ECM3420 Learning from Data (Year 3 only)
Course Work 2: Clustering

Submission Deadline: 12:00 midday on Wednesday 2nd December 2020 (Week 11)
Weight: 20%

Task 1: Incremental K-Means [25%]

Your task is to implement an incremental K-Means clustering algorithm. In the basic K-means algorithm, centroids are updated after all points are assigned to a centroid. In your incremental K-Means, centroids are update after each assignment (incremental approach).

- In the first/initial iteration:
  - The initial k centroids will equal to k randomly selected pints (the same point can't be selected more than once).
  - For each data point (from 0 to n-1 where n is the number data points):
    - Calculate the Euclidian distance between the point and each of the k centroids.
    - Assign the point to the nearest cluster and update the centroid of this cluster.
- For the next iterations:
  - Keep the cluster assignment from the previous iteration.
  - For each point (from 0 to n-1)
    - Calculate the Euclidian distance between the point and each of the k centroids.
    - Notice that this requires either zero or two updates to cluster centroids at each step now, since a point either moves to a new cluster (two updates) or stays in its current cluster (zero updates).
- Stopping conditions:
  - Maximum number of iterations (max_itr) is reached.
  - No centroid updates in one full iteration.
- You are expected to write your own code for the for the Incremental K-Means and for calculating the Euclidian distance (don't use sklearn or any other function for calculating the)
- You can use other libraries such as pandas, numpy, scipy, and matplotlib

- The output of this task is a function with the following definition:

```
incremental_kmeans(x, k, max_itr=100, random_state=None)
```

  - Parameters:
    - x: the data do be clustered (data points)
    - k: the number of clusters
    - max_itr: The maximum number of iterations
    - random_state: Determines random number generation for centroid initialization. Use an int to make the randomness deterministic.
  - Returns:

- `cluster_labels: the cluster membership labels for each element in the data x`
- `n_iter  int:  Number of iterations run`

Task 2: Incremental K-Means Run Time Analysis [20%]

Incremental updates are slightly more expensive. However, Incremental K-means could converge rather quickly, and therefore, the number of points switching clusters quickly becomes relatively small. In this tak you should compare your Incremental K-means to the standard K-means (sklearn) in terms of running time and number of iterations.

- Use the iris dataset (uisng sklearn.datasets.load_iris)
- Run your Incremental K-means and standard K-Means *m* times
  - Use *m=5*
  - Repeat the experiment using number of clusters $k = \{2, 3, 4, 5\}$
  - Use sklearn to run the standard K-Means.
  - For each run use a different random_state
  - For each single run, calculate/log:
    - The time in milliseconds needed to cluster the data for each algorithm. Use the "time" package (Don not use "timeit"). Record the time before and after running the clustering algorithm (log the difference in time).
    - The number of iterations needed.
- Use the "best practice" when applying clustering algorithms in general.
- For the standard k-mena use sklearn.
- The output of this task is the fowling:

  - One table similar to the following [10%]:

		K=2	K=3	K=4	K=5
Inc K-Means	Average time				
	Average Iterations				
Std K-Means	Average time				
	Average Iterations				

  - One chart showing the run time of your Incremental K-means; x-access is the k value, y-access is the time in milliseconds. For each k value on the x-access plot a box-plot showing the "run time" distribution of the m runs [10%].

Task 3: Cluster Evaluation [10%]

- Write a function to calculate the "Jaccard score". Don't use sklearn or any other packagde for calculating the Jaccard score. You can use other libraries such as pandas and numpy.
- The output of this task is a function with the following definition [15]:

```
jaccard_index_cw2(y_true, y_pred)
```

  - `Parameters:`

        - `y_true: 1d array-like, or label indicator array. Ground truth (correct) labels/clusters.`
        - `y_pred: 1d array like, or label indicator array. Predicted labels/clusters.`
    - `Returns:`
        - `Score: float`

Task 4: Incremental K-Means Cluster Analysis [45%]

Run your Incremental K-Means and Standard K-Means algorithms on the iris dataset (uisng sklearn.datasets.load_iris). Use K=3 (number of labels). Use the "best practice" when applying clustering algorithms in general.

Using iris labels/target - for each method, count how many data points (instances) from each label (iris target) belongs to each cluster as shown in the following tables:

Inc. K-Means	Label 1	Label 2	Label 3
Cluster 1	# instances		
Cluster 2	...		
Cluster 3			

Std. K-Means	Label 1	Label 2	Label 3
Cluster 1	# instances		
Cluster 2	...		
Cluster 3			

The output of this task is the following:

- The two tables shown above [10%].
- Calculate the "Jaccard score" using your function `jaccard_index_cw2` and using `sklearn jaccard_score`.
- Use the tables, scores (Jaccard or other scores) and results to compare the performance of K-Means and Standard K-Means in splitting the iris dataset when compared with the original labels.
    - Draw one chart of your choice. The mark for this chart will be based on the quality of your selected chart [15%].
    - Write up to 300 words to reflect on the results in this task of K-Means and Standard K-Means [20%].

Submission

You are expected to complete this assessment using a Python 3 and packages mentioned in each task.

You are expected to submit the following items:

- Use Python 3 and submit a single Jupyter notebook (please use the attached template).
- PDF version of the Jupiter notebook file.

Please submit your BART assignment here: https://bart.exeter.ac.uk/

It is vital that you check the time and date of your deadline on BART prior to your submission. Assignments that need to be submitted through BART must be done so using the link above. BART will then guide you through the steps you need to complete for your submission. Failure to do so will result in your mark being capped for a late submission. [Full details are on the ELE page]

Marking Scheme

- [25%] Task 1: Incremental K-Means
  - The quality of the code e.g. reusability, functions, comments, correctness, efficiency etc.
- [20%] Incremental K-Means Run Time Analysis
  - On the correctness and quality of the Python code and results
  - Quality, clarity and correctness of the output chart and tables.
- [10%] Cluster Evaluation
  - The quality of the code e.g. reusability, functions, comments, correctness, efficiency etc.
  - The correctness of the Jaccard scores show in task 4.
- [45%] Incremental K-Means Cluster Analysis:
  - The quality of the code e.g. reusability, functions, comments, correctness, efficiency etc.
  - Implementation of the experiment
  - Interpretation of the experiments and results in this task.
  - Reflection on the results.
  - Presentation style and references quality.