Series temporais

Mohammed Ashour, Antonio Moscoso Sánchez,
Nicolás Vilela Pérez

20 de novembro do 2022

Horas dedicadas fóra de clase: 10

Primeiramente hai que comentar que xunto con este informe adxúntase un script de Python co nome `codigo.py` que contén todo o código empregado durante a realización do mesmo.

Para a realización deste informe partiuse de 3 arquivos CSV que recopilaron cada día os datos da irradiación solar, presión atmosférica e temperatura en Santiago de Compostela desde o 01/01/2006 ata o 24/10/2022, ambos incluídos.

O primeiro que se fixo sobre estes arquivos foi observalos, para así detectar rapidamente que os valores nulos están representados polo número `-9999`. Cabe comentar que tamén se modificaron as cabeceiras das columnas dos arquivos para que todos tiveran a mesma estrutura: `Fecha|X`, tomando X os valores `IRRA`, `P` ou `TM`, segundo os datos indicados sexan de irradiación solar, presión atmosférica ou temperatura, respectivamente.

Unha vez analizados os arquivos, para manipulalos o primeiro que hai que facer é cargalos a través do paquete `pandas`.

```
df_IRRA = pd.read_csv('IRRA_Santiago.csv', sep='|', index_col=
    False, encoding='unicode_escape', na_values=-9999)
df_P = pd.read_csv('P_Santiago.csv', sep='|', index_col=False,
    encoding='unicode_escape', na_values=-9999)
df_TM = pd.read_csv('TM_Santiago.csv', sep='|', index_col=False,
    encoding='unicode_escape', na_values=-9999)
```

Unha vez cargados, bórranse os valores atípicos ou *outliers*. Para esta labor definiuse a seguinte función:

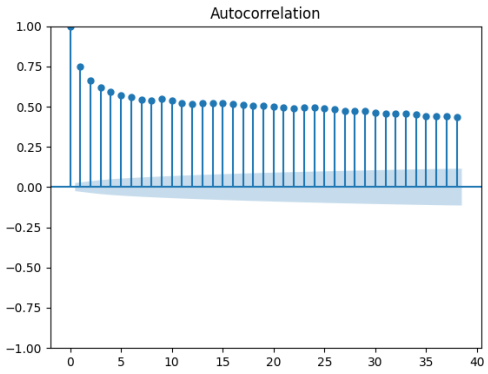
```
def remove_outliers(df):
    """Function to remove outliers"""

    # Calculate the Q3 and Q1
    Q3 = float(df.quantile(0.75, numeric_only=True))
    Q1 = float(df.quantile(0.25, numeric_only=True))
```

```
    # Calculate the IQR
    IQR = Q3 - Q1

    # Calculate the upper and lower limits of the dataframe
    upper_limit = Q3 + 1.5 * IQR
    lower_limit = Q1 - 1.5 * IQR

    # Add index of rows that are outliers to a list
    indexesToDelete = []
    for j in range(len(df)):
        if df.iloc[j, 1] > upper_limit or df.iloc[j, 1] < lower_limit:
            indexesToDelete.append(j)

    # Drop from the dataframe the rows whose indexes are in the list
    df.drop(indexesToDelete, inplace=True)

    return df
```

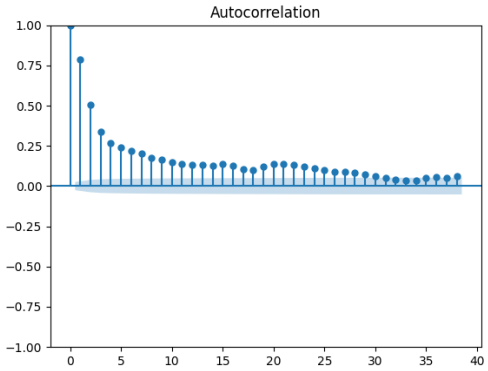
Cabe comentar que se calcularon as medias dos diferentes datos para substituír os valores nulos por devanditos valores nos cálculos e representacións posteriores que requerían que non houbese devandito tipo de valores.

```
# Calculate the mean of the data
mean_IRRA = float(df_IRRA.mean(numeric_only=True))
mean_P = float(df_P.mean(numeric_only=True))
mean_TM = float(df_TM.mean(numeric_only=True))
```

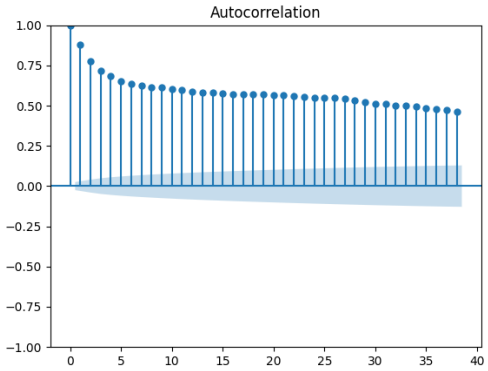
Móstranse a continuación os diagramas ACF, que nos axudarán a saber se as series temporais son estacionarias ou non.



(a) *Irradiación solar*



(b) *Presión atmosférica*

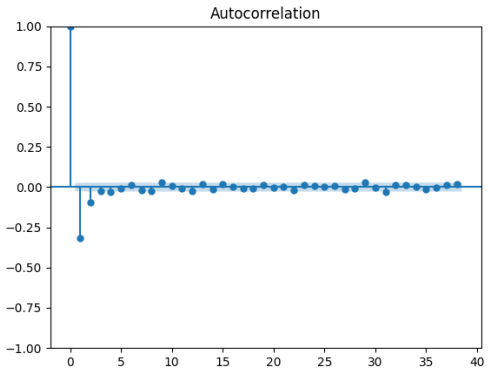


(c) *Temperatura*

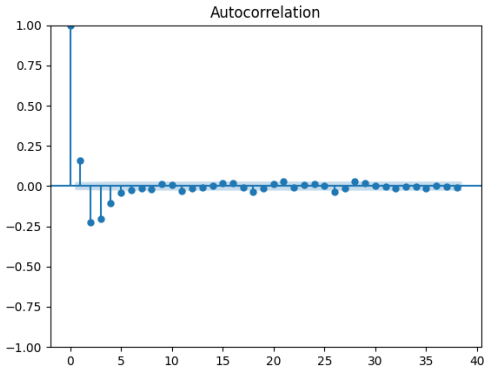
Figura 1: *Diagramas ACF das series temporais*

Como se pode observar nos diagramas da figura 1, o valor da función de autocorrelación decae lentamente, particularidade das series non estacionarias.

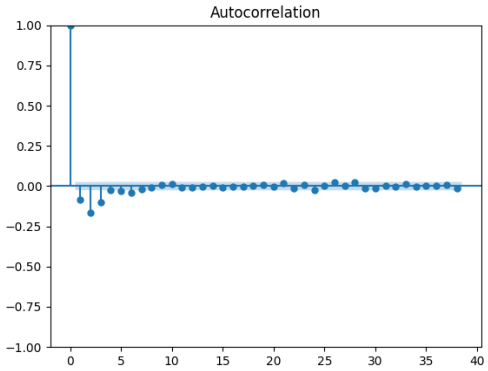
Para facer que unha serie sexa estacionaria hai que calcular as diferenzas. Neste caso calculáronse a primeira e a segunda diferenza. Móstranse a continuación os diagramas ACF correspondentes a estas diferenzas:



(a) *Irradiación solar*

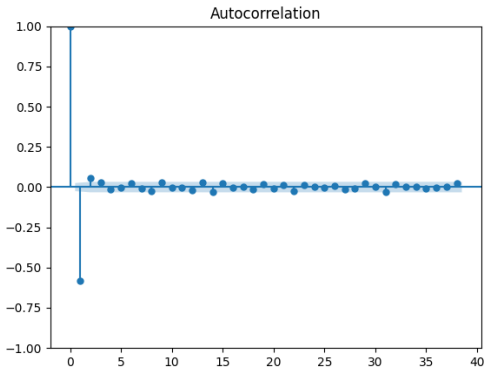


(b) *Presión atmosférica*

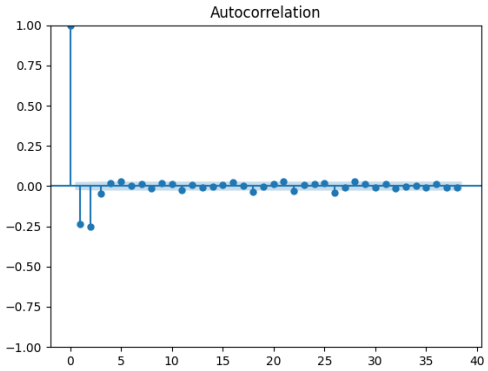


(c) *Temperatura*

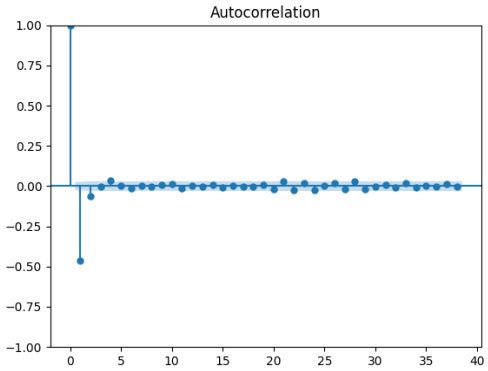
Figura 2: *Diagramas ACF da primeira diferenza*



(a) *Irradiación solar*



(b) *Presión atmosférica*

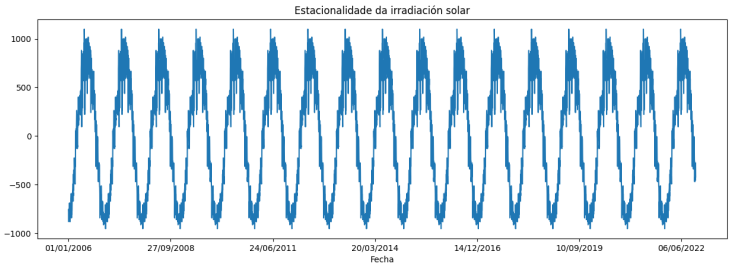


(c) *Temperatura*

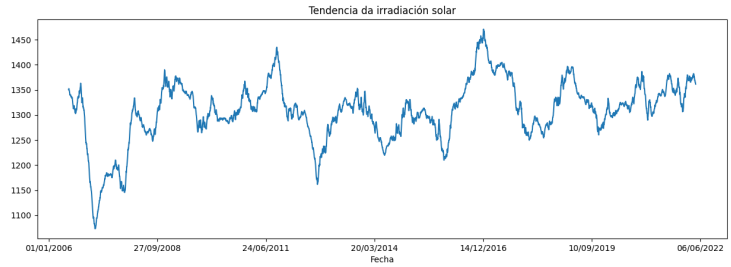
Figura 3: *Diagramas ACF da segunda diferenza*

Como se pode apreciar nos diagramas das figuras 2 e 3, as series son agora estacionarias, tras aplicar dúas diferenzas. Realizouse a segunda diferenza xa que a veces para facer unha serie estacionaria non chega ca primeira (neste caso, pódese apreciar que a segunda diferenza é máis estacionaria que a primeira).

A continuación descompuxéronse as series temporais en compoñentes estacionais e tendencia, obtendo os seguintes gráficos:

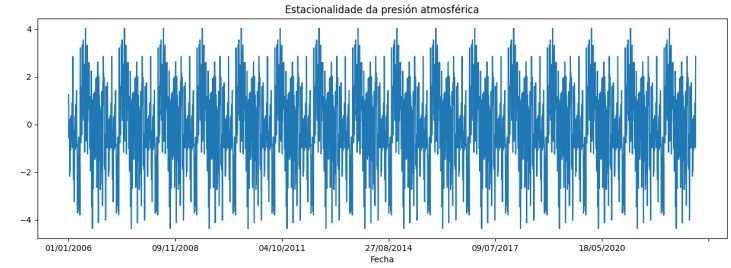


(a) *Estacionalidade*

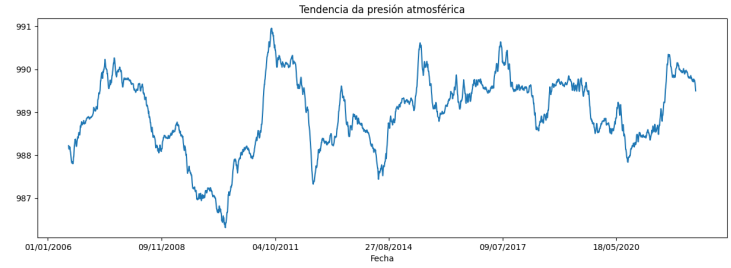


(b) *Tendencia*

Figura 4: *Descomposición da serie da irradiación solar*

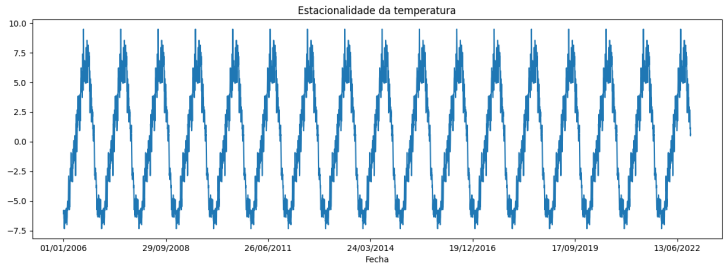


(a) *Estacionalidade*

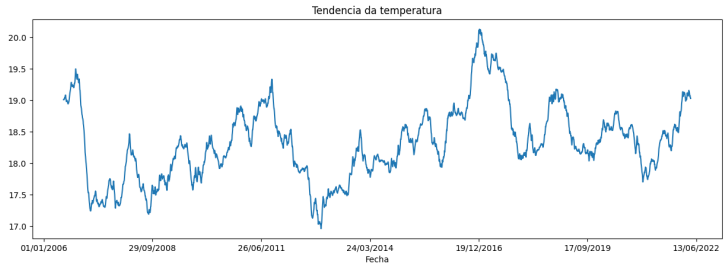


(b) *Tendencia*

Figura 5: *Descomposición da serie da presión atmosférica*



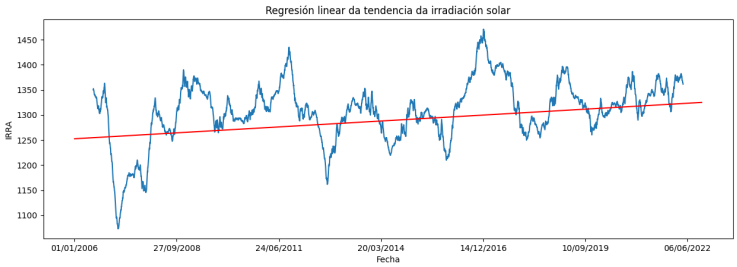
(a) *Estacionalidade*



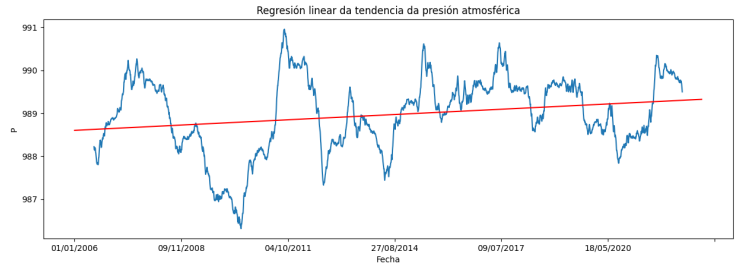
(b) *Tendencia*

Figura 6: *Descomposición da serie da temperatura*

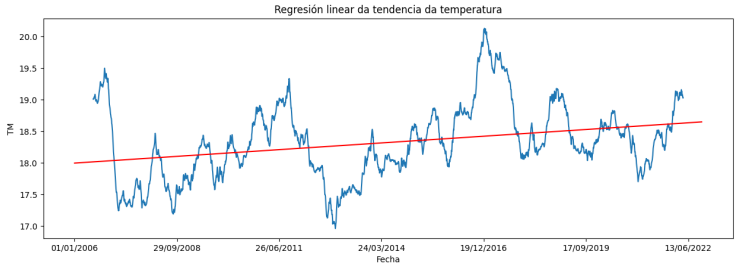
Unha vez obtidas as tendencias a partir das series temporais, realizarase o cálculo da interpolación lineal ás devanditas tendencias. Tras executar o código correspondente obtivéronse os seguintes gráficos:



(a) *Regresión da irradiación solar*



(b) *Regresión da presión atmosférica*

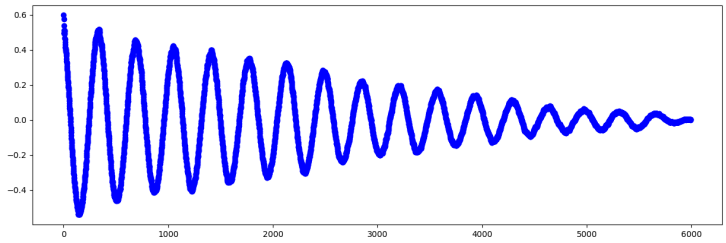


(c) *Regresión da temperatura*

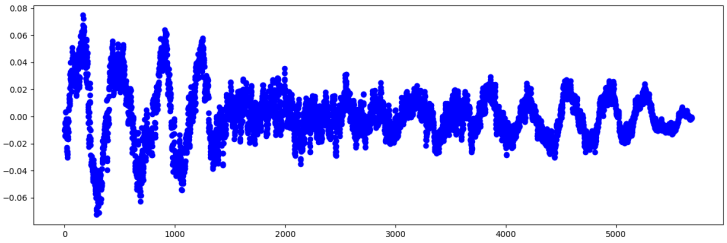
Figura 7: *Interpolación lineal da tendencia de cada unha das descomposicións*

Pódese comprobar na figura 7 que mediante a regresión obtense un axuste lineal da función de tendencia anteriormente calculada. Aínda que para as primeiras e últimas datas non existan valores para a función, coa regresión vese que existen valores para todos os valores do eixo X.

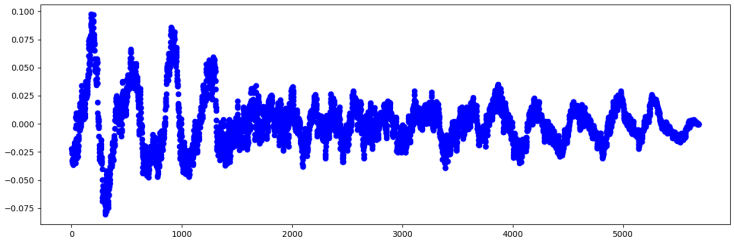
Posteriormente realizarase o cálculo da correlación cruzada entre as series temporais dos arquivos CSV anteriormente importados. Tras representar a correlación cruzada entre as tres series, obtivéronse as seguintes gráficas:



(a) *Correlación cruzada entre irradiación solar e temperatura atmosférica*



(b) *Correlación cruzada entre irradiación solar e presión atmosférica*



(c) *Correlación cruzada entre temperatura atmosférica e presión atmosférica*

Figura 8: *Interpolación lineal do trend de cada unha das descomposicións*

Pódese comprobar na figura 8 que existe unha gran correlación cruzada entre a irradiación solar e a temperatura atmosférica, pois pódese observar que os valores nos que oscila a gráfica son notablemente altos (tanto positivos como negativos en función do crecemento ou decrecemento conxunto das gráficas, tendo valores máximos que chegan aproximadamente a 0.6). Por outro lado, as correlacións cruzadas entre a presión atmosférica e a temperatura ou irradiación solar non son fortes (vendo as propias gráficas, os valores están ao redor do 0 no eixo Y). Ademais, os valores máximos de correlación cruzada (eixo Y) non superan o 0.1 en ambos casos. Polo tanto, existe unha alta correlación entre a irradiación solar e a temperatura atmosférica, mentres que a presión atmosférica ten unha correlación moi baixa coa temperatura atmosférica e a irradiación solar.

Seguidamente farase unha explicación dos procesos que se levaron a cabo para filtrar os conxuntos de datos cunha media móbil.
En primeiro lugar creouse unha función que realiza este proceso:

```
1 def mean_filter(dataseries: pd.Series, window_size=5):
2     """Function applies rolling mean into given dataseries"""
```

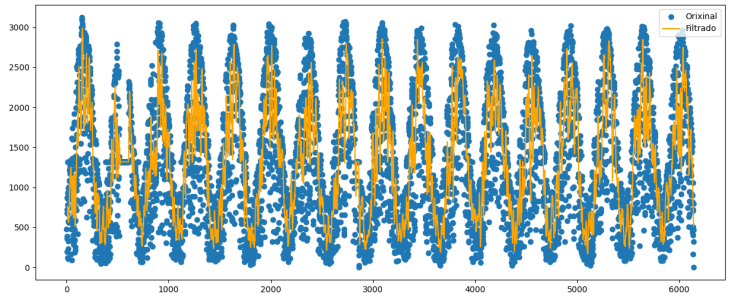
```
3
4       filtered = dataseries.rolling(window_size).mean()
5       return filtered
6
7       return df
```

A función `rolling` agrupa as observación dunha semana atrás e emprega o valor promedio para analizar a tendencia xeral. Desta maneira, evítanse valores que poden ser pouco representativos da tendencia do modelo. O valor *window* corresponde ao tamaño dos subconxuntos de datos que se seleccionan para calcular os valores agregados que darán lugar á nova serie de datos.

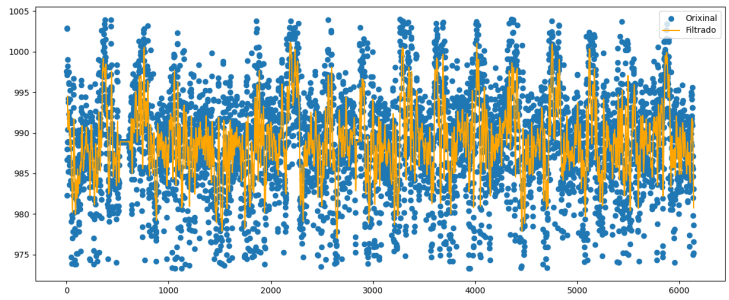
Despois de obter o conxunto de datos filtrado, procedemos a realizar a representación dos dous conxuntos nunha gráfica, e a almacenala como imaxe PNG:

```
plt.figure(figsize=(15, 6))
plt.scatter(df.index, df.valor)
plt.plot(df_avg.valor, color = '#ffa500')
plt.legend(["Orixinal", "Filtrado"])
plt.savefig("df_mean_filter.png")
plt.clf()
```

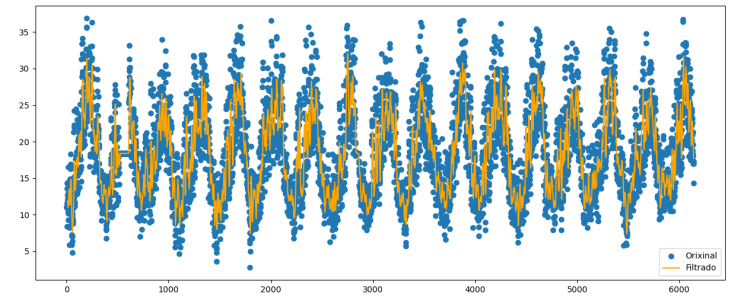
Os resultados obtidos para as funcións son os seguintes:



(a) *Filtrado da irradiación solar*



(b) *Filtrado da presión atmosférica*



(c) *Filtrado da temperatura atmosférica*

Figura 9: *Filtrado en tendencia coa media móbil para cada conxunto de datos*

Como se pode observar nas distintas gráficas, o filtrado de datos crea unha representación máis "fina"do conxunto de datos orixinal, evitando así posibles datos pouco representativos.

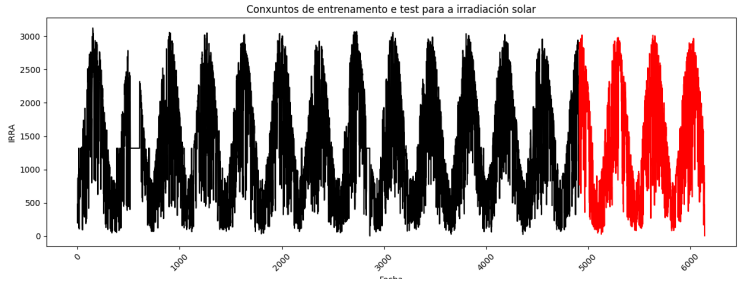
O último exercicio que se levará a cabo será a predición de valores futuros nos ditintos conxuntos de datos. Para isto empregaranase os paquetes `sklearn` e `skforecast`.

En primerio lugar, divídese o conxunto de datos en dous conxuntos: entrenamento e test. O 80 % dos datos conformará o conxunto de entrenamento, e o 20 % restante, o de test. Co primerio conxunto entrenarase o modelo, e o de test será o empregando para comparar coa predición. A separación realízase da seguinte

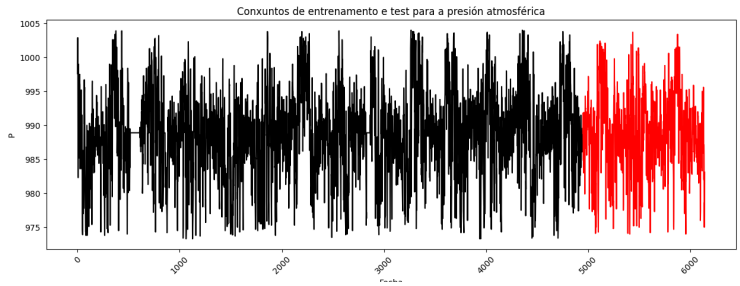
maneira:

```
train = df_TM[:int(0.8 * (len(df_TM)))]
test = df_TM[int(0.8 * (len(df_TM))):]
```

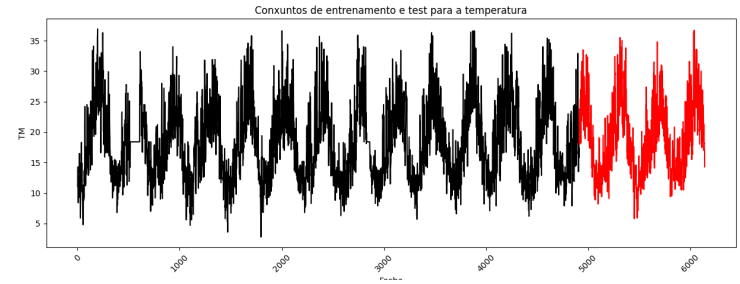
A representación gráfica dos conxuntos de datos de entrenameto e test é a seguinte:



(a) *Entrenamento e test da irradiación solar*



(b) *Entrenamento e test da presión atmosférica*



(c) *Entrenamento e test da temperatura atmosférica*

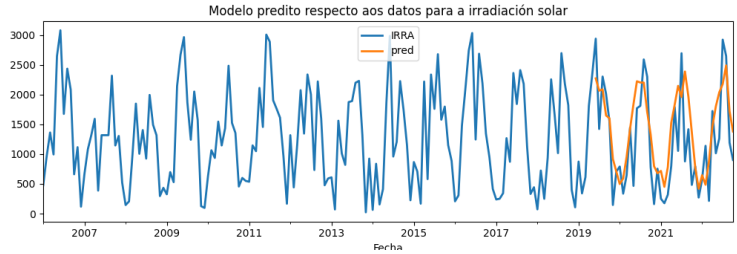
Figura 10: *Entrenamento e test para cada conxunto de datos*

Cos dous conxuntos de datos creados, en primeiro lugar defínese un obxecto da clase `ForecasterAutoreg` para crear o prognosticador. Neste caso, defínese un regresor de tipo `Ridge`, e establécese o valor de `lags` para que empregue todos os datos anteriores.
Despois de definir este obxecto, entrénase o prognosticador cos datos de entrenamento, para continuar relaizando o proceso de *backtesting*, no que se avalía o comportamento do modelo predictivo. Neste proceso tamén se realiza a predición do conxunto de test:

```
forecaster = ForecasterAutoreg(
        regressor=Ridge(),
```

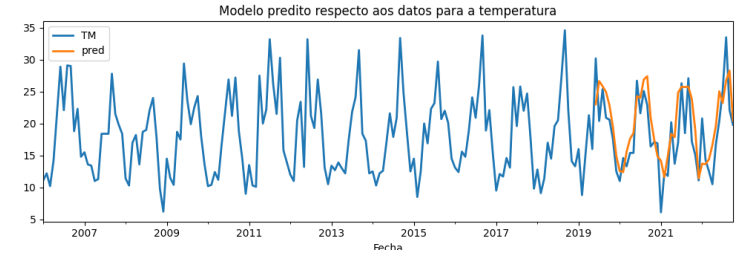
```
        lags=150,
        transformer_y=StandardScaler()
    )
    forecaster.fit(y=data_train[c2].ffill())

    _, predicciones = backtesting_forecaster(
        forecaster=forecaster,
        y=data[c2].ffill(),initial_train_size = len(data_train),
                           fixed_train_size   = False,
                           steps      = 24,
                           metric     = 'mean_absolute_error',
                           refit      = False,
                           verbose    = False)
```

As distintas predicción que se obteñen son as seguintes:



(a) *Entrenamento e test da irradiación solar*

(b) *Entrenamento e test da presión atmosférica*



(c) *Entrenamento e test da temperatura atmosférica*

Figura 11: *Entrenamento e test para cada conxunto de datos*

Os erros absolutos obtidos para cada caso son os seguintes:

- Irradiación solar: 490.65198854504644
- Presión atmosférica: 4.28202712730119
- Temperatura: 3.3822717772778184

Tendo en conta a escala de valores de cada conxunto de datos, os erros obtidos son bastante baixos, polo que os modelos de predición adáptanse adecuadamente aos conxuntos de datos.