AI/ML for Climate Workshop

International Livestock Research Institute (ILRI)

---

hide: - toc

🌦 ML-based Subseasonal & Seasonal Prediction

Monthly Climate/Ocean Indices from Physical Sciences Laboratory (PSL), NOAA

Ocean Time-series: ENSO Indices

- Source: NOAA Extended Reconstructed SST V5
- Temporal Resolution: Monthly
- Duration: Jan 1950 to Sep 2025
- Units: ℃
- Climatology: 1981-2010
- Temporal Coverage: Jan 1950 to Sep 2025

1) Niño 1+2 region 0N-10S, 90W-80W - https://psl.noaa.gov/data/correlation/nina1.anom.csv

2) Niño 3 region (5N-5S,150W-90W)

- https://psl.noaa.gov/data/correlation/nina3.anom.csv

3) Niño 3.4 region (5N-5S, 170-120W) - https://psl.noaa.gov/data/correlation/nina34.anom.csv

4) Nino 4 (5N-5S, 160E-150W) - https://psl.noaa.gov/data/correlation/nina4.anom.csv

5) Multivariate ENSO Index (MEI V2): `SST` and `5 atmospheric variables` are combined using `EOFs` of the `tropical Pacific` to create a `single index`.

```
- https://psl.noaa.gov/data/correlation/meiv2.csv
```

6) Oceanic Niño; Index (ONI) V2: The ONI is the NOAA official index indicating the state of the ENSO. [Niño 3.4 region (5°N-5°S, 120°-170°W)]

- https://psl.noaa.gov/data/correlation/oni.csv

7) Dipole Mode Index (DMI):

Region: - Western equatorial Indian Ocean (50E-70E and 10S-10N) - South eastern equatorial Indian Ocean (90E-110E and 10S-0N)

- https://psl.noaa.gov/data/timeseries/month/data/dmi.had.long.csv

8) Pacific Warm-Pool Time-series: Area averaged SST [60E-170E, 15S-15N] - Dataset: NOAA ERSSTV5 1948-present.

```
 - https://psl.noaa.gov/data/timeseries/month/data/pacwarmpool.ersst.csv
```

9) Bivariate EnSo Timeseries: An ENSO index based on ocean (SST) and atmospheric (SOI) data.

```
- https://psl.noaa.gov/data/correlation/censo.csv
```

10) Real-time Multivariate MJO (RMM) index: standard for historical daily data from 1974 to near-real-time.

```
 - https://psl.noaa.gov/mjo/mjoindex/omi.1x.txt

 - RMM1/RMM2 are PCs of OLR/U850/U200 anomalies.

 - Amplitude >1 = active MJO; phases indicate propagation.

 - omi.1x.txt has four column and need parse convert to CSV: year, month, day, PC1, PC2
```

Preprocess the Predictors

- Gather monthly indices
- Extract to 1981–2024
- Create appropriate candidate features and Lagged Indices without leakage:
    - 3 month lead

- 2 month lead
- 1 month lead

OND model (issued Sep 30): candidate features

- n12_JAS, n12_AS, n12_Sep
- n3_JAS, n3_AS, n3_Sep
- n34_JAS, n34_AS, n34_Sep
- n4_JAS, n4_AS, n4_Sep
- meiv2_JAS, meiv2_AS, meiv2_Sep
- oni_JAS, oni_AS, oni_Sep
- pacwarmpool_JAS, pacwarmpool_AS, pacwarmpool_Sep
- censo_JAS, censo_AS, censo_Sep
- dmi_JAS, dmi_AS, dmi_Sep
- season persistence (e.g., MJJ, JJA, JAS rainfall anomaly)

MAM model (issued Feb 28): candidate features

- n12_NDJ, n12_DJ, n12_Feb
- n3_NDJ, n3_DJ, n3_Feb
- n34_NDJ, n34_DJ, n34_Feb
- n4_NDJ, n4_DJ, n4_Feb
- meiv2_NDJ, meiv2_DJ, meiv2_Feb
- oni_NDJ, oni_DJ, oni_Feb
- pacwarmpool_NDJ, pacwarmpool_DJ, pacwarmpool_Feb,
- censo_NDJ, censo_DJ, censo_Feb,
- dmi_NDJ, dmi_DJ, dmi_Feb
- season persistence (OND, NDJ, DJF anomaly)

🚀 Interactive Learning

launch binder

Click the Binder button above to launch an interactive Jupyter notebook for NumPy and Pandas climate data analysis!

Import Libraries

```
import os, glob, math, re, io, warnings, requests, yaml
from typing import Dict, Tuple, List

import numpy as np
import pandas as pd
import xarray as xr
import geopandas as gpd
import regionmask
from shapely.geometry import mapping
from tqdm import tqdm

xr.set_options(keep_attrs=True)

# Set working directory

os.chdir("C:\\Users\\yonas\\Documents\\ICPAC\\ea_seasonal_pred\\seasonal-ml-pred")

# Paths (matches your repo)
RAW_DIR        = "data/raw/chirps_monthly_global"
PROCESSED_DIR  = "data/processed"
EXTERNAL_DIR   = "data/external/indices"
LOGS_DIR       = "reports/logs"

os.makedirs(RAW_DIR, exist_ok=True)
os.makedirs(PROCESSED_DIR, exist_ok=True)
os.makedirs(EXTERNAL_DIR, exist_ok=True)
os.makedirs(LOGS_DIR, exist_ok=True)

# Project settings
START_YEAR, END_YEAR = 1981, 2024
CLIM_START, CLIM_END = 1991, 2020

# Download switches (set to False if you already have files)
DOWNLOAD_CHIRPS = True
DOWNLOAD_INDICES = True
```

Defining the Paths

```
import os, io, re, warnings, requests
import numpy as np
import pandas as pd
```

```
EXTERNAL_DIR  = "data/external/indices"
PROCESSED_DIR = "data/processed"
os.makedirs(EXTERNAL_DIR, exist_ok=True)
os.makedirs(PROCESSED_DIR, exist_ok=True)

# Set working directory
os.chdir("C:\\Users\\yonas\\Documents\\ICPAC\\ea_seasonal_pred\\seasonal-ml-pred")

# Time window you care about
START_YEAR, END_YEAR = 1981, 2024

# Your curated PSL sources (monthly, mostly anomalies; units often °C)
SOURCES = {
    # ENSO SST regions
    "nino12":  "https://psl.noaa.gov/data/correlation/nina1.anom.csv",
    "nino3":   "https://psl.noaa.gov/data/correlation/nina3.anom.csv",
    "nino34":  "https://psl.noaa.gov/data/correlation/nina34.anom.csv",
    "nino4":   "https://psl.noaa.gov/data/correlation/nina4.anom.csv",
    # MEI v2
    "meiv2":   "https://psl.noaa.gov/data/correlation/meiv2.csv",
    # ONI (NOAA official 3-mo running N3.4 anomaly)
    "oni":     "https://psl.noaa.gov/data/correlation/oni.csv",
    # IOD / DMI (HadISST)
    "dmi":     "https://psl.noaa.gov/data/timeseries/month/data/dmi.had.long.csv",
    # Pacific Warm Pool (ERSSTV5)
    "pacwarmpool": "https://psl.noaa.gov/data/timeseries/month/data/pacwarmpool.ersst.c
    # Bivariate ENSO index
    "censo":   "https://psl.noaa.gov/data/correlation/censo.csv",
}
```

Downloading and parsers to handle PSL CSV

```
def _fetch_text(url: str, timeout: int = 90) -> str:
    r = requests.get(url, timeout=timeout)
    r.raise_for_status()
    return r.text

def _clean_df(df: pd.DataFrame) -> pd.DataFrame:
    # drop all-empty cols/rows, strip names
    df = df.copy()
    df.columns = [str(c).strip() for c in df.columns]
    df = df.dropna(how="all")
    return df

def _parse_wide_year_months(df: pd.DataFrame) -> pd.Series:
    """
    Wide table: one row per YEAR and columns JAN..DEC (sometimes also 'ANNUAL').
    """
    df = _clean_df(df)
    # Normalize month names
```

```
    month_alias = {
        'JAN':'1','FEB':'2','MAR':'3','APR':'4','MAY':'5','JUN':'6',
        'JUL':'7','AUG':'8','SEP':'9','OCT':'10','NOV':'11','DEC':'12'
    }
    # Guess year column
    ycol = None
    for c in df.columns:
        if re.fullmatch(r"(?i)year|yr|yyyy", c):
            ycol = c
            break
    if ycol is None:
        # sometimes first column is Year
        ycol = df.columns[0]
    # Keep month columns only
    mcols = [c for c in df.columns if c.upper()[:3] in month_alias]
    if not mcols:
        raise ValueError("No JAN..DEC columns found in wide table.")
    melted = []
    for _, row in df.iterrows():
        y = int(row[ycol])
        for mc in mcols:
            mnum = int(month_alias[mc.upper()[:3]])
            val = pd.to_numeric(row[mc], errors="coerce")
            melted.append({"date": pd.Timestamp(y, mnum, 1), "value": val})
    ser = (pd.DataFrame(melted)
             .dropna(subset=["value"])
             .set_index("date")["value"]
             .sort_index()
             .asfreq("MS"))
    return ser

def _parse_long_year_month_value(df: pd.DataFrame) -> pd.Series:
    """
    Long tidy table with YEAR, MONTH (or MON) and a value/ANOM column.
    """
    df = _clean_df(df)
    cols = {c.lower(): c for c in df.columns}
    # Try to find year & month col names
    year_c = next((cols[k] for k in ["year","yr","yyyy"] if k in cols), None)
    mon_c  = next((cols[k] for k in ["month","mon","mm"] if k in cols), None)
    if year_c is None or mon_c is None:
        raise ValueError("YEAR/MONTH columns not found in long table.")

    # Choose a value column by priority
    val_candidates = [k for k in df.columns if k not in [year_c, mon_c]]
    # Prefer typical names
    pref = [c for c in val_candidates if re.search(r"(?i)value|anom|index|dmi|mei|oni|s
    val_c = pref[0] if pref else val_candidates[0]

    tmp = df[[year_c, mon_c, val_c]].copy()
    tmp.columns = ["year", "month", "value"]
    tmp["date"] = pd.to_datetime(dict(year=tmp["year"].astype(int),
                                      month=tmp["month"].astype(int),
                                      day=1), errors="coerce")
```

```
    ser = (tmp.dropna(subset=["date"])
              .set_index("date")["value"]
              .astype(float)
              .sort_index()
              .asfreq("MS"))
    return ser

def _parse_single_date_value(df: pd.DataFrame) -> pd.Series:
    """
    Two-column style: first column is a date (YYYY-MM or YYYY-MM-DD), second is value.
    """
    df = _clean_df(df)
    # choose date-like column
    date_c = None
    for c in df.columns:
        # Try datetime conversion on a sample
        try:
            ts = pd.to_datetime(df[c], errors="coerce")
            if ts.notna().mean() > 0.8:
                date_c = c
                break
        except Exception:
            continue
    if date_c is None:
        raise ValueError("No parseable date column found.")
    # choose value column (first numeric other than date)
    val_c = None
    for c in df.columns:
        if c == date_c:
            continue
        if pd.to_numeric(df[c], errors="coerce").notna().mean() > 0.8:
            val_c = c
            break
    if val_c is None:
        raise ValueError("No numeric value column found.")

    ser = (pd.DataFrame({
                "date": pd.to_datetime(df[date_c], errors="coerce"),
                "value": pd.to_numeric(df[val_c], errors="coerce")
            })
            .dropna(subset=["date","value"])
            .set_index("date")["value"]
            .sort_index())

    # Normalize to month-start - fix the period conversion
    ser.index = pd.to_datetime(ser.index.to_period("M").to_timestamp())
    ser = ser.asfreq("MS")
    return ser

def parse_psl_csv_text(txt: str) -> pd.Series:
    """
    Try multiple parsers to handle PSL CSV flavors:
    - comment lines (#) ignored
    - wide (YEAR + JAN..DEC)
```

```
    - long (YEAR, MONTH, VALUE)
    - single date/value columns
    Returns monthly Series indexed by Timestamp (MS).
    """
    # Load with comment skipping and flexible parser
    df = pd.read_csv(io.StringIO(txt), comment="#", skip_blank_lines=True)
    if df.shape[1] >= 13 and any(m in [c.upper()[:3] for c in df.columns] for m in ["JA
        # Wide monthly table
        try:
            return _parse_wide_year_months(df)
        except Exception as e:
            warnings.warn(f"Wide parse failed: {e}")
    # Try long YEAR/MON/value
    try:
        return _parse_long_year_month_value(df)
    except Exception:
        pass
    # Try date + value
    try:
        return _parse_single_date_value(df)
    except Exception as e:
        raise ValueError(f"CSV did not match known formats: {e}")

def load_or_download_indices(sources: dict, download: bool = True) -> dict:
    out = {}
    for key, url in sources.items():
        cache = os.path.join(EXTERNAL_DIR, f"{key}_monthly.csv")
        if download:
            try:
                txt = _fetch_text(url)
                ser = parse_psl_csv_text(txt)
                # Drop gross missing sentinels if any
                ser = ser.replace([-99, -99.9, -9.9e1, -999, -9999], np.nan)
                # Save normalized cache
                ser.to_frame("value").to_csv(cache, index_label="date")
            except Exception as e:
                warnings.warn(f"Download/parsing failed for {key}: {e}")
                if os.path.exists(cache):
                    ser = pd.read_csv(cache, parse_dates=["date"]).set_index("date")["v
                else:
                    raise
        else:
            ser = pd.read_csv(cache, parse_dates=["date"]).set_index("date")["value"]

        # Standardize: month-start, sorted, float
        ser.index = ser.index.to_period("M").to_timestamp()
        ser = ser.sort_index().asfreq("MS").astype(float)
        out[key] = ser
    return out

indices = load_or_download_indices(SOURCES, download=True)
```

```
# Quick sanity peek
{k: (v.index.min(), v.index.max(), f"{v.isna().mean():.2%} NaN") for k, v in indices.it
```

Output:

```
{'nino12': (Timestamp('1948-01-01 00:00:00'),
  Timestamp('2025-12-01 00:00:00'),
  '2.88% NaN'),
 'nino3': (Timestamp('1948-01-01 00:00:00'),
  Timestamp('2025-12-01 00:00:00'),
  '2.88% NaN'),
 'nino34': (Timestamp('1948-01-01 00:00:00'),
  Timestamp('2025-12-01 00:00:00'),
  '2.88% NaN'),
 'nino4': (Timestamp('1948-01-01 00:00:00'),
  Timestamp('2025-12-01 00:00:00'),
  '2.88% NaN'),
 'meiv2': (Timestamp('1979-01-01 00:00:00'),
  Timestamp('2025-12-01 00:00:00'),
  '1.24% NaN'),
 'oni': (Timestamp('1950-01-01 00:00:00'),
  Timestamp('2025-12-01 00:00:00'),
  '0.55% NaN'),
 'dmi': (Timestamp('1870-01-01 00:00:00'),
  Timestamp('2025-12-01 00:00:00'),
  '0.43% NaN'),
 'pacwarmpool': (Timestamp('1854-01-01 00:00:00'),
  Timestamp('2025-12-01 00:00:00'),
  '0.15% NaN'),
 'censo': (Timestamp('1948-01-01 00:00:00'),
  Timestamp('2025-12-01 00:00:00'),
  '1.28% NaN')}
```

Verify the Download

```
# Print only the key of the indices
for k in indices.keys():
    print(k)
```

Output:

```
nino12
nino3
nino34
nino4
meiv2
oni
```

```
dmi
pacwarmpool
censo
```

Copy the indices data frame

```
n12 = indices["nino12"].copy()
n3 = indices["nino3"].copy()
n34 = indices["nino34"].copy()
n4 = indices["nino4"].copy()
meiv2 = indices["meiv2"].copy()
oni = indices["oni"].copy()
pacwarmpool = indices["pacwarmpool"].copy()
censo = indices["censo"].copy()
dmi = indices["dmi"].copy()
```

Select the time period

```
n12 = n12.loc["1981-01": f"{END_YEAR}-12"]
n3 = n3.loc["1981-01": f"{END_YEAR}-12"]
n34 = n34.loc["1981-01": f"{END_YEAR}-12"]
n4 = n4.loc["1981-01": f"{END_YEAR}-12"]
meiv2 = meiv2.loc["1981-01": f"{END_YEAR}-12"]
oni = oni.loc["1981-01": f"{END_YEAR}-12"]
pacwarmpool = pacwarmpool.loc["1981-01": f"{END_YEAR}-12"]
censo = censo.loc["1981-01": f"{END_YEAR}-12"]
dmi = dmi.loc["1981-01": f"{END_YEAR}-12"]
```

Convert to the data frame

```
df_indices = pd.DataFrame({
    "n12": n12,
    "n3": n3,
    "n34": n34,
    "n4": n4,
    "meiv2": meiv2,
    "oni": oni,
    "pacwarmpool": pacwarmpool,
    "censo": censo,
    "dmi": dmi
})

df_indices
```

Output:

```
             n12    n3   n34    n4  meiv2   oni  pacwarmpool  censo    dmi
date
1981-01-01 -1.50 -0.71 -0.36 -0.13  -0.36 -0.26       -0.299  -0.26 -0.201
1981-02-01 -1.17 -0.88 -0.64 -0.17  -0.23 -0.50       -0.332  -0.05 -0.024
1981-03-01 -0.55 -0.59 -0.64 -0.17   0.33 -0.47       -0.229   0.82  0.027
1981-04-01 -0.73 -0.52 -0.53 -0.52   0.43 -0.37       -0.358   0.18  0.092
1981-05-01 -0.77 -0.54 -0.57 -0.56  -0.24 -0.26       -0.231  -0.36 -0.018
...          ...   ...   ...   ...    ...   ...          ...    ...    ...
2024-08-01 -0.42 -0.29 -0.12  0.41  -0.73 -0.11        0.635  -0.63  0.267
2024-09-01 -0.75 -0.20 -0.26  0.15  -0.65 -0.21        0.640   0.11  0.115
2024-10-01 -0.32 -0.16 -0.27  0.12  -0.52 -0.26        0.654  -0.48 -0.196
2024-11-01  0.06 -0.16 -0.25  0.12  -0.68 -0.37        0.658  -0.38 -0.383
2024-12-01 -0.03 -0.39 -0.60 -0.28  -0.91 -0.53        0.492  -0.97 -0.331

[528 rows x 9 columns]
```

Check for any missing values

```
df_indices.isna().sum()
```

Output:

```
n12            0
n3             0
n34            0
n4             0
meiv2          0
oni            0
pacwarmpool    0
censo          0
dmi            0
dtype: int64
```

```
# Plot the indices to visualize trends
import matplotlib.pyplot as plt
df_indices.plot(subplots=True, figsize=(10, 12), title="Climate Indices Time Series")
plt.tight_layout()
plt.show()
```

Output:

```
<Figure size 1000x1200 with 9 Axes>
```

Check and fix the datetime

```
def to_month_start(df: pd.DataFrame) -> pd.DataFrame:
    out = df.copy()
    # Ensure datetime-like index
    out.index = pd.to_datetime(out.index, errors="coerce")
    # Anchor to the START of each month using Period->Timestamp
    out.index = out.index.to_period("M").to_timestamp(how="start")
    # Ensure a complete monthly index (MonthStart) and keep missing months as NaN
    out = out.sort_index().asfreq("MS")
    return out

df_indices = to_month_start(df_indices)
df_indices.index.freq  # should be <MonthBegin>
```

Output:

```
<MonthBegin>
```

```
df_indices
```

Output:

```
             n12    n3   n34    n4  meiv2   oni  pacwarmpool  censo    dmi
date
1981-01-01 -1.50 -0.71 -0.36 -0.13  -0.36 -0.26       -0.299  -0.26 -0.201
1981-02-01 -1.17 -0.88 -0.64 -0.17  -0.23 -0.50       -0.332  -0.05 -0.024
1981-03-01 -0.55 -0.59 -0.64 -0.17   0.33 -0.47       -0.229   0.82  0.027
1981-04-01 -0.73 -0.52 -0.53 -0.52   0.43 -0.37       -0.358   0.18  0.092
1981-05-01 -0.77 -0.54 -0.57 -0.56  -0.24 -0.26       -0.231  -0.36 -0.018
...          ...   ...   ...   ...    ...   ...          ...    ...    ...
2024-08-01 -0.42 -0.29 -0.12  0.41  -0.73 -0.11        0.635  -0.63  0.267
2024-09-01 -0.75 -0.20 -0.26  0.15  -0.65 -0.21        0.640   0.11  0.115
2024-10-01 -0.32 -0.16 -0.27  0.12  -0.52 -0.26        0.654  -0.48 -0.196
2024-11-01  0.06 -0.16 -0.25  0.12  -0.68 -0.37        0.658  -0.38 -0.383
2024-12-01 -0.03 -0.39 -0.60 -0.28  -0.91 -0.53        0.492  -0.97 -0.331

[528 rows x 9 columns]
```

Features Enginerrign

- OND
- {index}_JAS, {index}_AS, {index}_Sep

- MAM
- {index}_NDJ, {index}_DJ, {index}_Feb

```
def _grab(ser: pd.Series, dates: list[pd.Timestamp]) -> pd.Series:
    """Return values at specific dates (preserving order), possibly containing NaNs."""
    return ser.reindex(dates)

def features_OND_all(df: pd.DataFrame, y0: int, y1: int) -> pd.DataFrame:
    """
    Issuance: end of Sep (use JAS, AS, Sep for each predictor).
    For each column 'X', create: X_JAS, X_AS, X_Sep.
    """
    recs = []
    for y in range(y0, y1 + 1):
        J, A, S = pd.Timestamp(y, 7, 1), pd.Timestamp(y, 8, 1), pd.Timestamp(y, 9, 1)
        # Require at least Sep present for this year; if not, skip this season-year ent
        if S not in df.index:
            continue
        rec = {"season_year": y}
        for col in df.columns:
            ser = df[col]
            vJAS = _grab(ser, [J, A, S]).mean(skipna=False)  # strict: NaN if any month
            vAS  = _grab(ser, [A, S]).mean(skipna=False)
            vSep = ser.reindex([S]).iloc[0] if S in ser.index else np.nan
            rec[f"{col}_JAS"] = vJAS
            rec[f"{col}_AS"]  = vAS
            rec[f"{col}_Sep"] = vSep
        recs.append(rec)
    out = pd.DataFrame(recs).set_index("season_year").sort_index()
    return out

def features_MAM_all(df: pd.DataFrame, y0: int, y1: int) -> pd.DataFrame:
    """
    Issuance: end of Feb (use NDJ, DJ, Feb around year y) for each predictor.
    For each column 'X', create: X_NDJ, X_DJ, X_Feb.
    """
    recs = []
    for y in range(y0, y1 + 1):
        prev = y - 1
        N, D, J, F = pd.Timestamp(prev, 11, 1), pd.Timestamp(prev, 12, 1), pd.Timestamp
        # Require at least Feb present for this year; if not, skip this season-year
        if F not in df.index:
            continue
        rec = {"season_year": y}
        for col in df.columns:
            ser = df[col]
            vNDJ = _grab(ser, [N, D, J]).mean(skipna=False)  # strict: NaN if any missi
            vDJ  = _grab(ser, [D, J]).mean(skipna=False)
            vFeb = ser.reindex([F]).iloc[0] if F in ser.index else np.nan
            rec[f"{col}_NDJ"] = vNDJ
            rec[f"{col}_DJ"]  = vDJ
            rec[f"{col}_Feb"] = vFeb
```

```
        recs.append(rec)
    out = pd.DataFrame(recs).set_index("season_year").sort_index()
    return out
```

Check the missng values

```
features_OND_all_df = features_OND_all(df_indices, START_YEAR, END_YEAR)
features_MAM_all_df = features_MAM_all(df_indices, START_YEAR, END_YEAR)

# Optional QC: how many NaNs per year?
nan_summary_ond = features_OND_all_df.isna().sum(axis=1).rename("num_nan_features")
nan_summary_mam = features_MAM_all_df.isna().sum(axis=1).rename("num_nan_features")
display(pd.concat([nan_summary_ond, nan_summary_mam], axis=1).head())

# Persist
ond_all_csv  = os.path.join(PROCESSED_DIR, "features_OND_all.csv")
mam_all_csv  = os.path.join(PROCESSED_DIR, "features_MAM_all.csv")

features_OND_all_df.to_csv(ond_all_csv)
features_MAM_all_df.to_csv(mam_all_csv)


print("WROTE:")
print("  -", ond_all_csv)
print("  -", mam_all_csv)
```

Output:

```
             num_nan_features  num_nan_features
season_year
1981                        0                18
1982                        0                 0
1983                        0                 0
1984                        0                 0
1985                        0                 0
WROTE:
  - data/processed\features_OND_all.csv
  - data/processed\features_MAM_all.csv
```

```
features_OND_all_df
```

Output:

```
              n12_JAS  n12_AS  n12_Sep    n3_JAS  n3_AS  n3_Sep   n34_JAS  \
season_year
1981        -0.936667  -0.960    -0.79 -0.520000 -0.480   -0.23 -0.453333
```

```
1982          0.860000   1.055      1.31  1.083333  1.420    1.89  0.863333
1983          2.700000   2.150      1.37  0.526667  0.390    0.13 -0.300000
1984         -0.420000  -0.315     -0.07 -0.510000 -0.375   -0.34 -0.360000
1985         -1.206667  -1.110     -0.99 -0.946667 -0.870   -0.86 -0.666667
1986         -0.136667   0.040      0.18  0.110000  0.235    0.30  0.246667
1987          0.983333   0.960      1.10  1.406667  1.555    1.66  1.503333
1988         -1.373333  -1.375     -1.29 -1.543333 -1.285   -1.09 -1.310000
1989         -0.580000  -0.475     -0.65 -0.383333 -0.350   -0.30 -0.466667
1990         -0.530000  -0.465     -0.49  0.080000  0.115    0.12  0.176667
1991          0.476667   0.415      0.46  0.566667  0.435    0.36  0.620000
1992         -0.060000  -0.055      0.03 -0.070000 -0.075   -0.07  0.070000
1993          0.546667   0.470      0.38  0.266667  0.280    0.32  0.236667
1994         -0.683333  -0.590     -0.15 -0.196667 -0.155   -0.05  0.416667
1995         -0.433333  -0.345     -0.28 -0.793333 -0.970   -1.04 -0.560000
1996         -0.926667  -0.760     -0.65 -0.520000 -0.460   -0.50 -0.370000
1997          3.850000   3.960      3.96  2.506667  2.715    2.84  1.860000
1998          0.866667   0.590      0.22 -0.666667 -0.810   -0.95 -1.176667
1999         -0.463333  -0.580     -1.05 -0.903333 -0.975   -1.11 -1.160000
2000         -0.360000  -0.400     -0.23 -0.473333 -0.400   -0.32 -0.560000
2001         -0.850000  -0.985     -1.13 -0.360000 -0.405   -0.59 -0.096667
2002         -0.096667  -0.030      0.20  0.586667  0.685    0.86  0.900000
2003         -0.483333  -0.220     -0.45  0.176667  0.205    0.15  0.246667
2004         -0.600000  -0.510     -0.27  0.223333  0.345    0.45  0.686667
2005         -0.496667  -0.570     -0.74  0.066667  0.050   -0.08 -0.106667
2006          0.623333   0.835      0.97  0.410000  0.610    0.83  0.306667
2007         -1.050000  -1.085     -0.97 -1.133333 -1.225   -1.27 -0.806667
2008          1.136667   1.210      1.02  0.360000  0.400    0.30 -0.226667
2009          0.700000   0.675      0.51  0.756667  0.790    0.78  0.573333
2010         -1.186667  -1.265     -1.22 -1.146667 -1.215   -1.29 -1.353333
2011         -0.283333  -0.475     -0.74 -0.400000 -0.500   -0.61 -0.626667
2012          0.176667   0.065      0.06  0.476667  0.430    0.34  0.366667
2013         -1.176667  -0.965     -0.77 -0.560000 -0.490   -0.33 -0.316667
2014          0.890000   0.845      0.80  0.286667  0.275    0.37  0.066667
2015          2.156667   1.965      2.28  2.146667  2.295    2.50  1.866667
2016          0.353333   0.405      0.57 -0.466667 -0.395   -0.26 -0.546667
2017         -0.490000  -0.575     -0.81 -0.310000 -0.485   -0.72 -0.113333
2018         -0.250000  -0.175     -0.08  0.133333  0.145    0.35  0.226667
2019         -0.720000  -0.735     -0.71 -0.230000 -0.295   -0.27  0.140000
2020         -1.183333  -1.135     -1.22 -0.730000 -0.800   -0.99 -0.573333
2021         -0.400000  -0.495     -0.70 -0.400000 -0.440   -0.43 -0.490000
2022         -0.940000  -0.810     -1.02 -0.720000 -0.815   -0.96 -0.910000
2023          2.896667   2.855      2.42  1.900000  2.050    2.10  1.323333
2024         -0.650000  -0.585     -0.75 -0.210000 -0.245   -0.20 -0.113333

             n34_AS  n34_Sep    n4_JAS  ...  oni_Sep  pacwarmpool_JAS  \
season_year                             ...                             
1981         -0.360    -0.19 -0.613333  ...    -0.16        -0.176333   
1982          1.110     1.49  0.183333  ...     1.58        -0.547000   
1983         -0.395    -0.52 -0.483333  ...    -0.46        -0.179667   
1984         -0.290    -0.34 -0.583333  ...    -0.24        -0.425667   
1985         -0.630    -0.70 -0.593333  ...    -0.40        -0.421667   
1986          0.425     0.53  0.066667  ...     0.71        -0.382667   
1987          1.610     1.65  0.620000  ...     1.65        -0.164667   
1988         -1.095    -1.00 -0.876667  ...    -1.19         0.125333   
```

```
1989         -0.415    -0.30 -0.706667  ...    -0.24         -0.161667
1990          0.220     0.22  0.123333  ...     0.39         -0.190000
1991          0.550     0.42  0.416667  ...     0.62         -0.141000
1992         -0.005    -0.06  0.183333  ...    -0.13         -0.244333
1993          0.240     0.35  0.166667  ...     0.15         -0.444000
1994          0.505     0.48  0.583333  ...     0.55         -0.456000
1995         -0.735    -0.84 -0.153333  ...    -0.81         -0.007667
1996         -0.335    -0.45 -0.380000  ...    -0.35          0.020000
1997          2.010     2.13  0.550000  ...     2.14         -0.376667
1998         -1.270    -1.26 -0.940000  ...    -1.31          0.273000
1999         -1.155    -1.09 -1.070000  ...    -1.16         -0.222667
2000         -0.505    -0.52 -0.613333  ...    -0.55         -0.089000
2001         -0.125    -0.20  0.116667  ...    -0.19         -0.033000
2002          0.980     1.09  0.626667  ...     1.01         -0.039667
2003          0.265     0.27  0.186667  ...     0.26          0.076000
2004          0.785     0.81  0.553333  ...     0.70         -0.237667
2005         -0.045    -0.04 -0.046667  ...    -0.11          0.041667
2006          0.455     0.60  0.416667  ...     0.54         -0.127333
2007         -0.915    -1.11 -0.393333  ...    -1.07          0.029667
2008         -0.190    -0.25 -0.753333  ...    -0.24         -0.067333
2009          0.620     0.68  0.433333  ...     0.71          0.157667
2010         -1.495    -1.60 -1.160000  ...    -1.56          0.312000
2011         -0.725    -0.80 -0.623333  ...    -0.83         -0.095667
2012          0.425     0.38  0.010000  ...     0.37         -0.025667
2013         -0.280    -0.18 -0.176667  ...    -0.26          0.075333
2014          0.130     0.29  0.260000  ...     0.23          0.135000
2015          2.070     2.21  0.976667  ...     2.16          0.123000
2016         -0.580    -0.58 -0.053333  ...    -0.63          0.409333
2017         -0.295    -0.43  0.100000  ...    -0.38          0.280333
2018          0.280     0.47  0.420000  ...     0.49          0.012667
2019          0.035     0.03  0.663333  ...     0.19          0.135000
2020         -0.710    -0.83 -0.290000  ...    -0.89          0.526333
2021         -0.540    -0.55 -0.350000  ...    -0.67          0.279000
2022         -1.020    -1.07 -1.080000  ...    -1.01          0.284333
2023          1.475     1.60  0.916667  ...     1.56          0.270333
2024         -0.190    -0.26  0.356667  ...    -0.21          0.643333

             pacwarmpool_AS  pacwarmpool_Sep  censo_JAS  censo_AS  censo_Sep  \
season_year
1981                -0.2175           -0.186  -0.360000    -0.235      -0.18
1982                -0.5175           -0.495   1.786667     1.905       1.95
1983                -0.1355           -0.082  -0.070000    -0.285      -0.64
1984                -0.3760           -0.288  -0.250000    -0.265      -0.21
1985                -0.4170           -0.381  -0.390000    -0.460      -0.28
1986                -0.4730           -0.564   0.493333     0.650       0.74
1987                -0.1490           -0.126   1.843333     1.725       1.62
1988                 0.1915            0.222  -1.636667    -1.640      -1.64
1989                -0.1340           -0.061  -0.370000    -0.220      -0.38
1990                -0.2220           -0.275   0.330000     0.485       0.52
1991                -0.1715           -0.238   0.823333     0.885       0.97
1992                -0.2575           -0.269   0.193333    -0.050      -0.02
1993                -0.4460           -0.416   0.803333     0.750       0.61
1994                -0.4755           -0.545   1.166667     1.210       1.12
1995                -0.0075           -0.062  -0.213333    -0.310      -0.40
```

```
1996           0.0155        0.039  -0.353333   -0.390     -0.42
1997          -0.4010       -0.371   2.166667    2.315      2.19
1998           0.2860        0.245  -1.046667   -1.020     -0.99
1999          -0.2180       -0.161  -0.666667   -0.595     -0.43
2000           0.0005        0.066  -0.400000   -0.520     -0.61
2001          -0.0335       -0.023   0.186667    0.120     -0.17
2002          -0.0885       -0.068   1.036667    1.095      0.96
2003           0.0685        0.075   0.190000    0.210      0.24
2004          -0.2410       -0.190   0.796667    0.795      0.73
2005           0.0355       -0.008   0.103333    0.095     -0.14
2006          -0.1360       -0.126   0.776667    0.905      0.82
2007           0.0345        0.053  -0.353333   -0.560     -0.69
2008           0.0230        0.120  -0.376667   -0.560     -0.75
2009           0.1455        0.063   0.613333    0.625      0.30
2010           0.2910        0.319  -2.293333   -2.430     -2.57
2011          -0.0700       -0.087  -0.920000   -0.970     -1.15
2012           0.0095        0.027   0.453333    0.420      0.22
2013           0.0870        0.120  -0.406667   -0.255     -0.18
2014           0.1465        0.159   0.623333    0.755      0.87
2015           0.1315        0.147   2.453333    2.535      2.63
2016           0.3795        0.344  -0.790000   -0.975     -1.08
2017           0.2970        0.293  -0.510000   -0.610     -0.71
2018           0.0140        0.011   0.460000    0.615      0.96
2019           0.0815        0.030   0.670000    0.630      1.01
2020           0.4990        0.453  -0.770000   -1.035     -0.99
2021           0.3095        0.272  -0.876667   -0.755     -0.81
2022           0.2940        0.357  -1.393333   -1.615     -1.79
2023           0.2550        0.210   1.726667    1.945      2.12
2024           0.6375        0.640   0.110000   -0.260      0.11

              dmi_JAS  dmi_AS  dmi_Sep
season_year
1981        -0.648333 -0.6925   -0.757
1982         0.321000  0.3490    0.442
1983         0.267000  0.1380   -0.069
1984        -0.490667 -0.5530   -0.608
1985        -0.366667 -0.3485   -0.238
1986        -0.363000 -0.2715   -0.142
1987         0.299000  0.3450    0.393
1988        -0.277667 -0.3415   -0.394
1989        -0.331000 -0.2730   -0.225
1990        -0.273667 -0.2875   -0.183
1991         0.142333  0.0785    0.099
1992        -0.693333 -0.8005   -0.833
1993        -0.209000 -0.2375   -0.170
1994         0.632333  0.6700    0.529
1995        -0.164333 -0.1620   -0.180
1996        -0.678667 -0.6965   -0.712
1997         0.617333  0.7025    0.771
1998        -0.487000 -0.5380   -0.496
1999         0.028333 -0.0135   -0.050
2000         0.037333  0.0175   -0.096
2001        -0.221333 -0.2620   -0.223
2002        -0.060667  0.0390    0.286
```

```
2003         0.059333  0.0235   -0.061
2004        -0.179667 -0.1190   -0.106
2005        -0.421000 -0.4395   -0.534
2006         0.232333  0.3285    0.428
2007         0.172000  0.2355    0.235
2008         0.152000  0.1050    0.086
2009        -0.133000 -0.1035   -0.103
2010        -0.110333 -0.1650   -0.268
2011         0.256333  0.2730    0.202
2012         0.550333  0.5525    0.453
2013        -0.228000 -0.2520   -0.310
2014        -0.293333 -0.2585   -0.145
2015         0.362000  0.4305    0.294
2016        -0.546333 -0.4405   -0.437
2017         0.301000  0.1915    0.034
2018         0.259667  0.3630    0.604
2019         0.642000  0.6645    0.893
2020        -0.017667 -0.1865   -0.190
2021        -0.128333 -0.0785   -0.058
2022        -0.254333 -0.2840   -0.322
2023         0.756333  0.8855    0.946
2024         0.138333  0.1910    0.115

[44 rows x 27 columns]
```

```
# Feature Engineering
features_OND_all_df.columns
```

Output:

```
Index(['n12_JAS', 'n12_AS', 'n12_Sep', 'n3_JAS', 'n3_AS', 'n3_Sep', 'n34_JAS',
       'n34_AS', 'n34_Sep', 'n4_JAS', 'n4_AS', 'n4_Sep', 'meiv2_JAS',
       'meiv2_AS', 'meiv2_Sep', 'oni_JAS', 'oni_AS', 'oni_Sep',
       'pacwarmpool_JAS', 'pacwarmpool_AS', 'pacwarmpool_Sep', 'censo_JAS',
       'censo_AS', 'censo_Sep', 'dmi_JAS', 'dmi_AS', 'dmi_Sep'],
      dtype='object')
```

```
# Feature Engineering
features_MAM_all_df
```

Output:

```
              n12_NDJ  n12_DJ  n12_Feb    n3_NDJ  n3_DJ  n3_Feb   n34_NDJ  \
season_year
1981              NaN     NaN    -1.17       NaN    NaN   -0.88       NaN
1982        -0.300000  -0.220    -0.79 -0.086667  0.075   -0.00 -0.160000
1983         2.946667   2.905     2.03  2.753333  2.930    2.30  2.156667
1984         0.073333  -0.090    -0.88 -0.816667 -0.710   -0.15 -0.996667
```

```
1985        -0.436667  -0.590    -1.38 -1.173333 -1.265   -0.90 -1.230000
1986        -0.390000  -0.190    -0.10 -0.743333 -0.765   -0.53 -0.480000
1987         0.650000   0.795     0.97  0.906667  0.970    1.14  1.090000
1988         0.610000   0.345    -0.31  0.890000  0.840   -0.01  0.970000
1989        -0.670000  -0.525    -0.06 -1.573333 -1.525   -0.96 -1.993333
1990        -0.373333  -0.440    -0.16 -0.420000 -0.335    0.01 -0.190000
1991        -0.563333  -0.520    -0.38 -0.050000  0.025    0.00  0.313333
1992         0.673333   0.650     0.61  1.190000  1.310    1.27  1.576667
1993        -0.060000  -0.055     0.30 -0.310000 -0.235    0.39 -0.096667
1994         0.080000  -0.035    -0.33  0.093333  0.145   -0.08  0.086667
1995         0.730000   0.795     0.10  0.826667  0.795    0.45  1.126667
1996        -0.593333  -0.750    -0.44 -0.953333 -0.895   -0.67 -0.970000
1997        -1.043333  -1.015    -0.22 -0.853333 -0.950   -0.68 -0.526667
1998         4.090000   3.900     2.64  3.216667  3.190    2.49  2.360000
1999        -0.366667  -0.480    -0.54 -1.070000 -1.175   -0.88 -1.613333
2000        -0.876667  -0.700    -0.62 -1.570000 -1.605   -1.14 -1.696667
2001        -0.926667  -0.795    -0.48 -0.593333 -0.540   -0.28 -0.813333
2002        -0.896667  -0.900    -0.07 -0.596667 -0.555   -0.24 -0.310000
2003         0.733333   0.615    -0.12  1.026667  0.890    0.44  1.146667
2004         0.096667   0.005    -0.07  0.610000  0.595    0.30  0.343333
2005         0.353333   0.170    -0.94  0.546667  0.505   -0.09  0.686667
2006        -0.906667  -0.645     0.22 -1.053333 -1.015   -0.35 -0.806667
2007         0.720000   0.685     0.11  1.000000  1.005    0.08  0.943333
2008        -1.290000  -1.070     0.22 -1.580000 -1.560   -1.24 -1.600000
2009        -0.300000  -0.310    -0.68 -0.396667 -0.535   -0.57 -0.736667
2010         0.403333   0.440     0.12  1.160000  1.200    0.88  1.573333
2011        -0.896667  -0.690    -0.16 -1.426667 -1.380   -0.77 -1.593333
2012        -0.476667  -0.430     0.36 -0.870000 -0.745   -0.19 -1.040000
2013        -0.646667  -0.765    -0.73 -0.366667 -0.540   -0.56 -0.206667
2014        -0.336667  -0.225    -0.61 -0.253333 -0.280   -0.79 -0.266667
2015         0.350000   0.125    -0.53  0.600000  0.515    0.05  0.656667
2016         2.093333   2.000     1.30  2.756667  2.715    1.97  2.650000
2017         0.493333   0.680     1.36 -0.390000 -0.275    0.42 -0.563333
2018        -1.246667  -1.270    -0.97 -1.160000 -1.200   -0.90 -0.976667
2019         0.896667   0.940     0.52  0.806667  0.760    0.59  0.813333
2020        -0.173333  -0.205     0.02  0.263333  0.230    0.10  0.546667
2021        -0.686667  -0.740    -0.86 -0.883333 -0.720   -0.73 -1.190000
2022        -1.306667  -1.400    -1.58 -1.203333 -1.290   -1.14 -0.980000
2023        -0.716667  -0.510     0.48 -0.773333 -0.690   -0.10 -0.823333
2024         1.656667   1.450     1.12  1.990000  1.935    1.52  1.953333

             n34_DJ  n34_Feb    n4_NDJ  ...  oni_Feb  pacwarmpool_NDJ  \
season_year                             ...
1981            NaN    -0.64       NaN  ...    -0.50              NaN
1982         -0.035    -0.17 -0.143333  ...     0.07        -0.262667
1983          2.275     1.94  0.490000  ...     1.92        -0.239667
1984         -0.850    -0.19 -0.890000  ...    -0.42        -0.328000
1985         -1.250    -0.72 -0.833333  ...    -0.85        -0.364000
1986         -0.535    -0.71 -0.280000  ...    -0.47        -0.333333
1987          1.130     1.13  0.333333  ...     1.19        -0.348333
1988          0.920     0.28  0.743333  ...     0.54        -0.029000
1989         -1.965    -1.47 -1.740000  ...    -1.43        -0.242333
1990         -0.060     0.21 -0.166667  ...     0.21        -0.190667
1991          0.420     0.32  0.576667  ...     0.26        -0.183667
```

```
1992          1.765     1.78  0.706667  ...     1.63        -0.371333
1993         -0.005     0.41  0.093333  ...     0.30        -0.496667
1994          0.130     0.06  0.126667  ...     0.07        -0.349000
1995          1.135     0.73  0.796667  ...     0.72        -0.275333
1996         -0.905    -0.86 -0.520000  ...    -0.75        -0.145667
1997         -0.585    -0.37 -0.136667  ...    -0.36        -0.120667
1998          2.335     2.03  0.633333  ...     1.93        -0.081333
1999         -1.690    -1.32 -1.536667  ...    -1.30        -0.101667
2000         -1.755    -1.55 -1.320000  ...    -1.41        -0.150000
2001         -0.825    -0.63 -0.870000  ...    -0.52        -0.037000
2002         -0.280    -0.04  0.200000  ...     0.03        -0.018000
2003          0.985     0.64  0.753333  ...     0.63         0.151000
2004          0.350     0.23  0.410000  ...     0.31         0.007667
2005          0.700     0.36  0.836667  ...     0.58         0.011000
2006         -0.915    -0.67 -0.300000  ...    -0.77         0.006333
2007          0.920     0.13  0.830000  ...     0.22         0.049000
2008         -1.630    -1.67 -1.220000  ...    -1.52        -0.108000
2009         -0.875    -0.79 -0.826667  ...    -0.79         0.006000
2010          1.630     1.25  1.140000  ...     1.22         0.052667
2011         -1.570    -1.11 -1.480000  ...    -1.19         0.018333
2012         -0.965    -0.67 -0.976667  ...    -0.72         0.120000
2013         -0.390    -0.52  0.093333  ...    -0.43         0.164667
2014         -0.330    -0.62 -0.066667  ...    -0.46         0.149333
2015          0.610     0.42  0.746667  ...     0.47         0.213000
2016          2.615     2.26  1.343333  ...     2.14         0.416333
2017         -0.465    -0.08 -0.270000  ...    -0.16         0.187333
2018         -0.980    -0.78 -0.453333  ...    -0.85         0.204667
2019          0.770     0.71  0.863333  ...     0.72         0.146667
2020          0.560     0.37  0.826667  ...     0.48         0.263000
2021         -1.075    -1.00 -0.983333  ...    -0.93         0.157333
2022         -1.000    -0.89 -0.596667  ...    -0.93         0.294333
2023         -0.785    -0.46 -0.830000  ...    -0.43         0.139667
2024          1.920     1.52  1.516667  ...     1.48         0.409000

             pacwarmpool_DJ  pacwarmpool_Feb  censo_NDJ  censo_DJ  censo_Feb  \
season_year
1981                    NaN           -0.332        NaN       NaN      -0.05
1982                -0.3165           -0.305  -0.166667    -0.175      -0.04
1983                -0.1730           -0.153   2.503333     2.440       2.76
1984                -0.3375           -0.404  -0.390000    -0.330      -0.47
1985                -0.3995           -0.275  -0.463333    -0.375      -0.84
1986                -0.3205           -0.318  -0.280000    -0.410       0.17
1987                -0.3410           -0.408   1.176667     1.105       1.30
1988                -0.0500            0.216   0.716667     0.715       0.50
1989                -0.2765           -0.331  -1.603333    -1.400      -1.12
1990                -0.2010            0.005   0.163333     0.215       0.97
1991                -0.1815           -0.191   0.323333     0.285       0.24
1992                -0.3810           -0.286   1.570000     1.870       1.42
1993                -0.5040           -0.584   0.446667     0.490       0.67
1994                -0.3725           -0.169   0.193333     0.170      -0.05
1995                -0.2535           -0.164   1.030000     1.050       0.69
1996                -0.1820           -0.128  -0.316667    -0.290      -0.36
1997                -0.1480           -0.329  -0.230000    -0.335      -0.61
1998                 0.0030            0.256   2.053333     2.145       2.36
```

```
1999            -0.1395          -0.079  -1.300000    -1.380      -1.15
2000            -0.1415          -0.186  -1.216667    -1.210      -1.55
2001            -0.0755          -0.042  -0.936667    -0.740      -0.97
2002            -0.0155          -0.115  -0.090000     0.090      -0.34
2003             0.1790           0.209   1.046667     1.045       0.90
2004            -0.0115           0.136   0.370000     0.340      -0.33
2005             0.0440           0.184   0.640000     0.565       1.87
2006            -0.0025           0.046  -0.466667    -0.680      -0.32
2007             0.1080           0.101   0.713333     0.825       0.31
2008            -0.0930          -0.197  -1.420000    -1.520      -2.10
2009             0.0040          -0.041  -1.020000    -1.085      -1.64
2010             0.0550           0.243   1.546667     1.600       1.72
2011            -0.0530          -0.110  -2.250000    -2.475      -2.58
2012             0.1285           0.044  -1.533333    -1.595      -0.67
2013             0.1875           0.220   0.106667     0.155      -0.05
2014             0.1225           0.011  -0.553333    -0.550      -0.28
2015             0.2220           0.151   0.996667     0.885       0.20
2016             0.4830           0.449   2.093333     2.270       2.59
2017             0.1650           0.083  -0.266667    -0.300       0.05
2018             0.1740           0.065  -0.823333    -0.670      -0.08
2019             0.1340           0.022   0.286667     0.135       1.34
2020             0.3465           0.403   0.693333     0.490       0.36
2021             0.1210           0.078  -1.450000    -1.665      -1.56
2022             0.2910           0.193  -1.093333    -1.040      -1.12
2023             0.0950           0.003  -1.193333    -1.490      -1.29
2024             0.4800           0.659   1.163333     0.895       1.84

              dmi_NDJ  dmi_DJ  dmi_Feb
season_year
1981              NaN     NaN   -0.024
1982        -0.067667  0.0625    0.166
1983        -0.120000 -0.3220   -0.587
1984        -0.222333 -0.1620   -0.149
1985        -0.387333 -0.3720   -0.627
1986        -0.152000 -0.2650   -0.135
1987        -0.199667 -0.1775    0.041
1988         0.151333  0.2525   -0.154
1989        -0.102667 -0.0625   -0.045
1990        -0.201333 -0.1330   -0.289
1991        -0.008667  0.0340   -0.097
1992        -0.065667 -0.1185   -0.389
1993        -0.304000 -0.2565    0.035
1994        -0.110000 -0.0905   -0.146
1995         0.230000  0.2015    0.164
1996        -0.075000  0.0345   -0.033
1997        -0.440000 -0.2615    0.079
1998         0.889000  0.6940    0.422
1999        -0.373000 -0.2330   -0.038
2000        -0.129667 -0.1365   -0.009
2001        -0.328333 -0.3395   -0.017
2002        -0.146000 -0.0965   -0.098
2003        -0.100333 -0.1985    0.017
2004         0.024667  0.1165    0.114
2005        -0.136333 -0.1280   -0.560
```

```
2006        -0.235667 -0.2175   -0.305
2007         0.299000  0.1980    0.150
2008        -0.045000 -0.0560   -0.072
2009        -0.050000 -0.0110    0.163
2010         0.129000  0.2270    0.023
2011        -0.171667 -0.0100    0.242
2012         0.084667 -0.0410   -0.078
2013         0.034333  0.1015    0.189
2014         0.079667  0.0200   -0.089
2015        -0.014667 -0.0270   -0.345
2016         0.295000  0.2690   -0.110
2017        -0.259333 -0.1980    0.101
2018         0.066000 -0.0455    0.215
2019         0.398667  0.3480    0.416
2020         0.417000  0.2080    0.054
2021         0.033667  0.0405    0.243
2022        -0.035667 -0.0880   -0.083
2023        -0.084000  0.0085    0.157
2024         0.845333  0.8080    0.328

[44 rows x 27 columns]
```

```
features_MAM_all_df.columns
```

Output:

```
Index(['n12_NDJ', 'n12_DJ', 'n12_Feb', 'n3_NDJ', 'n3_DJ', 'n3_Feb', 'n34_NDJ',
       'n34_DJ', 'n34_Feb', 'n4_NDJ', 'n4_DJ', 'n4_Feb', 'meiv2_NDJ',
       'meiv2_DJ', 'meiv2_Feb', 'oni_NDJ', 'oni_DJ', 'oni_Feb',
       'pacwarmpool_NDJ', 'pacwarmpool_DJ', 'pacwarmpool_Feb', 'censo_NDJ',
       'censo_DJ', 'censo_Feb', 'dmi_NDJ', 'dmi_DJ', 'dmi_Feb'],
      dtype='object')
```

Build persistence anomalies for OND target (issued end-Sep)

- Physics-guided windows (all available by Sep of year y):
  - MJJ (May–Jun–Jul of year y)
  - JJA (Jun–Jul–Aug of year y)
  - JAS (Jul–Aug–Sep of year y)

```
PROCESSED_DIR = "data/processed"
CLIM_START, CLIM_END = 1991, 2020
```

```
START_YEAR, END_YEAR = 1981, 2024

# Monthly Uganda area-mean rainfall
monthly_path = os.path.join(PROCESSED_DIR, "uganda_monthly_chirps_1981_2024.csv")
monthly = pd.read_csv(monthly_path, parse_dates=["time"], index_col="time")["rf_mm"]
monthly = monthly.asfreq("MS").sort_index()
monthly.head()
```

Output:

```
time
1981-01-01     0.037048
1981-02-01     0.795628
1981-03-01     0.191451
1981-04-01    10.987165
1981-05-01     1.123416
Freq: MS, Name: rf_mm, dtype: float64
```

Step 2: join features + targets and build"

```

```

```
def tri_total_for_year(monthly: pd.Series, months: list[int], year: int, year_offset: i
    """
    Sum of 3 specific calendar months for a given 'year + year_offset'.
    months: list of ints in 1..12 (e.g., [5,6,7] for MJJ)
    year_offset: 0 = same year; -1 = previous year (e.g., SON_prev for MAM features)
    Returns float (NaN if any month missing).
    """
    idx = [pd.Timestamp(year + year_offset, m, 1) for m in months]
    vals = monthly.reindex(idx)
    return float(vals.sum()) if vals.notna().all() else float("nan")

def tri_series(monthly: pd.Series, months: list[int], align: str, y0: int, y1: int) ->
    """
    Build a Series of 3-mo totals indexed by 'season_year' with chosen alignment:
      - align='same' : months in same calendar year y
      - align='prev' : months in previous year (label is y)
      - align='djf'  : special case Dec(y-1) + Jan(y) + Feb(y), labeled y
    """
    rec = {}
    for y in range(y0, y1 + 1):
        if align == "same":
            rec[y] = tri_total_for_year(monthly, months, y, 0)
        elif align == "prev":
            rec[y] = tri_total_for_year(monthly, months, y, -1)
        elif align == "djf":
            idx = [pd.Timestamp(y-1, 12, 1), pd.Timestamp(y, 1, 1), pd.Timestamp(y, 2,
```

```
            vals = monthly.reindex(idx)
            rec[y] = float(vals.sum()) if vals.notna().all() else float("nan")
        else:
            raise ValueError("align must be 'same', 'prev', or 'djf'")
    s = pd.Series(rec, name="total_mm")
    s.index.name = "season_year"
    return s

def climatology_and_anoms(series: pd.Series, base: Tuple[int, int] = (1991, 2020)) -> T
    """
    Compute climatological mean (broadcast to index), anomalies in mm, and standardizec
    """
    mask = (series.index >= base[0]) & (series.index <= base[1])
    base_vals = series[mask].dropna()
    clim_mean = base_vals.mean()
    clim_std  = base_vals.std(ddof=1)

    clim_ser  = pd.Series(clim_mean, index=series.index)
    anom_mm   = series - clim_mean
    anom_std  = (series - clim_mean) / clim_std if (clim_std and not np.isclose(clim_st
    return clim_ser, anom_mm, anom_std
```

```
# OND persistence
mjj_tot = tri_series(monthly, [5,6,7], align="same", y0=START_YEAR, y1=END_YEAR)
jja_tot = tri_series(monthly, [6,7,8], align="same", y0=START_YEAR, y1=END_YEAR)
jas_tot = tri_series(monthly, [7,8,9], align="same", y0=START_YEAR, y1=END_YEAR)
_, mjj_anom, _ = climatology_and_anoms(mjj_tot, (CLIM_START, CLIM_END))
_, jja_anom, _ = climatology_and_anoms(jja_tot, (CLIM_START, CLIM_END))
_, jas_anom, _ = climatology_and_anoms(jas_tot, (CLIM_START, CLIM_END))
```

```
# MAM persistence
son_prev_tot = tri_series(monthly, [9,10,11], align="prev", y0=START_YEAR, y1=END_YEAR)
ond_prev_tot = tri_series(monthly, [10,11,12], align="prev", y0=START_YEAR, y1=END_YEAF
djf_tot      = tri_series(monthly, [], align="djf",  y0=START_YEAR, y1=END_YEAR)
_, son_prev_anom, _ = climatology_and_anoms(son_prev_tot, (CLIM_START, CLIM_END))
_, ond_prev_anom, _ = climatology_and_anoms(ond_prev_tot, (CLIM_START, CLIM_END))
_, djf_anom, _      = climatology_and_anoms(djf_tot,      (CLIM_START, CLIM_END))
```

```
jas_anom.shape
```

Output:

```
(44,)
```

```
# Ensure indices are named 'season_year' for clean joins
for s in [mjj_anom, jja_anom, jas_anom, son_prev_anom, ond_prev_anom, djf_anom]:
    s.index.name = "season_year"

persist_OND = pd.DataFrame({
    "PERSIST_MJJ_anom_mm": mjj_anom,
    "PERSIST_JJA_anom_mm": jja_anom,
    "PERSIST_JAS_anom_mm": jas_anom,
})
persist_OND.index.name = "season_year"

persist_OND.head()
```

Output:

```
             PERSIST_MJJ_anom_mm  PERSIST_JJA_anom_mm  PERSIST_JAS_anom_mm
season_year
1981                   -2.042121             1.513391             3.086952
1982                   -8.787225            -1.924204            -5.021202
1983                   -0.441927             2.636551             2.906827
1984                  -10.111957            -6.647986            -6.963177
1985                   -4.830346             0.538307             1.014789
```

```
persist_OND.shape
```

Output:

```
(44, 3)
```

```
persist_MAM = pd.DataFrame({
    "PERSIST_SONprev_anom_mm": son_prev_anom,
    "PERSIST_ONDprev_anom_mm": ond_prev_anom,
    "PERSIST_DJF_anom_mm": djf_anom,
})
persist_MAM.index.name = "season_year"

persist_MAM.head()
```

Output:

```
             PERSIST_SONprev_anom_mm  PERSIST_ONDprev_anom_mm  \
season_year
1981                             NaN                      NaN
1982                        1.067961                -3.141819
1983                       -0.035007                 5.855969
1984                       -4.057695                -3.513931
```

```
1985                       -5.366506                -2.751584


             PERSIST_DJF_anom_mm
season_year
1981                             NaN
1982                       -1.641512
1983                       -0.308066
1984                        0.486139
1985                        1.288048
```

```
persist_MAM.shape
```

Output:

```
(44, 3)
```

Save combined feature table

```
# Align on overlapping years to be safe
years_ond = features_OND_all_df.index.intersection(persist_OND.index)
years_mam = features_MAM_all_df.index.intersection(persist_MAM.index)

features_OND_all_plus = features_OND_all_df.loc[years_ond].join(persist_OND.loc[years_c
features_MAM_all_plus = features_MAM_all_df.loc[years_mam].join(persist_MAM.loc[years_m

print("OND shape:", features_OND_all_plus.shape, "years:", features_OND_all_plus.index.
print("MAM shape:", features_MAM_all_plus.shape, "years:", features_MAM_all_plus.index.

# Quick check of the new columns added
new_cols_ond = [c for c in features_OND_all_plus.columns if c.startswith("PERSIST_")]
new_cols_mam = [c for c in features_MAM_all_plus.columns if c.startswith("PERSIST_")]
print("Added OND cols:", new_cols_ond)
print("Added MAM cols:", new_cols_mam)
```

Output:

```
OND shape: (44, 30) years: 1981 → 2024
MAM shape: (44, 30) years: 1981 → 2024
Added OND cols: ['PERSIST_MJJ_anom_mm', 'PERSIST_JJA_anom_mm', 'PERSIST_JAS_anom_mm']
Added MAM cols: ['PERSIST_SONprev_anom_mm', 'PERSIST_ONDprev_anom_mm', 'PERSIST_DJF_anc
```

```
ond_out_csv  = os.path.join(PROCESSED_DIR, "features_OND_all_plus_persist.csv")
mam_out_csv  = os.path.join(PROCESSED_DIR, "features_MAM_all_plus_persist.csv")
```

```
features_OND_all_plus.to_csv(ond_out_csv)
features_MAM_all_plus.to_csv(mam_out_csv)

print("WROTE:")
print("  -", ond_out_csv)
print("  -", mam_out_csv)
```

Output:

```
WROTE:
  - data/processed\features_OND_all_plus_persist.csv
  - data/processed\features_MAM_all_plus_persist.csv
```

```
def missing_report(df: pd.DataFrame, title: str = ""):
    col_na = df.isna().sum().sort_values(ascending=False)
    col_rate = (df.isna().mean()*100).round(1).sort_values(ascending=False)
    row_na = df.isna().sum(axis=1)
    row_rate = (df.isna().mean(axis=1)*100).round(1)
    print(f"=== Missingness report: {title} ===")
    print(f"Rows: {len(df)} | Cols: {df.shape[1]}")
    print(f"Rows with ≥1 missing: {int((row_na>0).sum())} ({(row_na>0).mean()*100:.1f}%
    print("\nTop 10 columns by % missing:")
    display(col_rate.head(10).to_frame("% missing"))
    print("\nYears with any missing (first 10):")
    display(row_rate[row_rate>0].head(10).to_frame("% missing in row"))
```

```
missing_report(features_OND_all_plus, "OND all + persistence")
```

Output:

```
=== Missingness report: OND all + persistence ===
Rows: 44 | Cols: 30
Rows with ≥1 missing: 0 (0.0%)

Top 10 columns by % missing:

         % missing
n12_JAS        0.0
n12_AS         0.0
n12_Sep        0.0
n3_JAS         0.0
n3_AS          0.0
n3_Sep         0.0
n34_JAS        0.0
n34_AS         0.0
n34_Sep        0.0
```

```
n4_JAS          0.0

Years with any missing (first 10):

Empty DataFrame
Columns: [% missing in row]
Index: []
```

```
missing_report(features_MAM_all_plus, "MAM all + persistence")
```

Output:

```
=== Missingness report: MAM all + persistence ===
Rows: 44 | Cols: 30
Rows with ≥1 missing: 1 (2.3%)

Top 10 columns by % missing:

           % missing
n12_NDJ          2.3
n12_DJ           2.3
n3_NDJ           2.3
n3_DJ            2.3
n34_DJ           2.3
n34_NDJ          2.3
meiv2_DJ         2.3
meiv2_NDJ        2.3
n4_NDJ           2.3
n4_DJ            2.3

Years with any missing (first 10):

             % missing in row
season_year
1981                     70.0
```

```
def plot_missingness(df: pd.DataFrame, title: str):
    m = df.isna().astype(int)
    fig, ax = plt.subplots(figsize=(10, 4))
    ax.imshow(m.values, aspect="auto", interpolation="nearest")
    ax.set_title(title)
    ax.set_xlabel("features")
    ax.set_ylabel("season_year (row)")
    ax.set_yticks([]); ax.set_xticks([])
    plt.show()
```

```
plot_missingness(features_OND_all_plus, "Missingness: OND features")
```

> *Output:*

```
<Figure size 1000x400 with 1 Axes>
```

```
plot_missingness(features_MAM_all_plus, "Missingness: MAM features")
```

> *Output:*

```
<Figure size 1000x400 with 1 Axes>
```
