CARACTERIZACIÓN ESPECTRAL DE MÓDULOS FOTOVOLTAICOS DE CONCENTRACIÓN A PARTIR DE VARIABLES ATMOSFÉRICAS MEDIANTE TÉCNICAS DE INTELIGENCIA ARTIFICIAL

Francisco Cruz

IRON HACK

MADRID

propuesto por

Dr.Pedro Manuel Rodrigo

UNIVERSIDAD PANAMERICANA
Campus Aguascalientes

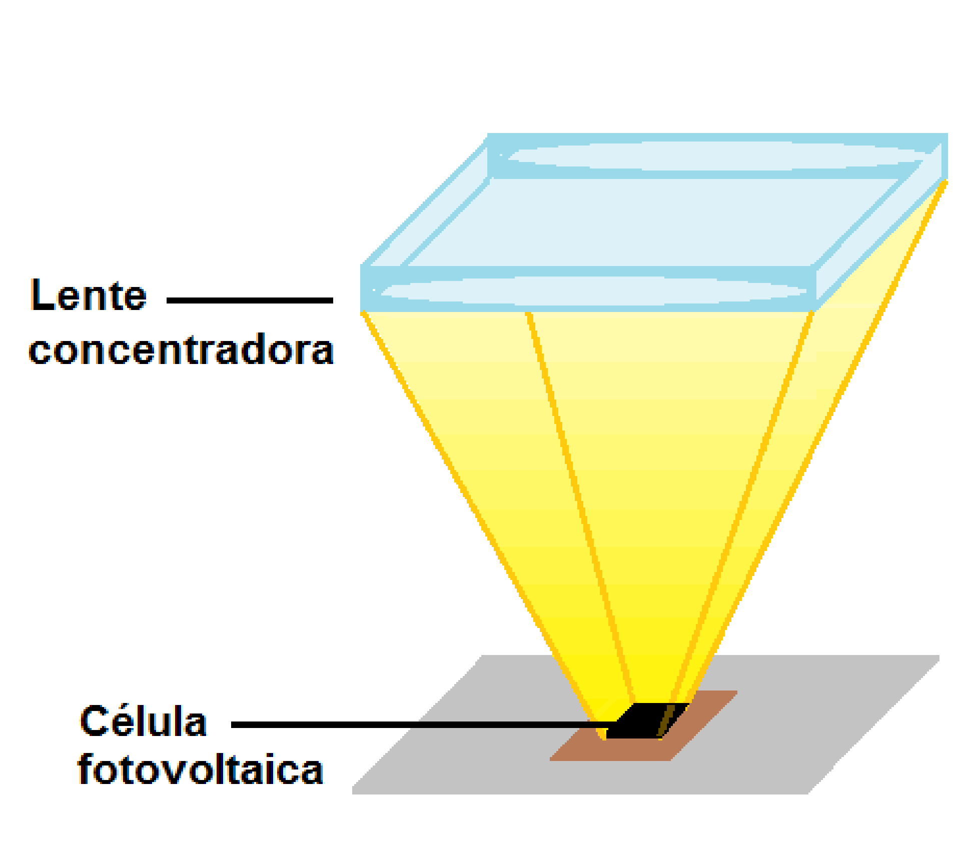
DEEP LEARNING VS MACHINE LEARNING

CONTENIDOS

- Cel.Foto.Concentración
- El problema
- Los datos
- Metodología|Tecnología
- Machine Learning
- Deep Learning
- Resultados
- Próximo pasos

SISTEMAS FOTOVOLTAICOS DE CONCENTRACIÓN

Elementos ópticos para concentrar la luz solar sobre las células



RECORD 46% EFICIENCIA EN APLICACIONES SOLARES

SISTEMAS FOTOVOLTAICOS DE CONCENTRACIÓN

$$P = P_{pk} \cdot \frac{DNI}{1000\, W/m^2} \cdot SF \cdot TF$$

P-PK: Power Peak
DNI: Direct Normal Irradiance
TF: Thermal Factor
SF: Spectral Factor

ES CAPITAL PODER PREDECIR LA PONTENCIA QUE VA A GENERAR UN SISTEMA RENOVABLE

EL PROBLEMA

Calcular SF es caro. Se necesitan simuladores solares y espectrorradiómetros.

$$SF = \frac{min_j\left\{\int E(\lambda)\cdot SR_j(\lambda)\cdot d\lambda\right\}}{min_j\left\{\int E^{\text{¿}}(\lambda)\cdot SR_j(\lambda)\cdot d\lambda\right\}}\cdot\frac{\int E^{\text{¿}}(\lambda)\cdot d\lambda}{\int E(\lambda)\cdot d\lambda}$$

Se sabe por física que SF depende de tres factores atmosféricos que son baratos de medir-

-Air Mass

-Aerosol Optical Depth

-Precipitable Water

LOS DATOS

	am.am	aod	pw	sf
0	19.094719	0.14548	4.670454	0.154467
1	11.037272	0.14548	4.670454	0.423916
2	7.723984	0.14548	4.670454	0.612491
3	5.934944	0.14548	4.670454	0.737127
4	4.819924	0.14548	4.670454	0.821330
5	4.060702	0.14548	4.670454	0.880082
6	3.511752	0.14548	4.670454	0.922377
7	3.097262	0.14548	4.670454	0.953616
8	2.773908	0.14548	4.670454	0.977192
9	2.515162	0.14548	4.670454	0.995313
10	2.303882	0.14548	4.670454	1.009508
11	2.128505	0.14548	4.670454	1.020666
12	1.980948	0.14548	4.670454	1.029719
13	1.855399	0.14548	4.670454	1.036994
14	1.747567	0.14548	4.670454	1.038944
15	1.654223	0.14548	4.670454	1.033348
16	1.572886	0.14548	4.670454	1.028373
17	1.501624	0.14548	4.670454	1.023837

129.546 registros en diferentes localizaciones:

-Solar Village en Arabia Saudi
-Alta Floresta en Brasil
-Frenchman Flat en USA
-Granada en España
-Pekín en China

LOS DATOS: PROCESADO

	am.am	aod	pw	sf
0	19.094719	0.14548	4.670454	0.154467
1	11.037272	0.14548	4.670454	0.423916
2	7.723984	0.14548	4.670454	0.612491
3	5.934944	0.14548	4.670454	0.737127
4	4.819924	0.14548	4.670454	0.821330
5	4.060702	0.14548	4.670454	0.880082
6	3.511752	0.14548	4.670454	0.922377
7	3.097262	0.14548	4.670454	0.953616
8	2.773908	0.14548	4.670454	0.977192
9	2.515162	0.14548	4.670454	0.995313
10	2.303882	0.14548	4.670454	1.009508
11	2.128505	0.14548	4.670454	1.020666
12	1.980948	0.14548	4.670454	1.029719
13	1.855399	0.14548	4.670454	1.036994
14	1.747567	0.14548	4.670454	1.038944
15	1.654223	0.14548	4.670454	1.033348
16	1.572886	0.14548	4.670454	1.028373
17	1.501624	0.14548	4.670454	1.023837

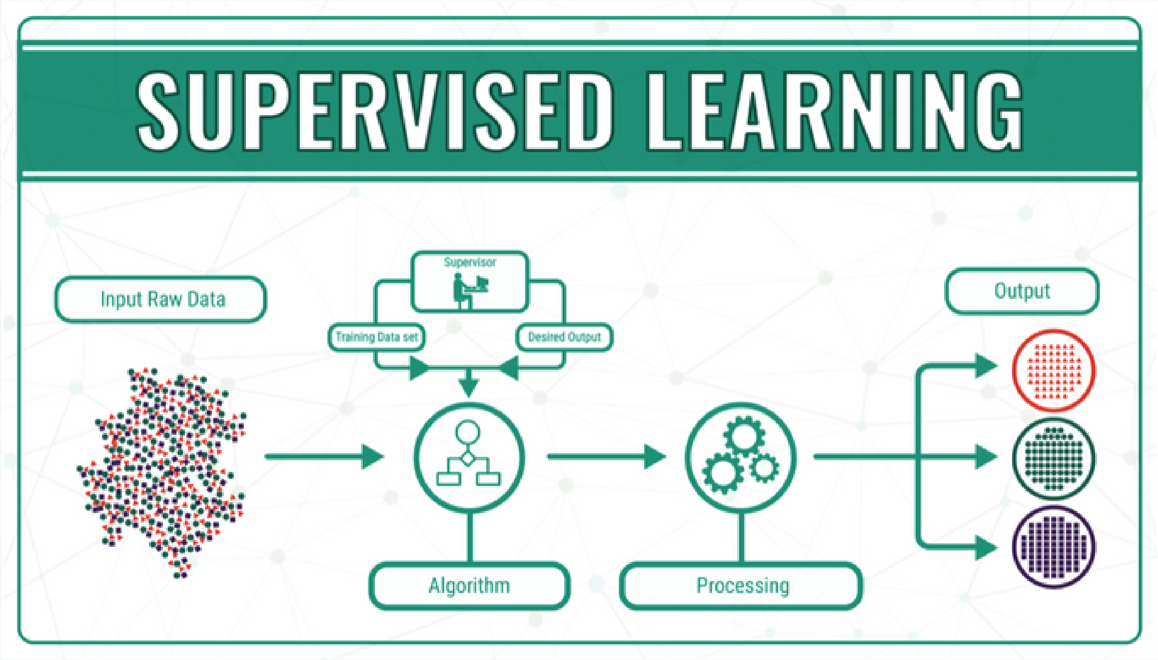
Se llevaron a cabo varias técnicas:

-Limpieza: Casteo a dato más preciso.

-Escalados: Coma flotante.

-Técnicas de normalización.

METODOLOGIA|TECNOLOGÍA



DASK

NVIDIA
CUDA

scikits
learn
machine learning in Python

Keras

MACHINE LEARNING

modelo+parámetros óptimos+
bucle infinito

```
def getTheForest():
    count=11
    while(True):
        model = RandomForestRegressor(bootstrap=bool, criterion='mse', max_depth=m,
                                      max_features=x, max_leaf_nodes=None,
                                      min_impurity_decrease=x, min_impurity_split=None,
                                      min_samples_leaf=x, min_samples_split=m,
                                      min_weight_fraction_leaf=x, n_estimators=s,
                                      n_jobs=None, oob_score=False, random_state=None,
                                      verbose=x, warm_start=False)

        data= pd.read_csv('./input/especNum.csv').drop(columns='Unnamed: 0')
        X_train, X_test, y_train, y_test = train_test_split(data.drop(columns='sf'), data.sf,test_size=0.5)
        model.fit(X_train,y_train)
        y_pred= model.predict(X_test)
        score=r2_score(y_test, y_pred)
        print(score)
        if score > 0.0:
            print(print('YEEEHHHH'))
            dump(model, './output/forest/forestFitted{}.joblib'.format(count))
            count+=1
        del model
```

NO problema de acoplamiento

60%

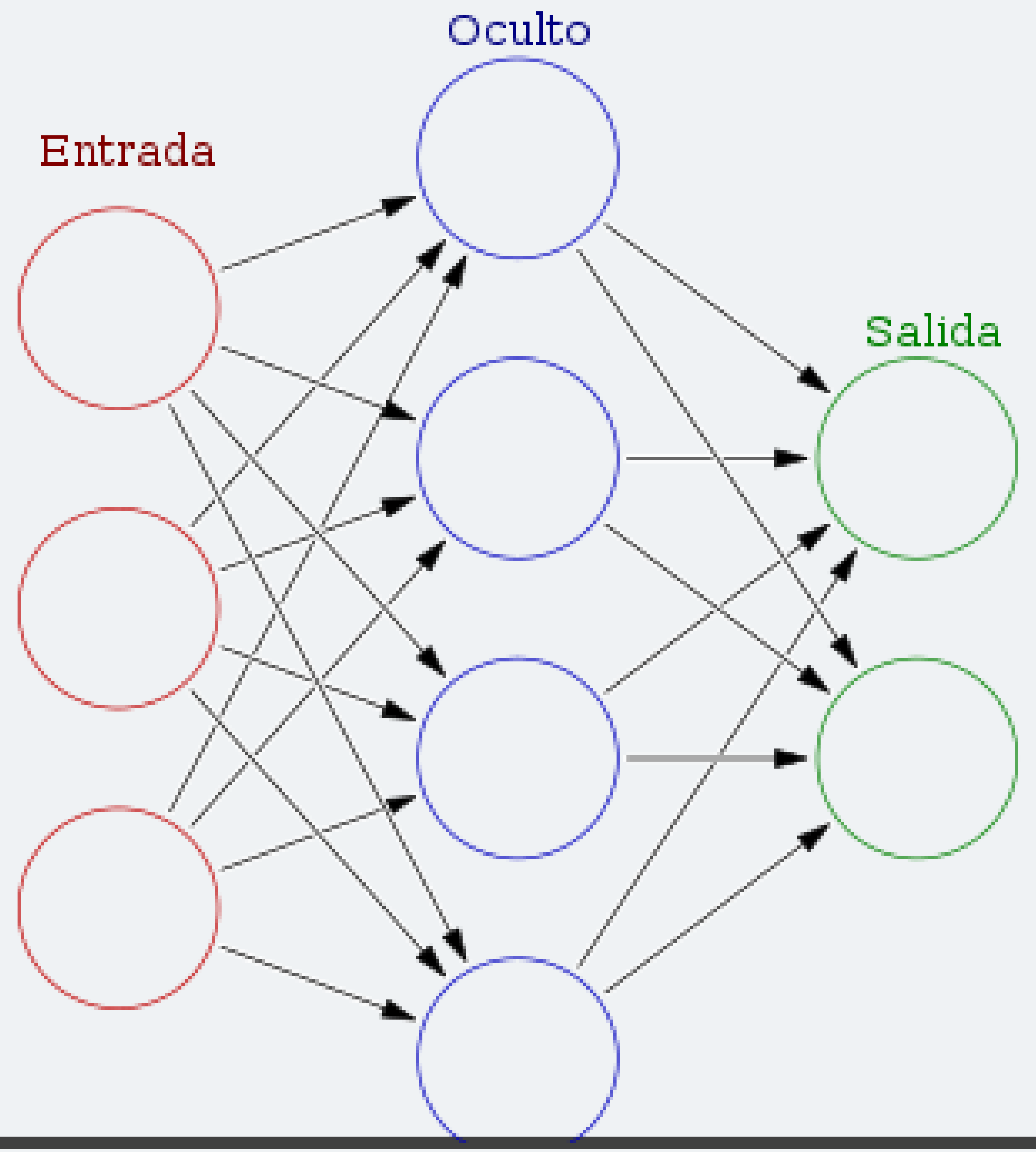
DATA

DEEP LEARNING

topología+parámetros óptimos+
bucle infinito

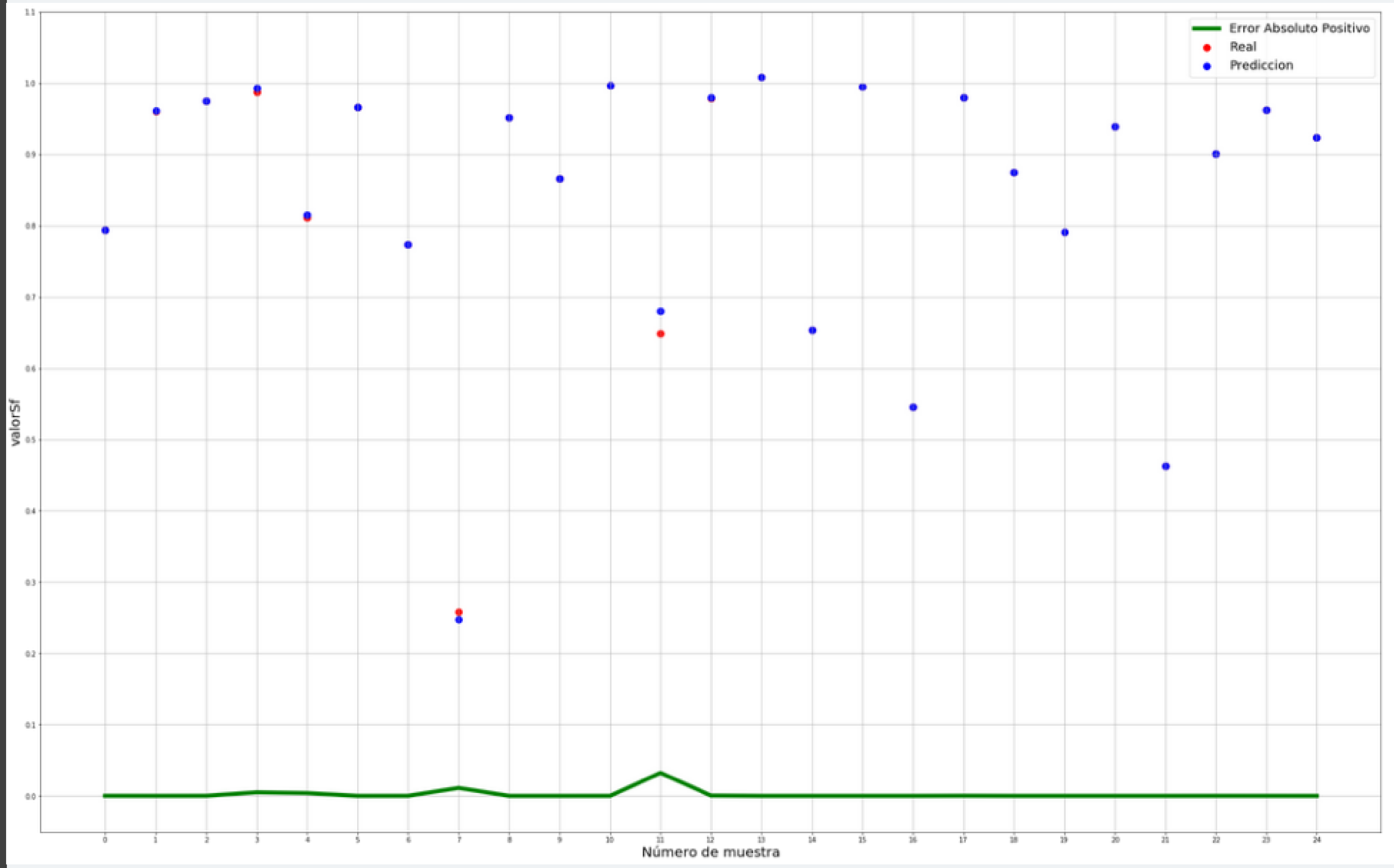
```
def eternal():
    early_stop = keras.callbacks.EarlyStopping(monitor='val_loss', patience=10)
    EPOCHS = 1000
    score=0.0
    count=1
    scaler=getScaler(pd.read_csv('./input/especNum.csv').drop(columns='Unnamed: 0'))
    while(True):
        train_dataset, test_dataset, train_labels ,test_labels=getData()
        model = build_model(train_dataset)
        train_dataset=scaler.transform(train_dataset)
        test_dataset=scaler.transform(test_dataset)
        model.fit(train_dataset, train_labels, epochs=EPOCHS,
                        validation_split = 0.2, verbose=0, callbacks=[early_stop, PrintDot()])
        y_pred= model.predict(test_dataset).flatten()
        newScore=r2_score(test_labels, y_pred)
        print(newScore)
        if newScore>score:
            print('YEEEHHHHHHHHHHHHHHHHHHHHHH')
            score=newScore
            model.save('./output/keras/kerasSGDScaler{}.h5'.format(count))
            count+=1
        elif score==1:
            raise Exception("APOCALYPSE!!!!!!!")
        del model
```

Red Neuronal

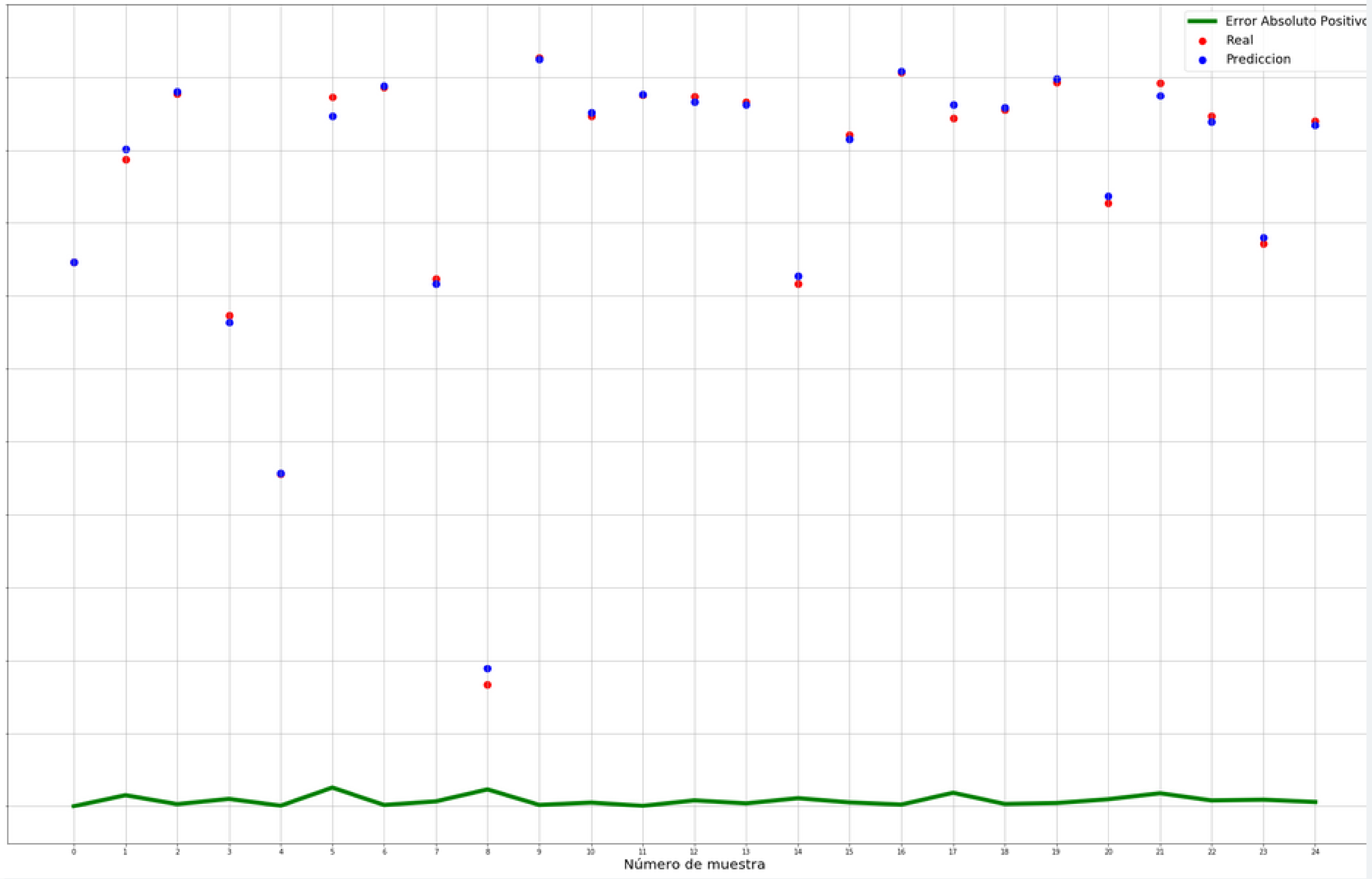


RESULTADOS

MACHINE LEARNING



DEEP LEARNING



25 predicciones aleatorias por modelo entre las 129.546 predicciones

RESULTADOS

MACHINE LEARNING	DEEP LEARNING
99.833%	99.176%
r2_score (%)	r2_score (%)
202 predicciones ErrorAbsP>0.1	1038 prediciones ErrorAbsP>0.1

129.546 predicciones totales

PRÓXIMOS PASOS

Contrastar con nuevos datos

Ecuación Subyacente

¿Publicar?

99% DONE?