AOSC447_HW_02 (Score: 91.0 / 100.0)

COLLEGE OF COMPUTER, MATHEMATICAL, & NATURAL SCIENCES
DEPARTMENT OF ATMOSPHERIC & OCEANIC SCIENCE

UNIVERSITY OF MARYLAND ™

Instruction, please READ ME first!

1. **Please don't download and then upload the file, EDIT IT DIRECTLY!**
2. **Please do NOT ADD OR DELETE ANY CELLS! Otherwise you may LOSE points.**
3. Before you submit your work, make sure everything runs as expected. **Restart** the kernel (in the menubar, select "Runtime" and then → "Restart and run all").
4. This assignment **takes about 10 secs to complete all**. If your code takes more than 10 minutes, you need to adjust it before submission. If your code takes more than 15 minutes, it will not be graded.
5. **Only edit cells saying "## YOUR CODE HERE".** **Remove all** not-implemented or in-executable codes before your submission, once you fill in your code. Otherwise you will receive a zero score for the invalid cell.
6. After you are done, you need to share it through colab (input our emails again, even though mine is listed). This just notifies us 🅰🅰 , it will NOT submit your work to the grading system.
7. To submit, complete the survey in the last cell, and you will get a confirmation email for a successful submission.
8. Don't hesitate to restart the kernel and run all cells. This will **not** accidentally submit it multiple times.
9. You are encouraged to write more comments for your codes and add function headers to increase readability. For the same reason, each line is limited to 79 characters.

Make sure you fill in any place that says YOUR CODE HERE or "YOUR ANSWER HERE".

```
 ___                           ___           ___
 /__/\                        /  /\         /  /\
|  |::\                      /  /:/_       /  /:/_
|  |:|:\    ___     ___     /  /:/ /\     /  /:/ /\
 __|__|:|\:\  /__/\   /  /\  /  /:/ /:/_   /  /:/ /::\
/__/::::| \:\ \  \:\ /  /:/ /__/:/ /:/ /\ /__/:/ /:/\:\
\  \:\~~\__\/  \  \:\  /:/ \  \:\/:/ /:/ \  \:\/:/~/:/
 \  \:\          \  \:\/:/    \  \::/ /:/   \  \::/ /:/
  \  \:\          \  \::/      \  \:\/:/     \__\/ /:/
   \  \:\          \__\/         \  \::/       /__/:/
    \__\/                       \__\/         \__\/
```

AOSC647: Machine Learning in Earth Science

Spring 2021

Prof. Xin-Zhong Liang

In this homework you will use several NOAA datasets associated with ENSO and put your gained knowledge and understanding from the classroom into practice of how to process the data and perform machine learning for prediction as well as how to visualize your results. You are given with instructions to prepare the working environment on Colab. After setting up your environment, write Python codes to do the following tasks.

Background on ENSO

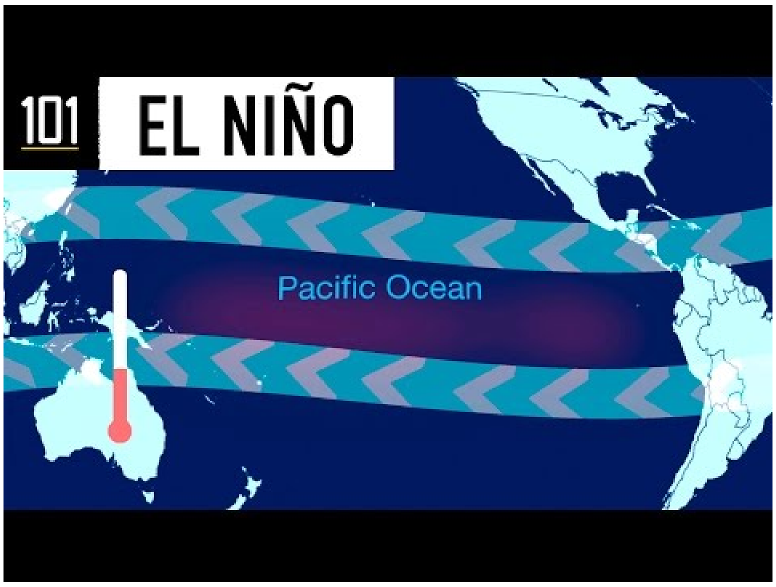
The El Niño-Southern Oscillation (ENSO) is a recurring climate pattern involving changes in the temperature of waters in the central and eastern tropical Pacific Ocean. On periods ranging from about three to seven years, the surface waters across a large swath of the tropical Pacific Ocean warm or cool by anywhere from 1°C to 3°C, compared to normal. This oscillating warming and cooling pattern, referred to as the ENSO cycle, directly affects rainfall distribution in the tropics and can have a strong influence on weather across the United States and other parts of the world. El Niño and La Niña are the extreme warm and cool phases of the ENSO cycle; between these two phases is a third phase called ENSO-neutral. See ENSOwhat for more details about the phenomena.

The National Oceanic and Atmospheric Administration (NOAA) uses the Oceanic Nino Index **(ONI)** as the primary indicator to monitor and identify ENSO events. It is the 3-month running mean of sea surface temperature (SST) anomalies in the Nino 3.4 region (5°S-5°N, 170°W-120°W). An El Niño event is defined to take place if ONI is at or above 0.5°C for at least 5 consecutive months. Reversely, an La Nina event is defined to take place if ONI is at or below -0.5°C for at least 5 consecutive months.

In [1]:

```
#@title El Niño 101 | National Geographic {display-mode: "form"}


from IPython.display import YouTubeVideo
YouTubeVideo('d6s0T0m3F8s')
# This code will be hidden when the notebook is loaded.
```

Out[1]:



Prepare your work enviroument

In this section, there is NO questions you need to answer, but you need to run the following cells to get ready for this HW.

*You need **mles** lib to complete this HW. This mles is only supported on the colab platform. Click the play button to download the mles.so, you will need its function. It will also download data and install mglearn.*

In [2]:

```
#@title Hove your mouse over the bracket on left and then click the play button to run me {displ
ay-mode: "form"}
import getpass
import time
from IPython.core.magic import register_cell_magic
from subprocess import check_call
import pandas as pd


import sys

@register_cell_magic
def playground(line, cell):
    if eval(line):
        get_ipython().ex(cell)
    else:
        return

tot_start = time.time()
user = getpass.getuser()
student = True if user !="easm" else False
print('Starting.')

if student: # It's you!
    from google.colab import output

    with output.use_tags('downloadmles'):
        sys.stdout.write('Download mles lib....\n')
        sys.stdout.flush();
        cmd = "wget --no-check-certificate 'https://docs.google.com/uc?
export=download&id=1tEIOLG0urw2R3rvf0EvkBxhV3oVLxob6' -O mles.cpython-37m-x86_64-linux-gnu.so"
        check_call(cmd,shell=True)

        sys.stdout.write('Installing mglearn....\n')
        sys.stdout.flush();
        cmd = "pip install mglearn"
        check_call(cmd,shell=True)

    output.clear(output_tags='downloadmles')
    print('Env is ready!')

    with output.use_tags('downloadcsv'):

        sys.stdout.write('download monthly index...\n')
        sys.stdout.flush();
        cmd = "wget --no-check-certificate 'https://docs.google.com/uc?
export=download&id=1yJK1eS2vrXzVVmnAswbbA9RJGOAtcjnl' -O climate_mon.csv"
        check_call(cmd,shell=True)

        sys.stdout.write('download weather data...\n')
        sys.stdout.flush();
        cmd = "wget --no-check-certificate 'https://docs.google.com/uc?
export=download&id=19Tw9F28B9qTsXufw7lg_xB1vQE27-s8h' -O weather_all.csv"
        check_call(cmd,shell=True)
    # Now clear the previous outputs.
    output.clear(output_tags='downloadcsv')
    print('Data is ready!')



else: #this is CS
    import os
    if not os.path.islink("mles.cpython-38-darwin.so"):
        os.symlink("/Users/easm/HW01/mles.cpython-38-darwin.so","./mles.cpython-38-darwin.so")
    for fname in ["climate_mon.csv","weather_all.csv"]:
        if not os.path.exists(fname):
            os.symlink(f"/Users/easm/HW02/{fname}",fname)



import mles
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
#matplotlib inline
import seaborn as sns
from sklearn import preprocessing
import scipy.stats as stats
from sklearn.neighbors import KNeighborsClassifier
import mglearn
from sklearn import model_selection
y_beg =1985
y_end =2018
```

```
Starting.
```

Task 1 Read data

Read the following csv files:

After run above cell you have downloaded csv files, in this task you are going to wrtie a function to read data from two CSV files provided into DataFrame:

1. climate_mon.csv $\Longrightarrow$ df_cindx

```
                 _ _                __
__      ___ __(_) |_ ___    __ _   / _|_   _ _ __
\ \ /\ / / '__| | __/ _ \  / _` | | |_| | | | '_ \
 \ V  V /| |  | | ||  __/ | (_| | |  _| |_| | | | |
  \_/\_/ |_|  |_|\__\___|  \__,_| |_|  \__,_|_| |_|

df_cindx_student = HW2_load_data(fname_idx = "climate_mon.csv" ):
```

Args:

1. fname_idx: string default value "climate_mon.csv", the csv file downloaded

Returns: Two DataFrames df_index, df_mets.

This function will:

1. Open and read climate_mon.csv data into df_cindx, assign the first column as index
2. Change df_index.index into a datetime objet.
3. Drop rows with if ALL values are NA.

DEMO

```
df_cidx = mles.HW2_load_data()
```

In [3]:

Student's answer (Top)

```
def HW2_load_data(fname_idx:str = "climate_mon.csv"):
    df_cindx = pd.read_csv(fname_idx,sep = ',',header = 0,index_col='date')
    df_cindx = df.set_index(pd.to_datetime(df.index))
    df_cindx.dropna(axis = 0,how = 'all',inplace=True)
```

In [4]:

```
%%playground student
### This is your playground, do whatever test at here!

### This is your playground, do whatever test at here!
```

In [5]:

Grade cell: `cell-e6fc9f21e520ef7a` Score: 5.0 / 5.0 (Top)

```
#@title Grading Cell, don't edit me, otherwise you would have to input all code by yourself aga
in! df_cidx loadded! {display-mode: "form"}

df_cidx_student = HW2_load_data()
df_cidx = mles.HW2_load_data()
assert df_cidx.equals(df_cidx_student)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-5-2af8e32ee977> in <module>
      1 #@title Grading Cell, don't edit me, otherwise you would have to input all code by yourse
lf again! df_cidx loadded! {display-mode: "form"}
      2 
----> 3 df_cidx_student = HW2_load_data()
      4 df_cidx = mles.HW2_load_data()
      5 assert df_cidx.equals(df_cidx_student)

<ipython-input-3-e00abd277f0c> in HW2_load_data(fname_idx)
      1 def HW2_load_data(fname_idx:str = "climate_mon.csv"):
      2     df_cindx = pd.read_csv(fname_idx,sep = ',',header = 0,index_col='date')
----> 3     df_cindx = df.set_index(pd.to_datetime(df.index))
      4     df_cindx.dropna(axis = 0,how = 'all',inplace=True)
      5 

NameError: name 'df' is not defined
```

Checkpoint

Until this point, you are supposed to get an identical DataFrame as the example above.

No matter what your previous outcome is, for the following task, you will all use this example DataFrame: df_cidx and df_mets to complete your assignment.

Run the following cell to load df_cidx, everytime you restart the notebook!

In [6]:

```
df_cidx = mles.HW2_load_data()
```

Task 2 Exploratory Data Analysis

```
                 _ _                __             
__      ___ __(_) |_ ___    __ _   / _|_   _ _ __  
\ \ /\ / / '__| | __/ _ \  / _` | | |_| | | | '_ \ 
 \ V  V /| |  | | ||  __/ | (_| | |  _| |_| | | | |
  \_/\_/ |_|  |_|\__\___|  \__,_| |_|  \__,_|_| |_|
```

```
df_student = Select_CorMax_Scale(df_cidx,vname="ONI",topn=5,plotTS=False)
```

Args:

1. df_cidx: DataFrame from above
2. vname: string, indicate which climate index to consider, default value "ONI"
3. topn: int, default 5, number of indices to select
4. plotTS: bool, default False, wether to plot or not

Returns: One DataFrames df see below

Returns: One DataFrames df see below.

This function will:

1. Calculate temporal correlations (i.e., correlation coefficients based on the monthly correspondences) between each pair of climate indices in df_cidx.
2. From the correlations calculated above, select the top N (topn) climate indices that have the largest absolute correlation values with the specific reference climate index (vname)
3. Drop rows with any number of NaN.
4. Use the StandardScaler to scale the selected climate indices, return this as a DataFrame.
5. Per user request, draw their normalized time series (i.e. mean = 0 and deviation =1) on a single plot using different colors or line styles to distinguish them.

Note:

1. Please discuss what you see from the figure. Explain what happened to 'Tropical Pacific SST EOF' at the end of the time series (sudden drop)?
2. Make sure do **NOT** alter your main DateFrame df_cidx, which will be used in the subsequent works.

DEMO

```
df = df_cidx.HW2.Select_CorMax_Scale(plotTS=True)
```

In [7]:

Student's answer (Top)

```
 from sklearn.preprocessing import StandardScaler
def Select_CorMax_Scale(df_in,vname="ONI",topn=5,plotTS=False):

  temp = df_in.copy(deep = True)
  a = df_cidx.corr()
  b = a[vname].nlargest(topn+1)
  list1 = b.index.to_list()
  list1.reverse()
  list1.pop(5)
  c = temp[list1]
  c.dropna(inplace = True)
  x = StandardScaler().fit_transform(c)
  finaldf = pd.DataFrame(data = x,index = c.index,columns = c.columns)
  finaldf.plot()
  return(finaldf)
```

In [8]:

```
%%playground student
### This is your playground, do whatever test at here!

print(Select_CorMax_Scale(df_cidx,vname="ONI",topn=5,plotTS=False))


### This is your playground, do whatever test at here!
```

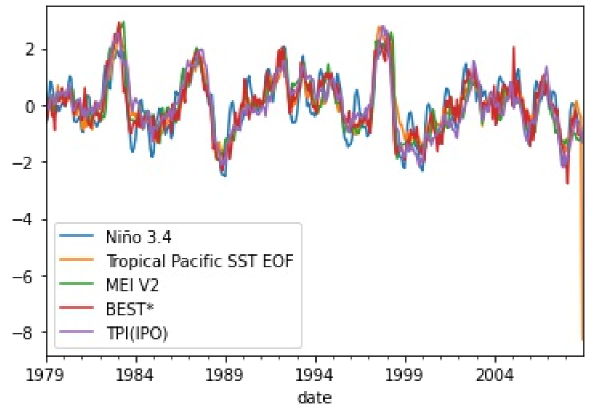
In [9]:

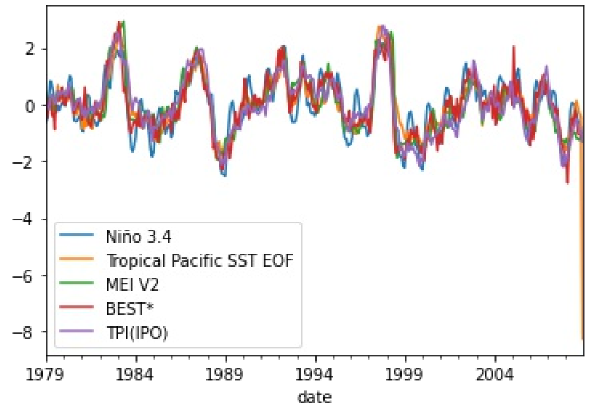
Grade cell: `cell-a39fa70128f7e2e9` Score: 15.0 / 15.0 (Top)

```
#@title Grading Cell, don't edit me, otherwise you would have to input all code by yourself aga
in!  {display-mode: "form"}

df_student = Select_CorMax_Scale(df_cidx,plotTS=True)
df = df_cidx.HW2.Select_CorMax_Scale(plotTS=True)

assert df_student.equals(df)
```

```
<ipython-input-7-0e983c31163b>:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  c.dropna(inplace = True)
```





```
              _ _                __
__      ___ __(_) |_ ___    __ _   / _|_   _ _ __
\ \ /\ / / '__| | __/ _ \  / _` | | |_| | | | '_ \
 \ V  V /| |  | | ||  __/ | (_| | |  _| |_| | | | |
  \_/\_/ |_|  |_|\__\___|  \__,_| |_|  \__,_|_| |_|
```

```
comparetwo(df_cidx,features)
```

Args:

1. df_cidx: DataFrame from above
2. features: list of strings, indicate which climate index to consider, length must be 2.

Returns: None

This function will:

1. Use statics t-tests to check whether the variance and mean values between selected features are equals.
2. Print out your answer.

Note:

DEMO

```
df_cidx.HW2.comparetwo(["ONI","SOI*"])
```

In [10]:

Student's answer (Top)

```python
def comparetwo(df,features):
  df = df[features].dropna(how='any')
  vars = True
  p = levene(df[features[0]], df[features[1]], center='mean')[1]
  if(p < 0.05):
    print("their variance are different")
    vars = False
  else:
    print("their variance are same")

  ptest = ttest_ind(df[features[0]], df[features[1]], equal_var=vars, nan_policy="omit")[1]
  if(ptest < 0.05):
    print("their means are different")
  else:
    print("their means are same")
```

In [11]:

```python
%%playground student
### This is your playground, do whatever test at here!
comparetwo(df_cidx,["ONI","SOI*"])
### This is your playground, do whatever test at here!
```

In [12]:

Grade cell: `cell-e3165d9bb6d8f034` Score: 3.0 / 5.0 (Top)

```python
#@title Grading Cell, don't edit me, otherwise you would have to input all code by yourself aga
in!  {display-mode: "form"}
print("student:")
comparetwo(df_cidx,["ONI","SOI*"])
print("demo:")
df_cidx.HW2.comparetwo(["ONI","SOI*"])
```

```
student:

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-12-999309f49a46> in <module>
      1 #@title Grading Cell, don't edit me, otherwise you would have to input all code by yourse
lf again!  {display-mode: "form"}
      2 print("student:")
----> 3 comparetwo(df_cidx,["ONI","SOI*"])
      4 print("demo:")
      5 df_cidx.HW2.comparetwo(["ONI","SOI*"])

<ipython-input-10-1242d2c11192> in comparetwo(df, features)
      2   df = df[features].dropna(how='any')
      3   vars = True
----> 4   p = levene(df[features[0]], df[features[1]], center='mean')[1]
      5   if(p < 0.05):
      6     print("their variance are different")

NameError: name 'levene' is not defined
```

ENSO event

Background info

An El Niño event is defined to take place if ONI is at or above 0.5C for at least 5 consecutive months. Sometimes a complete El Niño episode may continue for one to three consecutive years. As such, the onset of an El Niño episode is defined as the initial year when ONI starts to exceed 0.5, and the value of the highest ONI peak during the entire episode period is defined as its strength.

```
                 _ _              __
__      ___ __(_) |_ ___    __ _  / _|_   _ _ __
\ \ /\ / / '__| | __/ _ \  / _` | | |_| | | | '_ \
 \ V  V /| |  | | ||     / | (  | | |   | | | | | | |
 \_/\_/ |_|  |_|\__\___|  \__,_| |_|  \__,_|_| |_|
```

```
df_enso_student = getenso(df_cidx,low_th=0.5,mon_th=5,vname="ONI")
```

Args:

1. df_cidx: the DataFrame from above
2. low_th: the threshold to count consective months(default 0.5)
3. mon_th: the minimul number of consecutive months (with value greater than above low_th) to define an episode (default 5)
4. vname: the selected feature (default "ONI")

Returns: The output from this function is another DataFrame, which has two columns:

1. strength: The strength should be the same during an episode and assigned as a nan if it is not an El Niño year.
2. onset: Whether this is an onset year (True or False) Its index is the year value (1999,2000 etc.)

This function will: ***Calculate El Niño strength and whether this is an onset year based on the input DateFrame.***

1. Filter out nan values record in the selected feature (vname)
2. If in one year, there are more than mon_th consecutive months the feature is greater than the threshold value, this year is defined as the El Niño onset year.
3. For each El Niño onset period, the strength of this episode is defined as the maximum value of the selected feature.

Note:

1. If two episodes happen to be identified in the same year, define the strength of that year based on the first episode and also assign the year as an onset year.

Pay attention: The critical values I am using is 0.47, not a legitic one for general ENSO defination. However, to reporduce PNAS paper, I used this value.

DEMO You may use the following mles.getenso function to obtain a reference result to check your df_enso output:

```
df_enso = df_cidx.HW2.getenso(low_th=0.5,mon_th=5,vname="ONI")
```

In [13]:

Student's answer (Top)

```
def getenso(df_in,low_th=0.5,mon_th=5,vname="ONI"):
 temp = df_in.copy(deep = True)
 filtered = temp[temp[vname].notnull()]
 a = filtered[vname]
 flag = False
 flag2 = False
 countend = 0
 countend1 = np.empty(70,dtype = np.int64)
 onsetlist = []
 flaglist =  [False for i in range(70)]
 yearlist = []
 countbeglist = np.empty(70,dtype = np.int64)
 countbeg = 0
 maxlist =np.empty(70, dtype=np.float64)
 maxlist[:] = np.NaN
 filtered.reset_index(inplace = True)
 count = 0
 ninolist = []
 seasoncount = 0
 for i,x in filtered.groupby(filtered['date'].dt.year)[vname]:

    for j in range(0,12-mon_th+1,1):
     if((x.iloc[j]>=low_th) and (x.iloc[j+1]>=low_th) and (x.iloc[j+2]>=low_th) and (x.iloc[j+3]>
=low_th) and (x.iloc[j+4]>=low_th)):
       flag = True

    flaglist[count] = flag
    for k in range(8,12,1):
      if(x.iloc[k]>=low_th):
        countend = countend+1
      else:
        countend = 0
    for v in range(0,4,1):
      if(x.iloc[v]>=low_th):
        countbeg = countbeg+1
      else:
        countbeg = 0
    countbeglist[count] = countbeg
    countend1[count] = countend
    countend = 0
    countbeg = 0
    yearlist.append(i)
    flag = False
    count = count+1
 onsetlistarr = np.array(onsetlist)
 temp = True
 for m in range (0,70,1):
   if(m==69):
     temp = ((flaglist[m] is True))
   else:
     temp = ((countend1[m] + countbeglist[m+1])>=5 or (flaglist[m] is True))
   if(temp):
     onsetlist.append(True)
   else:
     onsetlist.append(False)
 count = 0
 slist = []
 for i,x in filtered.groupby(filtered['date'].dt.year)[vname]:
   a =onsetlist[count]
   if(a):
     max = x.max()
   else:
     max = np.NaN
   slist.append(max)

   count=count+1
 data = {'onset':onsetlist,
        'strength':slist}

# Creates pandas DataFrame.
 df = pd.DataFrame(data, index =yearlist)
 df.index.rename('date')
 return(df)
 #This approach of mine does not perfectly do what the question needs. I have calculated the ma
x value of each nino year instead of each period. I understand that I should have somehow group
ed the data into periods where each period contains the el nino phase. However, after several e
xperiments, i was not able to come up with the el-nino groupby statement for strength. Similarl
y, i also resorted to hardcoding the consecutive checks. I understand i should have used anothe
r loop there but I was having some issues implementing that. Regardless, it has been a busy wee
k last week with many midterms.
```

In [14]:

```
%%playground student

print(getenso(df_cidx,0.5,5,"ONI"))
print(df_cidx.HW2.getenso(low_th=0.5,mon_th=5,vname="ONI").head(50))
### This is your playground, do whatever test at here!
```

In [15]:

Grade cell: `cell-b17c82c02d23dec3` Score: 25.0 / 30.0 (Top)

```
#@title Grading Cell, don't edit me, otherwise you would have to input all code by yourself aga
in!{display-mode: "form"}
df_enso_student = getenso(df_cidx,low_th=0.5,mon_th=5,vname="ONI")
df_enso = df_cidx.HW2.getenso(low_th=0.5,mon_th=5,vname="ONI")
assert df_enso.equals(df_enso_student)
```

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<ipython-input-15-32935db9e808> in <module>
      2 df_enso_student = getenso(df_cidx,low_th=0.5,mon_th=5,vname="ONI")
      3 df_enso = df_cidx.HW2.getenso(low_th=0.5,mon_th=5,vname="ONI")
      3 df_enso = df_cidx.hw2.getenso(low_th=0.5,mon_th=5,vname='ONI')
----> 4 assert df_enso.equals(df_enso_student)

AssertionError:
```

Print out the strength along with the onset year and duration of all El Niño episodes. Select those recent episodes when their onset took place after year 1984, sort the resulting DataFrame structure by the El Niño strength in descending order, and save the final result into a DataFrame structure named df_onset.

date	onset	strength
2014	True	2.64
1997	True	2.4
1991	True	1.71
1986	True	1.7
2009	True	1.57
2002	True	1.31
1994	True	1.09
2018	True	0.85
2004	True	0.7

In [16]:

Student's answer   Score: 3.0 / 5.0 (Top)

```
# YOUR CODE HERE
df_onset = getenso(df_cidx)
print(df_onset.head(50))
df_onset = df_onset[df_onset.index.get_level_values(0) >1984]
df_onset = df_onset[df_onset['onset']==True]
df_onset.sort_values(by='strength', ascending=False)
```

```
      onset  strength
1950  False       NaN
1951   True      1.15
1952  False       NaN
1953   True      0.84
1954  False       NaN
1955  False       NaN
1956  False       NaN
1957   True      1.74
1958   True      1.81
1959  False       NaN
1960  False       NaN
1961  False       NaN
1962  False       NaN
1963   True      1.37
1964  False       NaN
1965   True      1.98
1966  False       NaN
1967  False       NaN
1968   True      0.98
1969   True      1.13
1970  False       NaN
1971  False       NaN
1972   True      2.12
1973  False       NaN
1974  False       NaN
1975  False       NaN
1976  False       NaN
1977  False       NaN
1978  False       NaN
1979  False       NaN
1980  False       NaN
1981  False       NaN
1982   True      2.23
1983   True      2.18
1984  False       NaN
1985  False       NaN
```

```
1986   True       1.22
1987   True       1.70
1988  False        NaN
1989  False        NaN
1990  False        NaN
1991   True       1.53
1992   True       1.71
1993  False        NaN
1994  False        NaN
1995  False        NaN
1996  False        NaN
1997   True       2.40
1998  False        NaN
1999  False        NaN
```

Out[16]:

```
      onset  strength
2015   True      2.64
1997   True      2.40
1992   True      1.71
1987   True      1.70
2009   True      1.57
1991   True      1.53
2002   True      1.31
1986   True      1.22
2018   True      0.85
2019   True      0.82
2004   True      0.70
2014   True      0.66
```

Checkpoint

Until this point, you are supposed to get an identical DataFrame df_enso as the example above.

No matter what your previous outcome is, for the following task, you will all use this example DataFrame: df_enso to complete your assignment.

Run the following cell to load the df_enso, everytime you restart the notebook!

In [17]:

```
df_enso = df_cidx.HW2.getenso(low_th=0.5,mon_th=5,vname="ONI")
```

Task 3 Build a model

1. From df_cindx use the two features ("AAO",'Tropical Pacific SST EOF') to predict the target data.
2. The target is defined as a binary class: 0 if 'strength' in df_enso is a NaN or otherwise 1. (i.e whether it is during an El Niño period).
3. Then apply the NeighborsClassifier algorithm to build a classifer that uses the two features to classify the El Niño years.
4. In doing so, split your data into train and test with a test_size of 0.33 and random_state of 42.
5. Try three different numbers of neighbors (1, 3, and 5) to evaluate the model performance.
6. Calculate the accuracy for the test set and print that score as part of your plot title.

Note You may use mglearn libarary discrete_scatter and plot_2d_separator to draw the data points and model decision boundaries. Arrange the three models’ results in a single row layout.

DEMO You may use the following function to obtain a reference result:

```
df_cidx.HW2.ensoclsf(df_enso)
```

In [18]:

Student's answer (Top)

```
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
def ensoclsf(df_cidx,df_enso):
 temp = df_cidx.copy(deep = True)
 tempenso = df_enso.copy(deep = True)
 df_sub = temp.groupby(temp.index.year).mean().loc[:,["AAO",'Tropical Pacific SST EOF']].dropna
()
 tempenso['strength'] = tempenso['strength'].fillna(0)
 tempenso['strength'][ tempenso['strength']!= 0] = 1
 df_sub['Target'] = tempenso.loc[df_sub.index,'strength']

 Target = df_sub['Target']
 print(Target)
 df_sub = temp.groupby(temp.index.year).mean().loc[:,["AAO",'Tropical Pacific SST EOF']].dropna
()
 print(df_sub)
 X_train,X_test,y_train,y_test = train_test_split(df_sub, Target, test_size=0.33, random_state=4
2)
 fig,axes = plt.subplots(1,3,figsize = (15,3))
 for n,ax in zip([1,3,5],axes):
  clf = KNeighborsClassifier(n_neighbors=n).fit(X_train,y_train)
  mglearn.plots.plot_2d_separator(clf,df_sub.values,fill = True,eps = 0.5,ax = ax,alpha = 0.4)
  mglearn.discrete_scatter(df_sub.iloc[:,0],df_sub.iloc[:,1],Target,ax = ax)
  train = clf.score(X_test,y_test)
  ax.set_title("neighb={}".format(n)+" score={}".format(train))
  ax.set_xlabel("AAO")
  ax.set_ylabel("Tropical Pacific SST EOF")
  ax.legend(loc = 3)
```

In [19]:

```
%%playground student
### This is your playground, do whatever test at here!


ensoclsf(df_cidx,df_enso)
df_cidx.HW2.ensoclsf(df_enso)



### This is your playground, do whatever test at here!
```

In [20]:

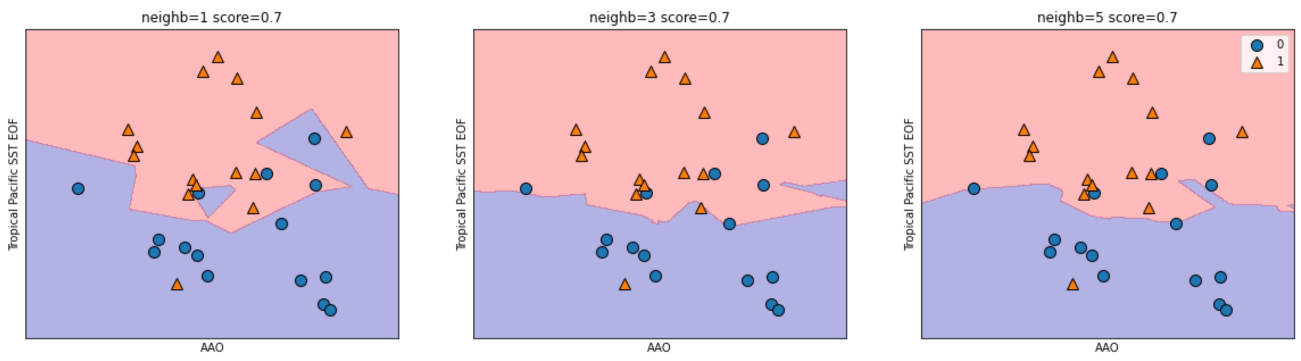
Grade cell: `cell-969ba909c40a0893` Score: 25.0 / 25.0 (Top)

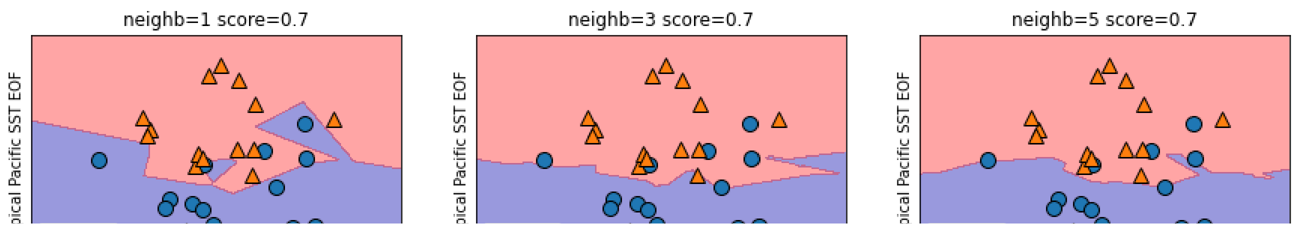
```
#@title Grading Cell, don't edit me, otherwise you would have to input all code by yourself aga
in! {display-mode: "form"}

df_cidx.HW2.ensoclsf(df_enso)
ensoclsf(df_cidx,df_enso)
```

```
<ipython-input-18-ceff193aef14>:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  tempenso['strength'][ tempenso['strength']!= 0] = 1
```

```
date
1979    0.0
1980    0.0
1981    0.0
1982    1.0
1983    1.0
1984    0.0
1985    0.0
1986    1.0
1987    1.0
1988    1.0
```

```
1989    0.0
1990    0.0
1991    1.0
1992    1.0
1993    0.0
1994    1.0
1995    1.0
1996    0.0
1997    1.0
1998    1.0
1999    0.0
2000    0.0
2001    0.0
2002    1.0
2003    1.0
2004    1.0
2005    1.0
2006    0.0
2007    0.0
2008    0.0
Name: Target, dtype: float64
           AAO  Tropical Pacific SST EOF
date
1979  0.578250                  0.381833
1980 -0.962167                  0.340333
1981 -0.440083                 -0.295917
1982  0.196167                  1.270083
1983  0.066583                  1.678417
1984 -0.270500                 -0.382917
1985  0.477667                 -0.782833
1986 -0.250167                  0.260583
1987 -0.149833                  1.761917
1988 -0.321500                 -0.831583
1989  0.640000                 -0.746500
1990 -0.181250                  0.283083
1991 -0.582083                  0.854083
1992 -0.638500                  1.061500
1993  0.567000                  0.950250
1994  0.059917                  0.535250
1995  0.174500                  0.095917
1996 -0.193583                 -0.487083
1997 -0.060583                  1.939000
1998  0.778583                  1.029917
1999  0.631833                 -1.075417
2000 -0.120417                 -0.735500
2001  0.353167                 -0.101167
2002 -0.601750                  0.741583
2003 -0.223083                  0.443250
2004  0.188583                  0.516667
2005 -0.197417                  0.380250
2006  0.257167                  0.510500
2007 -0.468333                 -0.443417
2008  0.670750                 -1.154333
```





Follow the same procedure outlined in the above Task, more systematically test the sensitivity of the model performance to the number of neighbors from 1 to 10. Draw the variations of the model accuracy scores for both the training set and the test set with respect to the number of neighbors. Make sure the two lines are distinguished.

DEMO You may use the following function to obtain a reference result:

```
df_cidx.HW2.sensitivitytest(df_enso)
```

In [21]:

Student's answer (Top)

```
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
def sensitivitytest(df_cidx,df_enso):
  temp = df_cidx.copy(deep = True)
  tempenso = df_enso.copy(deep = True)
  df_sub = temp.groupby(temp.index.year).mean().loc[:,["AAO",'Tropical Pacific SST
EOF']].dropna()
  tempenso['strength'] = tempenso['strength'].fillna(0)
  tempenso['strength'][ tempenso['strength']!= 0] = 1
  df_sub['Target'] = tempenso.loc[df_sub.index,'strength']
  Target = df_sub['Target']
  print(Target)
  df_sub = temp.groupby(temp.index.year).mean().loc[:,["AAO",'Tropical Pacific SST
EOF']].dropna()
  print(df_sub)
  X_train,X_test,y_train,y_test = train_test_split(df_sub, Target, test_size=0.33, random_state=
42)
  settings = range(1,11)
  arraytrain = []
  atest = []
  for n in [1,2,3,4,5,6,7,8,9,10]:
   clf = KNeighborsClassifier(n_neighbors=n).fit(X_train,y_train)
   train = clf.score(X_train,y_train)
   train1 = clf.score(X_test,y_test)
   arraytrain.append(train)
   atest.append(train1)
  plt.plot(settings,arraytrain,label = 'training accuracy')
  plt.plot(settings,atest,label = 'test accuracy')
  plt.ylabel('Accuracy')
  plt.xlabel('n_neighbors')
  plt.legend()
```

In [22]:

```
%%playground student
### This is your playground, do whatever test at here!

sensitivitytest(df_cidx,df_enso)
### This is your playground, do whatever test at here!
```

In [23]:

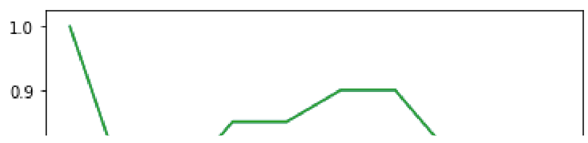
Grade cell: `cell-b2c32b77ea2cfa69` Score: 15.0 / 15.0 (Top)

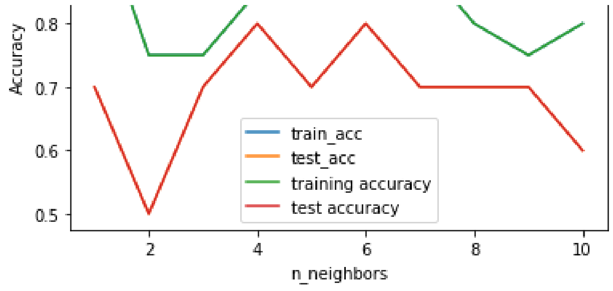
```
#@title Grading Cell, don't edit me, otherwise you would have to input all code by yourself aga
in! {display-mode: "form"}

df_cidx.HW2.sensitivitytest(df_enso)
sensitivitytest(df_cidx,df_enso)
```

```
<ipython-input-21-c9b443a5e064>:8: SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  tempenso['strength'][ tempenso['strength']!= 0] = 1
```

```
date
1979    0.0
1980    0.0
1981    0.0
1982    1.0
1983    1.0
1984    0.0
1985    0.0
1986    1.0
1987    1.0
1988    1.0
1989    0.0
1990    0.0
1991    1.0
1992    1.0
1993    0.0
1994    1.0
1995    1.0
1996    0.0
1997    1.0
1998    1.0
1999    0.0
2000    0.0
2001    0.0
2002    1.0
2003    1.0
2004    1.0
2005    1.0
2006    0.0
2007    0.0
2008    0.0
Name: Target, dtype: float64
           AAO  Tropical Pacific SST EOF
date
1979  0.578250                  0.381833
1980 -0.962167                  0.340333
1981 -0.440083                 -0.295917
1982  0.196167                  1.270083
1983  0.066583                  1.678417
1984 -0.270500                 -0.382917
1985  0.477667                 -0.782833
1986 -0.250167                  0.260583
1987 -0.149833                  1.761917
1988 -0.321500                 -0.831583
1989  0.640000                 -0.746500
1990 -0.181250                  0.283083
1991 -0.582083                  0.854083
1992 -0.638500                  1.061500
1993  0.567000                  0.950250
1994  0.059917                  0.535250
1995  0.174500                  0.095917
1996 -0.193583                 -0.487083
1997 -0.060583                  1.939000
1998  0.778583                  1.029917
1999  0.631833                 -1.075417
2000 -0.120417                 -0.735500
2001  0.353167                 -0.101167
2002 -0.601750                  0.741583
2003 -0.223083                  0.443250
2004  0.188583                  0.516667
2005 -0.197417                  0.380250
2006  0.257167                  0.510500
2007 -0.468333                 -0.443417
2008  0.670750                 -1.154333
```





In [24]:

```
tot_end = time.time()
print("total time to complete the notebook %s"%(tot_end-tot_start))
if tot_end-tot_start>300:
    print("double check your code, it is too slow!")
```

```
total time to complete the notebook 4.828179121017456
```

rookie task

1. download your data from https://drive.google.com/file/d/1yJK1eS2vrXzVVmnAswbbA9RJGOAtcjnl/view?usp=sharing
2. create your notebook to complete the following tasks
3. make proper data clean
4. do proper DEA
5. try any regression or classification model you learned in this class
6. once you are done, write up some conclusions you learned from the data
7. paste the sharable link of your notebook to the following cell, so I can see it

NOTE

1. the total boost from the rookie program is 81
2. that is to say if you did other tasks in this notebook and earned 80 points, then the rookie program can only help you 1 point
3. meanwhile, if the previous tasks have exceeded 81 points, then the rookie will not help you

Student's answer Score: 0.0 / 0.0 (Top)

YOUR ANSWER HERE

Comments:
No response.

Submission & Survey Finally, please restart and run all before complete the survey to submit your assignment!

You will get a confirmation email after you submit it. For447 students, the assignments number might be less, don't worry, just select the one shows up in your notebook. Thank you!

In [25]:

```
#@title RunMeAndCompleteThesurveyToSubmit! {display-mode: "form"}

# REMOVE THIS WHEN READY TO SUBMIT
# raise NotImplementedError

from IPython.display import IFrame
IFrame(src="https://docs.google.com/forms/d/e/1FAIpQLSe8Cz089znSlFY3J7btPjDiWFqQdwoksk4ES6OB89Bt8
-g/viewform?usp=sf_link", width='100%', height='500px')
```

Out[25]:

```
<IPython.lib.display.IFrame at 0x7fb74aaf5d00>
```