SI 671/721 (Fall 2021) Data Mining: Methods and Applications

Instructor: Paramveer Dhillon
GSIs: Anmol Panda (anmolp@umich.edu), Yulin Yu (yulinyu@umich.edu)
Homework 2: Mining Time Series - do the top 5 countries with the most cumulative COVID-19 cases demonstrate similar patterns?
Due: 10/28/2021 11:59 PM ET
Total Points: 100 points

Summary

We will continue to explore the data we used in the Time Series lab from the Johns Hopkins University CSSE COVID-19 dataset. However, this time, we are interested in the number of daily new cases **exclusively from the top 5 countries that have the most cumulative cases as of August 21, 2020**.

To explore and analyze this dataset, this assignment will focus on extracting the seasonal component from the countries' time series, computing the similarity between them, and calculating the Dynamic Time Warping (DTW) Cost.

Details

Data

For this assignment, we will be reusing the ***time_series_covid19_confirmed_global.csv*** file from the Time Series lab.

Packages

We recommend using the following Python packages in this assignment:

- numpy
- pandas
- matplotlib
- statsmodels
- math

Assignment Structure

This homework is divided into the following parts:

Part 1: Load & Transform the Data

Part 2: Extract Seasonal Components
Part 3: Time Series Similarities
Part 4: Dynamic Time Warping (DTW) Cost

1. Load & Transform the Data [20 points]

a) **[15 points]** To begin, create a function called *`load_data`* that reads in the csv file and produces a *`pd.DataFrame`* that looks like:

	?	?	?	?	?
2020-01-23	0.0	0.0	0.0	0.0	0.0
2020-01-24	1.0	0.0	0.0	0.0	0.0
2020-01-25	0.0	0.0	0.0	0.0	0.0
2020-01-26	3.0	0.0	0.0	0.0	0.0
2020-01-27	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...
2020-08-17	35112.0	19373.0	55018.0	4839.0	2541.0
2020-08-18	44091.0	47784.0	64572.0	4718.0	2258.0
2020-08-19	47408.0	49298.0	69672.0	4790.0	3916.0
2020-08-20	44023.0	45323.0	68900.0	4767.0	3880.0
2020-08-21	48693.0	30355.0	69876.0	4838.0	3398.0

where

- the index of the DataFrame is a *`pd.DatetimeIndex`*;
- the column names "?" are the top 5 countries with the most cumulative cases as of August 21, 2020, sorted in descending order from left to right;
- the values of the DataFrame are daily new cases; and
- the DataFrame doesn't contain any `NaN` values.

This function should return a *`pd.DataFrame`* of shape (212, 5), whose index is a *`pd.DatetimeIndex`* and whose column labels are the top 5 countries.

b) **[5 points]** Then, using your newly created *'load_data'* function, plot one line for each country that is in the top 5 for most cumulative cases where the x-axis is the date and the y-axis is the number of cases. Please do so within one figure.

2. Extract Seasonal Components [15 points]

Recall from lecture and lab that an additive Seasonal Decomposition decomposes a time series into the following components:

Y(t) = T(t) + S(t) + R(t)

where *T(t)* represents trends, *S(t)* represents seasonal patterns and *R(t)* represents residuals. In the rest of the assignment, we will work with the seasonal component *S(t)* to understand the similarities among the seasonal patterns of the five time series we have, so let's write a function that extracts this very seasonal component.

a) **[10 points]** Complete a function, *'sea_decomp',* that accepts a *`pd.DataFrame`* and returns another *`pd.DataFrame`* of the same shape that looks like:

	?	?	?	?	?
2020-01-23	2431.761670	3380.626554	441.179428	-54.886371	322.986535
2020-01-24	3446.796153	3457.641332	621.396176	23.689984	362.434811
2020-01-25	578.564626	586.665963	594.066127	55.034811	391.346141
2020-01-26	-2728.454422	-6031.950950	46.655454	137.908703	76.880131
2020-01-27	-3293.854422	-7144.674760	-1234.673118	1.842036	-507.496059
...	...	...	...	...	...
2020-08-17	-3293.854422	-7144.674760	-1234.673118	1.842036	-507.496059
2020-08-18	-719.521088	1549.577621	-544.749308	-28.929392	-662.877011
2020-08-19	284.707483	4202.114239	76.125240	-134.659770	16.725452
2020-08-20	2431.761670	3380.626554	441.179428	-54.886371	322.986535
2020-08-21	3446.796153	3457.641332	621.396176	23.689984	362.434811

where

- the index of the DataFrame is a *`pd.DatetimeIndex`*;
- the column names "?" are the top 5 countries with the most cumulative cases as of August 21, 2020, sorted in descending order from left to right;
- the values of the DataFrame are the seasonal components *S(t)* as returned by the *`seasonal_decompose`* function from the *`statsmodels`* package; and
- the DataFrame doesn't contain any `NaN` values.

This function should return a `*pd.DataFrame*` of shape (len(df), 5), whose index is a `*pd.DatetimeIndex*` and whose column labels are the top 5 countries.

b) **[5 points]** Then, using this function, please plot one line for each country in the top 5 showing the seasonal component - you should have a total of 5 line graphs where the x-axis is the date and the y-axis is the seasonal component.

3. Time Series Similarities [40 points]

3.1 Euclidean Distance [20 points]

Now, we may start to ask questions like, "which country in the top 5 countries are the most similar to Country A in terms of seasonal patterns?". In addition to the seasonal components that reflect seasonal patterns, we also need a measure of similarity between two time series in order to answer questions like this. One of such measures is the good old Euclidean Distance.

Recall that the Euclidean Distance between two vectors *x* and *y* is the length of the vector *x* - *y*:

$$\text{EucDist}(x, y) = \|x - y\|_2 = \sqrt{(x-y)^T (x-y)} = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

a) **[15 points]** Complete a function, *'calc_euclidean_dist',* that accepts a `*pd.DataFrame*`, whose columns are time series for each country, and that returns all pairwise Euclidean Distance among these time series, similar to the following:

	?	?	?	?	?
?	0.000000	233760.757213			
?	233760.757213	0.000000			
?			0.000000		
?				0.000000	
?					0.000000

where

- the index and the column names "?" are the top 5 countries with the most cumulative cases as of August 21, 2020, sorted in descending order from top to bottom and from left to right; and

- the values of the DataFrame are pairwise Euclidean Distance, for example, `233760.757213` is the Euclidean Distance between the time series of the Rank 1 country and the Rank 2 country

This function should return a `*pd.DataFrame*` of shape (5, 5) whose index and column labels are the top 5 countries.

b) **[5 points]** Then, use this new function to calculate the pairwise Euclidean Distance matrix for the extracted seasonal components from the top 5 countries with the most cumulative cases.

3.2 Cosine Similarity [20 points]

Another commonly used similarity measure is the Cosine Similarity. Recall that the Cosine Similarity between two vectors *x* and *y* is the cosine of the angle between *x* and *y*:

$$\text{CosSim}(x, y) = \frac{x^T y}{\|x\|_2 \|y\|_2} = \left(\frac{x}{\|x\|_2}\right)^T \left(\frac{y}{\|y\|_2}\right)$$

a) **[15 points]** Complete a function, *'calc_cos_sim'*, that accepts a *`pd.DataFrame`,* whose columns are the time series for each country, and that returns all pairwise Cosine Similarity among these time series, similar to the following:

	?	?	?	?	?
?	1.000000	0.898664			
?	0.898664	1.000000			
?			1.000000		
?				1.000000	
?					1.000000

where

- the index and the column names "?" are the top 5 countries with the most cumulative cases as of August 21, 2020, sorted in descending order from top to bottom and from left to right; and

- the values of the DataFrame are pairwise Cosine Similarity, for example, `0.898664` is the Cosine Similarity between the time series of the Rank 1 country and the Rank 2 country

This function should return a `*pd.DataFrame*` of shape (5, 5), whose index and column labels are the top 5 countries.

b) **[5 points]** Now, use this new function to calculate the pairwise Cosine Similarity between seasonal patterns.

4. Dynamic Time Warping (DTW) Cost [25 points]

4.1 Define a Function to Calculate DTW Cost [10 points]

Last but not least, the cost of aligning two time series can also be used as a similarity measure. Two time series are more similar if it incurs less cost to align them. One of the commonly used alignment costs is the Dynamic Time Warping (DTW) cost, which we will explore in this problem.

Recall from lecture that the DTW cost is defined by the following recursive relations:

$$\mathrm{DTW}(1,1) = d(x_1, y_1)$$

$$\mathrm{DTW}(i,j) = d(x_i, y_j) + \min \begin{cases} \mathrm{DTW}(i, j-1) & \text{Repeat } x_i \\ \mathrm{DTW}(i-1, j) & \text{Repeat } y_j \\ \mathrm{DTW}(i-1, j-1) & \text{Both proceed} \end{cases}$$

where we define $d(x_i, y_j) = (x_i, y_j)^2$.

a) **[10 points]** With reference to the demo of the DTW algorithm in the lecture slides, implement a function, *'calc_pairwise_dtw_cost',* below that computes the DTW cost for two time series. **We don't take the square root of the results just yet, until later when we compare the DTW costs with the Euclidean Distance.**

This function should EITHER return a `*np.ndarray*` of shape (len(y), len(x)) which represents the DTW cost matrix, OR a single `*float*` that represents the overall DTW cost, depending whether the parameter `*ret_matrix=True*`.

4.2 Compute Pairwise DTW Cost [15 points]

Now let's compute all pairwise DTW costs for our five time series.

a) **[10 points]** Implement a function, *'calc_dtw_cost',* below that accepts a *`pd.DataFrame`*, whose columns are the time series for each country, and that returns all pairwise DTW costs among these time series, similar to the following:

	?	?	?	?	?
?	0.000000e+00	9.575974e+09			
?	9.575974e+09	0.000000e+00			
?			0.000000e+00		
?				0.000000e+00	
?					0.000000e+00

where

- the index and the column names "?" are the top 5 countries with the most cumulative cases as of August 21, 2020, sorted in descending order from top to bottom and from left to right; and
- the values of the DataFrame are pairwise DTW costs, for example, `9.575974e+09` is the DTW cost between the time series of the Rank 1 country and the Rank 2 country

This function should return a *`pd.DataFrame`* of shape (5, 5), whose index and column labels are the top 5 countries.

b) [**5 points]** Now, use this function to calculate the pairwise DTW costs between seasonal patterns. **Please take the square root so that we can compare it with the Euclidean Distance.**

What can you say about the similarities among these seasonal patterns? Do the results of the pairwise Euclidean Distance, Cosine Similarity and DTW Cost calculations tell the same story?

Submission

All submissions should be made electronically by 11:59 PM EST on 10/28/2021.

Here are the main deliverables:

- A PDF version of your executed Jupyter Notebook
- The actual Jupyter notebook, so that we can check your results

Please make sure to provide appropriate conclusions drawn from the code/results throughout the notebook.

Academic Honesty

Unless otherwise specified in the homework, all submitted work must be your own original work. Any excerpts, statements, or phrases from the work of others must be clearly identified as a quotation, and a proper citation provided. Any violation of the University's policies on Academic and Professional Integrity may result in serious penalties, which might range from failing a homework, to failing a course, to being expelled from the program. Violations of academic and professional integrity will be reported to the concerned authorities. Consequences of academic misconduct are determined by the faculty instructor; additional sanctions may be imposed.