In order for the features to be used by a machine learning algorithm, the features are transformed

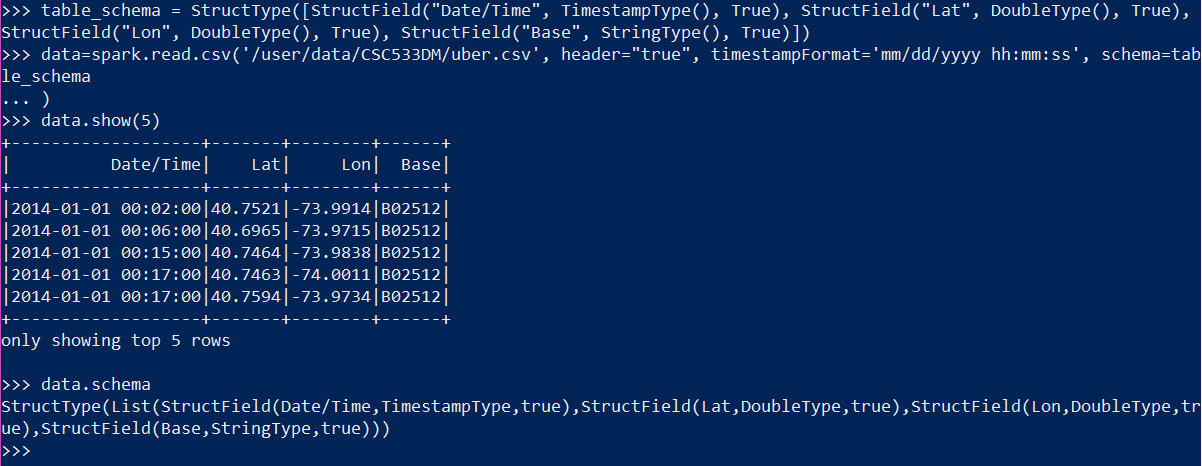
and put into Feature Vectors, which are vectors of numbers representing the value for each

feature. VectorAssemblers are used to transform and return a new DataFrame with two of

the feature columns in a vector column.

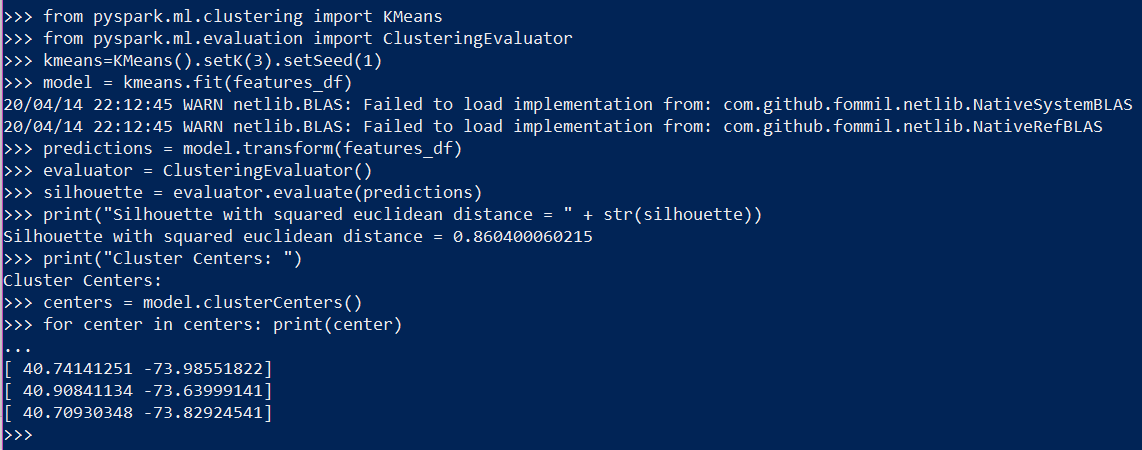
In this assignment, I have browsed through the clusters of Uber data based on longitude and latitude, then analyzing the cluster centers by date/time.

I have built three clustering (k-means) models with the Uber dataset, K=3, K=5, K=8, predicted & evaluated the models, metric used: silhouette, printed cluster centers for the models, compared and explained which one is best.

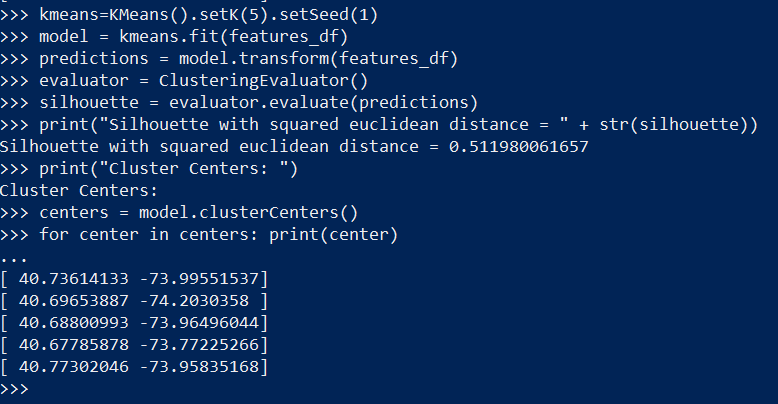


**Assignment 2:**

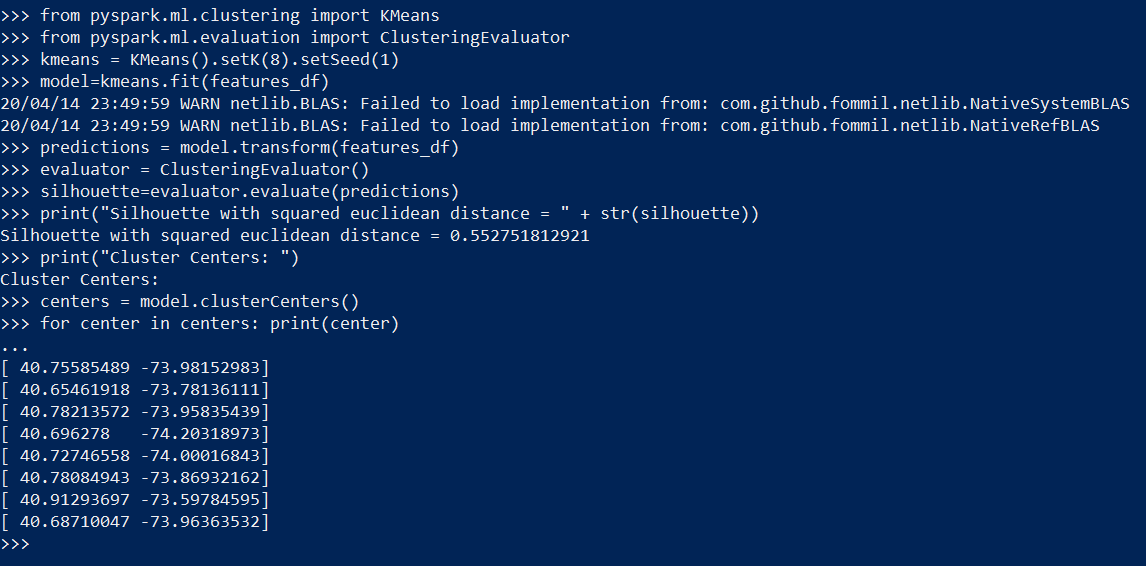
For K=3 :



For K=5 :



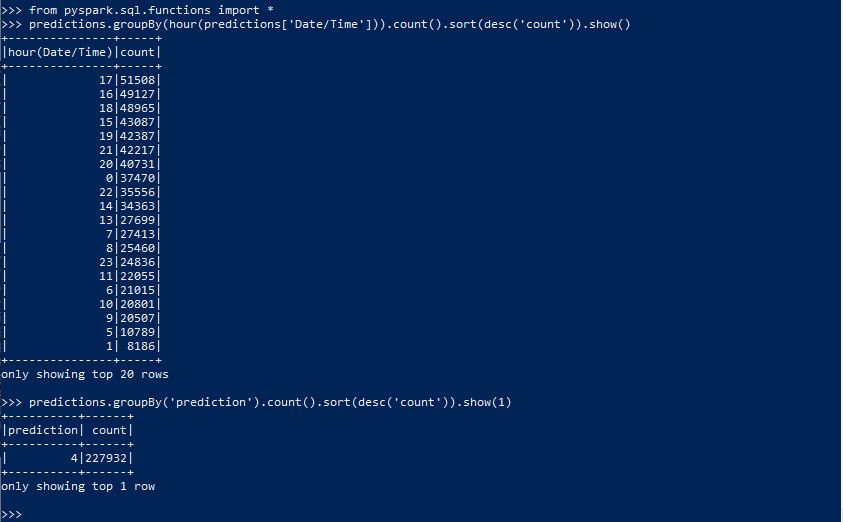
For K=8 :



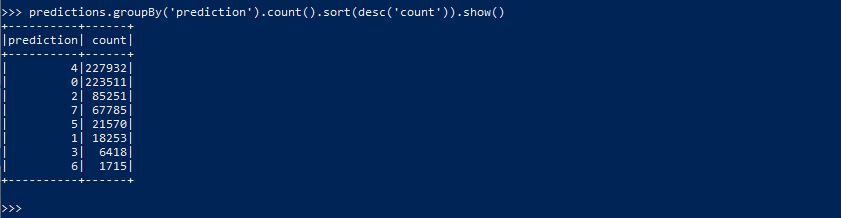
K=3 is the most efficient as it’s silhouette score is closest to 1, which is a sign of optimal clustering results.

Please note: Silhouette is a measure for the validation of the consistency within clusters. It ranges between 1 and -1, where a value close to 1 means that the points in a cluster are close to the other points in the same cluster and far from the points of the other clusters.

**Assignment 3:**



**Assignment 4:**



**Assignment 5:**

**K means++ is an optimized version of the traditional K means version.**

In K-Means, cluster centers are allocated randomly and more efficient solutions are then sought.

In K-Means++, one cluster center is allocated randomly, and other centers are searched for given the reference of the first one. Both approaches use initialization that is random to begin with, and therefore can present varying results with different runs.

K-Means++ can be thought of as an initialization procedure for K-Means. K-Means ++ works in a way that initial centroids are picked via an algorithm that optimizes centroids that are further away from each other. A random point is picked, which can serve to be the first centroid, and then another point is picked with respect to underlying probabilities that depend on the distance of the first point. The process is then repeated after achieving two centroids. The probablity of each point relies on the distance to the closest centroid in it’s vicinity.

This tends to introduce an overhead during algorithm initialization, but in retrospect tends to reduce the probability of a faulty initialization which would ultimately lead to an incorrect/poor quality clustering result.

**To summarize: The main difference between K-Means and K++ is that in K means++ where the after the first initialization, the centroids are optimized which aid in speedier convergence of the model which in turn reduces time for the algorithm to run, along with enhancing likelihood of optimum result quality.**

**This optimization is done by choosing centroids that are far from each other which increases probability. The result is then clustered with compact, tight-knit clusters that are apart from each other, and accentuates the speediness of the model to converge relatively quickly as it decreases the number of iterations performed.**

For the k-means algorithm, it is interesting to note that by choosing the initial cluster centers

carefully, we may be able to not only speed up the convergence of the algorithm, but also

guarantee the quality of the final clustering. The k-means++ algorithm is a variant of k-means,

which chooses the initial centers as follows. First, it selects one center uniformly at random from

the objects in the data set. Iteratively, for each object p other than the chosen center, it chooses

an object as the new center. This object is chosen at random with probability proportional to

dist(p)2, where dist(p)) is the distance from p, to the closest center that has already been chosen.

The iteration continues until k centers are selected.

Explain why this method will not only speed up the convergence of the k-means algorithm, but

also guarantee the quality of the final clustering results.

*(We are currently using scalable/parallel version of k-means++)*