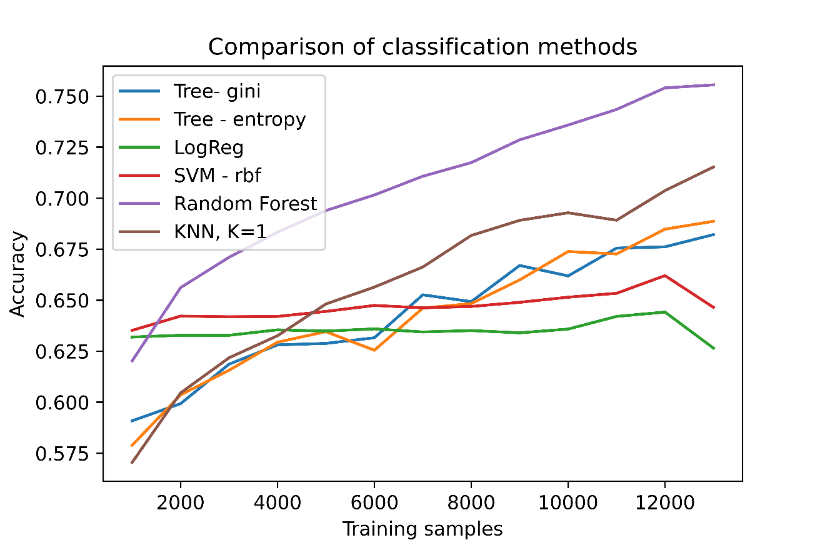
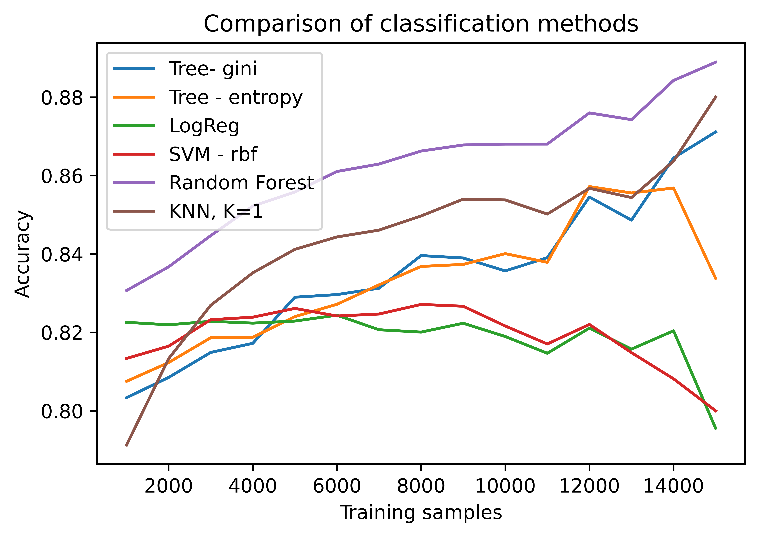
**Week 9 Writeup**

# RECAP: Week 8 Summary

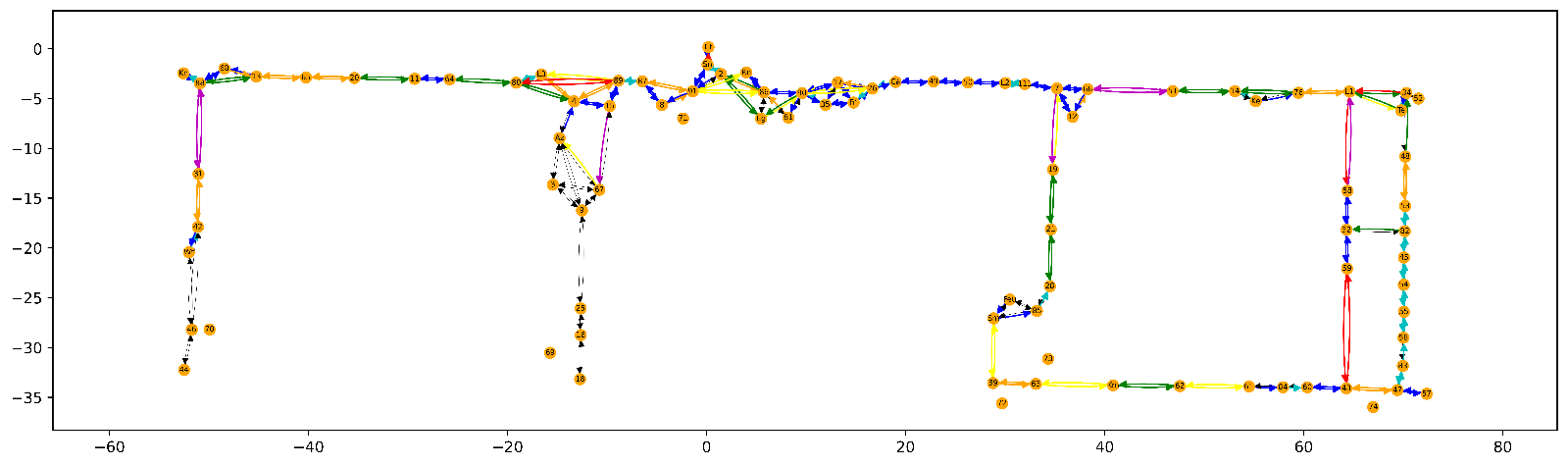
1. **[MON]** – Prior to last week’s meeting. [Train AAF, Test AAF]
2. **[TUE]** – Fix issues from week 7
   1. Blenheim congestion counting. Similar numbers – differences were due to counting congestion over the whole duration of an edge transition
   2. Anomalies in KS clustering: human effects or lack of data
   3. Effect of increasing no. of clusters on classification accuracy: accuracy increases, but improvement vs naïve classifier decreases.
3. **[WED]** – Can we merge data from all years? (AAF)
   1. Compare pdfs, cdfs, KS score: reasonable overlap
   2. Better performance when used for classification

**Left – merged data (AAF)** **Right – Y4 data (AAF)**

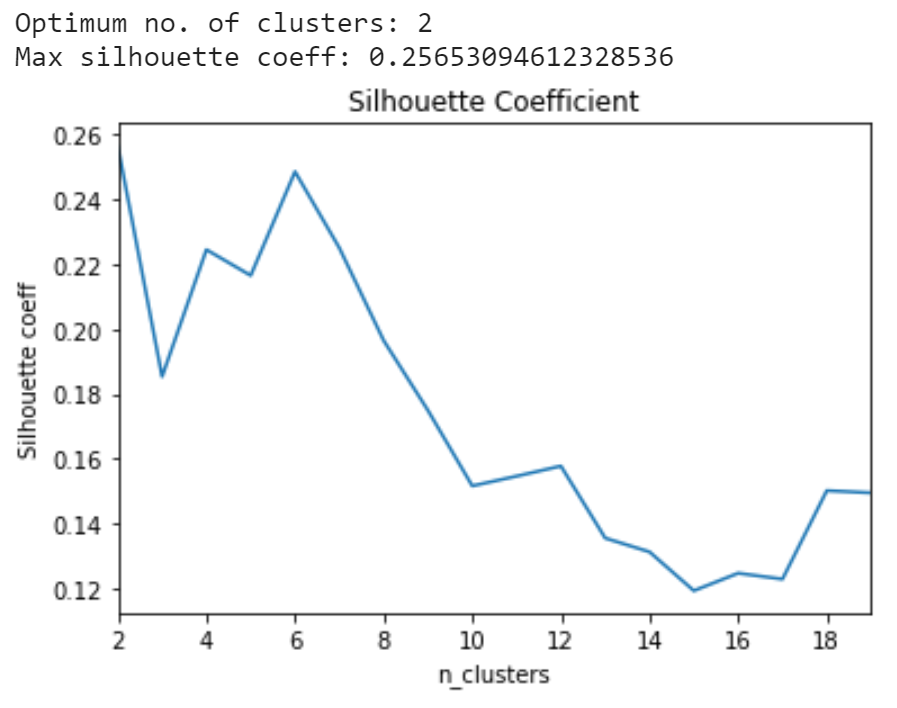
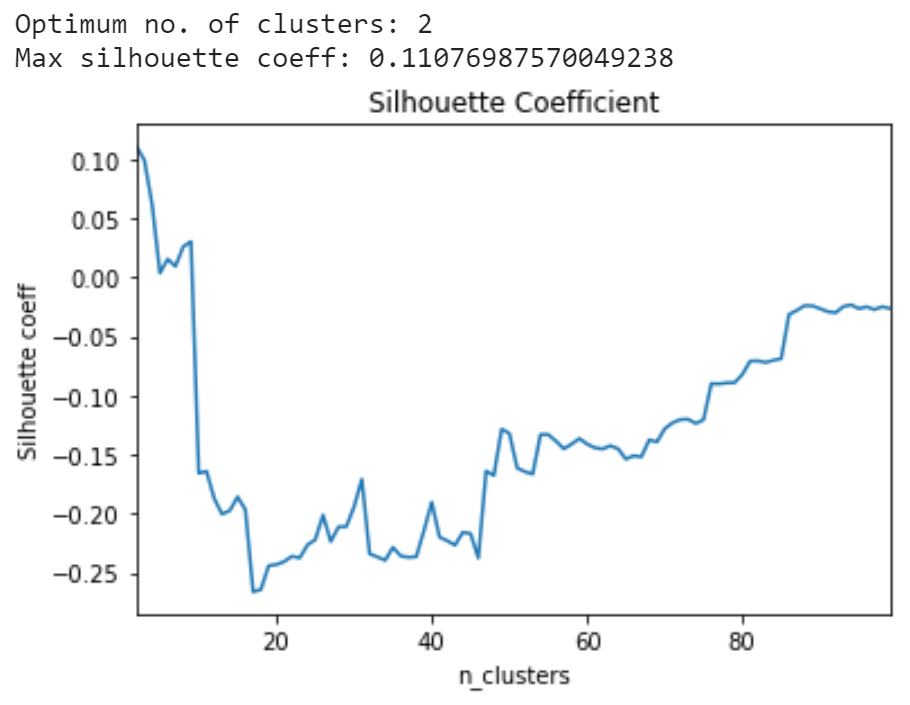
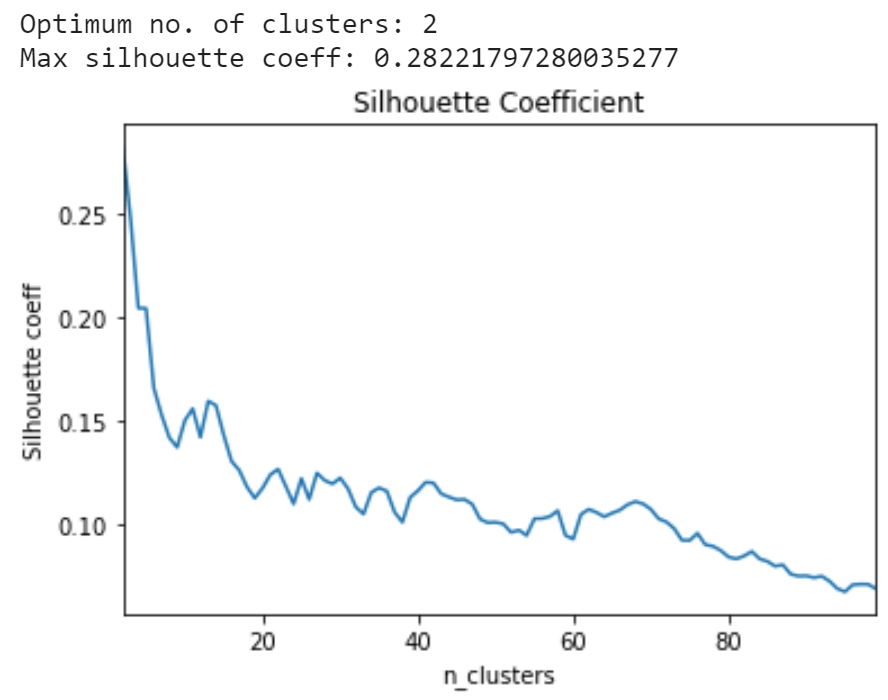


1. **[THU]** – Compare linkage metrics for HAC clustering
   1. Average & complete are similar. Single has more clusters in AAF
   2. Only average has clusters for TSC (other metrics suggest 1 cluster)
   3. Similar classification accuracy for “average”, “single”, “complete” in AAF

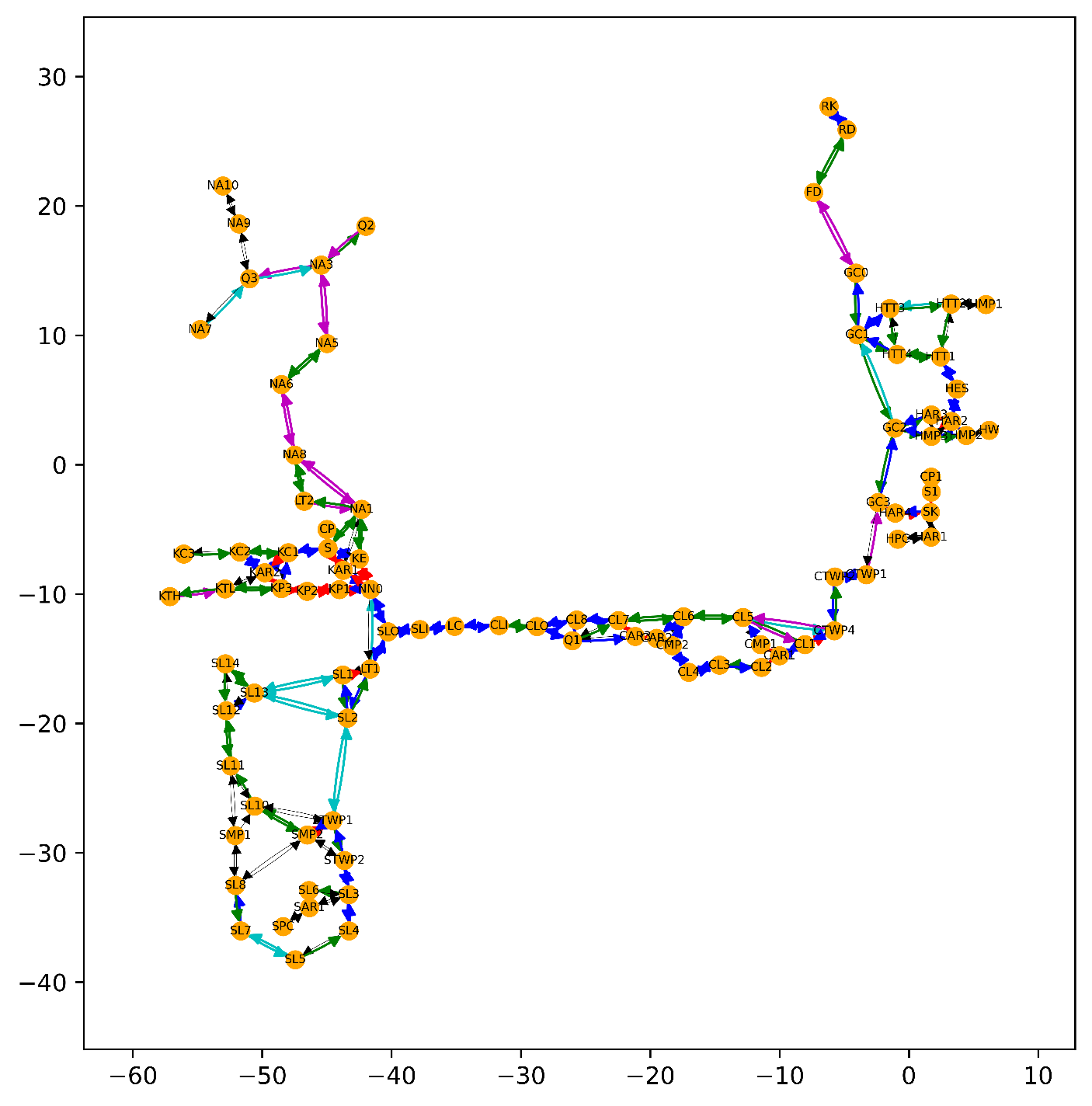
**AAF (average linkage) – 7 clusters**



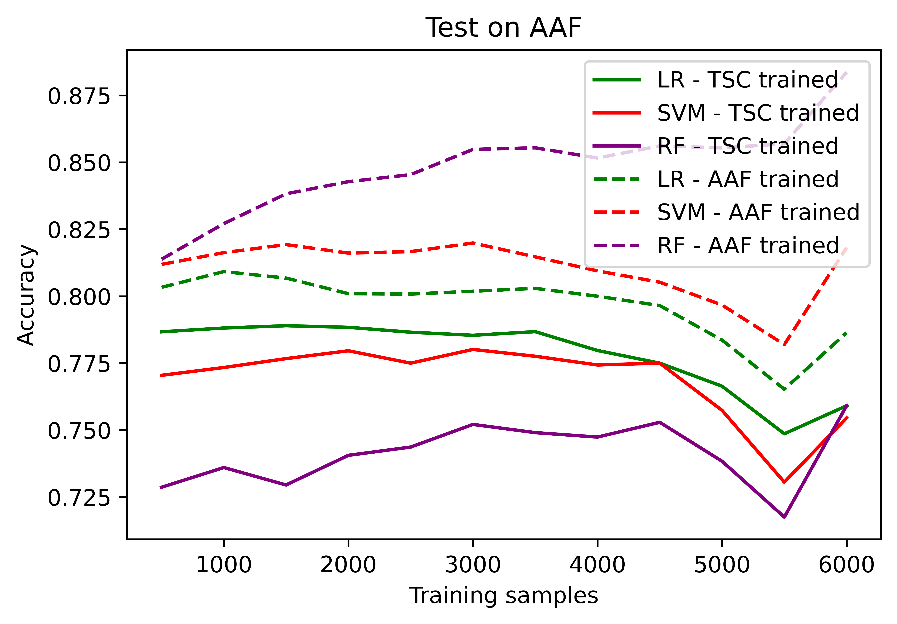
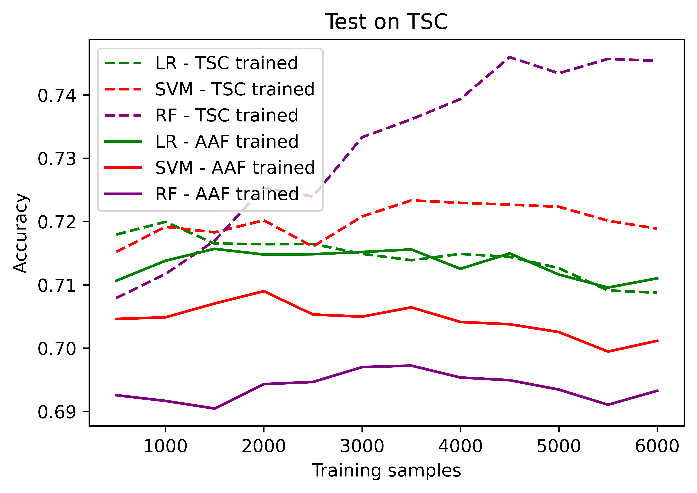
**Average (TSC): Single (TSC): Complete (TSC):**

**TSC (average metric) – 6 clusters**



1. **[FRI]** – Test on a different map to training dataset
   1. The three best classification methods are Logistic Regression (**LR**), Support Vector Machine (**SVM**), Random Forest (**RF**) when Train/test on DIFFERENT maps
   2. The three best classification methods are Random Forest (RF), Nearest-Neighbour (KNN),Decision Tree (DT) when Train/test on SAME map



# PLANS: Week 9

1. **[MON]** – Finish pipeline: Turn binary classification into clusters
2. **[TUE]** – Test performance by looking at sum/mean of KS distances using our generalisation methods. Compare to edge length clustering & KS clustering
3. **[WED]** – Try to improve performance of clustering and classifier algorithms
4. **[THU]** – Test binary classifier on Walmart 🡸🡺 Blenheim. Walmart 🡸🡺 AAF/TSC. Does this work?
5. **[FRI]** – Can you perform clustering or classification between maps (e.g. AAF & TSC) and directly transfer the data?
6. **[EXTRA]** – Plan alternative environments

Other considerations:

* Can we merge TSC data (esp Betty vs Bob)
* More features / more algorithms
* Walmart to Blenheim & vice versa
* Walmart to AAF/TSC – not expected to work with different robots
* Plan alternative environments
* Try Bayesian update without nice prior – e.g. johnson su
* Try using different GOF metrics to rank MLE distributions
* Complete linkage? Decide clusters not using silhouette score – use a more informed
  + Top-down clustering. Split from 1 cluster 🡺 2 clusters 🡺 3 clusters etc
  + Check whether the complete linkage is greater than a KS cutoff
    - To set KS cutoff – try visually
  + Codebook vectors (no. of clusters). Quantisation error is metric to determine performance
    - Sum of distances to cluster centre in training set
  + Check what is the max KS distance between any 2 items in a cluster
  + Preprocessing step to remove high KS / low probability
* Alternative validation step: check KS scores between edges
  + Compare generalised distribution vs fitted distribution
  + Compare against 3 different types of clustering:
    - edge length clustering (needs to beat this one)
    - KS-based clusters
    - Learnt classifier clusters
  + Sum/mean of KS distances over all edges
* For better generalisation: train on multiple maps & test on one
* Does evaluation require new maps

1. Finish pipeline
2. Test performance
3. Tweak clustering / classifier
4. Binary classifier between maps. Can you transfer data from one edge to another
   1. Be careful how we’re generating training maps
   2. Maybe ignore clustering and use a KS-distance threshold
   3. Careful: certain maps use different robots
5. New maps

Next Monday is bank holiday 🡺 Change to Tues 31

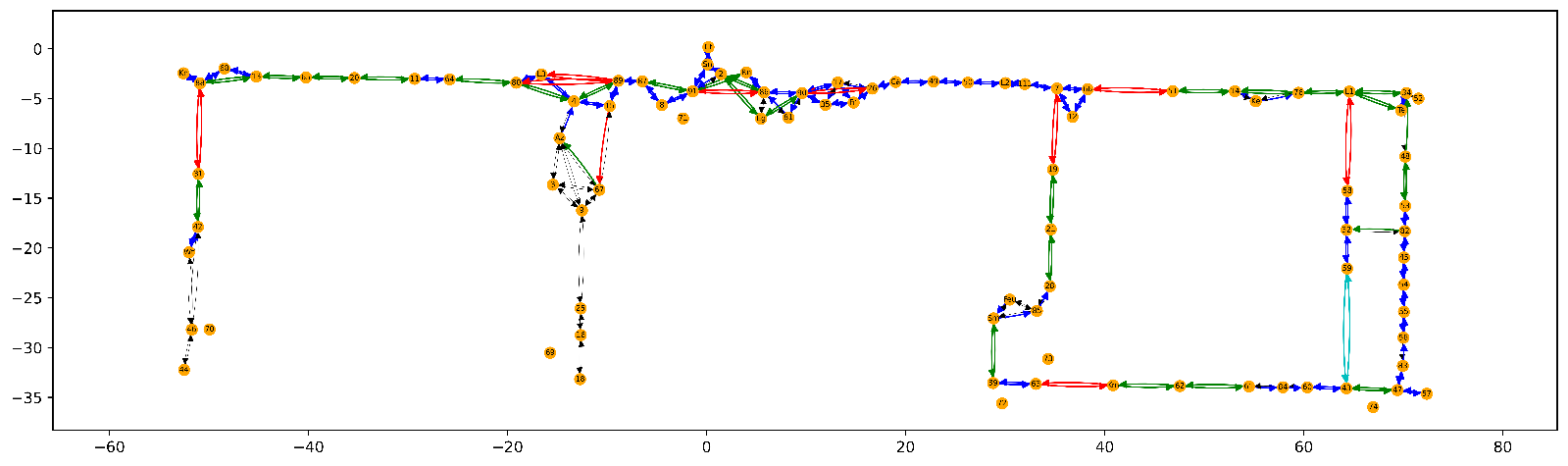
# TUE: Compare against different methods Train on TSC, Test on AAF

3 methods:

1. Binary classifier
2. Edge length
3. KS stat

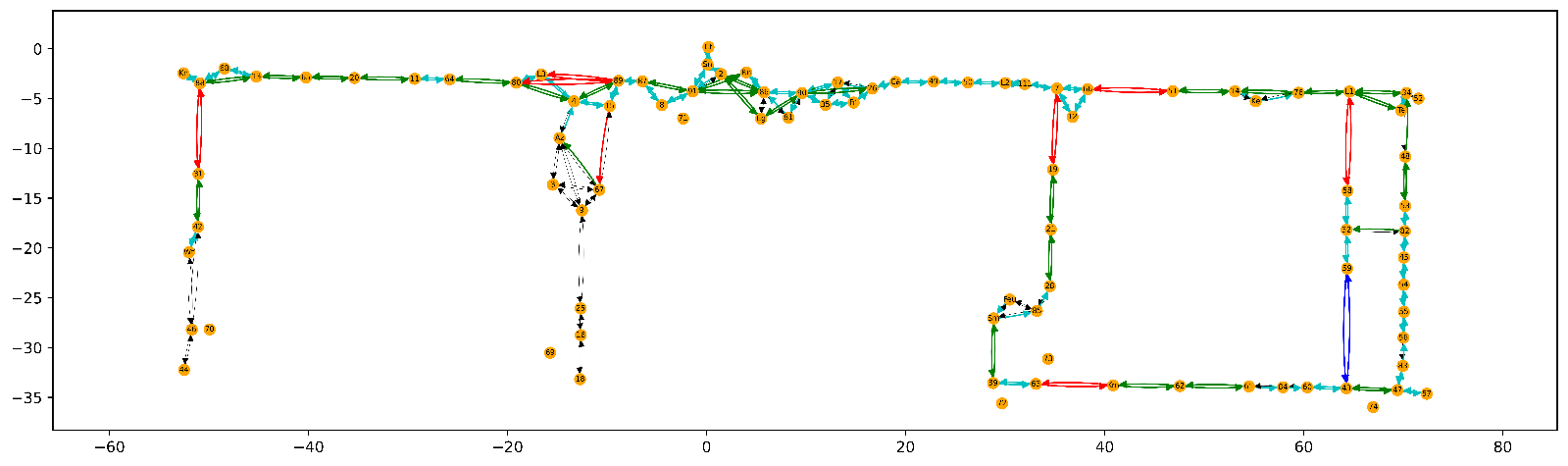
#### Binary classifier

Clusters\_dist\_aaf.png



#### Edge length

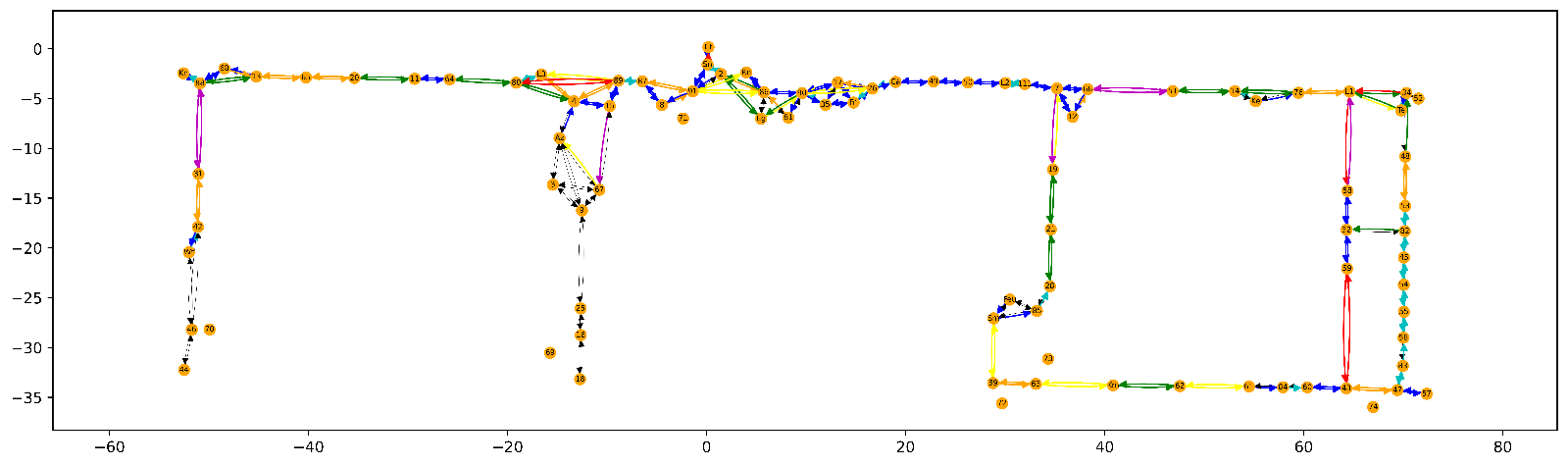
Clusters\_length\_aaf.png: cluster 0 (red), cluster 1 (green), cluster 2 (blue), cluster 3 (cyan)



Identical except for 91\_86, 90\_26

#### KS stat

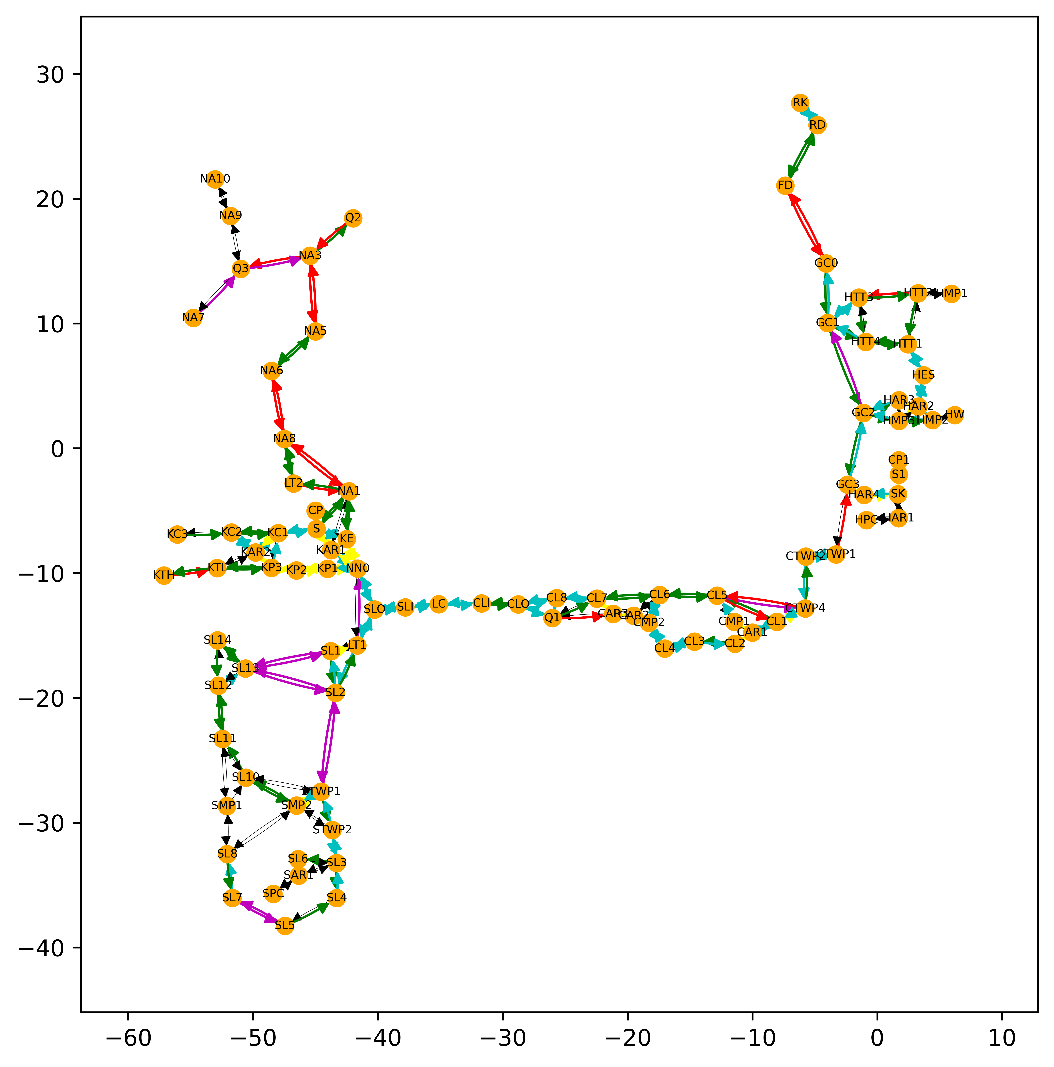
Clusters\_ks\_aaf.png:



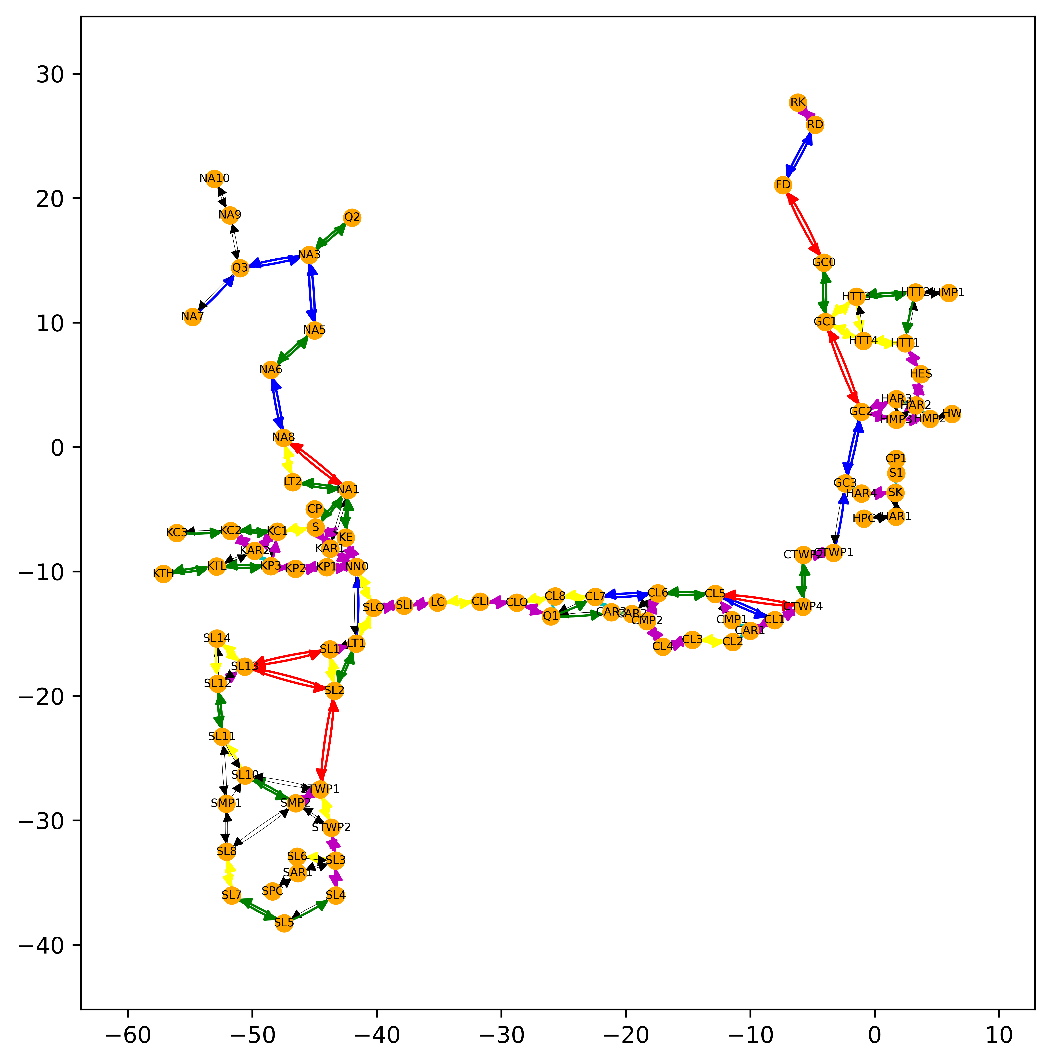
More clusters than edge\_length or binary classifier

## Train on AAF, Test on TSC

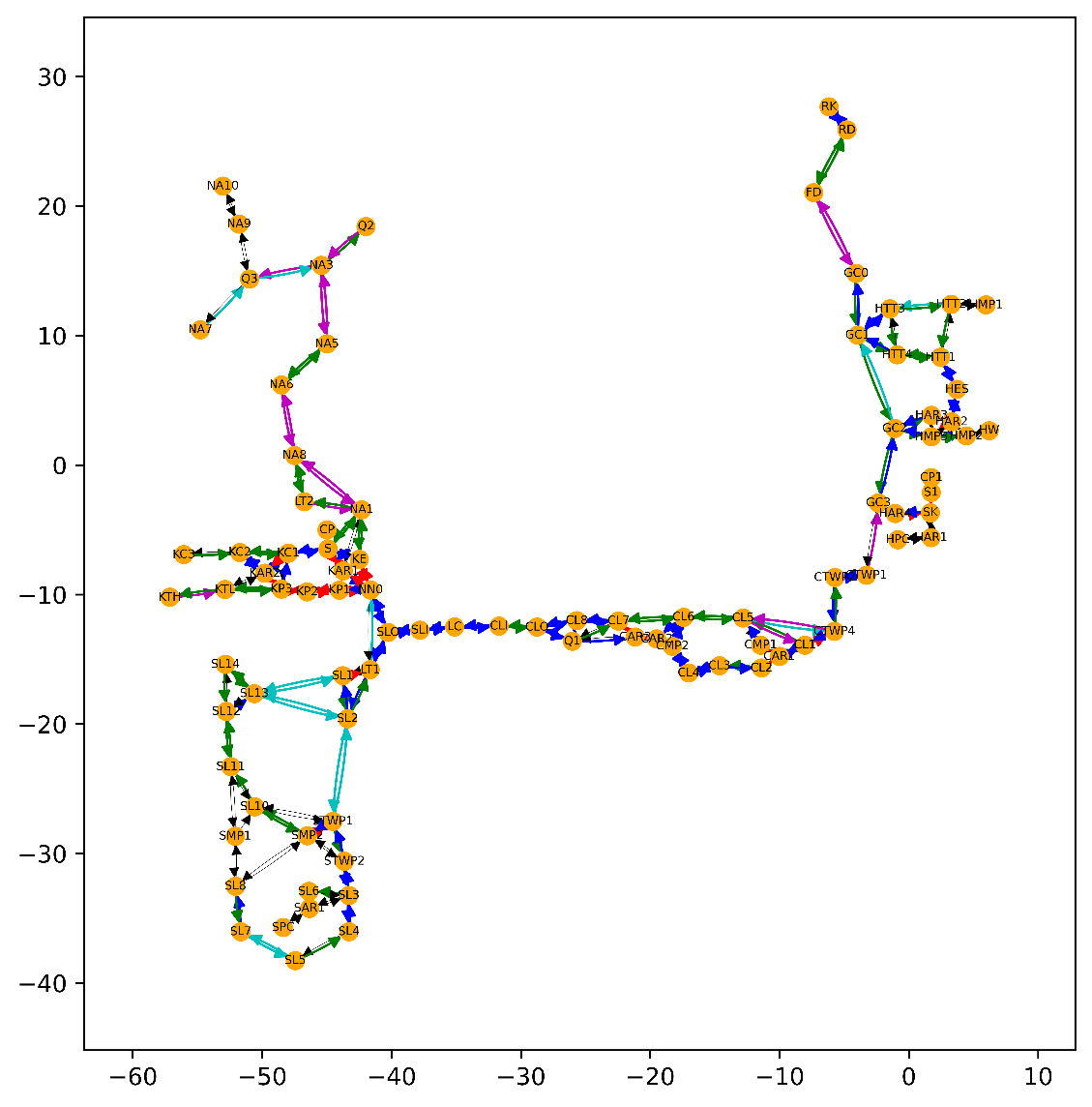
#### Binary Classifier



#### Edge Length



#### KS Statistic



## Evaluating performance

2 methods of generalisation:

1. Fit against edge with most data in a cluster
2. Fit against mergefit of all edges in cluster

#### Method 1 – fit against 1 edge per cluster

**Input:** dataframe containing data + cluster labels

**Output:** mean, max, min, median KS per cluster

Process:

* For each cluster (that has more than 1 edge)
* Pick edge with most data & fit
* Pick each other edge in cluster & fit
* Calculate mean KS between **fitted** distributions

# WED: Performance Evaluation

Yesterday, I converted the binary classification output into HAC clustering input, by using the probability of class 0 as the distance (i.e. probability that 2 edges are not in the same cluster).

* The feature importance is distributed as
* 0.4-0.5 for edge length
* ~0.2 for sum of angles & max angle
* the rest is split approx. evenly between total connections, connections at origin & connections at target
* For AAF, there are a lot of straight edges that have max\_angle & sum\_of\_angles = 0, so the clustering based on binary classification (attachment 1) is very similar to clustering based on edge length (attachment 2). The differences are in edges 91\_86 and 90\_26
* For TSC, there is a lot more differentiation between clusters based on binary classification (attachment 3) and based on edge length (attachment 4).

I also tried a simple method of comparing performance on the TSC maps (where there are more differences in the clustering), taking the mean KS score of the edge with most data points (fitted as lognormal) against all other edges in the cluster (also fitted as lognormal using the Bayesian process).

* For this method, the clustering based on edge length has KS score = 0.216
* The clustering based on edge length has KS score = 0.176
* It seems that edge length has better performance than the binary classification method in this case

#### Method 2 – fit all edges in cluster against all other edges (Section 6c & Section 7)

**Must run get\_lognorm\_fit first**

**Input:** dataframe containing data + cluster labels + dataframe containing fitted lognorm params

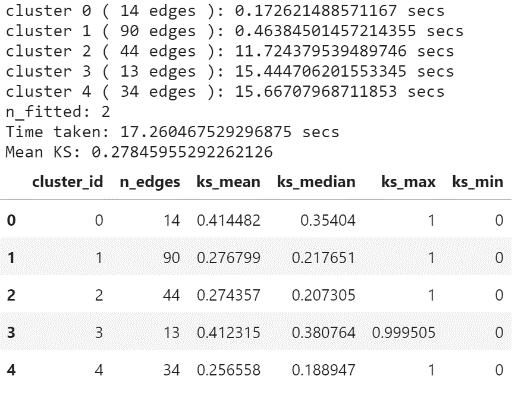
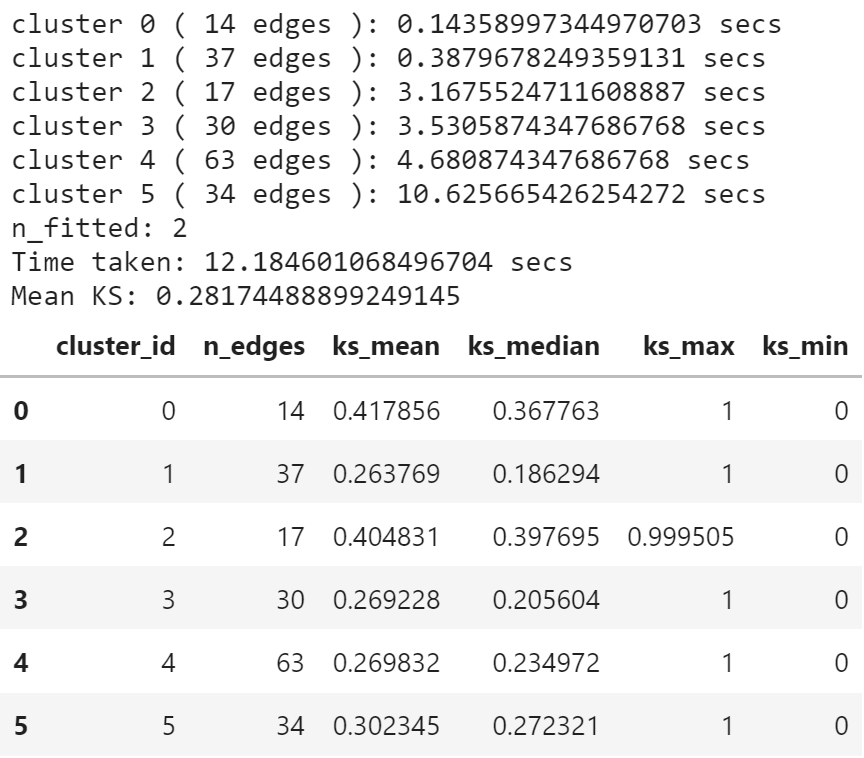
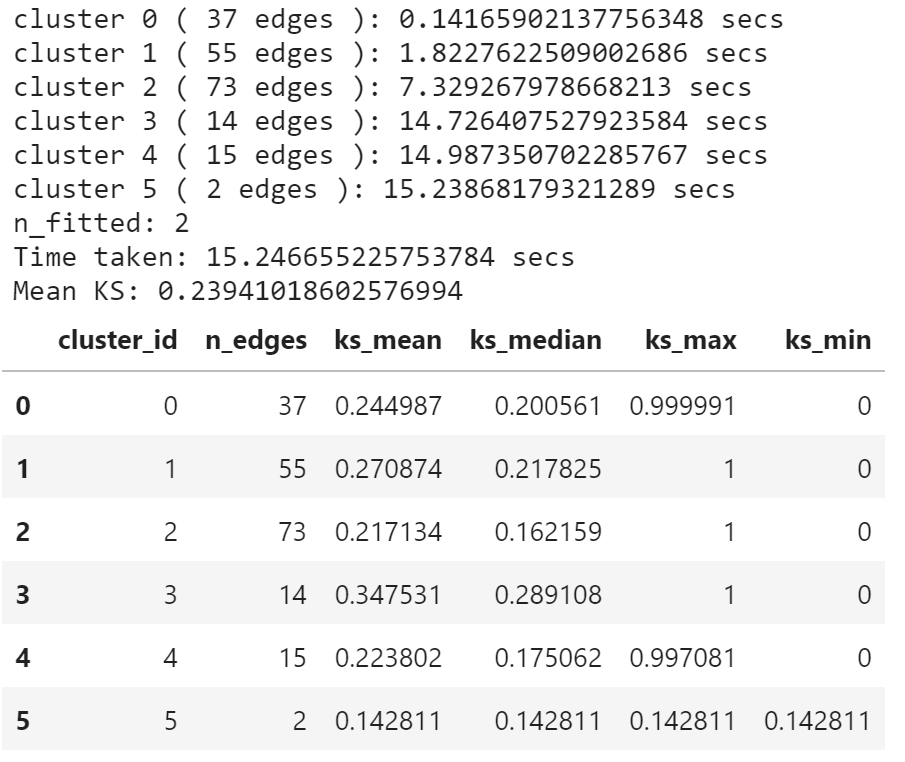
**Output:** mean, max, min, median KS per cluster

3 modes (n\_fitted = 0, 1, 2)

* n\_fitted = 0: use 2-sample KS test on raw observation data
* n\_fitted = 1: edge with most data is fitted. Comparison edges are not
* n\_fitted = 2: both edges are fitted using Bayesian lognormal optimisation

**For n\_fitted = 2 for [Train on AAF, Test on TSC]**

Binary classification clusters: Edge length clusters: KS clusters:

Performance (KS score of all fitted edges against all other fitted edges)

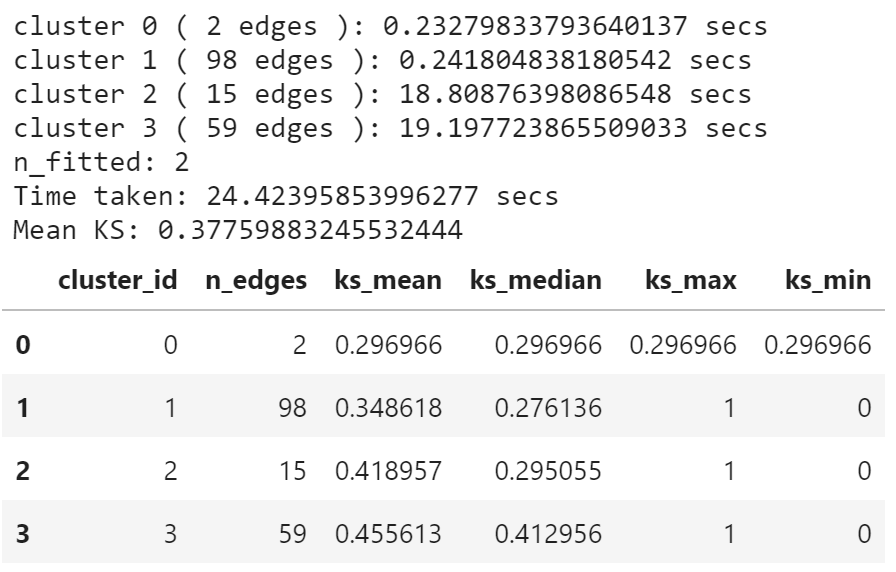
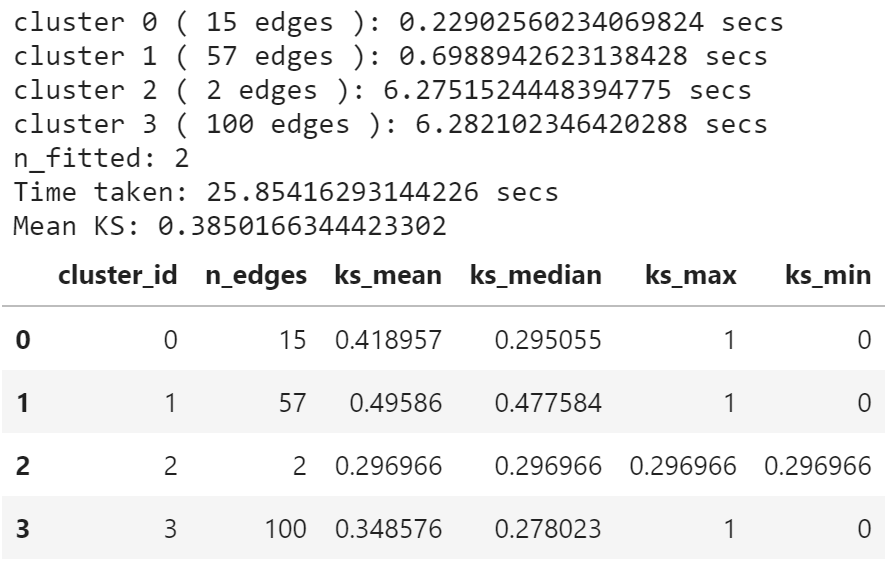
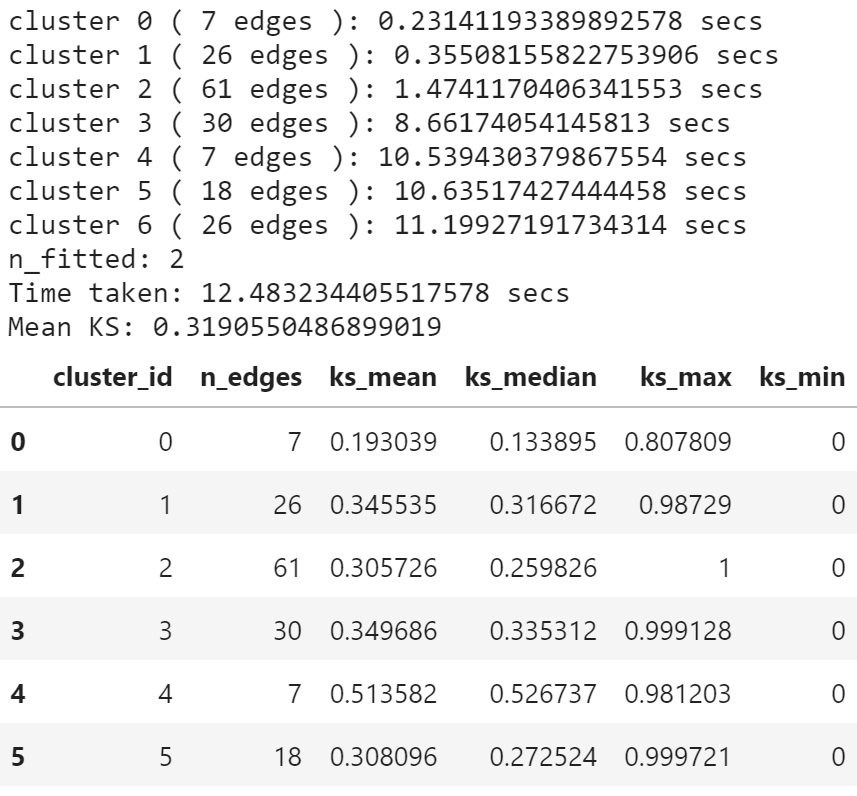
* Binary classification: 0.278
* Edge length clusters: 0.282
* KS clusters: 0.239

What is good is that for this TSC map where other factors than edge length play a role (the map is not as linear), the binary classification clusters outperform the edge length clusters.

KS clusters is slightly better (which makes sense since we clustered using HAC to minimise average linkage). However, KS clustering can only be performed when you have data for the edges. Binary classification clustering & edge length clustering can be done using only spatial features.

**For n\_fitted = 2 for [Train on TSC, Test on AAF]**

Binary classification clusters: Edge length clusters: KS clusters:

Performance (KS score of all fitted edges against all other fitted edges)

* Binary classification: 0.378
* Edge length clusters: 0.385
* KS clusters: 0.319

Only 2 edges that are not the same: 91\_86 & 90\_26. However, binary classification clustering puts these edges in a more suitable cluster than edge length clustering and the KS score decreases (good).

The reason for the high similarity between binary classification clustering and edge length clustering is that there are few angles in the AAF map and many edges have sum\_of\_angles or max\_angle = 0. Therefore edge length plays a dominant role in the binary classifier.

KS clusters is better (which makes sense since we clustered using HAC to minimise average linkage). However, KS clustering can only be performed when you have data for the edges. Binary classification clustering & edge length clustering can be done using only spatial features.

# THU: Trying out different clustering methods

## Summary

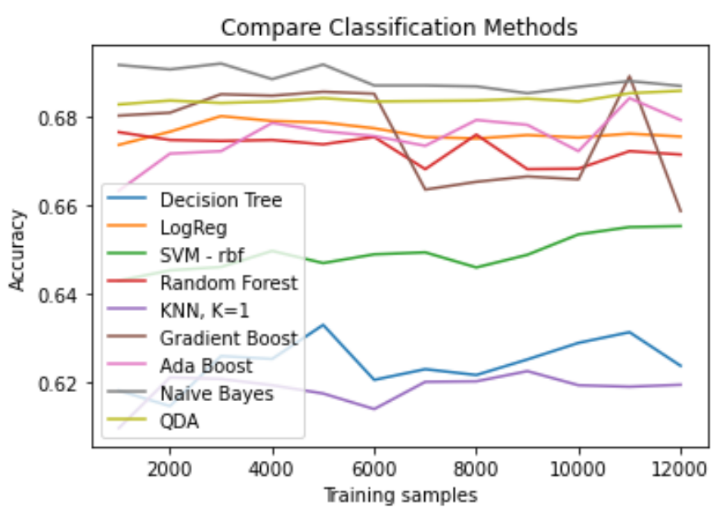
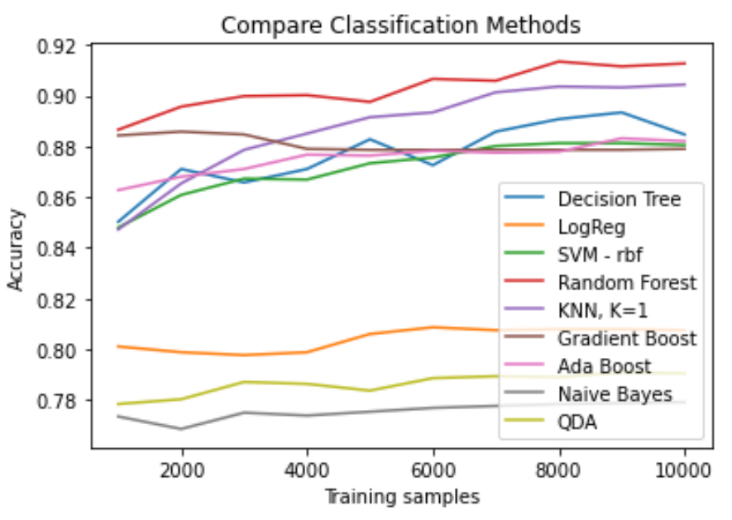
I mainly had a look at alternative classifiers and clustering.Regarding classifiers, there might be a problem of overfitting or concept drift with my current implementation. Comparing 2 situations:

* Train on random data from 4 maps (AAF, Walmart, Blenheim, LABS), and test on unseen data from the same maps (attachment 1)
* ~0.9 accuracy
* AUC (Area Under Receiving Operating Characteristic Curve) of 0.96 (where 1 is the best possible score)
* Train on random data from 4 maps (AAF, Walmart, Blenheim, LABS), and test data from TSC map (attachment 2)
* ~0.7 accuracy
* AUC of 0.75

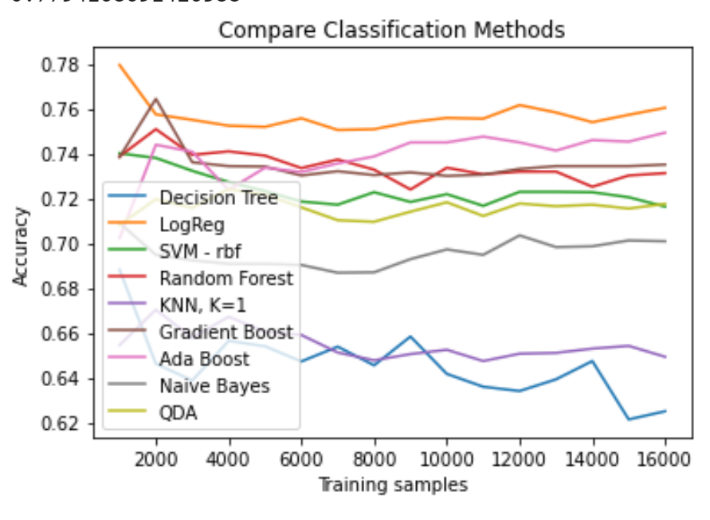
If this is due to overfitting, I can try to tune the parameters to reduce the likelihood of overfit. However, I did a quick try with random forest yesterday (just by hand as opposed to grid/random search) and this did not create a large increase in performance.  
The other possibility is that the TSC map is different to the previously seen maps (in that there are more non-linear connections).

* This seems to agree with the fact that when we train on {TSC, Walmart, Blenheim, LABS} and test on AAF (attachment 3), the performance is better
* accuracy = 0.77
* AUC = 0.87
* Similar numbers occur when we train on TSC and test on AAF
* Something we also saw in the clustering algorithms for TSC is the optimum number of clusters is less clear, so our "ground truth" cluster labels may be wrong

[Train AAF,LABS,Walmart,Blenheim], [Test same]: [Train TSC,LABS,Walmart,Blenheim], [Test AAF]:



[Train AAF,LABS,Walmart,Blenheim], [Test TSC]:



Regarding clustering, I tried out some alternative algorithms from the Sklearn library that can also take a distance matrix as input (Affinity Propagation, DBSCAN, OPTICS, Spectral Clustering). Of these, only Spectral Clustering (attachment 1) had good performance (i.e. found sensible numbers of clusters). However this also clusters the same anomalous edges (as HAC).

I also tried implementing the KS thresholding: the process was to remove edge that had a KS score greater than the threshold value when compared to all other edges. I set the threshold to 0.3 to remove a somewhat sensible amount of edges - this value was also a middle value between the mean KS score amongst class 1 (dissimilar edges) and amongst class 0 (similar edges).

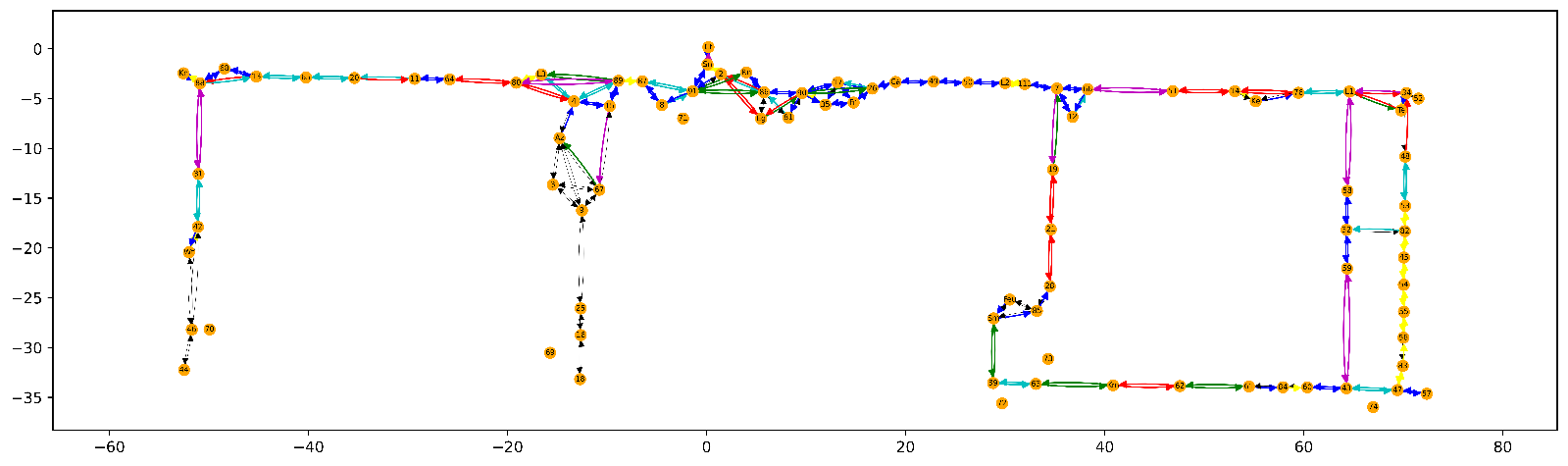
* Attachment 2 shows AAF map with KS threshold of 0.3. This has removed some of the anomalous edges 34\_Lift1 & Station\_ChargingPoint.

Looking at evaluation,

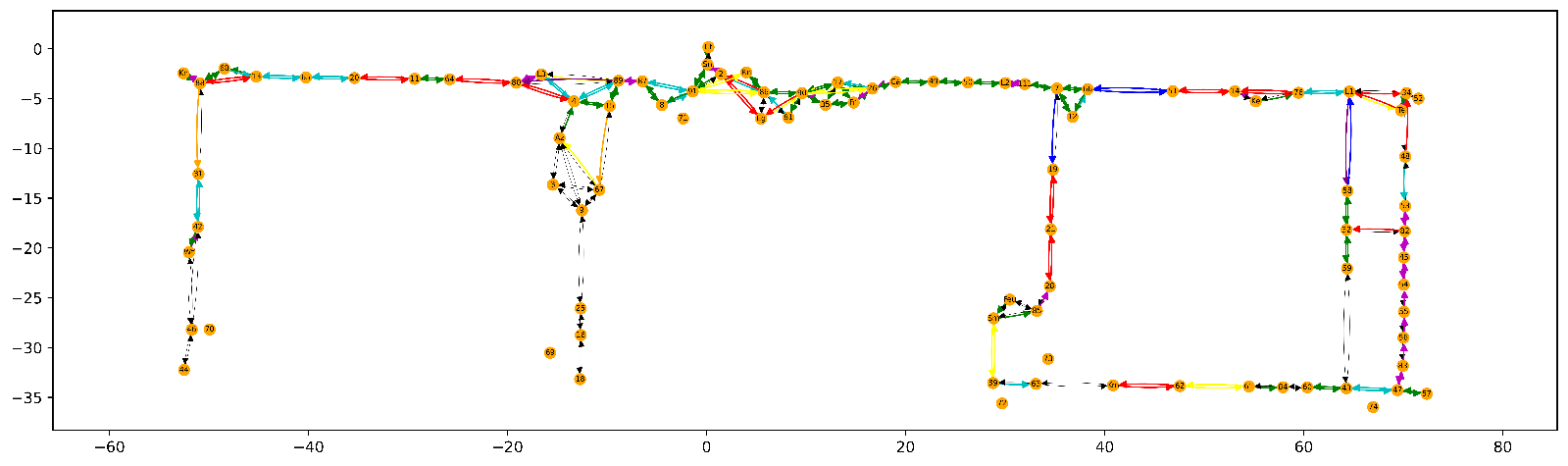
* Binary classification clustering has mean KS = 0.266
* Edge length clustering has mean KS = 0.271
* KS clustering has mean KS = 0.228

Realistically, we wouldn't expect the binary classification to perform better than KS clustering given that we have used the KS clustering to generate labels. However, the improvement over edge length clustering is still small.

Spectral clustering (AAF) – 6 clusters: "TEST\_SpectralClustering\_AAF.png"



KS clustering with threshold (AAF) – 8 clusters: "clusters\_threshold\_aaf.png"



## Clustering methods

Note: all of these algorithms are in Sklearn & can take a distance matrix as input

**Affinity Propagation (bad)**

* Tuning params: damping, preference
* Only gets 2 clusters 🡺 bad
* No change in number of clusters as damping increases

**DBSCAN (Promising)**

* Tuning params: eps & min\_samples
* Very big variance in number of clusters when tuning params are changed
* Predicts 15 clusters for AAF, 17 clusters for TSC
* Must be careful that beyond a certain eps (~0.5), we only get 1 cluster 🡺 we cannot take calculate score

**OPTICS (BAD)**

* Tuning params: max\_eps, min\_samples
* Only finds 2 clusters

**Spectral Clustering (Promising)**

* Tuning params: n\_clusters
* Predicts 6 clusters for AAF, 4 clusters for TSC

Note: HAC can also specify max KS distance. However, this gives the same result as using n\_clusters as the tuning parameter.

Divisive clustering (also called DIANA) has implementations in R & Github

# SUN: New Method

For a classifier based on thresholded KS values, would valid approaches for this be:

1. Directly predicting a KS score from the spatial features

* Training input: difference in spatial features (edge\_length, sum\_angles etc)
* Training output: KS score between 2 edges

1. Specifying KS values that indicate whether 2 edges are similar

* Training input: difference in spatial features (edge\_length, sum\_angles etc)
* Training output: class 0 (dissimilar) or class 1 (similar)

class 0 if KS score between 2 edges > 0.5         (0.5, 0.3 are just example)

class 1 if KS score between 2 edges < 0.3

otherwise ignore this pair of edges since KS value is ambiguous

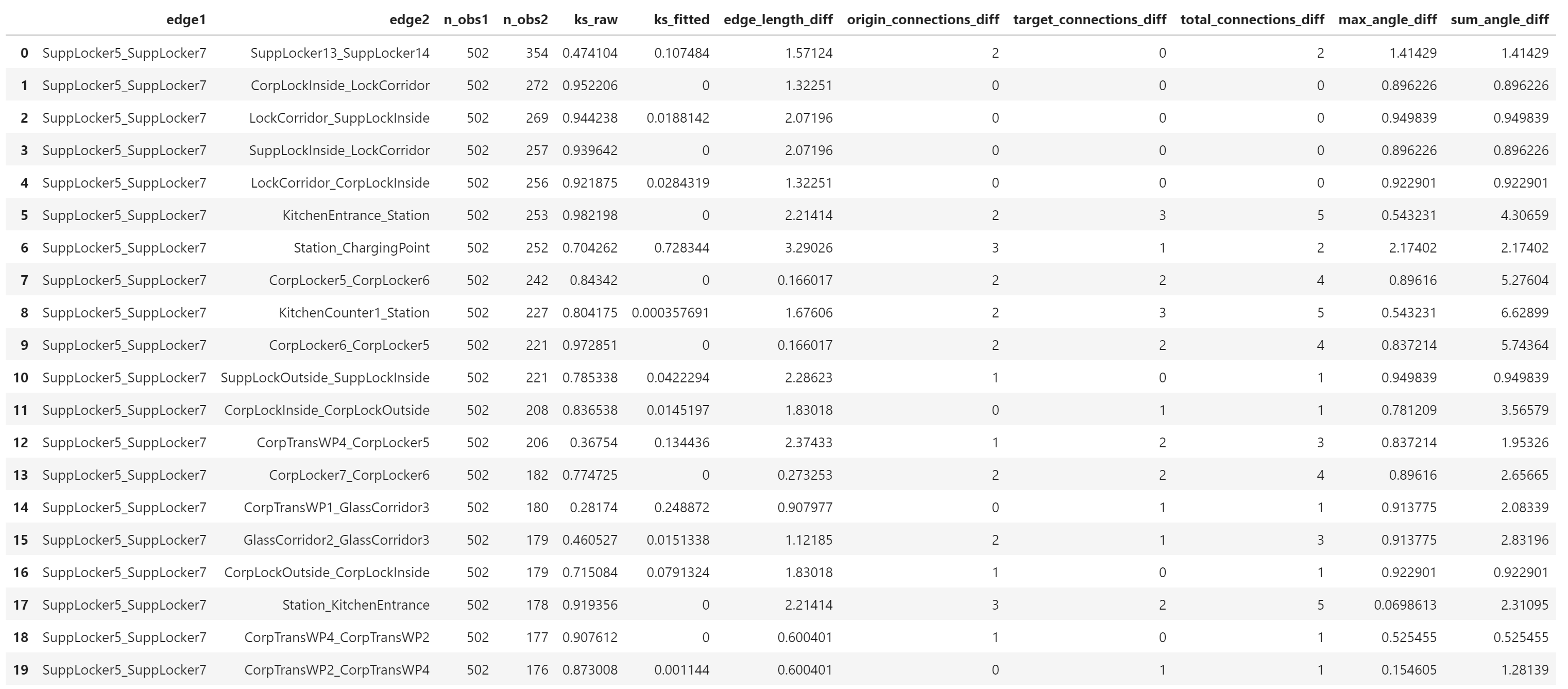
This could allow us to go directly to the second clustering step on the test map, using the predicted KS score or predicted class probability?  
Alternatively, if we trained a regressor to predict KS values, we could start to look for the training edge(s) with lowest predicted KS values wrt to a test edge and share the data?

Method 1 sounds better, but needs new methods (regression instead of classification).

**Potential classifiers:**

* Beta Regression
* GLM
* Tobit
* Logistic regression, Random Forest, SVM
* Ridge, Lasso

Looking at the output of the KS raw vs KS fitted, it seems that KS fitted seems to have large deviation from the raw scores



Therefore I have decided to use the raw KS for my regression input. This is loaded from **regression\_dataloader\_nofit.py**