Learning Outcomes:

1. Understanding unsupervised learning algorithms.
2. Developing and Applying KMeans data clustering algorithm.
3. Analyzing results from KMeans clustering.
4. Converting features in lower dimension using PCA/SVD.
In this assignment, you will apply unsupervised learning to text data that you pre-processed in homework one. You will need to develop a clustering algorithm that will group in similar documents vectors your produced in homework one.

**Pre-requisites:** Review the notes from Chapter 7 in the course website and read chapter 6 from the PA for D book.

You must write your own *K-means clustering algorithm* that takes as input the data matrix of TF/IDF or a reduced matrix using PCA, you had in HW#1 and an integer K that specifies the number of clusters.

You will need another K-means pre-built in a library such as in order to compare you result: [Java Machine Learning Library](http://www.javaml.org).

It is preferable to use an Integrated Development Environment such as Eclipse to be able to import the Java library with no issues.

**Tasks:**

1. Develop a routine (method) that takes a document as an import, a similarity measure, and return the K most similar document to that document. The routine should take the name of the document as input and a similarity measure. The similarity measure could be either the *Cosine similarity* or the *Euclidian distance*.

   Apply the routine to three documents of your choice from one of the 15 folders and register your analysis on which distance is more accurate. You might need to call the routines your developed from HW#1 to convert the matrix into a vector of n terms. Report your results.

2. Implement a routine K-means clustering algorithm that takes as input the data matrix of word-document matrix, a similarity measure and a K (number of desired clusters) the algorithm will output the K clusters. It is up to you to design this routine in terms of inputs and outputs, as long as it performs K-means and outputs the results. See 6 on how you will apply 2.

3. You will need to use another K-means pre-built in a library such [Java Machine Learning Library](http://www.javaml.org) to compare your results.

4. **Kmeans++**
   
   **Determining Initial Centroid**

   The quality of the final clustering depends heavily centroid initialization. To get the hang of how it affects the results, visit this [this visualization](http://www.someurl.com).

   You are expected to use a Java library for this. Kmeans++ algorithm should only be used for determining the initial centroids. Since you understood how critical it is to use to the right initial centroid.

   In Kmeans++ the first cluster center is chosen uniformly at random from the data points that are being clustered, after which each subsequent cluster center is chosen from the remaining data points with probability proportional to its squared distance from the point's closest existing cluster center.
5. **Plot the Clusters**

Visualize the final clusters. Give different color to each cluster.

**Hint:** How do we convert N-dimensions to 2D plot (you may want to apply PCA or SVD)

6. **Measuring Model Performance**

In this case, we know which documents belongs to which group, so measure the performance of the KMeans model using *Confusion Matrix* and print out the accuracy of the model along with the confusion matrix. You might use cluster analysis, silhouette score or **precision and recall**/F-measure to measure the performance of your clustering algorithm. Here are the scenarios you would apply Kmeans to and measure the performance for:

- Apply SVD to your matrix from HW#1. Register extracted insights.
- Apply Kmeans to all the documents (full matrix) with K=2 for document from two different folders of your choice with Cosine Distance.
- Apply Kmeans to all the documents (reduced matrix with PCA) with K=2 for document from two different folders of your choice with Cosine Distance.
- Apply Kmeans to all the documents with (full matrix) K=3 for document from three different folders of your choice with Cosine Distance.
- Apply Kmeans to all the documents with (full matrix) K=2 with Cosine Distance.
- Apply Kmeans to all the documents with(full matrix) K=15 with Euclidean Distance.
- Apply Kmeans to all the documents with (full matrix) K=15 with Cosine Distance.
- Apply Kmeans to all the documents with(reduced matrix with PCA) K=15 with Euclidean Distance.
- Apply Kmeans to all the documents with (reduced matrix with PCA) K=15 with Cosine Distance.

If the results are not very good, you will have a chance to review the matrix you generated in HW1.

You must use Java. You can use library for PCA and SVD, Kmeans++ including and not limited to:

- [http://commons.apache.org/proper/commons-math/download_math.cgi](http://commons.apache.org/proper/commons-math/download_math.cgi)