Machine Learning con Python

Random Forest Regressor

Vamos a tratar de realizar un modelo de predicción, mediante Random Forest Regressor, con los datos de la estación meteorológica, tratados en el proyecto anterior con R. Empezamos importando los datos.

```
import numpy as np
import pandas as pd
import datetime as dt
import matplotlib.pyplot as plt

archivo = '/Users/alexandremartinez/Desktop/Datos_estacion.xls'

df = pd.read_excel(archivo)
df.head()
```

```
##   Date (Europe/Madrid)  Tempin (ºC)  ...  Solarrad (W/m )  Uvi ()
## 0  2021-12-25 11:30:00         19.1  ...            112.9     1.0
## 1  2021-12-25 12:10:00         20.0  ...            160.6     1.0
## 2  2021-12-25 12:40:00         20.4  ...            389.0     3.0
## 3  2021-12-25 13:10:00         20.4  ...            303.3     2.0
## 4  2021-12-25 13:20:00         20.7  ...            206.3     2.0
##
## [5 rows x 19 columns]
```

Creamos un nuevo 'dataframe' donde irán alojadas las variables que nos interesan para realizar la predicción.

```
dataset = pd.DataFrame()

dataset['temp'] = df['Temp (ºC)'].values
dataset['rocio'] = df['Dew (ºC)'].values
dataset['hum'] = df['Hum (%)'].values
dataset['press'] = df['Bar (hPa)'].values
dataset['rain'] = df['Rain (mm)'].values
dataset['rad'] = df['Solarrad (W/m )'].values


dataset = dataset.dropna()
```

Tras el tratado y filtrado de datos podemos proseguir con el modelo de predicción. Usaremos los paquetes 'train_test_split' y 'RandomForestRegressor', alojados en la librería scikit-learn, de modo a poder dividir los datos adecuadamente y aplicar un modelo predictivo, respectivamente.

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split

X = dataset.drop(['rain', 'temp'], axis=1) #Eliminamos las variables rain y temp
y = dataset['temp']

xtrain, xtest, ytrain, ytest = train_test_split(X, y, test_size=0.2, random_state=42)

model = RandomForestRegressor()
model.fit(xtrain, ytrain)
```

```
## RandomForestRegressor()
```

```
score = model.score(xtest,ytest)

print('La precisión del modelo es:',round(score*100, 3), '%')
```

```
## La precisión del modelo es: 99.921 %
```

Observamos un resultado en la precisión del modelo bastante buena y por ello lo pondremos a prueba prediciendo la temperatura en las últimas horas que registró la estación meteorológica.

```
#prediccion

archivo_pred = '/Users/alexandremartinez/Downloads/pred.xls'

df_pred = pd.read_excel(archivo_pred, index_col=0, parse_dates=True)

dataset_pred = pd.DataFrame()
dataset_pred.index = df_pred.index
dataset_pred['temp'] = df_pred['Temp (ºC)'].values
dataset_pred['rocio'] = df_pred['Dew (ºC)'].values
dataset_pred['hum'] = df_pred['Hum (%)'].values
dataset_pred['press'] = df_pred['Bar (hPa)'].values
dataset_pred['rain'] = df_pred['Rain (mm)'].values
dataset_pred['rad'] = df_pred['Solarrad (W/m )'].values

dataset_pred = dataset_pred.dropna()
```

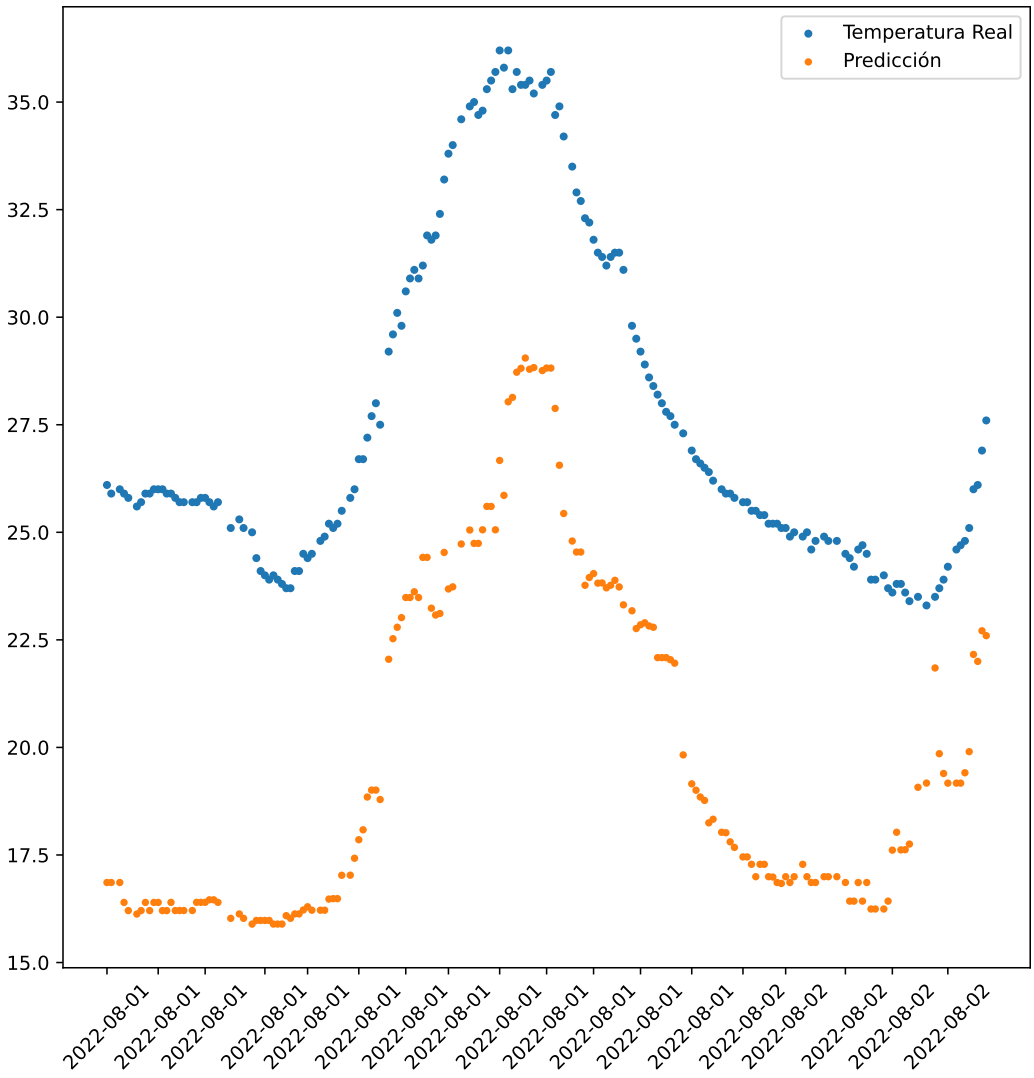
```
X_pred = dataset_pred.drop(['rain', 'temp'], axis=1)

plt.subplots(figsize=(9,9))
```

```
## (<Figure size 1800x1800 with 1 Axes>, <AxesSubplot: >)
##
## <string>:1: MatplotlibDeprecationWarning: The resize_event function was deprecated in Matplotlib 3.6
```

```
plt.scatter(dataset_pred.index, dataset_pred['temp'], s=10, label='Temperatura Real')
plt.scatter( dataset_pred.index , model.predict(X_pred), s = 8, label = 'Predicción')
plt.legend()

plt.xticks([dataset_pred.index[0], dataset_pred.index[10],dataset_pred.index[20],dataset_pred.index[30]
```

```
## ([<matplotlib.axis.XTick object at 0x163db7460>, <matplotlib.axis.XTick object at 0x163db7430>, <matp
```

```
plt.show()
```



Observamos una predicción bastante buena en forma pero un tanto imprecisa en cuanto a valor exacto. Aún así, el resultado tiene bastante buena pinta.

¿Podríamos mejorar el modelo?

La respuesta es un gran SÍ y lo podríamos hacer con la función GridSearchCV. Aquí dejo un ejemplo de cómo debería de escribirse.

```
#from sklearn.model_selection import GridSearchCV

#param = {
#  'n_estimators': np.arange(100, 500, 100),
#  'criterion': ['squared_error', 'absolute_error', 'poisson']
#}
```

```
#grid = GridSearchCV(RandomForestRegressor(), param, cv = 5)

#grid.fit(xtrain, ytrain)

#print('El mejor estimador es:', grid.best_params_)
```

Veamos si nuestro modelo es tan bueno prediciendo la lluvia. Realizamos los mismos ajustes que antes y obtenemos:

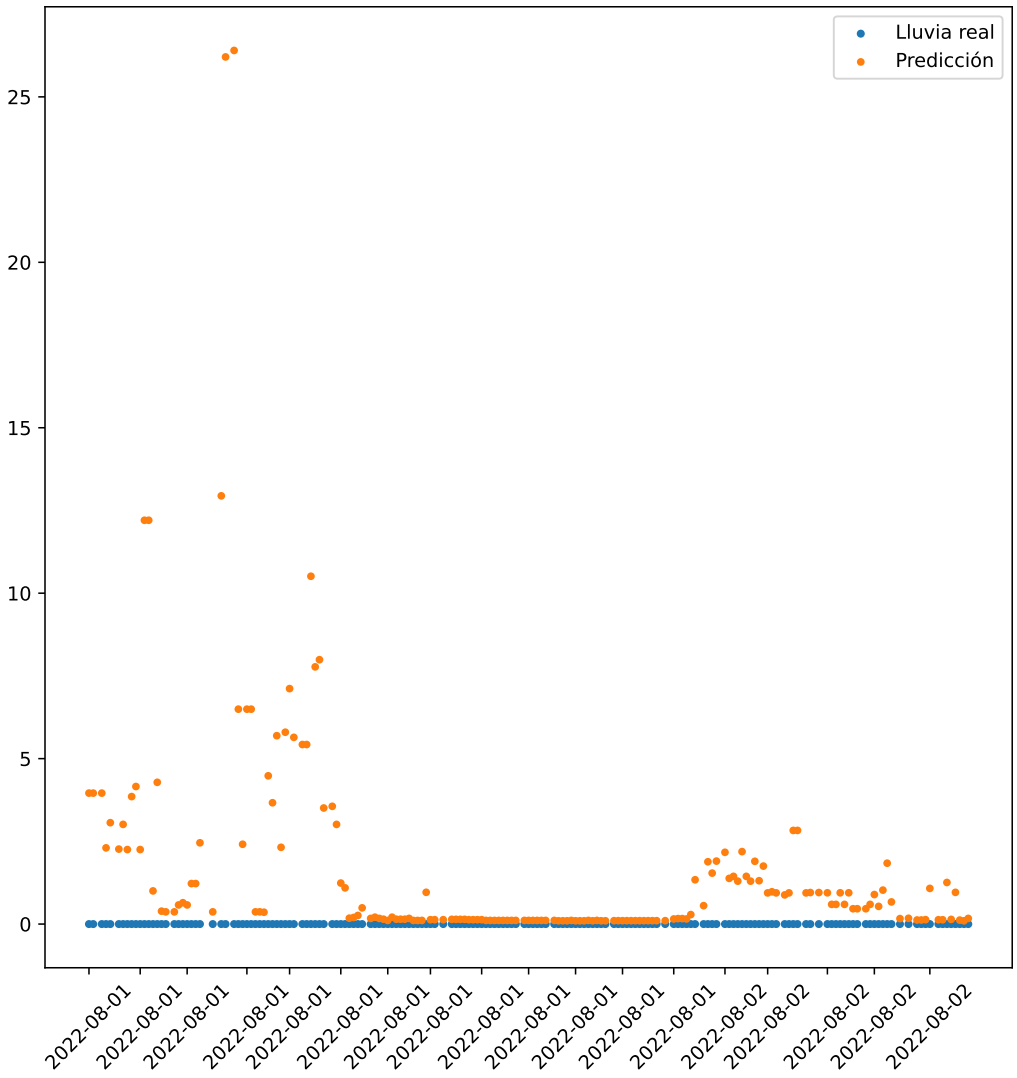
```
X_pred = dataset_pred.drop(['rain'], axis=1)

plt.subplots(figsize=(9,9))
```

```
## (<Figure size 1800x1800 with 1 Axes>, <AxesSubplot: >)
##
## <string>:1: MatplotlibDeprecationWarning: The resize_event function was deprecated in Matplotlib 3.6
```

```
plt.scatter(dataset_pred.index, dataset_pred['rain'], s=10, label='Lluvia real')
plt.scatter( dataset_pred.index , model.predict(X_pred), s = 9, label='Predicción')
plt.legend()
plt.xticks([dataset_pred.index[0], dataset_pred.index[10],dataset_pred.index[20],dataset_pred.index[30]
```

```
## ([<matplotlib.axis.XTick object at 0x16ce77610>, <matplotlib.axis.XTick object at 0x16ce775e0>, <matp
```

```
plt.show()
```



Remarcamos una predicción un tanto burda. Esto se puede deber a una escasez cantidad de datos de precipitación, representativos, o a un modelo que no es adecuado.