BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
PILANI

FOUNDATIONS OF DATA SCIENCE ASSIGNMENT

2019-20

Project Report

on

Multivariate Time Series Analysis

Submitted to:

Dr. Navneet Goyal

Submitted by:

2017A7PS0276P Kavya Gupta

2017A7PS0001P Bhoomi Sawant

2017A7PS0160P Aditi Mandloi

Clustering

DATASET

We have used the Agro-climatic data by county (ACDC) which is designed by Seong Do Yun and Benjamin M. Gramig. The dataset and metadata is taken from the paper “Agro-Climatic Data by County: A Spatially and Temporally Consistent U.S. Dataset for Agricultural Yields, Weather and Soils”. We have preprocessed the data according to our requirements. The final data used for clustering task consists of 14 attributes. First attribute is **stco** which represents an area, second attribute is **year,** next 10 attributes, **whc, sand, silt, clay, om, kwfactor, kffactor, spH, slope, tfactor,** represents soil conditions, **ppt1** and **ppt2** represents semi-annual precipitation in mm.

APPROACH

We have used TimeSeriesKmeans from the tslearn package because it allows us to cluster variable sized time series data using DTW metric. The number of clusters has been specified to 5 in the parameters passed to TimeSeriesKmeans.

```python
from tslearn.generators import random_walks
import matplotlib.pyplot as plt

from tslearn.clustering import TimeSeriesKMeans
from tslearn.datasets import CachedDatasets
from tslearn.preprocessing import TimeSeriesScalerMeanVariance, \
    TimeSeriesResampler
from tslearn.utils import to_time_series_dataset
import pandas as pd
import numpy as np

# X_bis = to_time_series_dataset([[[1, 2, 3, 4],[1, 2, 3, 4],[2, 5, 6, 7]],[[1, 7, 9, 4],[2, 2, 3, 4],[6, 12, 6, 2]],[[1, 1, 3, 2],[1, 3, 3, 2],[2, 8, 6, 7]]])
df = pd.read_csv('cluster.csv')
print(df.head())
X = []
for i in range(0,100):
  pd1 = df.iloc[4*i:4*i+4,2:]
  print(pd1.head())
  X.append(pd1.values)
print(X[0])

X_k = to_time_series_dataset(X)
k = TimeSeriesKMeans(n_clusters=5, max_iter=10,metric="dtw", random_state=0).fit(X_k)

print(k.cluster_centers_.shape)
print(k.cluster_centers_)
```

RESULTS FOR CLUSTERING (5 CLUSTERS) - Centroids of the clusters formed

```
array([[[2.15015000e+01, 3.45451364e+01, 3.58078636e+01, 2.96470455e+01,
         8.32022273e+01, 2.73045455e-01, 3.03545455e-01, 5.10272727e+00,
         4.26479091e+01, 4.12213636e+00, 6.46189864e+02, 7.35172136e+02],
        [2.12745455e+01, 3.47810909e+01, 3.57252273e+01, 2.94936818e+01,
         8.21462273e+01, 2.73409091e-01, 3.03590909e-01, 5.10509091e+00,
         4.25141818e+01, 4.08472727e+00, 8.45069455e+02, 9.40751409e+02],
        [2.09646522e+01, 3.40141304e+01, 3.60670000e+01, 2.99189565e+01,
         8.63689130e+01, 2.69000000e-01, 3.04478261e-01, 5.12260870e+00,
         4.24455652e+01, 4.05686957e+00, 5.26225130e+02, 6.96901348e+02],
        [2.12692727e+01, 3.47597727e+01, 3.57540455e+01, 2.94862727e+01,
         8.21035909e+01, 2.73636364e-01, 3.03909091e-01, 5.10600000e+00,
         4.25676364e+01, 4.08368182e+00, 8.75591773e+02, 8.95487000e+02]],
```

```
  [[2.06642857e+01, 4.73590000e+01, 3.16787143e+01, 2.09622857e+01,
    5.99898571e+01, 3.21714286e-01, 3.32428571e-01, 8.22414286e+00,
    1.94982714e+02, 4.49800000e+00, 1.43823000e+02, 9.86817143e+01],
   [2.07314000e+01, 4.87130000e+01, 3.14382000e+01, 1.98490000e+01,
    6.23738000e+01, 3.21400000e-01, 3.29400000e-01, 8.19440000e+00,
    1.81952000e+02, 4.50340000e+00, 5.47654000e+01, 3.83410000e+01],
   [2.07040000e+01, 4.85862000e+01, 3.14880000e+01, 1.99256000e+01,
    6.25878000e+01, 3.21200000e-01, 3.29800000e-01, 8.19440000e+00,
    1.81716000e+02, 4.50160000e+00, 4.56120000e+01, 5.41178000e+01],
   [2.06154000e+01, 4.87518000e+01, 3.13832000e+01, 1.98652000e+01,
    6.27292000e+01, 3.20400000e-01, 3.29200000e-01, 8.19400000e+00,
    1.81716600e+02, 4.50300000e+00, 3.42670000e+01, 5.05882000e+01]],

  [[2.38160952e+01, 4.76141190e+01, 2.43897857e+01, 2.79960476e+01,
    8.71662381e+01, 2.37452381e-01, 2.45523810e-01, 5.31864286e+00,
    5.37647381e+01, 4.78423810e+00, 6.68054310e+02, 6.85090595e+02],
   [2.38263810e+01, 4.73431667e+01, 2.46186190e+01, 2.80381667e+01,
    8.63706905e+01, 2.38285714e-01, 2.46571429e-01, 5.32111905e+00,
    5.37736190e+01, 4.77983333e+00, 8.74830405e+02, 7.44804643e+02],
   [2.41715909e+01, 4.56630227e+01, 2.54374545e+01, 2.88993864e+01,
    8.72432955e+01, 2.42909091e-01, 2.50590909e-01, 5.37943182e+00,
    5.49867955e+01, 4.78827273e+00, 4.86457182e+02, 5.94504500e+02],
   [2.38705714e+01, 4.73548571e+01, 2.45652143e+01, 2.80799048e+01,
    8.63982619e+01, 2.38071429e-01, 2.46095238e-01, 5.33171429e+00,
    5.42788333e+01, 4.78240476e+00, 5.76413429e+02, 5.57186952e+02]],

  [[2.30203333e+01, 2.75147143e+01, 4.00351905e+01, 3.24501429e+01,
    1.07489810e+02, 2.92857143e-01, 3.35333333e-01, 5.62790476e+00,
    5.05673333e+01, 4.10309524e+00, 6.26836333e+02, 6.73143762e+02],
   [2.29595238e+01, 2.70417143e+01, 4.03381905e+01, 3.26200476e+01,
    1.08383190e+02, 2.93047619e-01, 3.36095238e-01, 5.65076190e+00,
    5.05732857e+01, 4.09495238e+00, 6.73926476e+02, 7.26002714e+02],
   [2.26473846e+01, 2.60826154e+01, 4.08780769e+01, 3.30393077e+01,
    1.11369154e+02, 2.86884615e-01, 3.40692308e-01, 5.63338462e+00,
    5.10353462e+01, 4.04553846e+00, 4.77530962e+02, 6.00891077e+02],
   [2.33068846e+01, 2.65787308e+01, 4.15325769e+01, 3.18887308e+01,
    1.09761846e+02, 3.02615385e-01, 3.44692308e-01, 5.67711538e+00,
    5.18107692e+01, 4.05488462e+00, 7.54306000e+02, 8.07221269e+02]],
```

```
[[2.07694167e+01, 4.56209167e+01, 3.08904167e+01, 2.34886667e+01,
  1.07569417e+02, 2.83500000e-01, 3.10666667e-01, 7.95241667e+00,
  1.64711500e+02, 4.66683333e+00, 3.04839667e+02, 2.86150583e+02],
 [2.14042000e+01, 4.41252000e+01, 3.16844000e+01, 2.41905000e+01,
  9.63015000e+01, 2.90600000e-01, 3.10200000e-01, 8.05310000e+00,
  1.75281000e+02, 4.60470000e+00, 1.48478800e+02, 1.63878200e+02],
 [2.16161000e+01, 4.39816000e+01, 3.20642000e+01, 2.39539000e+01,
  9.40944000e+01, 2.94100000e-01, 3.11800000e-01, 8.05230000e+00,
  1.72364400e+02, 4.61980000e+00, 1.66981800e+02, 2.07196400e+02],
 [2.16085000e+01, 4.39716000e+01, 3.20534000e+01, 2.39749000e+01,
  9.43058000e+01, 2.94000000e-01, 3.11800000e-01, 8.05140000e+00,
  1.72461300e+02, 4.62240000e+00, 1.04882700e+02, 1.48858600e+02]]])
```

Regression

DATASET

For regression also, we have used the above mentioned ACDC data. The preprocessed data contains the attributes stco, year, corn, soyabean, cotton, wheat, ppt1 and ppt2. We have performed the regression task for stco number 1039, similar approach can be used for other location as well.

APPROACH

We have used Vector Auto Regression model for multivariate time series forecasting.
Vector autoregression (**VAR**) is a stochastic process model used to capture the interdependencies among multiple time series. **VAR** models generalize the univariate **autoregressive** model (AR model) by allowing for more than one evolving variable
 Each variable uses all other variable’s past values for regression. The code snippet and results of prediction with rmse values are as follows:

```
[ ] import pandas as pd
    import matplotlib.pyplot as plt
    %matplotlib inline
```

```
[ ] from google.colab import drive
    drive.mount('/gdrive')
```

```
[ ] import os
    os.chdir("/gdrive/My Drive")
    df = pd.read_csv("stco1039.csv" ,parse_dates=[1])
    df = df.drop(columns=['stco'])
    df.dtypes
```

```
year        datetime64[ns]
corn               float64
soybean            float64
cotton               int64
wheat              float64
ppt1               float64
ppt2               float64
dtype: object
```

```
[ ]
```

```
[ ] # !ls
```

```
[ ] df.head()
    # df.rename(columns={" Year": "Y", 'corn ': 'corn', 'soybean ':"soya"})#, 'cotton ', 'wheat ', 'ppt1 ', 'ppt2'""})
```

	stco	year	corn	soybean	cotton	wheat	ppt1	ppt2
0	1039	1982	45.0	22.2	767	28.8	688.315	711.813
1	1039	1983	56.6	23.1	429	29.3	991.910	945.394
2	1039	1984	65.3	19.4	848	34.3	728.646	711.832
3	1039	1985	64.2	27.6	888	27.8	644.544	799.323
4	1039	1986	45.0	24.5	546	22.5	526.135	551.347

```
[ ] df.tail()
```

	stco	year	corn	soybean	cotton	wheat	ppt1	ppt2
12	1039	1994	89.8	26.7	631	38.3	1149.777	1158.566
13	1039	1995	78.0	27.0	507	45.0	844.837	1074.552
14	1039	1996	75.0	36.0	672	50.0	866.788	809.495
15	1039	1997	87.0	24.0	688	40.0	673.559	758.928
16	1039	1998	32.0	22.0	520	44.0	871.234	1178.166

```
[ ] df['year'] = pd.to_datetime(df.year , format = '%d/%m/%Y %H.%M.%S')
    # df.set_index('', inplace = True)
    data = df.drop(['year'], axis=1)
    data.index = df.year
    data.head()
```

year	corn	soybean	cotton	wheat	ppt1	ppt2
1982-01-01	45.0	22.2	767	28.8	688.315	711.813
1983-01-01	56.6	23.1	429	29.3	991.910	945.394
1984-01-01	65.3	19.4	848	34.3	728.646	711.832
1985-01-01	64.2	27.6	888	27.8	644.544	799.323
1986-01-01	45.0	24.5	546	22.5	526.135	551.347

```
[ ] df.columns
    data.columns
```

```
Index(['corn', 'soybean', 'cotton', 'wheat', 'ppt1', 'ppt2'], dtype='object')
```

```
[ ] column = data.columns
    for a in column:
        for b in range(0,len(data)):
           if data[a][b] == -200:
               data[a][b] = data[a][b-1]
```

```
[ ] train = data[:int(0.8*(len(data)))]
    valid = data[int(0.8*(len(data))):]

    from statsmodels.tsa.vector_ar.var_model import VAR

    model = VAR(endog=train)
    model_fit = model.fit()

    prediction = model_fit.forecast(model_fit.y, steps=len(valid))
```

```
[ ] prediction
```

```
array([[ 62.87484536,  16.64905877, 697.29376698,  35.16002962,
         719.78895983, 668.33024152],
       [ 55.26554321,  23.83744251, 732.80274636,  25.79855211,
         863.0759682 , 877.04998169],
       [ 75.66375842,  22.06930797, 693.18032567,  33.34158968,
         836.34649052, 800.03819184],
       [ 58.67245898,  22.31119018, 723.64534358,  31.63338565,
         752.29245977, 754.9293395 ]])
```

```
prediction
```

```
array([[ 62.87484536,  16.64905877, 697.29376698,  35.16002962,
        719.78895983, 668.33024152],
       [ 55.26554321,  23.83744251, 732.80274636,  25.79855211,
        863.0759682 , 877.04998169],
       [ 75.66375842,  22.06930797, 693.18032567,  33.34158968,
        836.34649052, 800.03819184],
       [ 58.67245898,  22.31119018, 723.64534358,  31.63338565,
        752.29245977, 754.9293395 ]])
```

```
import math
from sklearn.metrics import mean_squared_error

pred = pd.DataFrame(index=range(0,len(prediction)),columns=data.columns)
for b in range(0,6):
    for a in range(0, len(prediction)):
       pred.iloc[a][b] = prediction[a][b]

for i in data.columns:
    print('rmse value for', i, 'is : ', math.sqrt(mean_squared_error(pred[i], copy[i])))
```

```
rmse value for corn is :  19.092862074772455
rmse value for soybean is :  8.045094777660696
rmse value for cotton is :  142.65951900555157
rmse value for wheat is :  14.83073392809225
rmse value for ppt1 is :  118.63562765240914
rmse value for ppt2 is :  295.9725963996251
```

```
copy = valid.copy()
# copy.head()
copy = copy.reset_index(drop=True)
print(copy.head())
```

```
   corn  soybean  cotton  wheat     ppt1      ppt2
0  78.0     27.0     507   45.0  844.837  1074.552
1  75.0     36.0     672   50.0  866.788   809.495
2  87.0     24.0     688   40.0  673.559   758.928
3  32.0     22.0     520   44.0  871.234  1178.166
```

```
print(valid.head())
pred.head()
```

```
            corn  soybean  cotton  wheat     ppt1      ppt2
year
1995-01-01  78.0     27.0     507   45.0  844.837  1074.552
1996-01-01  75.0     36.0     672   50.0  866.788   809.495
1997-01-01  87.0     24.0     688   40.0  673.559   758.928
1998-01-01  32.0     22.0     520   44.0  871.234  1178.166
```

	corn	soybean	cotton	wheat	ppt1	ppt2
0	62.8748	16.6491	697.294	35.16	719.789	668.33
1	55.2655	23.8374	732.803	25.7986	863.076	877.05
2	75.6638	22.0693	693.18	33.3416	836.346	800.038
3	58.6725	22.3112	723.645	31.6334	752.292	754.929

CLASSIFICATION

DATASET

Has the following attributes :

Date, Temperature, Humidity, Light, CO2, HumidityRatio, Occupancy

APPROACH

We have performed 1NN classification on the time series data using DTW as the distance metric with a combination of a distance-metric adopted from LB_Keogh.

```python
import pandas as pd
import numpy as np
import matplotlib.pylab as plt
import math
from sklearn.metrics import classification_report


def DTWDistance(s1, s2,w):
    DTW={}

    w = max(w, abs(len(s1)-len(s2)))

    for i in range(-1,len(s1)):
        for j in range(-1,len(s2)):
            DTW[(i, j)] = float('inf')
    DTW[(-1, -1)] = 0

    for i in range(len(s1)):
        for j in range(max(0, i-w), min(len(s2), i+w)):
            dist= (s1[i]-s2[j])**2
            DTW[(i, j)] = dist + min(DTW[(i-1, j)],DTW[(i, j-1)], DTW[(i-1, j-1)])

    return math.sqrt(DTW[len(s1)-1, len(s2)-1])


def euclid_dist(t1,t2):
    return math.sqrt(sum((t1-t2)**2))

def funcc(s1,s2,r):
    LB_sum=0
    for ind,i in enumerate(s1):

        lower_bound=min(s2[(ind-r if ind-r>=0 else 0):(ind+r)])
        upper_bound=max(s2[(ind-r if ind-r>=0 else 0):(ind+r)])

        if i>upper_bound:
            LB_sum=LB_sum+(i-upper_bound)**2
        elif i<lower_bound:
            LB_sum=LB_sum+(i-lower_bound)**2

    return math.sqrt(LB_sum)
```

```
ls =[]
def knn(train,test,w):
    preds=[]
    for ind,i in enumerate(test):
        min_dist=float('inf')
        closest_seq=[]
        #print ind
        for j in train:
            if funcc(i[:-1],j[:-1],5)<min_dist:
                dist=DTWDistance(i[:-1],j[:-1],w)
                if dist<min_dist:
                    min_dist=dist
                    closest_seq=j
                    print(closest_seq[-1])
                    ls.append(closest_seq[-1])
                    preds.append(closest_seq[-1])
    return classification_report(test[:,-1],preds)

def my_accuracy(y_pred, vl):
  cnt =0
  for j in range(len(y_pred)):
    if(y_pred[j] == vl[j]):
      cnt = cnt + 1
  return (cnt/len(y_pred)* 100)

train = np.genfromtxt('fdstrain.csv', delimiter=',')
test = np.genfromtxt('fdstest.csv', delimiter=',')

print(knn(train,test[:,0:5],4))
```

Form of results-

Class 0- Not occupied
Class 1- Occupied

```
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
0.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0
1.0
0.0
0.0
0.0
0.0
0.0
0.0
```