Banikoara_eda

July 21, 2025

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import yaml
import os
import tqdm
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler, StandardScaler
```

[2]:
```
sns.set_theme(style="white", context="talk", palette="muted")
plt.rcParams.update({
    'font.family': 'serif',
    'font.size': 14,
    'axes.titlesize': 15,
    'axes.labelsize': 13,
    'legend.fontsize': 11,
    'xtick.labelsize': 10,
    'ytick.labelsize': 10,
    'figure.dpi': 100
})
```

[3]:
```
# Read config.yaml
with open('../configs/config_banikoara.yaml', 'r') as config_file:
    config = yaml.safe_load(config_file)

# extract data params
data_params = config['data_params']
```

0.0.1 Read the csv file

[4]:
```
dataset_path = data_params['data_path'] + 'raw/' + data_params['dataset']
data = pd.read_csv(dataset_path)
data.head()
```

[4]:
```
        DATE     PS    T2M   RH2M    WD2M  WS2M  GWETPROF  CLOUD_AMT  \
0  31/1/1981  98.05  22.05  34.62   58.69  2.73      0.55        NaN
1  28/2/1981  97.85  26.33  24.62   73.88  2.71      0.52        NaN
2  31/3/1981  97.75  30.48  34.56  105.81  2.16      0.51        NaN
```

```
3  30/4/1981  97.66  30.79  50.38  215.50  2.62      0.50        NaN
4  31/5/1981  97.85  28.11  71.88  233.12  2.42      0.52        NaN

   TOA_SW_DWN  PRECTOTCORR_SUM  ALLSKY_SFC_SW_DWN  SPI6  Moving_Sum_6
0         NaN             0.00                NaN   NaN           NaN
1         NaN             0.00                NaN   NaN           NaN
2         NaN             5.27                NaN   NaN           NaN
3         NaN            42.19                NaN   NaN           NaN
4         NaN            94.92                NaN   NaN           NaN
```

[5]:
```
data.describe()
```

[5]:
```
               PS         T2M        RH2M        WD2M        WS2M    GWETPROF  \
count  492.000000  492.000000  492.000000  492.000000  492.000000  492.000000
mean    97.903963   27.561707   53.031992  155.935081    2.254614    0.533354
std      0.147023    2.589921   23.243314   72.824601    0.552562    0.052559
min     97.460000   21.070000   12.310000   24.440000    1.190000    0.480000
25%     97.807500   25.795000   31.560000   73.120000    1.787500    0.490000
50%     97.940000   26.870000   54.095000  197.095000    2.290000    0.520000
75%     98.010000   29.370000   75.327500  220.955000    2.652500    0.560000
max     98.350000   34.230000   86.620000  345.810000    3.820000    0.750000

        CLOUD_AMT  TOA_SW_DWN  PRECTOTCORR_SUM  ALLSKY_SFC_SW_DWN  \
count  456.000000  456.000000       492.000000         456.000000
mean    49.743224   35.247237        66.271524          20.531031
std     19.380463    2.661197        80.382083           1.601459
min     10.450000   30.230000         0.000000          15.840000
25%     31.760000   33.455000         0.000000          19.380000
50%     55.355000   36.680000        26.370000          20.645000
75%     65.800000   37.357500       126.560000          21.635000
max     84.090000   37.880000       326.950000          25.260000

               SPI6  Moving_Sum_6
count  4.870000e+02    487.000000
mean   7.842232e-17    399.859692
std    1.000000e+00    276.216636
min   -1.447631e+00      0.000000
25%   -9.703604e-01    131.830000
50%   -1.574739e-02    395.510000
75%    9.197321e-01    653.905000
max    2.275244e+00   1028.320000
```

[6]:
```
dataset_path
```

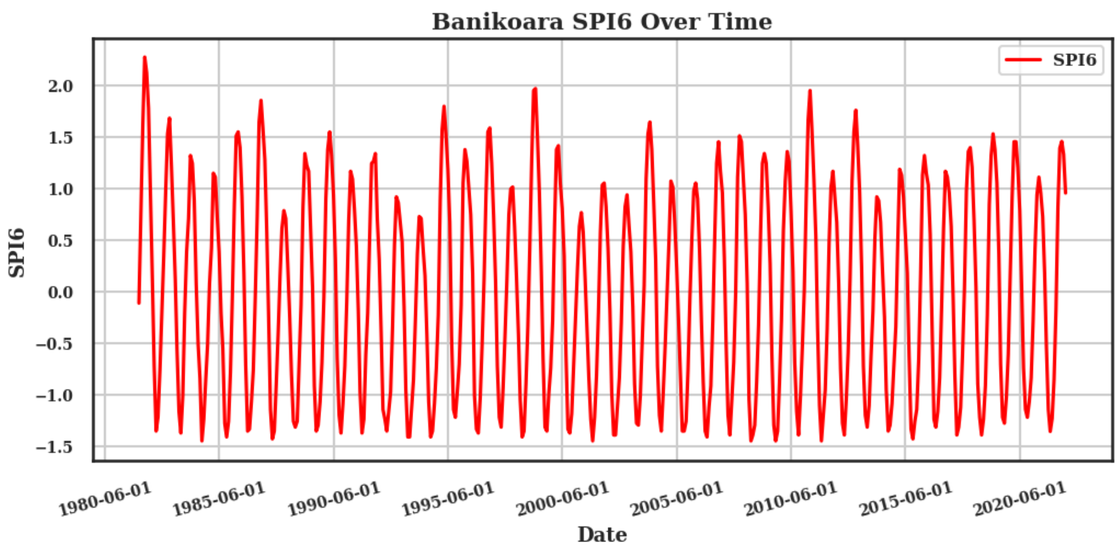
[6]: '../datasets/raw/Banikoara_with_SPI6.csv'

[7]:
```
# Ensure the date column is a datetime
data['DATE'] = pd.to_datetime(data['DATE'], dayfirst=True)

# Set the date column as the index and drop it from the columns
data = data.set_index('DATE')

data.head()
```

[7]:
```
               PS    T2M   RH2M    WD2M  WS2M  GWETPROF  CLOUD_AMT  \
DATE
1981-01-31  98.05  22.05  34.62   58.69  2.73      0.55        NaN
1981-02-28  97.85  26.33  24.62   73.88  2.71      0.52        NaN
1981-03-31  97.75  30.48  34.56  105.81  2.16      0.51        NaN
1981-04-30  97.66  30.79  50.38  215.50  2.62      0.50        NaN
1981-05-31  97.85  28.11  71.88  233.12  2.42      0.52        NaN

            TOA_SW_DWN  PRECTOTCORR_SUM  ALLSKY_SFC_SW_DWN  SPI6  Moving_Sum_6
DATE
1981-01-31         NaN             0.00                NaN   NaN           NaN
1981-02-28         NaN             0.00                NaN   NaN           NaN
1981-03-31         NaN             5.27                NaN   NaN           NaN
1981-04-30         NaN            42.19                NaN   NaN           NaN
1981-05-31         NaN            94.92                NaN   NaN           NaN
```

[8]:
```
import matplotlib.dates as mdates

# Plot SPI6 in function of Date
plt.figure(figsize=(12, 5))
plt.plot(data.index, data['SPI6'], label='SPI6', color='red')

# Adding grid
plt.grid(True)

# Adding Title
plt.title(data_params['city'] + ' SPI6 Over Time', fontweight='bold')

# Annotating x and y axis
plt.xlabel('Date', fontweight='bold')
plt.ylabel('SPI6', fontweight='bold')

# Rotate dates on x-axis and bold the ticks
plt.xticks(rotation=15, fontweight='bold')
plt.yticks(fontweight='bold')

# Adding legend with bold font
plt.legend(prop={'weight': 'bold'}, fontsize='large')
```

```python
# Set the major locator for x-axis to show every 5th year in June
five_years = mdates.YearLocator(5)  # every 5 years
june_locator = mdates.MonthLocator(6)  # Add ticks only for June
plt.gca().xaxis.set_major_locator(five_years)
plt.gca().xaxis.set_minor_locator(june_locator)
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))

# Adjust the format to only show June 1st for every 5 years
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y-06-01'))

# Saving the figure before showing
plt.savefig(data_params['save_path'] + data_params['city'] + '/spi6_plot.png')

# Show the plot
plt.show()
```



[9]:
```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 492 entries, 1981-01-31 to 2021-12-31
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   PS               492 non-null    float64
 1   T2M              492 non-null    float64
 2   RH2M             492 non-null    float64
 3   WD2M             492 non-null    float64
 4   WS2M             492 non-null    float64
 5   GWETPROF         492 non-null    float64
```

```
 6   CLOUD_AMT          456 non-null    float64
 7   TOA_SW_DWN         456 non-null    float64
 8   PRECTOTCORR_SUM    492 non-null    float64
 9   ALLSKY_SFC_SW_DWN  456 non-null    float64
 10  SPI6               487 non-null    float64
 11  Moving_Sum_6       487 non-null    float64
dtypes: float64(12)
memory usage: 50.0 KB
```

[10]:
```
data.describe()
```

[10]:
```
               PS         T2M        RH2M        WD2M        WS2M    GWETPROF  \
count  492.000000  492.000000  492.000000  492.000000  492.000000  492.000000
mean    97.903963   27.561707   53.031992  155.935081    2.254614    0.533354
std      0.147023    2.589921   23.243314   72.824601    0.552562    0.052559
min     97.460000   21.070000   12.310000   24.440000    1.190000    0.480000
25%     97.807500   25.795000   31.560000   73.120000    1.787500    0.490000
50%     97.940000   26.870000   54.095000  197.095000    2.290000    0.520000
75%     98.010000   29.370000   75.327500  220.955000    2.652500    0.560000
max     98.350000   34.230000   86.620000  345.810000    3.820000    0.750000

        CLOUD_AMT  TOA_SW_DWN  PRECTOTCORR_SUM  ALLSKY_SFC_SW_DWN  \
count  456.000000  456.000000       492.000000         456.000000
mean    49.743224   35.247237        66.271524          20.531031
std     19.380463    2.661197        80.382083           1.601459
min     10.450000   30.230000         0.000000          15.840000
25%     31.760000   33.455000         0.000000          19.380000
50%     55.355000   36.680000        26.370000          20.645000
75%     65.800000   37.357500       126.560000          21.635000
max     84.090000   37.880000       326.950000          25.260000

               SPI6  Moving_Sum_6
count  4.870000e+02    487.000000
mean   7.842232e-17    399.859692
std    1.000000e+00    276.216636
min   -1.447631e+00      0.000000
25%   -9.703604e-01    131.830000
50%   -1.574739e-02    395.510000
75%    9.197321e-01    653.905000
max    2.275244e+00   1028.320000
```

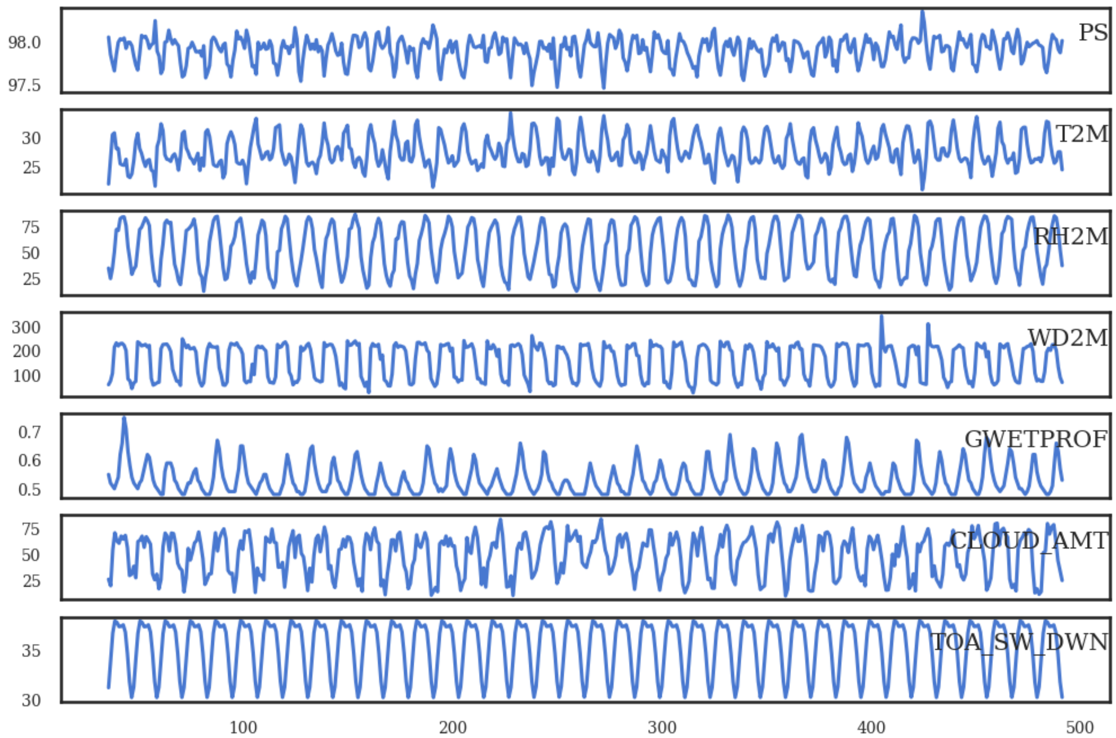
[11]:
```
data.isnull().sum()
```

[11]:
```
PS                   0
T2M                  0
RH2M                 0
WD2M                 0
WS2M                 0
```

```
GWETPROF             0
CLOUD_AMT           36
TOA_SW_DWN          36
PRECTOTCORR_SUM      0
ALLSKY_SFC_SW_DWN   36
SPI6                 5
Moving_Sum_6         5
dtype: int64
```

[12]:

```python
# specify columns to plot
groups = [0, 1, 2, 3, 5, 6, 7]
i = 1
# plot each column
plt.figure(figsize=(12,8))
for group in groups:
 plt.subplot(len(groups), 1, i)
 plt.plot(data.values[:, group])
 plt.title(data.columns[group], y=0.5, loc='right')
 i += 1
plt.show()
```



Normalization The scale of the features are big comparatively to the target SPI6. Let's normalize those features

[13]:
```python
feature_range = (-1, 1)

# Separate the target column from the other columns
features = data.drop(columns=['SPI6'])
target = data['SPI6']

# Initialize and fit scaler on train data if needed
scaler_type = data_params['scaling_type']
if scaler_type == 'minmax':
    scaler = MinMaxScaler(feature_range=feature_range)
elif scaler_type == 'standard':
    scaler = StandardScaler()
elif scaler_type == 'none':
    scaler = None
else:
    raise ValueError(f"Unsupported scaler_type: {scaler_type}")

normalized_features = scaler.fit_transform(features)

# Convert the normalized features back to a DataFrame
normalized_features_df = pd.DataFrame(normalized_features, columns=features.
 ↪columns, index=data.index)

# Reconstruct the DataFrame with the normalized columns and the target column
normalized_data = pd.concat([normalized_features_df, target], axis=1)

# Display the head of the new DataFrame
normalized_data.head()
```

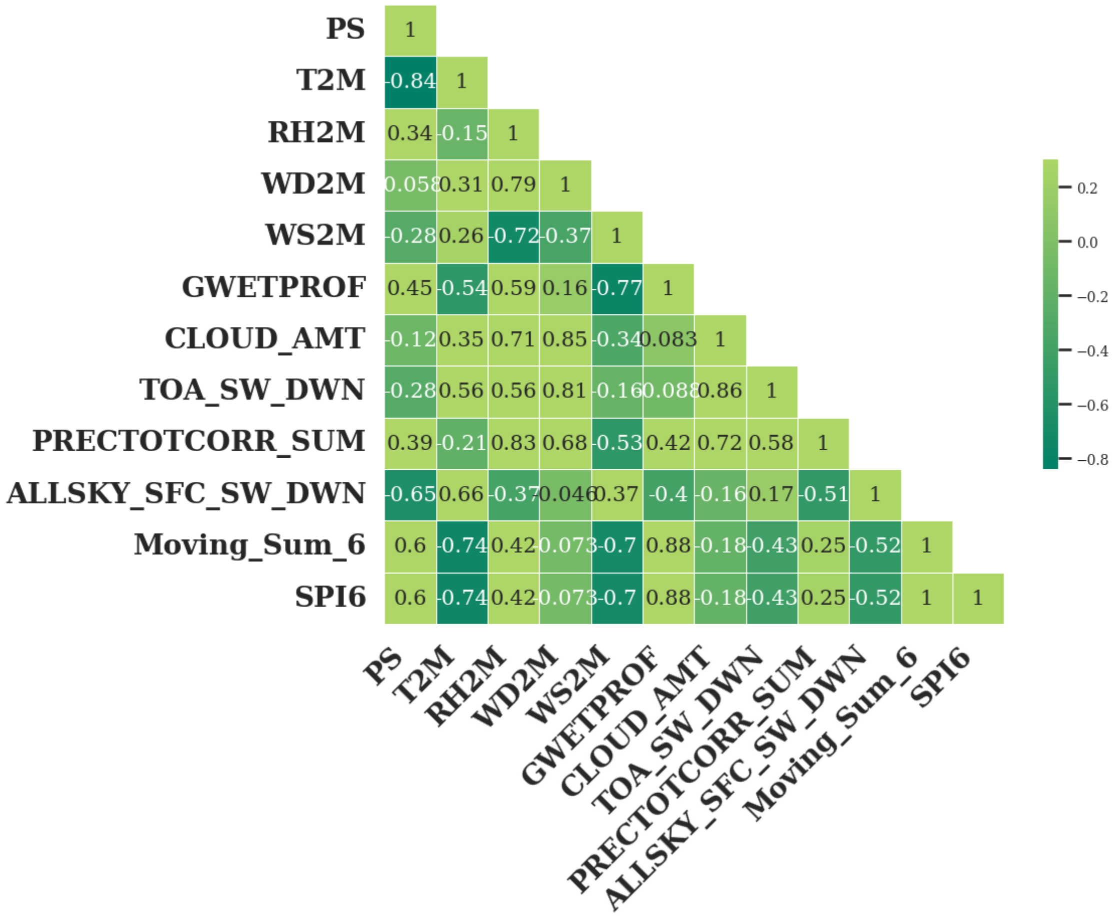
[13]:
```
                  PS       T2M      RH2M      WD2M      WS2M  GWETPROF  \
DATE
1981-01-31  0.325843 -0.851064 -0.399542 -0.786850  0.171103 -0.481481
1981-02-28 -0.123596 -0.200608 -0.668685 -0.692317  0.155894 -0.703704
1981-03-31 -0.348315  0.430091 -0.401157 -0.493606 -0.262357 -0.777778
1981-04-30 -0.550562  0.477204  0.024627  0.189034  0.087452 -0.851852
1981-05-31 -0.123596  0.069909  0.603284  0.298690 -0.064639 -0.703704

            CLOUD_AMT  TOA_SW_DWN  PRECTOTCORR_SUM  ALLSKY_SFC_SW_DWN  \
DATE
1981-01-31        NaN         NaN        -1.000000                NaN
1981-02-28        NaN         NaN        -1.000000                NaN
1981-03-31        NaN         NaN        -0.967763                NaN
1981-04-30        NaN         NaN        -0.741918                NaN
1981-05-31        NaN         NaN        -0.419361                NaN

            Moving_Sum_6  SPI6
DATE
```

```
1981-01-31          NaN    NaN
1981-02-28          NaN    NaN
1981-03-31          NaN    NaN
1981-04-30          NaN    NaN
1981-05-31          NaN    NaN
```

0.0.2 Pearson Correlation

[14]:

```
# Compute the correlation matrix
corr = normalized_data.corr()

# Generate a mask for the upper triangle
mask = np.triu(np.ones_like(corr, dtype=bool), k=1)

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(10, 8))
# plt.title('Heatmap of correlation between variables',fontsize=16)

# Generate a custom diverging colormap
# cmap = sns.diverging_palette(230, 20, as_cmap=True)
cmap = 'summer'

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask,annot=True, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5},
 ↪annot_kws={"size":15})
ax.set_xticklabels(ax.get_xticklabels(), rotation=45,
 ↪horizontalalignment='right',fontsize=20, fontweight='bold')
ax.set_yticklabels(ax.get_yticklabels(),fontsize=20, fontweight='bold')
#plt.savefig('banikoara_heatmap_correlation')
plt.show()
```



Let's focus on the corelations between each variable and target variable

[15]:
```
target_column = 'SPI6'
```

[16]:
```
correlations_data = normalized_data.corr()[target_column].
↪sort_values(ascending=False)
correlations_data
```

[16]:
```
SPI6                 1.000000
Moving_Sum_6         1.000000
GWETPROF             0.882607
PS                   0.598602
RH2M                 0.423335
PRECTOTCORR_SUM      0.254686
WD2M                -0.072987
CLOUD_AMT           -0.178796
TOA_SW_DWN          -0.425151
ALLSKY_SFC_SW_DWN   -0.515810
WS2M                -0.696878
```

```
T2M                  -0.737511
Name: SPI6, dtype: float64
```

[17]:
```
correlations_data = np.abs(normalized_data.corr()[target_column]).
    ↪sort_values(ascending=False)
correlations_data
```

[17]:
```
SPI6                  1.000000
Moving_Sum_6          1.000000
GWETPROF              0.882607
T2M                   0.737511
WS2M                  0.696878
PS                    0.598602
ALLSKY_SFC_SW_DWN     0.515810
TOA_SW_DWN            0.425151
RH2M                  0.423335
PRECTOTCORR_SUM       0.254686
CLOUD_AMT             0.178796
WD2M                  0.072987
Name: SPI6, dtype: float64
```

We can remove WD2M as the coefficient is very low

[18]:
```
normalized_data = normalized_data.drop(columns=['WD2M'])
normalized_data.head()
```

[18]:
```
                  PS       T2M      RH2M      WS2M  GWETPROF  CLOUD_AMT  \
DATE
1981-01-31  0.325843 -0.851064 -0.399542  0.171103 -0.481481        NaN
1981-02-28 -0.123596 -0.200608 -0.668685  0.155894 -0.703704        NaN
1981-03-31 -0.348315  0.430091 -0.401157 -0.262357 -0.777778        NaN
1981-04-30 -0.550562  0.477204  0.024627  0.087452 -0.851852        NaN
1981-05-31 -0.123596  0.069909  0.603284 -0.064639 -0.703704        NaN

            TOA_SW_DWN  PRECTOTCORR_SUM  ALLSKY_SFC_SW_DWN  Moving_Sum_6  SPI6
DATE
1981-01-31         NaN        -1.000000                NaN           NaN   NaN
1981-02-28         NaN        -1.000000                NaN           NaN   NaN
1981-03-31         NaN        -0.967763                NaN           NaN   NaN
1981-04-30         NaN        -0.741918                NaN           NaN   NaN
1981-05-31         NaN        -0.419361                NaN           NaN   NaN
```

[19]:
```
no_lagged_data_path = data_params['data_path'] + 'no_lagged/' +␣
    ↪data_params['city'] + '_no_lagged.csv'
normalized_data.to_csv(no_lagged_data_path)
```

0.0.3 Transform dataset for Time series forecasting

[20]:
```python
import sys
sys.path.append('../')

from models.utils import create_lagged_features
```

```
2025-07-20 23:49:15.818812: I tensorflow/core/util/port.cc:113] oneDNN custom
operations are on. You may see slightly different numerical results due to
floating-point round-off errors from different computation orders. To turn them
off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2025-07-20 23:49:15.998054: E
external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:479] Unable to register
cuFFT factory: Attempting to register factory for plugin cuFFT when one has
already been registered
2025-07-20 23:49:16.119375: E
external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:10575] Unable to
register cuDNN factory: Attempting to register factory for plugin cuDNN when one
has already been registered
2025-07-20 23:49:16.120251: E
external/local_xla/xla/stream_executor/cuda/cuda_blas.cc:1442] Unable to
register cuBLAS factory: Attempting to register factory for plugin cuBLAS when
one has already been registered
2025-07-20 23:49:16.266488: I tensorflow/core/platform/cpu_feature_guard.cc:210]
This TensorFlow binary is optimized to use available CPU instructions in
performance-critical operations.
To enable the following instructions: AVX2 AVX_VNNI FMA, in other operations,
rebuild TensorFlow with the appropriate compiler flags.
2025-07-20 23:49:18.744848: W
tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not
find TensorRT
```

[21]:
```python
col_names = list(normalized_data.columns)
print(col_names)
```

```
['PS', 'T2M', 'RH2M', 'WS2M', 'GWETPROF', 'CLOUD_AMT', 'TOA_SW_DWN',
'PRECTOTCORR_SUM', 'ALLSKY_SFC_SW_DWN', 'Moving_Sum_6', 'SPI6']
```

[22]:
```python
lagged_data = create_lagged_features(data= normalized_data,
                                     col_names=col_names,
                                     n_in=data_params['window_size'],
                                     n_out=data_params['n_output_steps'],
                                     dropnan=True)


lagged_data.head()
```

[22]:
```
             PS(t-5)  T2M(t-5)  RH2M(t-5)  WS2M(t-5)  GWETPROF(t-5)  \
DATE
1984-06-30  0.101124 -0.490881  -0.728973   0.201521      -0.851852
```

```
1984-07-31 -0.348315 -0.130699  -0.826672   0.277567      -0.925926
1984-08-31 -0.707865  0.462006  -0.416229   0.148289      -1.000000
1984-09-30 -0.662921  0.516717   0.029471   0.452471      -1.000000
1984-10-31 -0.415730  0.083587   0.576369   0.163498      -0.851852

            CLOUD_AMT(t-5)  TOA_SW_DWN(t-5)  PRECTOTCORR_SUM(t-5)  \
DATE
1984-06-30       -0.564910        -0.743791             -1.000000
1984-07-31       -0.734383        -0.014379             -1.000000
1984-08-31        0.198805         0.675817             -1.000000
1984-09-30        0.642042         0.989542             -0.645145
1984-10-31        0.431287         0.952941             -0.354825

            ALLSKY_SFC_SW_DWN(t-5)  Moving_Sum_6(t-5)  …    T2M(t)  \
DATE                                                    …
1984-06-30                0.144374          -0.497433  … -0.025836
1984-07-31                0.131635          -0.671795  … -0.167173
1984-08-31                0.581741          -1.000000  … -0.215805
1984-09-30                0.588110          -0.887175  … -0.325228
1984-10-31                0.507431          -0.682044  … -0.189970

             RH2M(t)   WS2M(t)  GWETPROF(t)  CLOUD_AMT(t)  TOA_SW_DWN(t)  \
DATE
1984-06-30  0.611627 -0.125475    -0.703704      0.369636       0.843137
1984-07-31  0.667070 -0.346008    -0.703704      0.557577       0.850980
1984-08-31  0.742700 -0.536122    -0.555556      0.477729       0.890196
1984-09-30  0.875656 -0.543726    -0.407407      0.575502       0.699346
1984-10-31  0.593056 -0.634981    -0.333333      0.257740       0.139869

            PRECTOTCORR_SUM(t)  ALLSKY_SFC_SW_DWN(t)  Moving_Sum_6(t)  \
DATE
1984-06-30           -0.419361              0.259023        -0.497433
1984-07-31            0.000031             -0.046709        -0.179477
1984-08-31           -0.419361              0.059448         0.005135
1984-09-30            0.225814             -0.125265         0.394877
1984-10-31           -0.709680             -0.016985         0.374358

             SPI6(t)
DATE
1984-06-30 -0.512133
1984-07-31  0.079721
1984-08-31  0.423364
1984-09-30  1.148846
1984-10-31  1.110651

[5 rows x 66 columns]
```

0.0.4 Now we have 42 columns. Lets reduce them

0.0.5 Select the most useful lags.

Let's plot correlation matrice by including for each the target TWS and others with a given lag_lenth

[23]:
```
normalized_data.columns
```

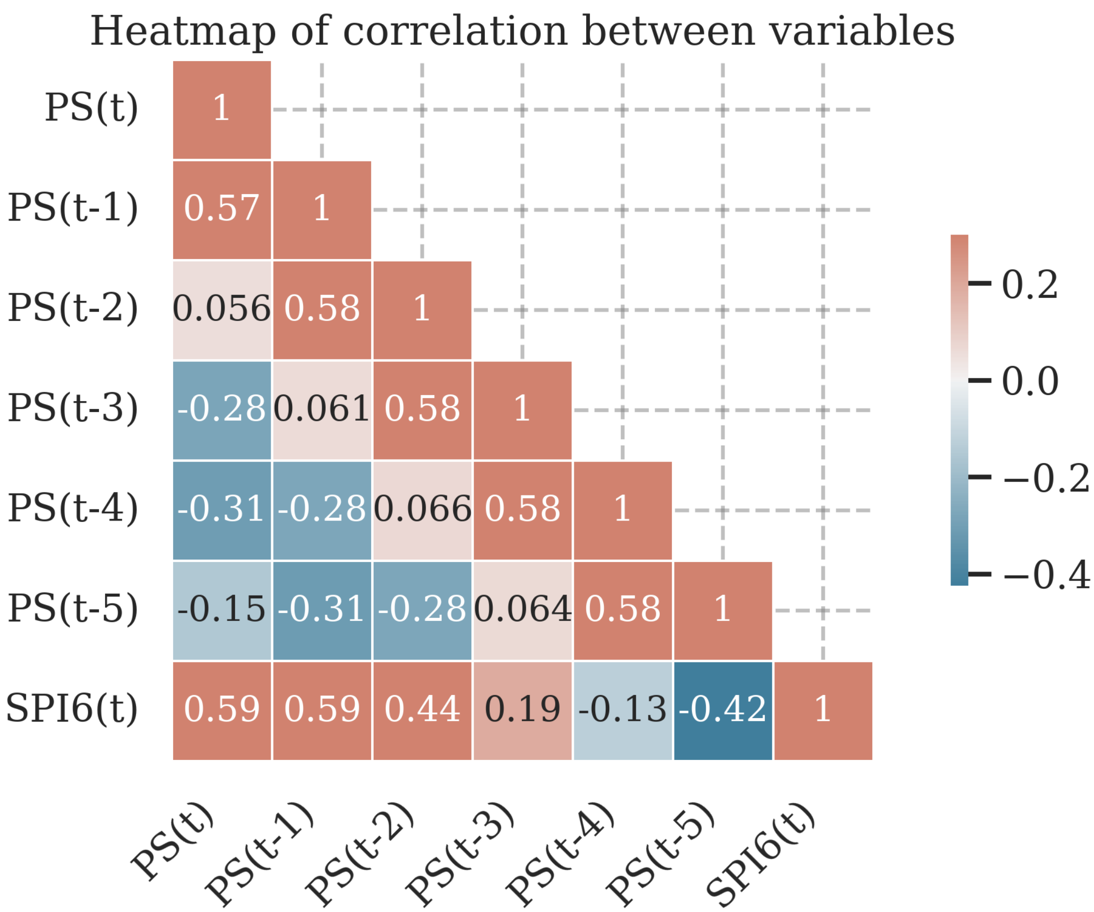
[23]:
```
Index(['PS', 'T2M', 'RH2M', 'WS2M', 'GWETPROF', 'CLOUD_AMT', 'TOA_SW_DWN',
       'PRECTOTCORR_SUM', 'ALLSKY_SFC_SW_DWN', 'Moving_Sum_6', 'SPI6'],
      dtype='object')
```

[24]:
```
lagged_data.columns
```

[24]:
```
Index(['PS(t-5)', 'T2M(t-5)', 'RH2M(t-5)', 'WS2M(t-5)', 'GWETPROF(t-5)',
       'CLOUD_AMT(t-5)', 'TOA_SW_DWN(t-5)', 'PRECTOTCORR_SUM(t-5)',
       'ALLSKY_SFC_SW_DWN(t-5)', 'Moving_Sum_6(t-5)', 'SPI6(t-5)', 'PS(t-4)',
       'T2M(t-4)', 'RH2M(t-4)', 'WS2M(t-4)', 'GWETPROF(t-4)', 'CLOUD_AMT(t-4)',
       'TOA_SW_DWN(t-4)', 'PRECTOTCORR_SUM(t-4)', 'ALLSKY_SFC_SW_DWN(t-4)',
       'Moving_Sum_6(t-4)', 'SPI6(t-4)', 'PS(t-3)', 'T2M(t-3)', 'RH2M(t-3)',
       'WS2M(t-3)', 'GWETPROF(t-3)', 'CLOUD_AMT(t-3)', 'TOA_SW_DWN(t-3)',
       'PRECTOTCORR_SUM(t-3)', 'ALLSKY_SFC_SW_DWN(t-3)', 'Moving_Sum_6(t-3)',
       'SPI6(t-3)', 'PS(t-2)', 'T2M(t-2)', 'RH2M(t-2)', 'WS2M(t-2)',
       'GWETPROF(t-2)', 'CLOUD_AMT(t-2)', 'TOA_SW_DWN(t-2)',
       'PRECTOTCORR_SUM(t-2)', 'ALLSKY_SFC_SW_DWN(t-2)', 'Moving_Sum_6(t-2)',
       'SPI6(t-2)', 'PS(t-1)', 'T2M(t-1)', 'RH2M(t-1)', 'WS2M(t-1)',
       'GWETPROF(t-1)', 'CLOUD_AMT(t-1)', 'TOA_SW_DWN(t-1)',
       'PRECTOTCORR_SUM(t-1)', 'ALLSKY_SFC_SW_DWN(t-1)', 'Moving_Sum_6(t-1)',
       'SPI6(t-1)', 'PS(t)', 'T2M(t)', 'RH2M(t)', 'WS2M(t)', 'GWETPROF(t)',
       'CLOUD_AMT(t)', 'TOA_SW_DWN(t)', 'PRECTOTCORR_SUM(t)',
       'ALLSKY_SFC_SW_DWN(t)', 'Moving_Sum_6(t)', 'SPI6(t)'],
      dtype='object')
```

[25]:
```
# Compute the correlation matrix : SPI6 and lag PS

dfm1 = lagged_data[['PS(t)','PS(t-1)', 'PS(t-2)', 'PS(t-3)','PS(t-4)',␣
 ↪'PS(t-5)','SPI6(t)']]

corr = dfm1.corr()

# Generate a mask for the upper triangle
mask = np.triu(np.ones_like(corr, dtype=bool), k=1)

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(11, 5))
plt.title('Heatmap of correlation between variables',fontsize=16)

# Generate a custom diverging colormap
```

```
cmap = sns.diverging_palette(230, 20, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask,annot=True, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
ax.set_xticklabels(ax.get_xticklabels(), rotation=45,␣
 ↪horizontalalignment='right')
plt.show()
```



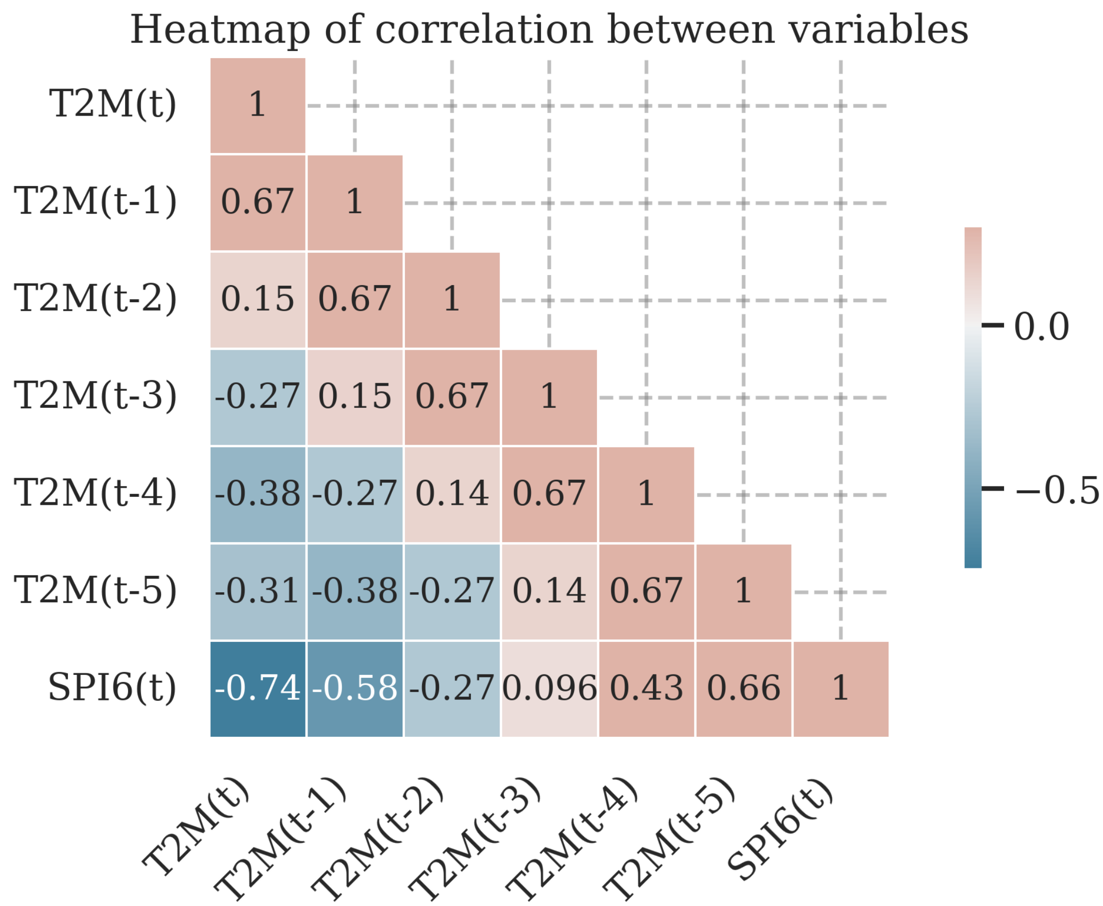
[26]:
```
np.abs(dfm1.corr()['SPI6(t)']).sort_values(ascending=False)
```

[26]:
```
SPI6(t)     1.000000
PS(t)       0.589333
PS(t-1)     0.586479
PS(t-2)     0.441077
PS(t-5)     0.424336
PS(t-3)     0.187110
PS(t-4)     0.129779
```

```
Name: SPI6(t), dtype: float64
```

le lag 0 : PS(t) est le plus correllé avec un coef de 0.59

[27]:
```python
# Compute the correlation matrix : SPI6 and lag T2M

dfm2 = lagged_data[['T2M(t)','T2M(t-1)', 'T2M(t-2)', 'T2M(t-3)','T2M(t-4)',␣
 ↪'T2M(t-5)','SPI6(t)']]

corr = dfm2.corr()

# Generate a mask for the upper triangle
mask = np.triu(np.ones_like(corr, dtype=bool), k=1)

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(11, 5))
plt.title('Heatmap of correlation between variables',fontsize=16)

# Generate a custom diverging colormap
cmap = sns.diverging_palette(230, 20, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask,annot=True, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
ax.set_xticklabels(ax.get_xticklabels(), rotation=45,␣
 ↪horizontalalignment='right')
plt.show()
```



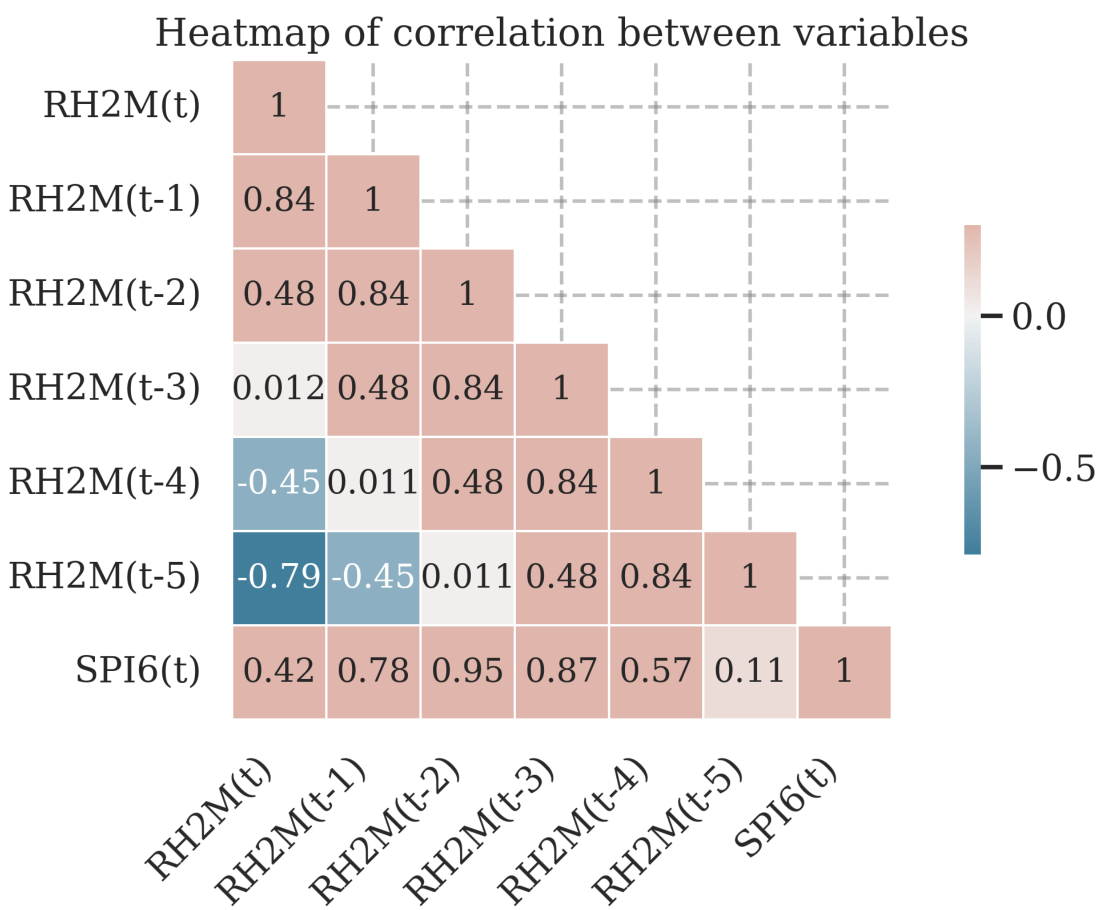
[28]:
```
np.abs(dfm2.corr()['SPI6(t)']).sort_values(ascending=False)
```

[28]:
```
SPI6(t)      1.000000
T2M(t)       0.742238
T2M(t-5)     0.664737
T2M(t-1)     0.578521
T2M(t-4)     0.432413
T2M(t-2)     0.270325
T2M(t-3)     0.095851
Name: SPI6(t), dtype: float64
```

le lag 0 T2M(t) est le plus correlé avec 0.73

[29]:
```
# Compute the correlation matrix : SPI6 and lag RH2M

dfm3 = lagged_data[['RH2M(t)','RH2M(t-1)', 'RH2M(t-2)',␣
↪'RH2M(t-3)','RH2M(t-4)', 'RH2M(t-5)','SPI6(t)']]

corr = dfm3.corr()
```

```python
# Generate a mask for the upper triangle
mask = np.triu(np.ones_like(corr, dtype=bool), k=1)

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(11, 5))
plt.title('Heatmap of correlation between variables',fontsize=16)

# Generate a custom diverging colormap
cmap = sns.diverging_palette(230, 20, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask,annot=True, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
ax.set_xticklabels(ax.get_xticklabels(), rotation=45, horizontalalignment='right')
plt.show()
```



[30]:
```
np.abs(dfm3.corr()['SPI6(t)']).sort_values(ascending=False)
```

[30]:
```
SPI6(t)      1.000000
RH2M(t-2)    0.947782
RH2M(t-3)    0.874279
RH2M(t-1)    0.781602
RH2M(t-4)    0.569526
RH2M(t)      0.415573
RH2M(t-5)    0.108428
Name: SPI6(t), dtype: float64
```

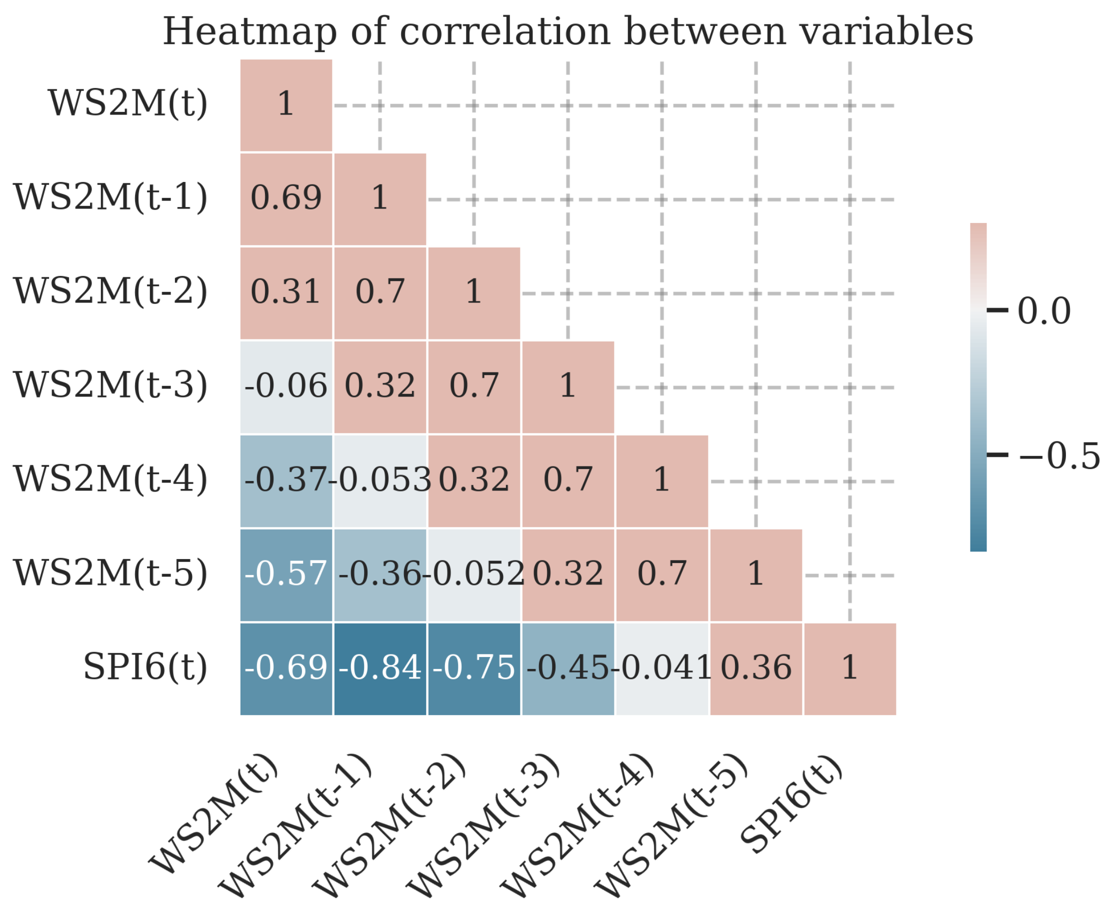
le lag 2 RH2M(t-2) est le plus correlé avec un coefficient de 0.948855

[31]:
```
lagged_data.columns
```

[31]:
```
Index(['PS(t-5)', 'T2M(t-5)', 'RH2M(t-5)', 'WS2M(t-5)', 'GWETPROF(t-5)',
       'CLOUD_AMT(t-5)', 'TOA_SW_DWN(t-5)', 'PRECTOTCORR_SUM(t-5)',
       'ALLSKY_SFC_SW_DWN(t-5)', 'Moving_Sum_6(t-5)', 'SPI6(t-5)', 'PS(t-4)',
       'T2M(t-4)', 'RH2M(t-4)', 'WS2M(t-4)', 'GWETPROF(t-4)', 'CLOUD_AMT(t-4)',
       'TOA_SW_DWN(t-4)', 'PRECTOTCORR_SUM(t-4)', 'ALLSKY_SFC_SW_DWN(t-4)',
       'Moving_Sum_6(t-4)', 'SPI6(t-4)', 'PS(t-3)', 'T2M(t-3)', 'RH2M(t-3)',
       'WS2M(t-3)', 'GWETPROF(t-3)', 'CLOUD_AMT(t-3)', 'TOA_SW_DWN(t-3)',
       'PRECTOTCORR_SUM(t-3)', 'ALLSKY_SFC_SW_DWN(t-3)', 'Moving_Sum_6(t-3)',
       'SPI6(t-3)', 'PS(t-2)', 'T2M(t-2)', 'RH2M(t-2)', 'WS2M(t-2)',
       'GWETPROF(t-2)', 'CLOUD_AMT(t-2)', 'TOA_SW_DWN(t-2)',
       'PRECTOTCORR_SUM(t-2)', 'ALLSKY_SFC_SW_DWN(t-2)', 'Moving_Sum_6(t-2)',
       'SPI6(t-2)', 'PS(t-1)', 'T2M(t-1)', 'RH2M(t-1)', 'WS2M(t-1)',
       'GWETPROF(t-1)', 'CLOUD_AMT(t-1)', 'TOA_SW_DWN(t-1)',
       'PRECTOTCORR_SUM(t-1)', 'ALLSKY_SFC_SW_DWN(t-1)', 'Moving_Sum_6(t-1)',
       'SPI6(t-1)', 'PS(t)', 'T2M(t)', 'RH2M(t)', 'WS2M(t)', 'GWETPROF(t)',
       'CLOUD_AMT(t)', 'TOA_SW_DWN(t)', 'PRECTOTCORR_SUM(t)',
       'ALLSKY_SFC_SW_DWN(t)', 'Moving_Sum_6(t)', 'SPI6(t)'],
      dtype='object')
```

[32]:
```
# Compute the correlation matrix : SPI6 and lag WS2M

dfm4 = lagged_data[['WS2M(t)','WS2M(t-1)', 'WS2M(t-2)',␣
 ↪'WS2M(t-3)','WS2M(t-4)', 'WS2M(t-5)','SPI6(t)']]

corr = dfm4.corr()

# Generate a mask for the upper triangle
mask = np.triu(np.ones_like(corr, dtype=bool), k=1)

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(11, 5))
plt.title('Heatmap of correlation between variables',fontsize=16)
```

```python
# Generate a custom diverging colormap
cmap = sns.diverging_palette(230, 20, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask,annot=True, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
ax.set_xticklabels(ax.get_xticklabels(), rotation=45,
 horizontalalignment='right')
plt.show()
```



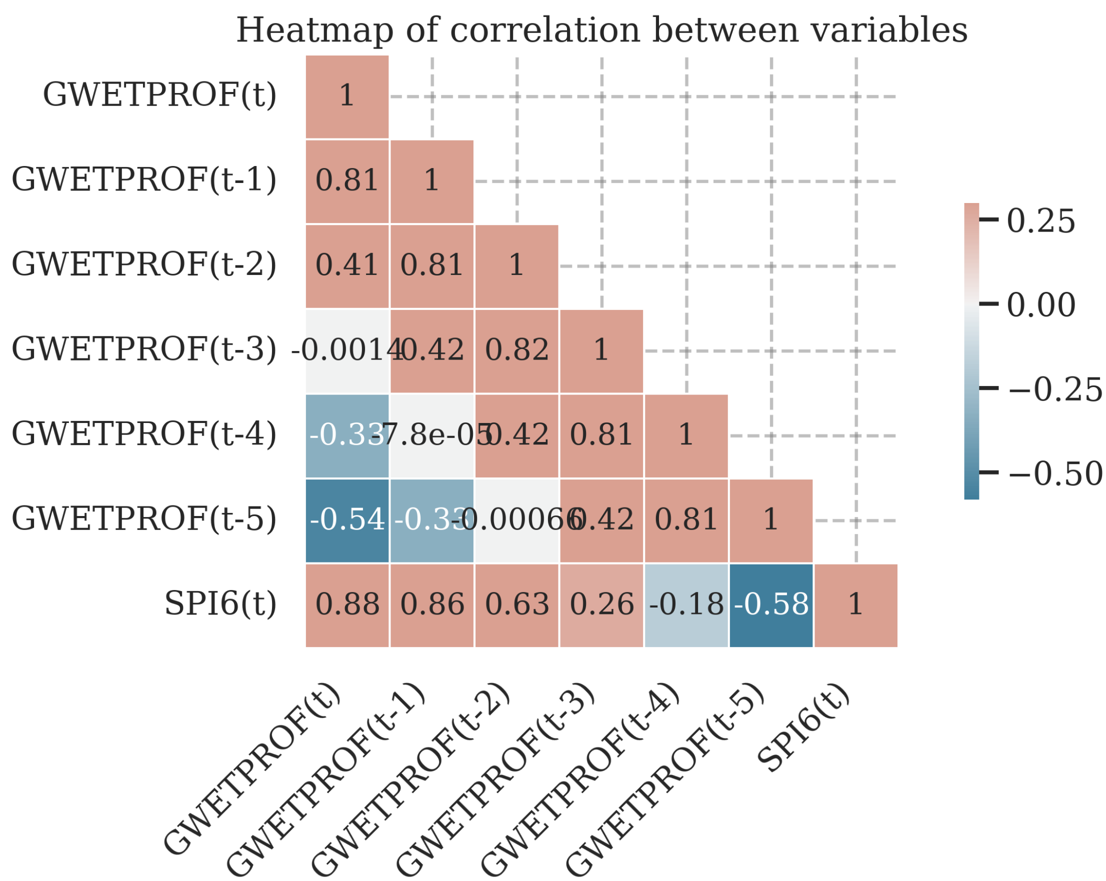
[33]:
```python
np.abs(dfm4.corr()['SPI6(t)']).sort_values(ascending=False)
```

[33]:
```
SPI6(t)      1.000000
WS2M(t-1)    0.835303
WS2M(t-2)    0.747661
WS2M(t)      0.693350
WS2M(t-3)    0.452175
WS2M(t-5)    0.361850
```

```
WS2M(t-4)    0.040524
Name: SPI6(t), dtype: float64
```

le lag 1 WS2M(t-1) est le plus correlé avec SPI6(t) : 0.83

[34]:
```python
# Compute the correlation matrix : SPI6 and lag GWETPROF

dfm5 = lagged_data[['GWETPROF(t)','GWETPROF(t-1)', 'GWETPROF(t-2)',␣
 ↪'GWETPROF(t-3)','GWETPROF(t-4)', 'GWETPROF(t-5)','SPI6(t)']]

corr = dfm5.corr()

# Generate a mask for the upper triangle
mask = np.triu(np.ones_like(corr, dtype=bool), k=1)

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(11, 5))
plt.title('Heatmap of correlation between variables',fontsize=16)

# Generate a custom diverging colormap
cmap = sns.diverging_palette(230, 20, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask,annot=True, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
ax.set_xticklabels(ax.get_xticklabels(), rotation=45,␣
 ↪horizontalalignment='right')
plt.show()
```



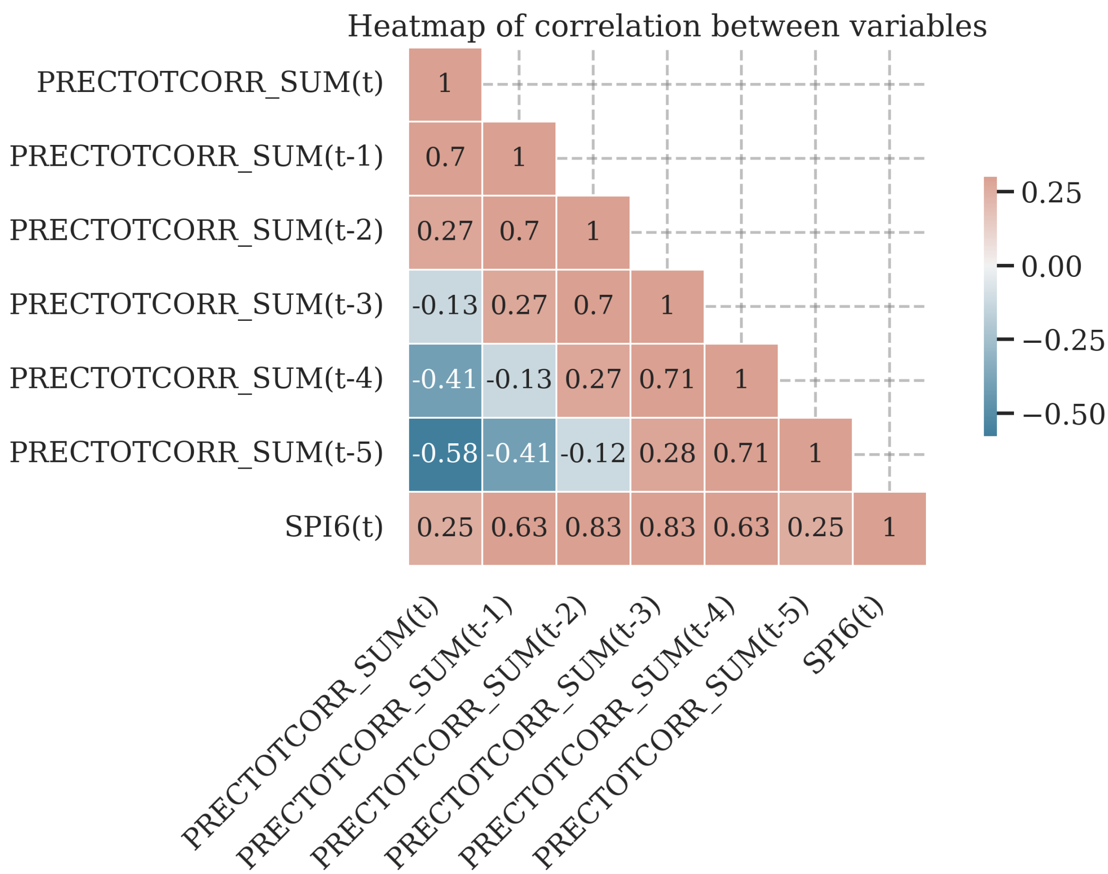
[35]:
```
np.abs(dfm5.corr()['SPI6(t)']).sort_values(ascending=False)
```

[35]:
```
SPI6(t)          1.000000
GWETPROF(t)      0.882901
GWETPROF(t-1)    0.859610
GWETPROF(t-2)    0.629462
GWETPROF(t-5)    0.580412
GWETPROF(t-3)    0.257053
GWETPROF(t-4)    0.183971
Name: SPI6(t), dtype: float64
```

le lag 0 est le plus correlé : GWETPROF(t) 0.883425

[36]:
```
# Compute the correlation matrix : SPI6 and lag PRECTOTCORR_SUM

dfm6 = lagged_data[['PRECTOTCORR_SUM(t)','PRECTOTCORR_SUM(t-1)',␣
 ↪'PRECTOTCORR_SUM(t-2)',
                        'PRECTOTCORR_SUM(t-3)','PRECTOTCORR_SUM(t-4)',␣
 ↪'PRECTOTCORR_SUM(t-5)','SPI6(t)']]
```

```
corr = dfm6.corr()

# Generate a mask for the upper triangle
mask = np.triu(np.ones_like(corr, dtype=bool), k=1)

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(11, 5))
plt.title('Heatmap of correlation between variables',fontsize=16)

# Generate a custom diverging colormap
cmap = sns.diverging_palette(230, 20, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask,annot=True, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
ax.set_xticklabels(ax.get_xticklabels(), rotation=45,␣
 ↪horizontalalignment='right')
plt.show()
```

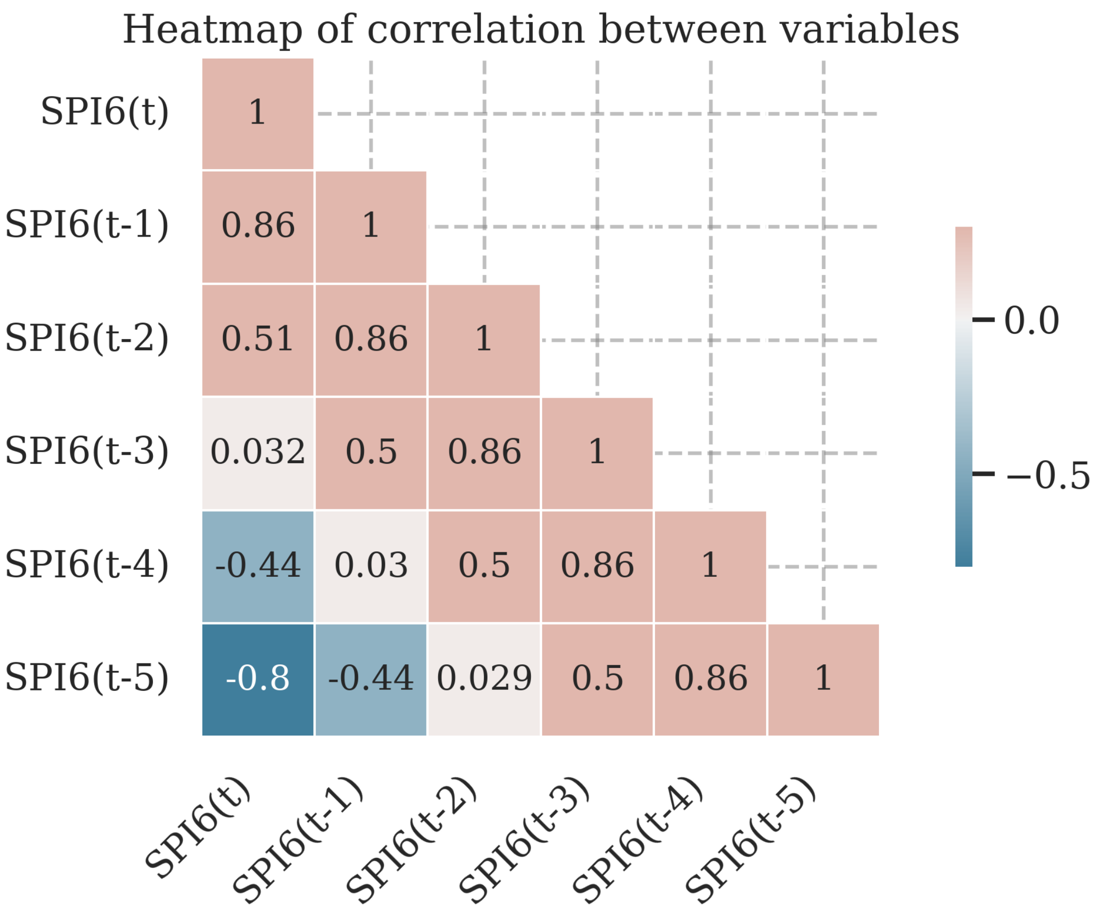


[37]:
```
np.abs(dfm6.corr()['SPI6(t)']).sort_values(ascending=False)
```

[37]:
```
SPI6(t)                   1.000000
PRECTOTCORR_SUM(t-2)      0.828760
PRECTOTCORR_SUM(t-3)      0.828122
PRECTOTCORR_SUM(t-1)      0.627837
PRECTOTCORR_SUM(t-4)      0.626535
PRECTOTCORR_SUM(t)        0.253252
PRECTOTCORR_SUM(t-5)      0.253221
Name: SPI6(t), dtype: float64
```

le lag 3 est le plus correllé PRECTOTCORR_SUM(t-3) 0.830974

[38]:
```
# Compute the correlation matrix : SPI6 and lag SPI6

dfm7 = lagged_data[['SPI6(t)','SPI6(t-1)', 'SPI6(t-2)',
                    'SPI6(t-3)','SPI6(t-4)', 'SPI6(t-5)']]

corr = dfm7.corr()

# Generate a mask for the upper triangle
mask = np.triu(np.ones_like(corr, dtype=bool), k=1)

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(11, 5))
plt.title('Heatmap of correlation between variables',fontsize=16)

# Generate a custom diverging colormap
cmap = sns.diverging_palette(230, 20, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask,annot=True, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
ax.set_xticklabels(ax.get_xticklabels(), rotation=45,␣
 ↪horizontalalignment='right')
plt.show()
```



[39]:
```
np.abs(dfm7.corr()['SPI6(t)']).sort_values(ascending=False)
```

[39]:
```
SPI6(t)      1.000000
SPI6(t-1)    0.861661
SPI6(t-5)    0.801372
SPI6(t-2)    0.505299
SPI6(t-4)    0.442887
SPI6(t-3)    0.031917
Name: SPI6(t), dtype: float64
```

le lag 1 est le plus correllé SPI6(t-1) 0.861646 New dataset by choosing each new lag

[40]:
```
final_lagged_data = lagged_data[['PS(t)', 'T2M(t)', 'RH2M(t-2)','WS2M(t-1)',
                                 'GWETPROF(t)', 'PRECTOTCORR_SUM(t-3)',␣
 ↪'SPI6(t-1)', 'SPI6(t)']]

final_lagged_data.head()
```

[40]:
```
                PS(t)    T2M(t)  RH2M(t-2)  WS2M(t-1)  GWETPROF(t)  \
DATE
1984-06-30  0.033708 -0.025836   0.029471   0.163498    -0.703704
1984-07-31  0.056180 -0.167173   0.576369  -0.125475    -0.703704
1984-08-31  0.191011 -0.215805   0.611627  -0.346008    -0.555556
1984-09-30  0.078652 -0.325228   0.667070  -0.536122    -0.407407
1984-10-31 -0.056180 -0.189970   0.742700  -0.543726    -0.333333

            PRECTOTCORR_SUM(t-3)  SPI6(t-1)   SPI6(t)
DATE
1984-06-30             -1.000000  -0.855776 -0.512133
1984-07-31             -0.645145  -0.512133  0.079721
1984-08-31             -0.354825   0.079721  0.423364
1984-09-30             -0.419361   0.423364  1.148846
1984-10-31              0.000031   1.148846  1.110651
```

0.0.6 Create lags with raw data to prevent data leakage

To prevent data leakage, we save the not normalized data, and then we will split, then normalize the training data, and finally, apply the normalization parameters to the test data.

[41]:
```python
final_lagged_data.columns
```

[41]:
```
Index(['PS(t)', 'T2M(t)', 'RH2M(t-2)', 'WS2M(t-1)', 'GWETPROF(t)',
       'PRECTOTCORR_SUM(t-3)', 'SPI6(t-1)', 'SPI6(t)'],
      dtype='object')
```

[42]:
```python
lagged_data_raw = create_lagged_features(data= data,
                                      col_names=data.columns,
                                      n_in=data_params['window_size'],
                                      n_out=data_params['n_output_steps'],
                                      dropnan=True)

lagged_data_raw.head()
```

[42]:
```
            PS(t-5)  T2M(t-5)  RH2M(t-5)  WD2M(t-5)  WS2M(t-5)  GWETPROF(t-5)  \
DATE
1984-06-30    97.95     24.42      22.38      71.88       2.77           0.50
1984-07-31    97.75     26.79      18.75      55.69       2.87           0.49
1984-08-31    97.59     30.69      34.00     250.00       2.70           0.48
1984-09-30    97.61     31.05      50.56     230.75       3.10           0.48
1984-10-31    97.72     28.20      70.88     213.25       2.72           0.50

            CLOUD_AMT(t-5)  TOA_SW_DWN(t-5)  PRECTOTCORR_SUM(t-5)  \
DATE
1984-06-30           26.47            31.21                  0.00
1984-07-31           20.23            34.00                  0.00
1984-08-31           54.59            36.64                  0.00
```

```
1984-09-30           70.91           37.84                   58.01
1984-10-31           63.15           37.70                  105.47

            ALLSKY_SFC_SW_DWN(t-5)  …  RH2M(t)  WD2M(t)  WS2M(t)  \
DATE                                …
1984-06-30                   21.23  …    72.19   223.25     2.34
1984-07-31                   21.17  …    74.25   207.94     2.05
1984-08-31                   23.29  …    77.06   210.00     1.80
1984-09-30                   23.32  …    82.00   213.44     1.79
1984-10-31                   22.94  …    71.50   195.00     1.67

            GWETPROF(t)  CLOUD_AMT(t)  TOA_SW_DWN(t)  PRECTOTCORR_SUM(t)  \
DATE
1984-06-30         0.52         60.88          37.28               94.92
1984-07-31         0.52         67.80          37.31              163.48
1984-08-31         0.54         64.86          37.46               94.92
1984-09-30         0.56         68.46          36.73              200.39
1984-10-31         0.57         56.76          34.59               47.46

            ALLSKY_SFC_SW_DWN(t)   SPI6(t)  Moving_Sum_6(t)
DATE
1984-06-30                 21.77 -0.512133           258.40
1984-07-31                 20.33  0.079721           421.88
1984-08-31                 20.83  0.423364           516.80
1984-09-30                 19.96  1.148846           717.19
1984-10-31                 20.47  1.110651           706.64

[5 rows x 72 columns]
```

[43]:
```
lagged_data_path = data_params['data_path'] + 'lagged/' + data_params['city'] +
    '_lagged_raw.csv'
lagged_data_raw[final_lagged_data.columns].to_csv(lagged_data_path)
```

[44]:
```
pd.read_csv('../datasets/lagged/Banikoara_lagged_raw.csv').describe()
```

[44]:
```
            PS(t)      T2M(t)   RH2M(t-2)   WS2M(t-1)  GWETPROF(t)  \
count  451.000000  451.000000  451.000000  451.000000   451.000000
mean    97.903592   27.623659   53.018825    2.247029     0.531840
std      0.146422    2.584615   23.327118    0.552785     0.051385
min     97.460000   21.070000   12.310000    1.190000     0.480000
25%     97.805000   25.800000   30.095000    1.760000     0.490000
50%     97.940000   26.870000   55.190000    2.290000     0.520000
75%     98.010000   29.410000   75.000000    2.635000     0.560000
max     98.350000   34.230000   86.620000    3.770000     0.690000

       PRECTOTCORR_SUM(t-3)   SPI6(t-1)     SPI6(t)
count            451.000000  451.000000  451.000000
```

mean	66.189379	-0.019990	-0.015971
std	80.123085	0.990260	0.990539
min	0.000000	-1.447631	-1.447631
25%	0.000000	-0.970360	-0.970360
50%	26.370000	-0.053906	-0.034827
75%	126.560000	0.900635	0.910174
max	326.950000	1.969759	1.969759

[ ]: