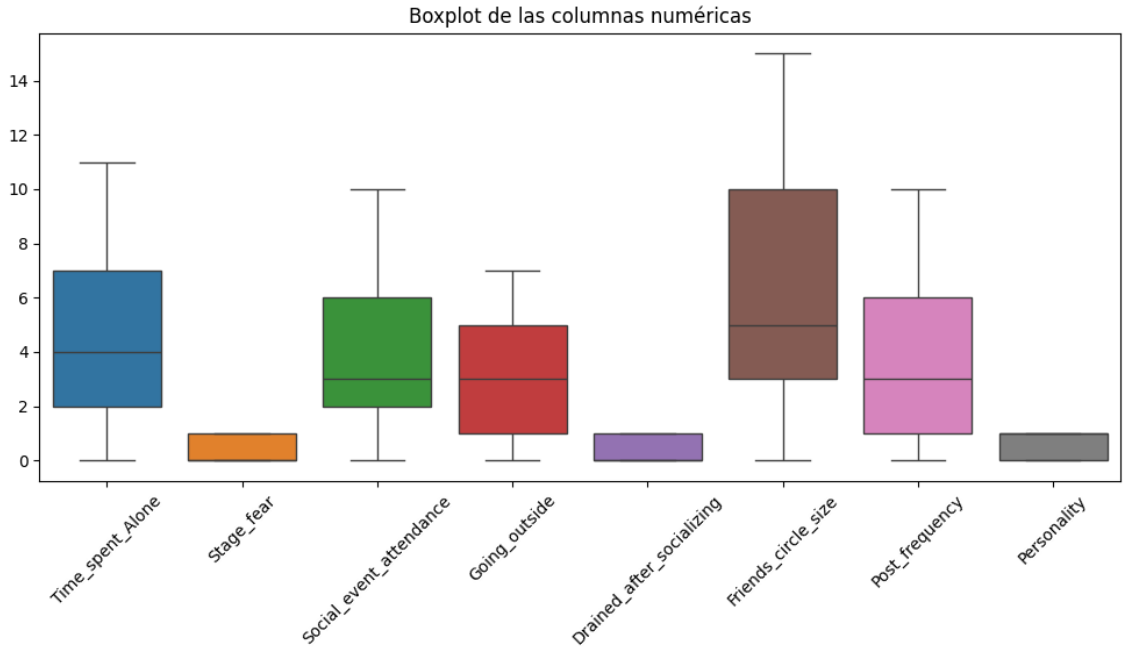
Para la primera opción, se perderían 423 datos lo cual representan un 14,59%. Ya que no es una base de datos muy grande es necesario mantener la mayor cantidad de entradas posibles para que nuestra red neuronal tenga más material para utilizar. Al optar por la segunda opción de rellenar la información, las columnas numéricas se llenaron con la mediana y para las columnas categóricas, la moda. Así obteniendo 2900 entradas en todas las columnas de la base de datos sin datos 'null'

Sin embargo, todavía poseemos información importante para nuestra red en tipo distinto al numérico. Para solucionar esto, utilicé la función '.map()' y así cambiar "Yes/Extrovert" por 1 y "No/Introvert" por 0. Esto ayudó a tener más material necesario para llegar a nuestro valor objetivo.

Ahora que poseemos los 2900 datos numéricos en todas las columnas, comenzamos a analizar si hay alguno que sea atípico.

	count	mean	std	min	25%	50%	75%	max
Time_spent_Alone	2900.0	4.494828	3.441971	0.0	2.0	4.0	7.0	11.0
Stage_fear	2900.0	0.486207	0.499896	0.0	0.0	0.0	1.0	1.0
Social_event_attendance	2900.0	3.942759	2.875987	0.0	2.0	3.0	6.0	10.0
Going_outside	2900.0	3.000000	2.221597	0.0	1.0	3.0	5.0	7.0
Drained_after_socializing	2900.0	0.485172	0.499866	0.0	0.0	0.0	1.0	1.0
Friends_circle_size	2900.0	6.235172	4.237255	0.0	3.0	5.0	10.0	15.0
Post_frequency	2900.0	3.552069	2.894794	0.0	1.0	3.0	6.0	10.0
Personality	2900.0	0.514138	0.499886	0.0	0.0	1.0	1.0	1.0

Aquí se puede observar que no hay gran varianza y solo “Friends circle size” es la única columna que posee una desviación estándar significativa en comparación al resto.



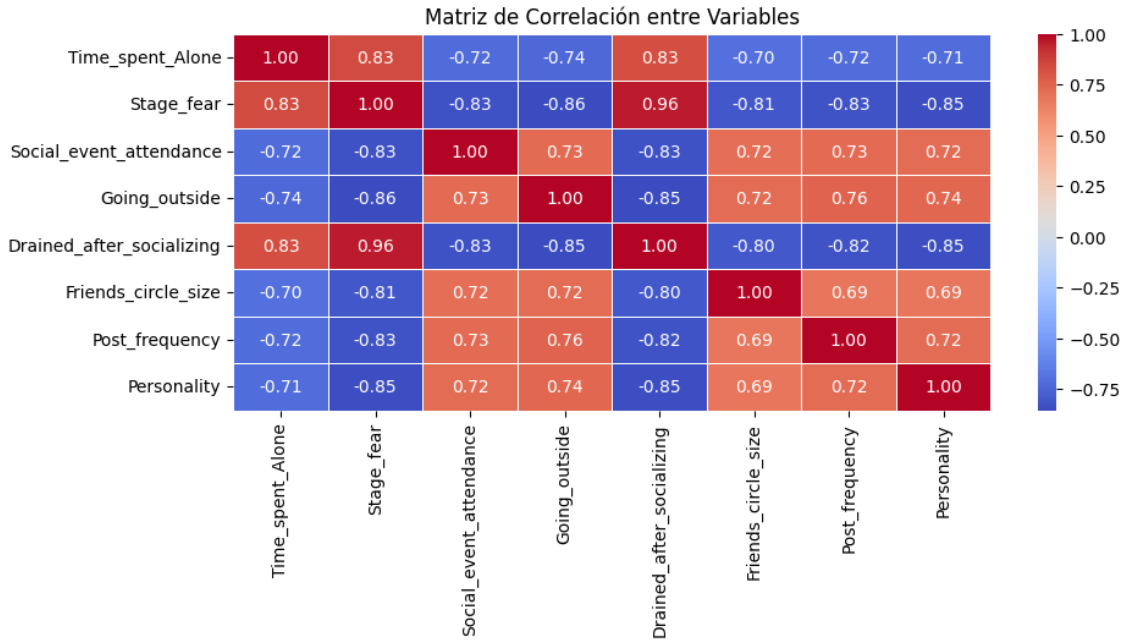
Luego de realizar el análisis de los rangos intercuartílicos no se hallaron datos atípicos, entonces no se tuvo que hacer una limpieza de la tabla.

En la normalización de los datos decidí utilizar la cuenta en la que Standard Scaler se basa y así traer todos los valores cercanos a 0. Esto ayuda a la red neuronal ya que es más fácil discriminar los valores entre -1 y 1 que entre céntimas.

```python
#Normalizamos los datos
columnas_a_normalizar = nuevo_df.columns[:-1] #todas las columnas menos la objetivo
final_df = nuevo_df.copy()

for col in columnas_a_normalizar:
    u = final_df[col].mean() #promedio
    s = final_df[col].std() #desviación estandar
    final_df[col] = (final_df[col] - u) / s
```

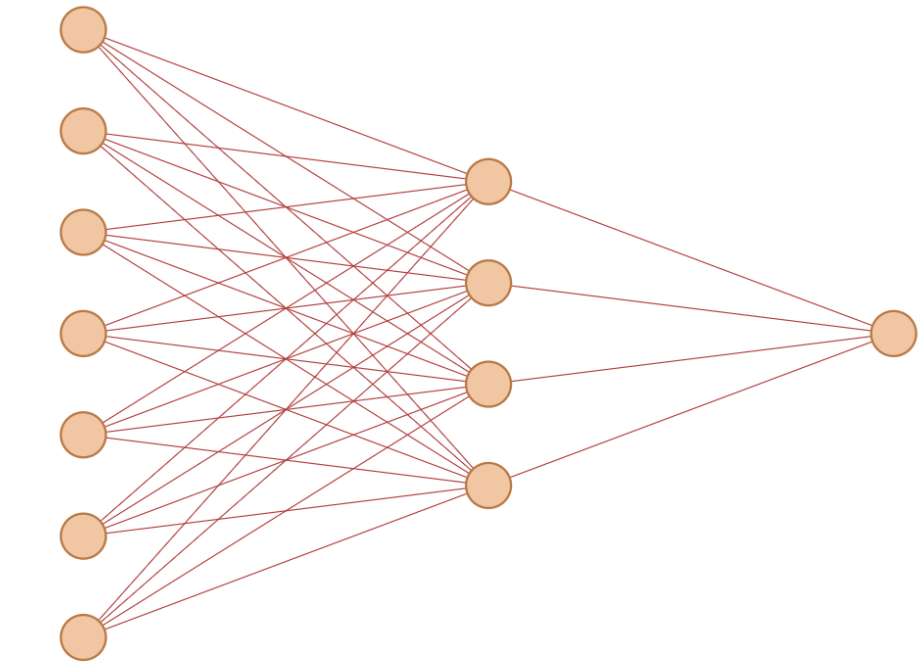
Con los datos normalizados podemos analizar la correlación que tienen entre sí las columnas y poder ver cuales influyen más sobre nuestro objetivo.



En este gráfico se puede observar como “Social event attendance”, “Going outside”, “Friends circle size” y “Post frequency” tienen números altos positivos en correlación con “Personality”. Esto tiene sentido ya que antes determinamos que alguien extrovertido posee el valor de 1 y si estas categorías aumentan, significa que el valor de “Personality” tiende a aproximarse a 1. Así también, si “Time spent alone” aumenta, disminuye la tendencia llevándolo a 0 significando en alguien introvertido por su correlación negativa.

Parte 2 – Desarrollo de la red neuronal

La red neuronal está compuesta de 7 entradas, 4 nodos en la capa oculta y 1 nodo en la capa de salida. En la capa oculta se utilizó la función de activación ReLU así los valores negativos pasan a ser 0 y se mantienen los números positivos. Para la capa de salida, se utilizó la función de activación Logística ya que convierte un número entre 0 y 1 facilitando la salida binaria de la red ya que se dividieron los resultados en valores mayores o menores a 0,5.



Para comenzar, se asignó como valores de entrada todas las filas de las columnas menos la última ya que esta es nuestra salida a comparar. Luego se dividieron 2/3 de estos datos para que se utilicen como entrenamiento y el restante 1/3 como prueba.

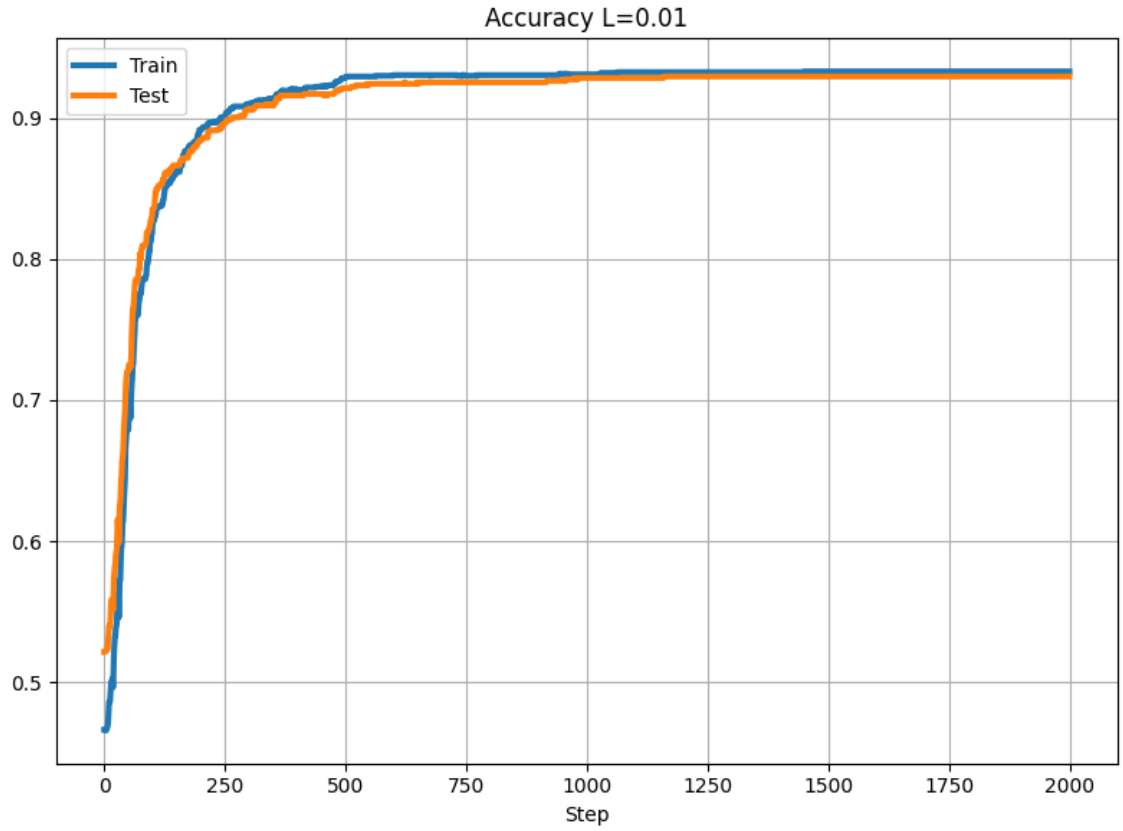
Para la inicialización de nuestra red se le otorgaron valores aleatorios de pesos y sesgos. El "* 2 – 1" permite que estos valores se encuentren entre -1 y 1.

```
w_hidden = np.random.rand(4,7) * 2 - 1
b_hidden = np.random.rand(4,1) * 2 - 1

w_output = np.random.rand(1, 4) * 2 - 1
b_output = np.random.rand(1, 1) * 2 - 1
```

Más adelante al definir nuestro *forward propagation* y *backward propagation* se analizó entre diferentes tasas de aprendizaje(L) y distintas cantidades de iteraciones cuál es la más conveniente para la red. En este caso resultó ser L = 0,01

con 2000 iteraciones, sin embargo, la red aprende lo suficientemente rápido y se iguala al entrenamiento. Podemos observar que en ambos casos la red supera un 90% de precisión en la predicción de datos sin hacer overfitting.



Para tener una mejor apreciación del trabajo de la red, se eligieron 10 entradas aleatorias que pasaron por *forward propagation* y así ver la elección de la red dependiendo de los datos recibidos.

```
print("Extrovertido = 1 / Introvertido = 0\n")
for i, row in filas_aleatorias2.iterrows():
    X_sample = row.values.reshape(1,-1)  # Datos de entrada en la Red Neuronal

    Z1, A1, Z2, A2 = forward_prop(X_sample.transpose())

    prediccion = A2

    resultado = ""

    if prediccion >= .5:
        resultado = "Extrovertido"
    else:
        resultado = "Introvertido"
```

Aquí podemos observar algunos de los resultados:

- Fila 2034
    - Activaciones de la capa de salida: [[0.81073767]]

    - Predicción: Extrovertido
    - Personality en el Dataframe es: 0
- Fila 63
    - Activaciones de la capa de salida: [[0.81073767]]
    - Predicción: Extrovertido
    - Personality en el Dataframe es: 1
- Fila 208
    - Activaciones de la capa de salida: [[0.0116535]]
    - Predicción: Introvertido
    - Personality en el Dataframe es: 0
- Fila 1523
    - Activaciones de la capa de salida: [[0.06307611]]
    - Predicción: Introvertido
    - Personality en el Dataframe es: 0

Parte 3 – Comparación con scikit- learn

Se implementó otra red neuronal con la librería Scikit-Learn similar a la red hecha manualmente. En esta también se dividieron los datos, 2/3 para el entrenamiento y 1/3 para la prueba.

```
nn = MLPClassifier(solver='sgd',
    hidden_layer_sizes=(4, ),
    activation='relu',
    max_iter=2000,
    learning_rate_init=.01)

nn.fit(X_train, Y_train)
```

Para que se puedan comparar se le determinó que sea una red que utiliza el mismo método que el nuestro que es el Descenso de Gradiente Estocástico y con la misma tasa de aprendizaje al igual que la misma cantidad de iteraciones.

La red implementada con Scikit-Learn tuvo los siguientes resultados:

```
Puntaje del conjunto de entrenamiento: 0.932230
Puntaje del conjunto de prueba: 0.937952
```

Mientras que la red hecha manualmente tuvo los siguientes resultados:

```
Precision del entrenamiento: 0.933264355923435
Precision de la prueba: 0.9296794208893485
```

No se haya mucha diferencia significativa entre sí, sin embargo, utilizar la librería trae más comodidad en la implementación y es más rápida al manipular los datos.

Parte 4 – Conclusión

Al realizar este trabajo aprendí bastante sobre el uso de librerías como Numpy, Matplotlib y Scikit-Learn ya que no las había utilizado antes. El crear de 0 la red me ayudó a comprender qué significa cada paso que realiza y el porqué hasta llegar a nuestro objetivo. También ayuda a comprender el boom de las Inteligencias Artificiales recientemente ya que en cuestión de segundos ya se obtiene una respuesta que puede llevar horas por otro medio.

Fue muy interesante implementar un nuevo método de programación con el que no estaba familiarizada y tener, aunque sea leve, una experiencia con algo que capaz tenga que utilizar en el futuro.