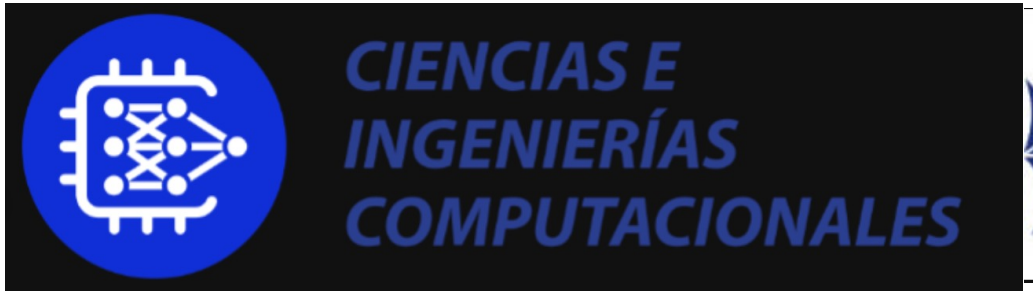
Predicción de la producción de biogas a partir de señales electroquímicas mediante inteligencia artificial

Proyecto Final

Antonio De León Rodríguez

Diplomado de Inteligencia Artificial Aplicada
2022-2023



Motivación

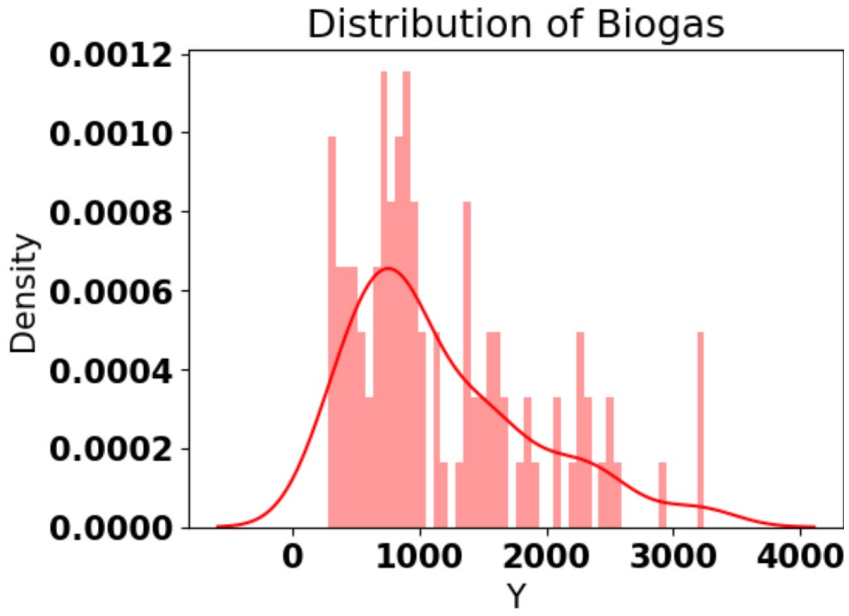
- La medición de la composición del biogas (*i.e.* hidrógeno o metano) en línea es complicado por cuestiones técnicas, por lo que la estimación mediante inteligencia artificial a partir de señales electroquímicas, sería de utilidad para la automatización, control y optimización de bioprocesos.

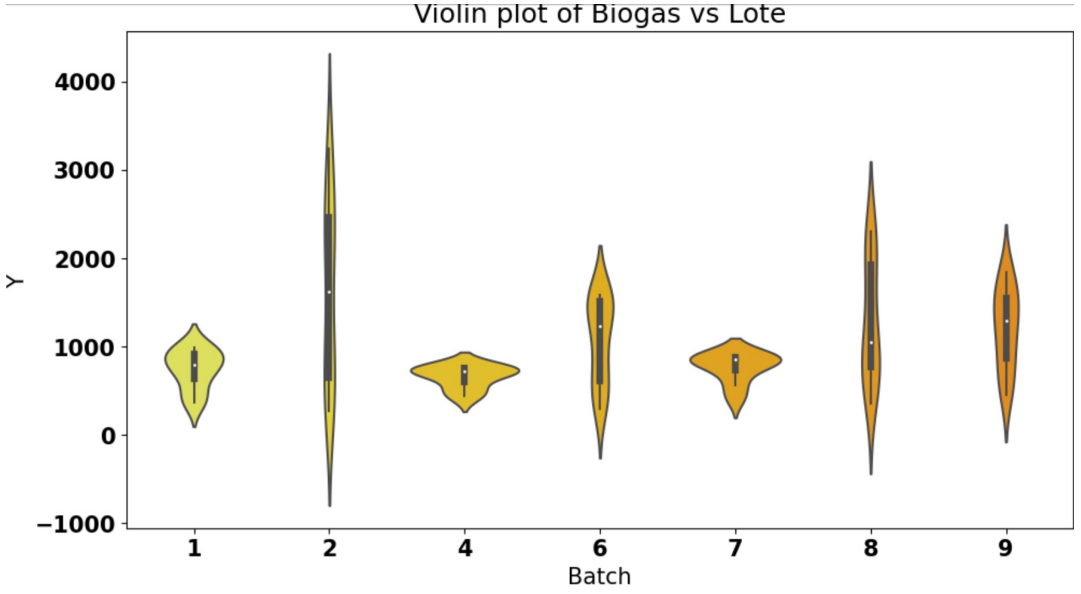
Análisis de datos

```
[6]: # explaration data analysis

     df.describe()
```

[6]:

	Batch	Time	X1	X2	X3	Y
count	102.000000	102.000000	102.000000	102.000000	102.000000	102.000000
mean	5.058824	85.191176	5.513902	86.570294	-503.121245	1173.996786
std	2.947808	72.800556	0.731490	40.618339	72.635445	729.131554
min	1.000000	9.000000	4.263000	25.900000	-621.024242	271.505254
25%	2.000000	28.000000	4.770000	44.450000	-563.301733	674.768239
50%	6.000000	67.000000	5.900000	89.650000	-511.383436	910.492938
75%	8.000000	116.750000	6.015000	116.625000	-453.397059	1572.722444
max	9.000000	342.000000	6.712000	165.300000	-321.614706	3245.397725



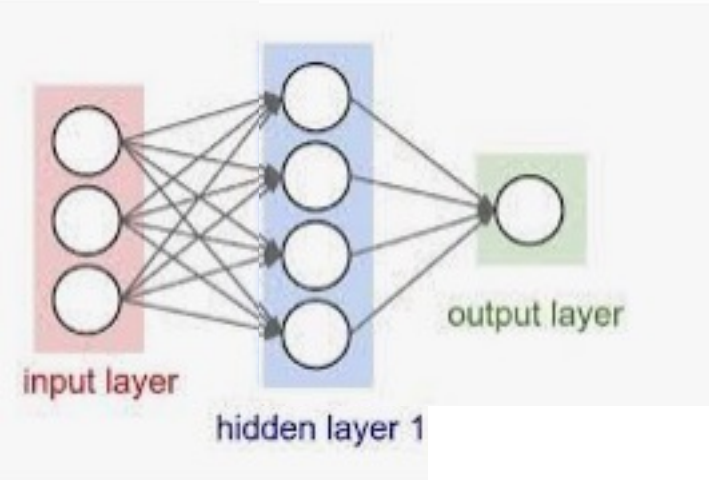


Los datos no presentan una distribución Gaussiana
Algunos lotes (2 y 8) tienen alta dispersión

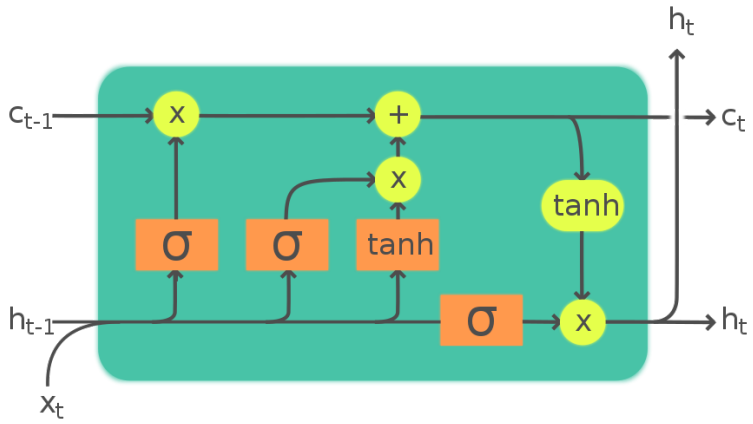
Metodología de solución

- ¿Qué técnicas de Aprendizaje de Máquina/IA se utilizaron para resolverlo?

Se utilizaron herramientas de ***Aprendizaje profundo***



Red neuronal densa



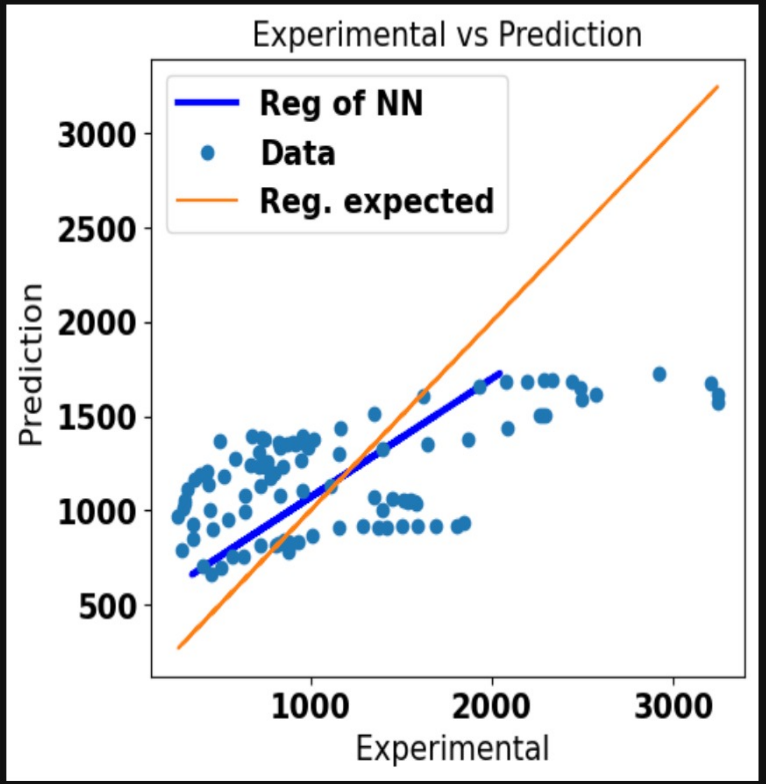
Red neuronal recurrente LSTM

Solución usando una red neuronal densa tipo regresor

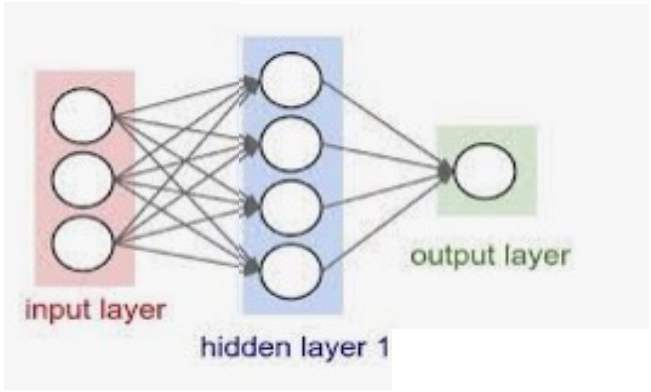
```
[19]: import tensorflow as tf
      from tensorflow import keras
      from tensorflow.keras import layers

      inputs = keras.Input(shape = (3))
      X = layers.Dense(100)(inputs) # 100
      outputs = layers.Dense(1)(X)  #1 neurona de salida.

      model = keras.Model(inputs, outputs)
      model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 3)]               0

 dense (Dense)               (None, 100)               400

 dense_1 (Dense)             (None, 1)                 101

=================================================================
Total params: 501
Trainable params: 501
Non-trainable params: 0
_________________________________________________________________
```

```
Intercept: [897.12461396]
Slope: [[0.2409753]]
Coefficient of determination: 0.10725196722161368
[83]: <matplotlib.legend.Legend at 0x7fb7012479a0>
```



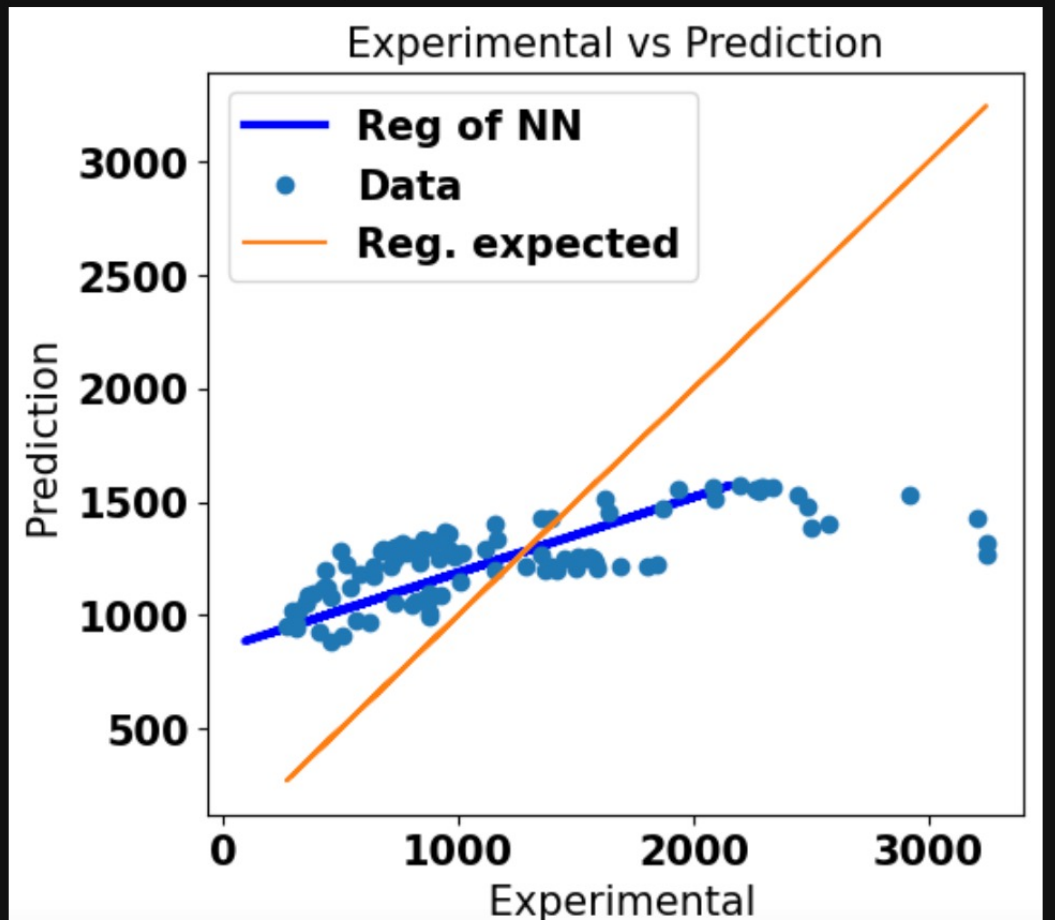
Solución usando una red neuronal densa con una capa oculta




```
#modelo de una capa oculta
inputs = keras.Input(shape = (3,))
X = layers.Dense(10)(inputs) # 10
hidden = layers.Dense(20)(X) #20
outputs = layers.Dense(1)(hidden)  #1 neurona de salida.

model = keras.Model(inputs, outputs)
model.summary()
```

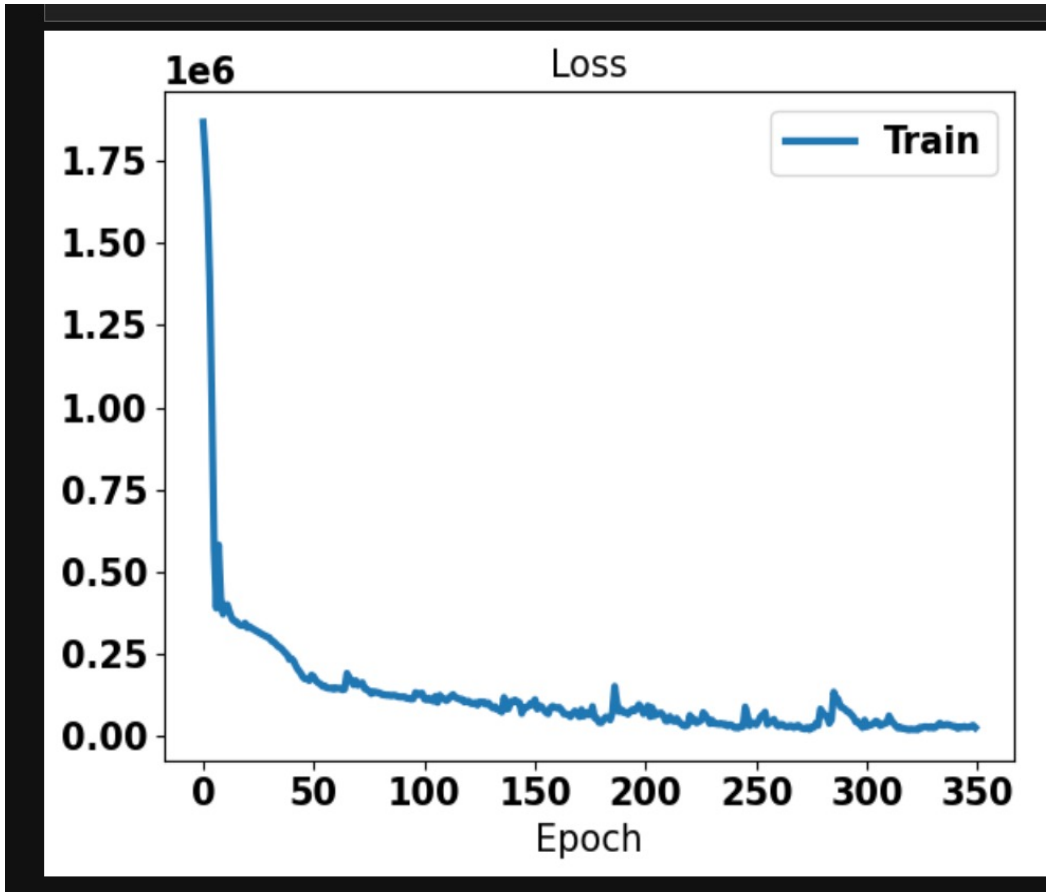
```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 3)]               0

 dense (Dense)               (None, 10)                40

 dense_1 (Dense)             (None, 20)                220

 dense_2 (Dense)             (None, 1)                 21

=================================================================
Total params: 281
Trainable params: 281
Non-trainable params: 0
_________________________________________________________________
```

```
Intercept: [1047.62631781]
Slope: [[0.16877917]]
Coefficient of determination: 0.042003602178628086
[24]: <matplotlib.legend.Legend at 0x7fe1533dee90>
```



Uso de una red neuronal recurrente LSTM *(Long short term memory)*

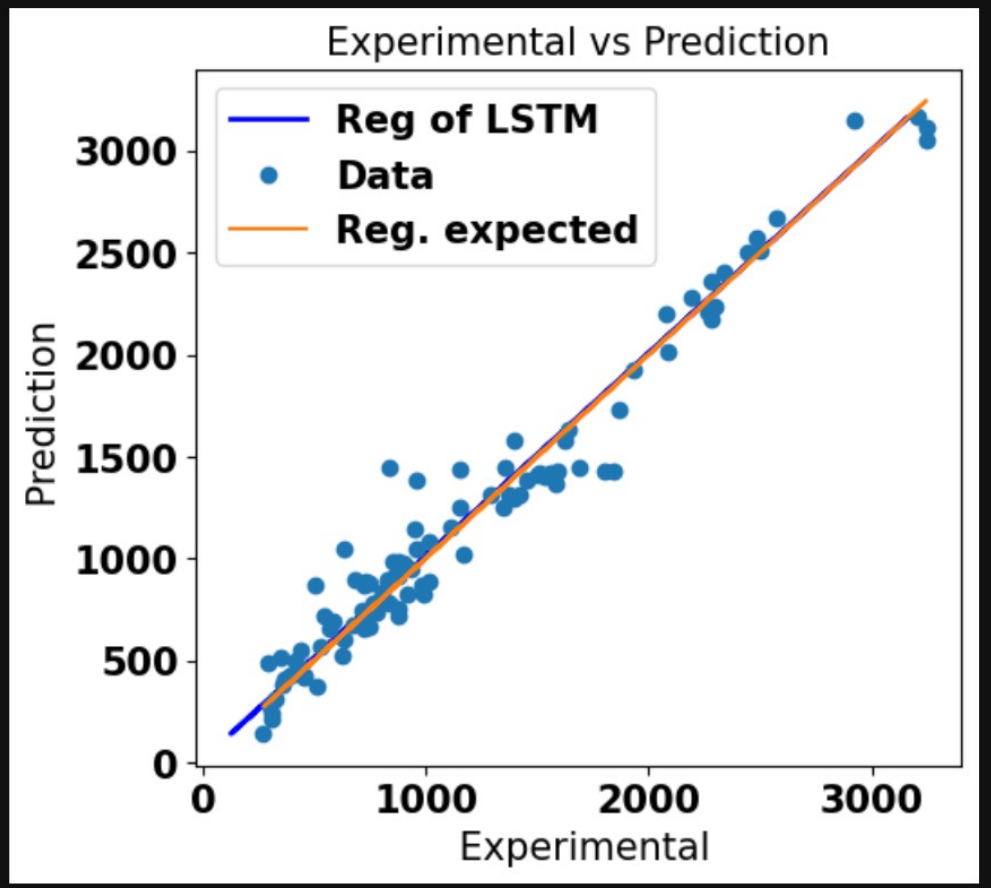
```
[87]: ### **Using The LSTM**

[93]: from keras.models import Sequential
      from keras.layers import LSTM
      from keras.layers import Dense

[106]: model2 = Sequential()
       model2.add(LSTM(1100, activation = 'relu', input_shape = (3,1)))  #LSTM 1000
       model2.add(Dense(1))
       model.summary()

Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 3)]               0

 dense (Dense)               (None, 100)               400

 dense_1 (Dense)             (None, 1)                 101

=================================================================
Total params: 501
Trainable params: 501
Non-trainable params: 0
_________________________________________________________________
```



350 epochs

La red neuronal recurrente LSTM dio un coef. de correlación de 0.9534 entre los datos experimentales y los datos del predictor

```
Intercept: [65.17624927]
Slope: [[0.95243999]]
Coefficient of determination: 0.9534412797791668
[115]: <matplotlib.legend.Legend at 0x7fb701604580>
```




```
[ ]:

[ ]: # Se salva la red LSTM entrenada

     modelFile2 = "V4.Biogas__Deep_Python.LSTM.keras"

     model2.save(modelFile2)

[ ]: ### ** MUCHAS GRACIAS ANA PAULINA, DANIEL, SALVADOR Y CESARË*
     ### ** POR TODO LO ENSEÑADO*
```

Discusión

- Los datos experimentales no presentaron una distribución Gaussiana y algunos lotes tienen alta dispersión.

- En las corridas realizadas con la red neuronal densa tipo regresor y de una capa oculta, no se obtuvieron buenos resultados por lo que se recomienda evaluar el uso de más capas ocultas o bien probar normalizando los datos de entrada.

- La red neuronal recurrente (LSTM) generó una pendiente de 0.9524 entre los datos experimentales y los del predictor con un coef. De determinación de 0.9534.

- Se recomienda evaluar esta red con datos que no ha visto para evaluar la capacidad predictora.

Conclusión

La red neuronal recurrente LSTM es sencilla de implementar y el modelo propuesto tiene una correlación adecuada para utilizarse como predictor.

Esta red neuronal puede ser de utilidad para la automatización, control y optimización de bioprocesos para la producción de biogas en procesos fermentativos.

Agradecimientos

- A los Doctores Ana Paulina Castañeda, Salvador Ruíz Correa, Daniel Salgado y Cesaré Ovando, **muchas gracias** por sus enseñanzas y paciencia.
- Al grupo de ciencias e ingenierías computacionales del CNS-IPICyT.
- A los compañeros del diplomado por su amistad y apoyo constante.
- Al Dr. Julio García por su ayuda personal.