

UNIVERSIDAD  
SAN MARTÍN DE  
PORRES

INGENIERÍA  
CIVIL



HIDROLOGÍA

# OBJETIVO

DESARROLLAR SOFTWARE

*QUE SIRVA PARA HALLAR LAS PRECIPITACIONES CON  
INTELIGENCIA ARTIFICIAL*



# RESUMEN

LA HIDROLOGÍA HA UTILIZADO MÉTODOS TRADICIONALES PARA PRONOSTICAR NIVELES DE INUNDACIÓN. SIN EMBARGO, ÉSTOS PUEDEN GENERAR PROBLEMAS DE PRECISIÓN, CAUSADOS POR EL COMPORTAMIENTO NO LINEAL DE LAS INUNDACIONES Y LAS LIMITACIONES AL NO INCLUIR TODAS LAS VARIABLES, COMO FLUJO, Y NIVEL DE AGUA Y PRECIPITACIÓN. EN CONSECUENCIA, ALGUNOS CIENTÍFICOS COMENZARON A UTILIZAR MÉTODOS NO CONVENCIONALES BASADOS EN MODELOS DE INTELIGENCIA ARTIFICIAL, PRONOSTICANDO LAS INUNDACIONES DE MANERA MÁS PRECISA Y RIGUROSA.

# PINART

DESARROLLO DE CÓDIGO CON PYTHON

# SOFTWARE

FOR CIVIL ENGINEERS

DESARROLLA LA FUNCION DE COMPLETACIÓN DE  
DATOS DE LAS PRECIPITACIONES CON USO DE LA  
INTELIGENCIA ARTIFICIAL







## CÓMO

DESCUBRE CÓMO IMPLEMENTARLO USANDO  
EL CODIGO PYTHON Y LOS PASOS A SEGUIR.



## QUÉ

DESCUBRE QUÉ SOLUCIONES UNA  
ALTERNATIVA PARA DESARROLLAR LA  
COMPLETACIÓN DE DATOS CON IA


# PINART



## QUIÉN

INGENIEROS CIVILES, HIDRAULICOS,  
GEOLOGOS, ETC.





CÓDIGO  
FUENTE  
PYTHON 3.7.1





# RECONOCIMIENTO DE DATOS PASO 1



- |   |   |   |
|---|---|---|
| ○ <code>import matplotlib as plt</code>   | → | ○ Librería para mostrar las gráficas                              |
| ○ <code>from string import ascii_letters</code>   | → | ○ Librería para importar códigos de números binarios              |
| ○ <code>import pandas as pd</code>  | → | ○ Librería para leer archivos Excel, txt, etc.                    |
| ○ <code>import numpy as np</code>   | → | ○ Librería con gran colección de funciones matemáticas            |
| ○ <code>TodasEstaciones = "DatosPrecipitacionesPinart.xlsx"</code>  | → | ○ Definir "TodasEstaciones" como archivo Excel                    |
| ○ <code>df = pd.read_excel(TodasEstaciones, index_col=0)</code>   | → | ○ Leer el archivo Excel desde la columna inicial                  |
| ○ <code>print(df.head())</code>   | → | ○ Imprimir en consola los 5 primeros datos                        |
| ○ <code>df.plot(subplots=['Est1','Est2','Est3','Est4','Est5','Estx'],<br/>figsize=(12, 8)); plt.legend(loc='best')</code> | → | ○ Llamar a la función Plot para mostrar las gráficas<br>iniciales |
| ○ <code>xticks(rotation='vertical')</code>  |   |   |

**LIBRERÍAS**

**IMPRIMIR DATOS**

**GRÁFICAR**



# Resultados: RECONOCIMIENTO DE DATOS

PASO 1



IMPRIMIR DATOS

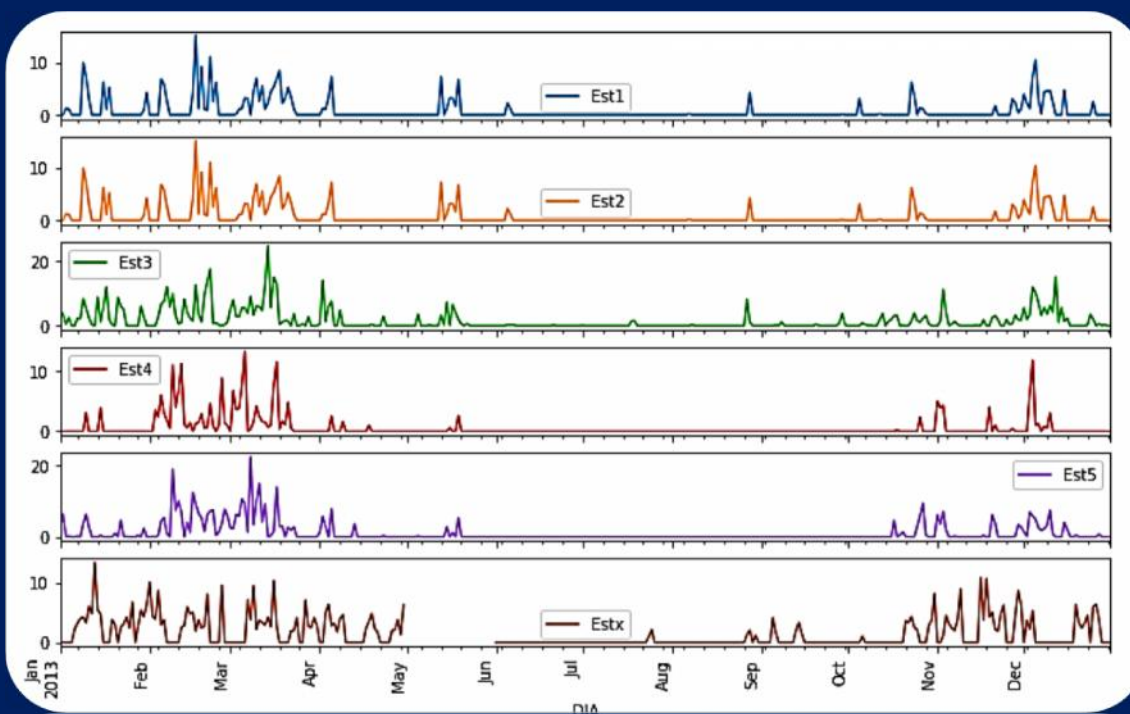
GRÁFICAR

*Populating the interactive namespace from numpy and matplotlib*

*Est1 Est2 Est3 Est4 Est5 Estx*

*DJA*

<i>2013-01-01</i>	<i>0.0</i>	<i>0.0</i>	<i>4.8</i>	<i>0.0</i>	<i>2.7</i>	<i>0.0</i>
<i>2013-01-02</i>	<i>0.0</i>	<i>0.0</i>	<i>3.5</i>	<i>0.0</i>	<i>6.4</i>	<i>0.0</i>
<i>2013-01-03</i>	<i>1.2</i>	<i>1.2</i>	<i>0.4</i>	<i>0.0</i>	<i>0.0</i>	<i>0.0</i>
<i>2013-01-04</i>	<i>1.1</i>	<i>1.1</i>	<i>2.5</i>	<i>0.0</i>	<i>0.2</i>	<i>0.0</i>
<i>2013-01-05</i>	<i>0.0</i>	<i>0.0</i>	<i>0.0</i>	<i>0.0</i>	<i>0.0</i>	<i>0.0</i>



*2013-01-02 0.0 0.0 0.0 0.0 0.0 0.0*





# AJUSTAR DATOS PARA SU ENTRENAMIENTO

## PASO 2



○ `x_train = df.loc['2013-01-01':'2013-12-31',['Est1','Est2','Est3','Est4','Est5']].astype(float32).values`



○ Datos completos en "X\_TRAIN" para su entrenamiento

○ `y_train = df.loc['2013-01-01':'2013-12-31',['Estx']].astype(float32).values`



○ Datos incompletos en "Y\_TRAIN" para su entrenamiento

○ `from sklearn.preprocessing import StandardScaler`



○ Librería para normalizar o escalar los datos de -1 @ 1 para disminuir el error por la diferencia de cantidades.

○ `scaler= StandardScaler().fit(x_train)`



○ Llamar a la función "StandardScaler" para escalar los datos de "X\_TRAIN"

○ `x_train= scaler.transform(x_train)`



○ Transformar los datos a un nuevo "X\_TRAIN"

○ `x_train[:20]`



○ Imprimir los nuevos datos escalados de -1 @ 1

## DEFINICIÓN DE VARIABLES

## ESCALAR DATOS



# Resultados: AJUSTAR DATOS PARA SU ENTRENAMIENTO

## PASO 2

### ESCALAR DATOS



```
array([[ -0.39871722, -0.39871722,  0.948019  , -0.31968224,  0.5298519  ],
       [ -0.39871722, -0.39871722,  0.5537036 , -0.31968224,  1.8278112  ],
       [  0.18849915,  0.18849915, -0.38658696, -0.31968224, -0.41730762],
       [  0.13956445,  0.13956445,  0.25038403, -0.31968224, -0.34714767],
       [ -0.39871722, -0.39871722, -0.5079148 , -0.31968224, -0.41730762],
       [ -0.39871722, -0.39871722, -0.5079148 , -0.31968224, -0.41730762],
       [ -0.39871722, -0.39871722,  0.18972012, -0.31968224, -0.34714767],
       [ -0.39871722, -0.39871722,  0.1593882  , -0.31968224, -0.41730762],
       [  4.4458175  ,  4.4458175  ,  2.0096374 , -0.31968224,  0.8806518  ],
       [  3.1245809  ,  3.1245809  ,  1.1300107 ,  1.3627384 ,  1.7927314  ],
       [  1.0693237  ,  1.0693237  ,  0.22005212, -0.31968224,  0.5298519  ],
       [ -0.39871722, -0.39871722, -0.41691893, -0.31968224, -0.41730762],
       [ -0.39871722, -0.39871722, -0.5079148 , -0.31968224, -0.41730762],
       [ -0.39871722, -0.39871722,  2.161297  , -0.31968224, -0.41730762],
       [ -0.39871722, -0.39871722, -0.14393134,  1.7969116 , -0.2769877  ],
       [  2.6352339  ,  2.6352339  ,  1.2513386 , -0.31968224, -0.41730762],
       [  0.13956445,  0.13956445,  3.1015875  , -0.31968224, -0.41730762],
       [  2.145887   ,  2.145887   ,  0.06839231, -0.31968224, -0.41730762],
       [ -0.39871722, -0.39871722, -0.41691893, -0.31968224, -0.41730762],
       [ -0.39871722, -0.39871722, -0.5079148 , -0.31968224, -0.0665078  ]],
      dtype=float32)
```





# CONFIGURACIÓN DE LA RED NEURAL

## PASO 3



- `from keras.models import Sequential`
- `from keras.layers import Dense`
- `import tensorflow.keras as kr`
- `lr= 0.01`
- `model = Sequential()`
- `model.add(Dense(32,activation='linear', input_shape=(5,)))`
- `model.add(Dense(4, activation='linear'))`
- `model.add(Dense(8,activation='linear'))`
- `model.add(Dense(1,activation='linear'))`
- `model.summary()`
- `model.compile(loss='mean_squared_error',  
optimizer=kr.optimizers.Adam(lr=lr), metrics=['accuracy'])`
- `model.fit(x_train, y_train, epochs=100)`

- **KERAS es una librería para reducir la carga y la cantidad de acciones que el usuario debería hacer y proporciona mensajes de error claros y procesables.**

- **CONFIGURANDO LA RED NEURAL (...)**

- **AJUSTE DE LOS DATOS DE LA ESTACION FALTANTE CON LOS DATOS COMPLETOS DE OTRAS ESTACIONES Y ENTRENAMIENTO**



# RED NEURAL: EXPLICACIÓN

## PASO 3



Una red neuronal es un modelo simplificado que emula el modo en que el cerebro humano procesa la información: Funciona simultaneando un número elevado de unidades de procesamiento interconectadas que parecen versiones abstractas de neuronas.

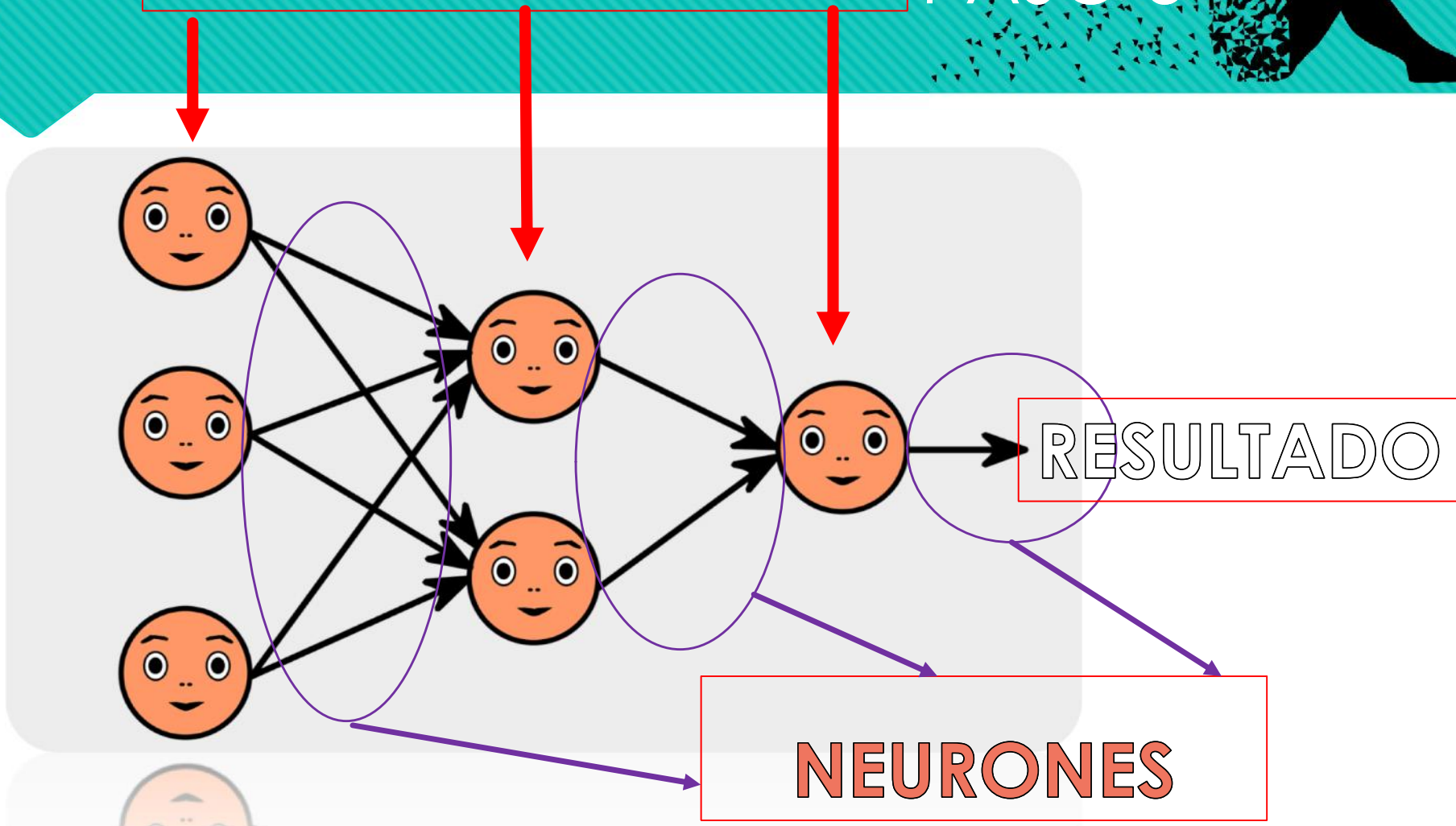






CAPAS

PASO 3





# CAPAS

## PASO 3

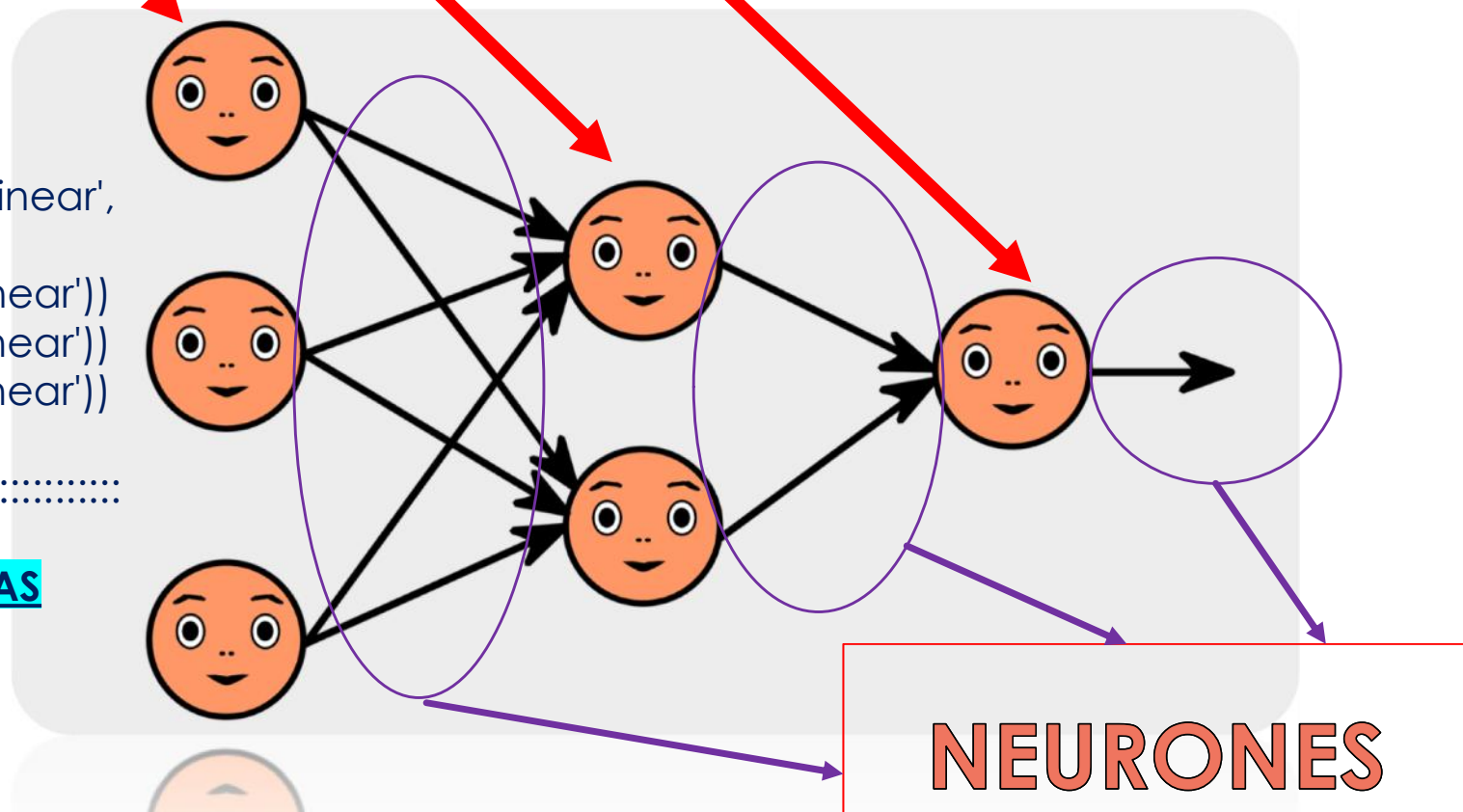


DEL CÓDIGO TENEMOS:

```
model = Sequential()  
model.add(Dense(32,activation='linear',  
input_shape=(5,)))  
model.add(Dense(4,activation='linear'))  
model.add(Dense(8,activation='linear'))  
model.add(Dense(1,activation='linear'))  
model.summary()
```

**MODEL.ADD = CAPAS= 4 CAPAS**

**32, 4, 8, 1 = NEURONES**







# RESULTADOS: CONFIGURACIÓN DE LA RED NEURAL

PASO 3



CONFIGURACIÓN DE LA RED NEURAL

AJUSTE Y ENTRENAMIENTO DE DATOS

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 2)	12
dense_1 (Dense)	(None, 4)	12
dense_2 (Dense)	(None, 8)	40
dense_3 (Dense)	(None, 1)	9

Total params: 73

Trainable params: 73

Non-trainable params: 0



# PREDICCIÓN DE LOS DATOS FALTANTES

## PASO 4



- `y_pred = model.predict(x_train)` → ○ Extraer los datos de predicción
- `print(y_pred.shape)` → ○ Imprimir los datos de predicción
- `plot(df.loc['2013-01-01':'2013-12-31'].index, y_pred, label='Predicted')` ○ Graficar los resultados comparando la estación faltante con la estación predicha por el programa
- `df['Estx'].loc['2013-01-01':'2013-12-31'].plot()` →
- `figsize(12,8)`
- `legend(loc='best')`





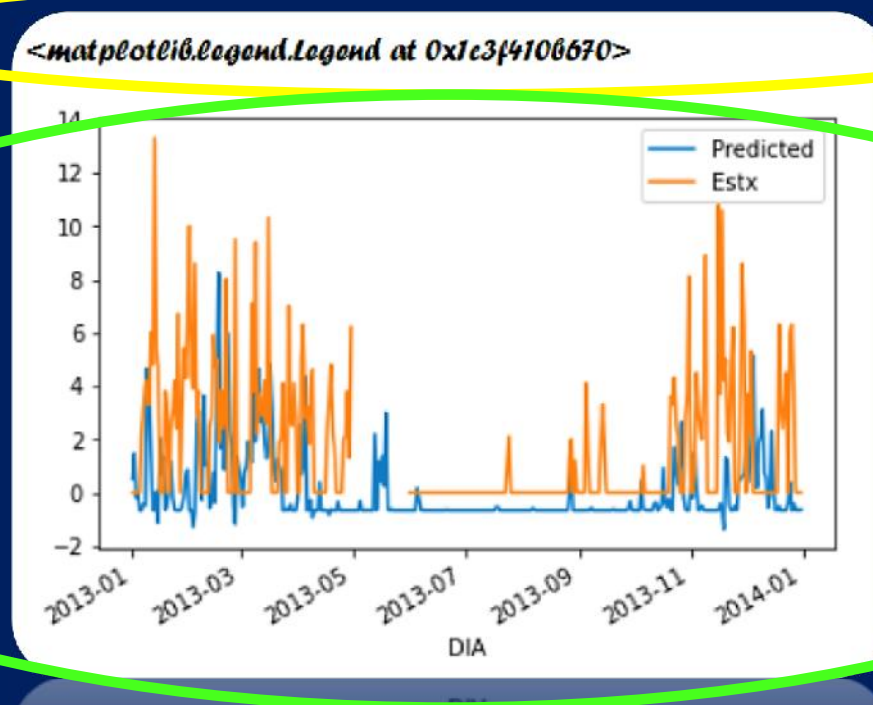
# RESULTADO: PREDICCIÓN DE LOS DATOS FALTANTES

PASO 4



EXTRACCIÓN DE LOS DATOS PREDICHOS

GRÁFICA DE LOS DATOS PREDICHOS





# GRAFICAR TODAS LAS ESTACIONES Y LA ESTACIÓN YA COMPLETADA

## PASO 4



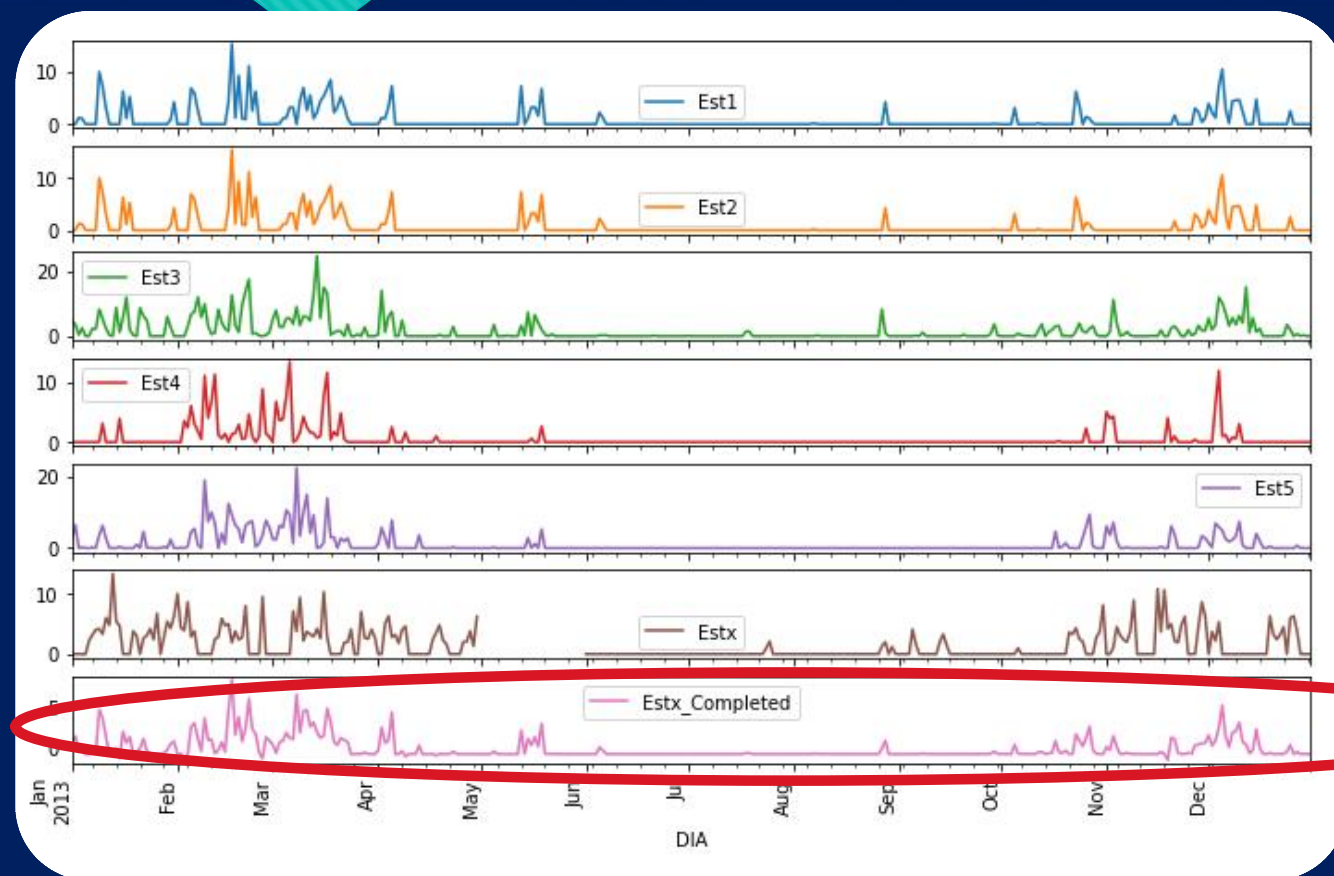
SE REPITE LAS ACCIONES DEL PASO 1 PERO AHORA SE DEBE IMPRIMIR LA ESTACIÓN PREDICHA TAMBIEN.

- `x_missing = df.loc['2013-01-01':'2013-12-31',['Est1','Est2','Est3','Est4','Est5']].astype(float32).values`
- `from sklearn.preprocessing import StandardScaler`
- `scaler= StandardScaler().fit(x_missing)`
- `x_missing= scaler.transform(x_missing)`
- `y_missing= model.predict(x_missing)`
- `y_missing= y_missing.reshape(365).tolist()`
- `df['Estx_Completed']=df['Estx']`
- `df['Estx_Completed'].loc['2013-01-01':'2013-12-31']=y_missing`
- `df.plot(subplots=['Est1','Est2','Est3','Est4','Est5','Estx"Estx_Completed'], figsize=(12, 8)); plt.legend(loc='best')`
- `xticks(rotation='vertical')`





# RESULTADOS TOTALES PINART



**RESULTADO**



# CONCLUSIONES

- LOS RESULTADOS MUESTRAN UNA COMPLETACIÓN DE DATOS FALTANTES DE ESTACIONES HIDROLÓGICAS CONSIDERANDO LOS DATOS DE LAS ESTACIONES CON INFORMACIÓN COMPLETA Y PREDICIENDO LOS DATOS RESPECTO AL COMPORTAMIENTO DE LA ESTACIÓN INCOMPLETA, Y ASÍ ENCONTRAR VALORES USANDO UN GRUPO DE ENTRENAMIENTOS LINEALES ENTRE SÍ.
- EL PROYECTO PUEDE MEJORAR SI SE USA UNA BASE DE DATOS EXTENSA EN LA CUAL PERMITA AL PROGRAMA ENCONTRAR LA INFORMACIÓN MÁS PRECISA. MIENTRAS MÁS INFORMACIÓN CONTENGA EL PROGRAMA MEJOR SERÁ EL RESULTADO OBTENIDO.
- LA CANTIDAD DE ENTRENAMIENTOS Y DE REDES NEURALES DEBEN SER DETERMINADAS POR EL DESARROLLADOR PARA QUE NO GENERE DESFASE EN EL ANÁLISIS POR EL COMPUTADOR. DEBIDO A QUE LA CAPACIDAD DEL ORDENADOR INFLUYE EN EL ANÁLISIS.



THANKS