You will be creating clusters for the data from the data quality report to try to identify potentially malicious network traffic.

Here is the data file:  [kddcup.txtPreview the documentView in a new window](https://psu.instructure.com/courses/1867326/files/87323248/download?verifier=IlBqfym3XrZn1zu7hzN5aLpxKGADq2J5stTcsSPp&wrap=1)

Do the following steps:

1) Convert the string fields to dummy fields using Pandas' get\_dummies function, i.e.:

df = pd.get\_dummies(df)

2) Normalize the data using 0-1 normalization.

3) Using MiniBatchKMeans with a batch size of at least 5,000, try to identify the point at which the within-cluster sum of square distances plot experiences "an elbow".  Use the associated k value for the remaining questions.

Note: If kmeans is your MiniBatchKMeans object, kmeans.inertia\_ will give you the within-cluster sum of squared distances.

4) Assign each record to a cluster for the k value identified in step 3.

5) From our Data Quality Report, we saw that 0 is the mode for most of the attributes (in fact it was the most common value by far for most attributes).  Therefore large attribute values are likely to be more interesting to us than small ones.  For each cluster, find the attributes whose mean values for the cluster are 0.9 or greater.

Note:  If kmeans is your MiniBatchKMeans object, kmeans.cluster\_centers\_ will give you a numpy array with the mean values for each attribute for observations in the cluster.  These values will be specified in the order of the columns in the dataframe.  You can easily get the three largest attribute values with commands like the following which summarizes the first cluster:

pd.DataFrame({'columns':df.columns,'means':kmeans.cluster\_centers\_[0]}).sort\_values('means')

6) Make predictions about which of the clusters in part 5 would be predominantly malicious traffic (i.e. more than 99% of the packets in the cluster would be malicious.)  It is ok to be wrong here, but commit to your predictions before going further.

7) We can verify your predictions vs. the ground in this case.  Load the following file into a dataframe called dfTarget: [kddcup.target.txtPreview the documentView in a new window](https://psu.instructure.com/courses/1867326/files/87434229/download?verifier=Ea1tRBOmuicYrs7VWYvJRAwT1n3mL1BCdeex5Cj7&wrap=1)

Assign each of the values from the original dataset to a cluster, via the command:

cluster = kmeans.predict(df)

Now print a crosstab that lists rows with clusters, and columns with the different attacks:

print(pd.crosstab(cluster,dfTarget["target"]).to\_string())

8) Based on the crosstab in part 7, answer the following questions:

a) How many clusters are comprised of at least 99% malicious packets?  Does this track with your predictions in 6?

b) How many observations are in the clusters identified in part a.

For steps 9 and 10, you will repeat your analysis from steps 9 and 10, but with different prepocessing steps.

9) Rerun the clustering, with the same value for k, but with non-normalized data values.  How do the results, using the metrics from step 8, compare to the model that used the normalized values?

10) Draw a graph based on the correlated attributes that you identified in the previous assignment.  The nodes in the graph should be the attribute values, and an edge should be present iff two attributes were correlated (either positively or negatively).  Remove nodes (and the edges incident to the removed nodes) from the graph until the remaining nodes have no edges.  Now remove the attributes from the removed nodes from the dataframe, with code like:

drop\_cols = ['correlatedAttr1','correlatedAttr2',correlatedAttr3']

cols = [c for c in df.columns if c not in drop\_cols]

df = df[cols]

After normalizing df, rerun the cluster, with the same value for k.  How did removing the correlated attributes affect the performance of the clustering using the metrics from step 8?