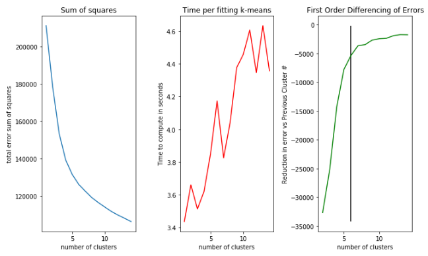
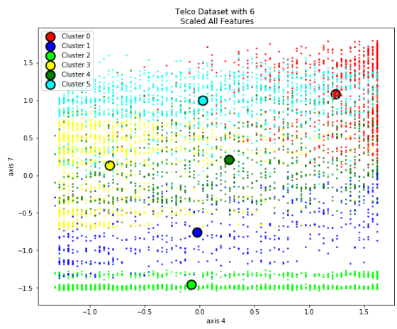
# Telco Dataset Feature Selection/Transformation

Scaling - We centered the entire dataset first. This helps us with making training easier on our algorithms as it puts it in a smaller space.

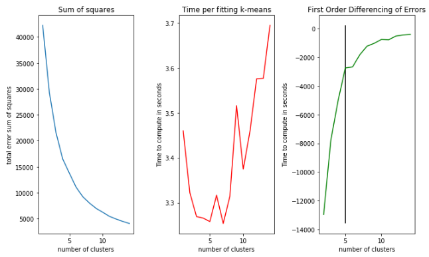
1a

## K Means Dimensionality Reductions

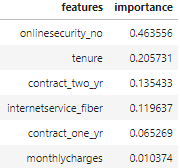
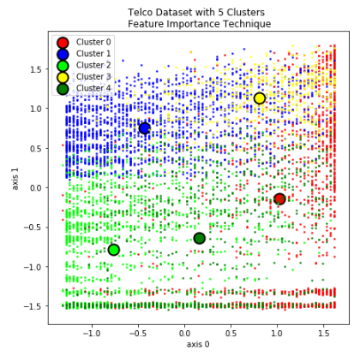
1. Clustered on Scaled dataset All Features (Task 1)
   1. To select our k, we performed a grid search between 1 and 15, k. We looked at the sum of square and plotted that against cluster number. We employed first differencing as way to find the optimal cluster via “elbow” method. This returned an optimal value of 6. We also must be cognizant of time too but this is a quick fit, < 4 seconds, so time is not a factor.

1b

* 1. These cluster are picking up a lot of the data density at the bottom and edges & top of the data space which then allows the middle cluster to grab the remaining points. The bottom two clusters appear to line up naturally, but the other clusters appear to have overlap.

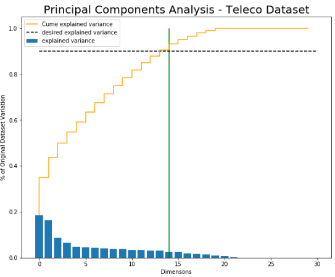
1.  Feature Selection – We used our final decision tree from homework 3 (depth of 3) to help us select our features. We choose all features which had an importance of > 0. This resulted in reducing our features from 33 to 6. We used entropy as our splitting criterion.

2

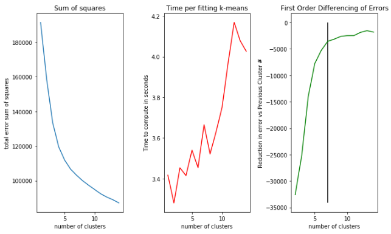
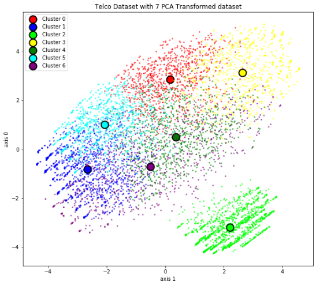
* 1. To select our k, we performed a grid search between 1 and 15, k. We looked at the sum of square and plotted that against cluster number. We employed first differencing as way to find the optimal cluster via “elbow” method. This returned an optimal value of 5. We also must be cognizant of time too, but this is a quick fit as it takes 3-4 seconds to fit.

2b

* 1. Looking at the clusters we got back they line up naturally and seem to make sense. We have 4 cluster near the edges and there is a lot of data density towards the sides and edges of our plot then towards the middle. What the edge clusters deem as too far, the middle cluster picks up. These clusters appear to have good separation between them, but we will need to look at silhouette score later to verify this. The clusters are different than 1b mainly driven by having 1 less cluster but this was optimal given our criteria.

1. PCA – We are using 90% of the variance explained from PCA as the number of features. This reduces our features from 33 to 15. We choose 90% because going from 80% to 90% only increased the dimensions by 3.

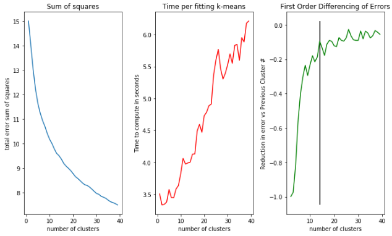
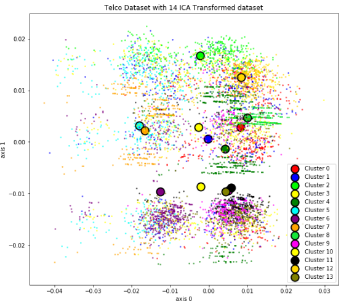
3a

* 1.  To select our k, we performed a grid search between 1 and 15, k. We looked at the sum of square and plotted that against cluster number. We employed first differencing as way to find the optimal cluster via “elbow” method. This returned an optimal value of 7. We also must be cognizant of time too. Time to compute is not an issue here as it takes 3-4 seconds to fit.

3

3b

* 1. We are getting clusters that appear to be defining a box and an outlier. This is very different than what we saw in 1 & 2 but makes sense as it’s taking the data and forcing it to be mutually orthogonal. This in turn projects our data into this box looking shape. 4 clusters have the edges of the box and 2 other clusters on opposite sides of the box. The other outlier cluster is its own cluster. This appears to be a very good cluster. There is some overlap between the clusters, except for the outlier in the bottom right which doesn’t have any overlap.

1. ICA – Finding the correct number of components with ICA is tough. ICA is trying to parse out the underlying signals from the data, i.e maximizing independence. This is done by keeping mutual information as high as possible with original features and keeping the new features independent. In the end, we ended going with the same number of dimensions as PCA, 15. This was done to keep things similar when comparing between feature transformation techniques.
   1.  To select our k, we performed a grid search between 1 and 40, k. We looked at the sum of square and plotted that against cluster number. We employed first differencing as way to find the optimal cluster via “elbow” method. This returned an optimal value of 14. We also must be cognizant of time too as computation time increases with # of clusters. Time to compute is not an issue here as it takes 3-6 seconds to fit with our solution taking 4 seconds to fit.

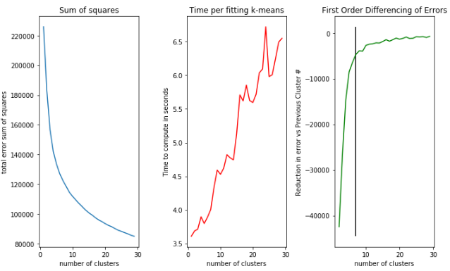
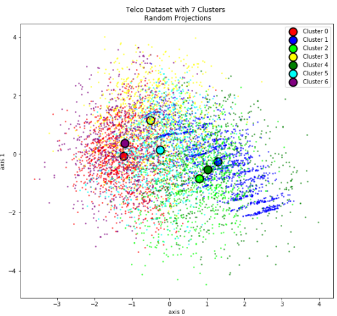
4a

4b

* 1. Since ICA is about independence, we can see how this projects our data into rather distinct clusters which is different than what we saw in 1b. ICA has taken out data and given us 3 different “levels” of data points within each data point we see 3 groupings. We would want/expect our clusters to pick up these groupings and line up naturally, but it is not. The reasoning, these algorithms, at times, have a hard time finding these clusters and can get stuck. When this happens, it sticks multiple groups together when it should be its own group. We are seeing this in our clusters as multiple clusters are being stuck together and others cluster being stuck on top of each other. In order to combat this, we employed 30 random restarts, but we are still getting sub-optimal clustering.

1. Random Projections – Finding the correct number of components with random projections can be tough because it’s a bit arbitrary. Random projections are taking random directions and projecting the data onto these random directions. The benefit of this RP being able to still pick up on correlation between features. Since our lecture notes state that RP features need to be slightly higher than PCA or ICA, we went with 20 features.

5a

* 1. To select our k, we performed a grid search between 1 and 30, k. We looked at the sum of square and plotted that against cluster number. We employed first differencing as way to find the optimal cluster via “elbow” method. This elbow is around 7 so we are choosing an optimal cluster of 7. We also must be cognizant of time as computation time increases with # of clusters. Time to compute is not an issue here as it takes 3-5 seconds to fit with our solution taking 4 seconds to fit

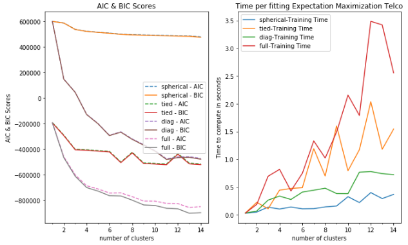
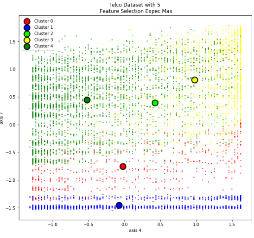
5b

* 1. These clusters are very different than what was returned by 1b. This makes sense, we are projecting the data into a multivariate gaussian distribution and the clusters we are returning appear to be very gaussian. One can almost see the multivariate distribution of the blue cluster. This shouldn’t surprise us as the underlying package we used is putting it in a gaussian space. The 7 clusters which we deemed as optimal by the elbow method show data points in clusters overlapping each other and clusters being bunched up with each other. The results we are seeing are somewhat close to ICA but not to the extreme of ICA. We can see how some of the correlation of the features are picked up by RP and projected onto the new space. This is evident by Cluster 1(blue) which sees data points in a somewhat straight line but having some distance between them.

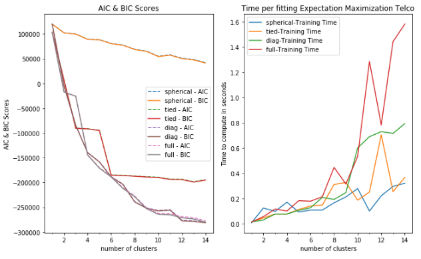
1. Comparing and Contrasting K-Means
   1. When we compare chart 1b, 2b, 3b, 4b , & 5b the difference between Feature selection, PCA, ICA and RP begin to come to light. 1b(all scaled features), 2b(feature selection) & 3b(PCA ) all appear to cluster our data very well. It’s finding the data densities and centering based on those densities. 4b(ICA) & 5b(RCA) don’t appear to be performing that well. ICA is having trouble finding the groupings of data is generalizing more towards the middle of our space. While RCA, appears to be find the random gaussians underneath but clusters appear to be overlapping each other and don’t appear to be separated that well. This could be due to only plotting in two dimensions. We will withhold judgement until we see silhouette scores. We see that across different transformations we are getting different clusters and cluster numbers, except for 1b & 2b which are similar. This shouldn’t surprise us given our data is in the same space just with a few less features. The other feature transformation being performed are giving different clusters as it’s putting our data in a different feature projection space.

## Expectation Maximization Dimensionality Reductions

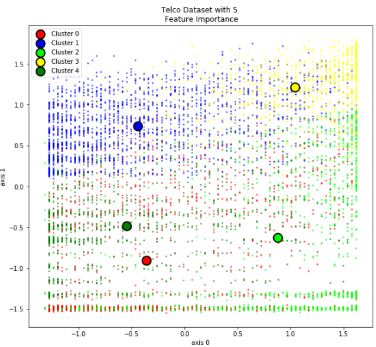
1a

1. Clustered on Scaled dataset All Features (Task 1) –
   1. To select our k in expectation maximization we performed a grid search between 1 and 15 k and over 4 different types of covariances spherical, tied, diagonal and full. We looked at the AIC & BIC values of each covariance type and plotted against cluster number. We are looking for the “elbow” with the lowest AIC/BIC score. This returned an optimal value of 5. We also must be cognizant of time, too. This this is quick as it takes around 1 second for fitting.

1b

* 1. Looking at the types clusters we got back, we see there is a lot of data points towards the edges of the chart. The Blue, Green and Yellow clusters and picking up on it and are clustering accordingly. The remaining two clusters are picking up the data more towards the middle. We got these clusters mainly by the data densities towards the outer edges of our data space. These results are very similar to what we saw in 1a of k-means.

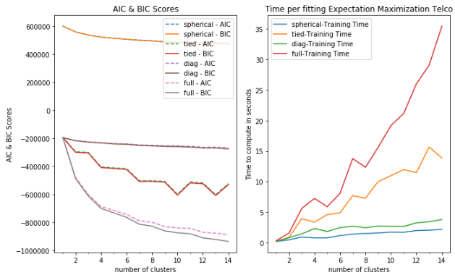
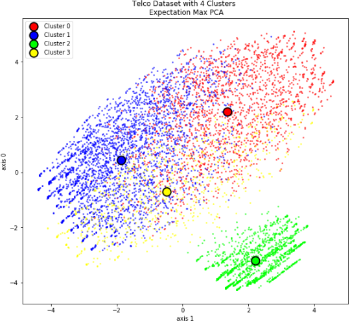
2a

1. Feature Selection – We used the same feature selection as we did in K-means.
   1. To select our k in expectation maximization we performed a grid search between 1 and 15 k and over 4 different types of covariances spherical, tied, diagonal and full. We looked at the AIC & BIC values of each covariance type and plotted against cluster number. We are looking for the “elbow” with the lowest AIC/BIC score. This returned an optimal value of 5 with either full or diagonal covariance. In order to keep things consistent with K-means we are choosing 5 clusters and using full covariance as full covariance allows each cluster its own general covar matrix. There isn’t a difference in time between them so we will use the more robust covariance matrix, full.

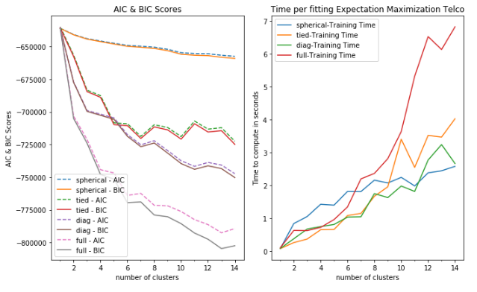
2b

* 1. The types of clusters returned are very similar to what K-means returned. Clusters are more towards the edges of the plot due to the data densities being the highest there. The biggest difference is cluster 4(dark green) which is more toward the left edge whereas k-means had this more towards the middle. I think we got the cluster we did because the edges have more data density than the center, so, expectation maximization is having a larger pull towards the edges than K-means.

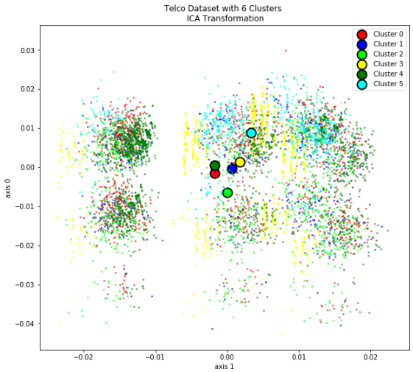
3a

1. PCA – Used same dimensions as we did in k-means, 15.
   1. To select our k in expectation maximization we performed a grid search between 1 and 15 k and over 4 different types of covariances spherical, tied, diagonal and full. We looked at the AIC & BIC values of each covariance type against plotted against cluster number. Again, we are looking for the “elbow” with the lowest AIC/BIC score. This returned an optimal value of 4 with full covariance. Full Covariance with 4 clusters does take a couple of seconds longer than others but the improvement in AIC/BIC is worth it.
   2.  We are getting clusters that appear to be defining a box and an outlier blob. This is very different data and clusters we saw in 1b & 2b. It makes sense as it’s taking the data and forcing it to be mutually orthogonal. We can see this in the box looking shape and the outlying cluster in the bottom. Expectation Max and K-means(3b) are both picking up on the bottom right cluster. The difference here is the three clusters in our box. Expec max is assigning the bottom of the box to cluster 3(yellow) and the top and sides to cluster 0 and 1. The cluster centers appear to have some good space between them but will need to look at homogeneity and silhouette scores later to determine this.

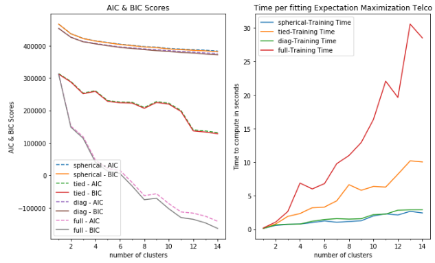
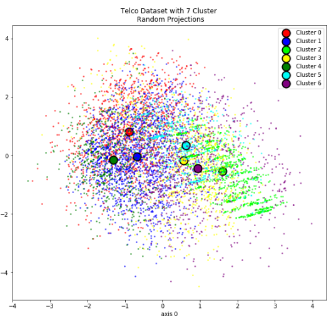
3b

1. ICA – Used same dimensions as we did in k-means, 15.
   1. To select our k in expectation maximization we performed a grid search between 1 and 15 k and over 4 different types of covariances spherical, tied, diagonal and full. We looked at the AIC & BIC values of each covariance type against plotted against cluster number. Again, we are looking for the “elbow” with the lowest AIC/BIC score. This returned an optimal value of 6 with full covariance. Full Covariance with 6 clusters takes roughly the same amount of time to compute vs other covariance matrix.

4a

* 1.  Since ICA is about independence, we can see how this projects our data into rather distinct clusters which is different than what we saw in 1b. We can see these groups of data which looks like it wants to be its own cluster. Given that ICA is about independence between features, it’s no surprise why our data is projected this way. We would want/expect our clusters to pick up these groupings and line up naturally, but it is not. The reasoning, these algorithms, at times, have a hard time finding these clusters and can get stuck. It then sticks multiple groups together when it should be its own group. We are seeing this in our clusters as multiple clusters are being stuck together. Given the spread of our data projection our cluster centers are all getting stuck in the middle and near each other. In this situation, we are seeing expectation maximization and k-means perform similar for ICA. They both are getting stuck. We employed 10 random restarts to help the algorithm get “unstuck”, but it doesn’t appear to be working.

4b

1. Random Projections – Used same dimensions as we did in k-means, 20.
   1. To select our k in expectation maximization we performed a grid search between 1 and 15 k and over 4 different types of covariances spherical, tied, diagonal and full. We looked at the AIC & BIC values of each covariance type against plotted against cluster number. Again, we are looking for the “elbow” with the lowest AIC/BIC score. This returned an optimal value between 6 and 8 with full covariance. Since we are returning a cluster number similar to k-means, we will choose 7 to compare against k-means. We need to be cognizant of time as different covariance matrix take longer than others. In this case, it did take at least 6 seconds for our optimal value and better aic/bic scores are fine.

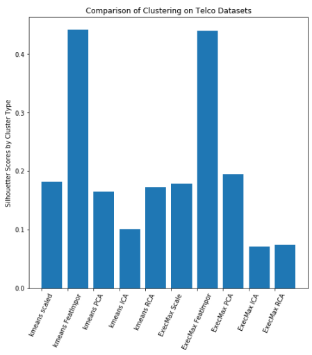
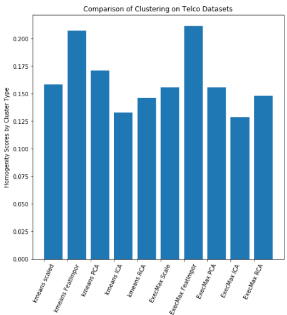
5a

* 1. These clusters are very different than what was returned by 1b. This makes sense, we are projecting the data into a multivariate gaussian distribution and the clusters we are returning appear to be very gaussian. After seeing the types of clusters returned in K-means it’s no surprise what was returned. This is due to random projection and expectation maximization assuming a gaussian process underneath. The cluster centers between k-means and expec max for RP are in similar locations. This is evident by Cluster 2(green) being almost identical to cluster 1(blue), 5b k-means. Given how close these centers are to each other they don’t appear to be compact, will need to look at homogeneity and silhouette scores later to determine this.

5b

1. Comparing and Contrasting K-Means & Expectation Maximization
   1. We see that across expectation maximization we are getting different clusters and cluster numbers, except for 1b & 2b which are similar. The reason is each feature transformation being performed is different as is putting our data in a different feature projection space.

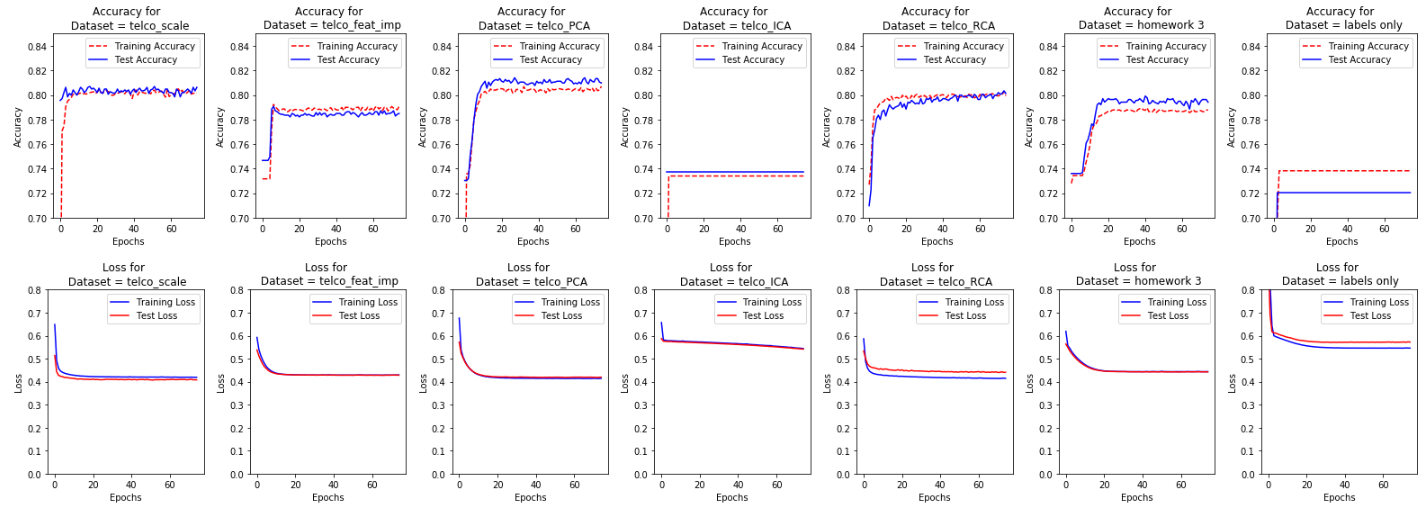
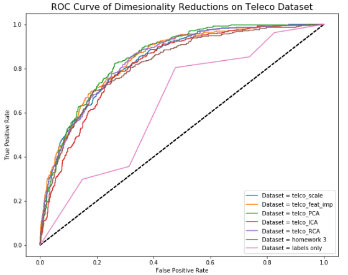
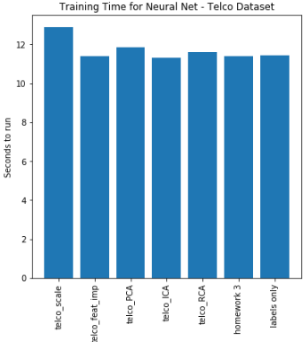
6b

* 1. When comparing all these feature transformations and cluster algorithms we want to see how they line up with our class labels and how compact the cluster are. In order to do this, we employed homogeneity score and a silhouette score. A cluster is homogenous if all class labels in the cluster are the same. We want our score to be as high as possible. Here the two best scores are Feature Importance for both k-means and expec max. It’s not really a surprise that within each clustering algorithm, k-means & expec max, we see feature selection as the highest. We used a greedy algorithm, a decision tree, to split our data based on the class labels. This selected our features knowing the se a good candidate for splits. PCA is the second best, this is probably a case of us throwing a way a variable which is useful but had low variance, so it was thrown away.

6c

* 1. We also need to see the compactness of our clusters. We used a measure called Silhouette score to tell us this. Silhouette score tells us how similar an object is to its own cluster compared to other clusters. Values close to 1 represent values identified well for its cluster where as a value of -1 would say its poorly matched and it belongs in a different cluster. All clustering algorithms have a value 0 which indicate that our clusters are not overlapping or are relatively compact. Clusters assigned via feature importance are having the highest values. This tells us this we are getting clusters that a very similar within and different between. We can say we have compact clusters for feature importance.

## Telco Neural Net

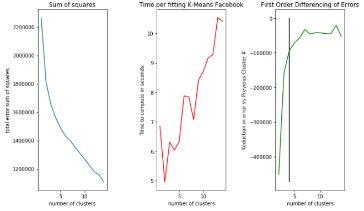
1. Run your neural network learner from assignment 3 on the data after dimensionality reduction (from task 2). Explain and plot your observations (error rates, etc.)
   1. Looking at both Accuracy and Loss we see that telco\_scaled, PCA, RCA and results from homework 3 all perform the same. Feature selection is not too far behind our top performers. Both ICA and labels only from task 1 had the highest loss and lowest accuracy. Also, all seem to take the same amount of time to run, 11-12 seconds.
   2. Looking at the ROC/ACU curves the following are showing a similar AUC score of 84%, all scaled(telco\_scale), feature selection, PCA & RCA. The next best scores are ICA and homework 3 and 81% and the last is labels only at 64%.

1a

1a

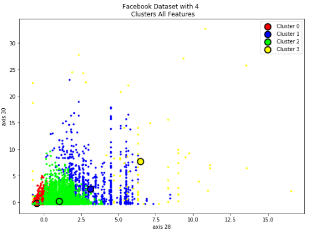
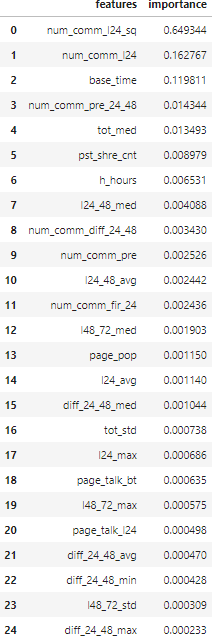
1b

# Facebook Dataset Feature Selection/Transformation

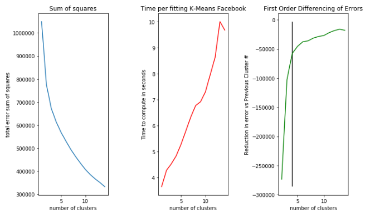
Scaling - We centered the entire dataset first. This helps us with making training easier on our algorithms as it puts it in a smaller space.

1a

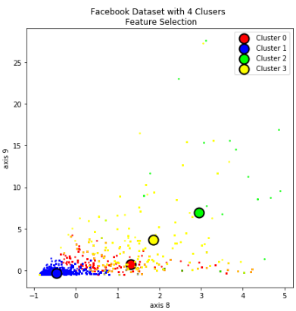
## K Means Dimensionality Reductions

1. Clustered on Scaled dataset All Features (Task 1) –
   1. To select our k, we performed a grid search between 1 and 15, k. We looked at the sum of square and plotted that against cluster number. We employed first differencing as way to find the optimal cluster via “elbow” method. This returned an optimal value of 4. We also must be cognizant of time, but this is a relatively quick fit.
   2. Looking at the types clusters we got back, we see the clusters gathered close to the origin are tighter. This is due to a lot of data points at the bottom and close to the origin. Our cluster become less dense as we move away from the origin due to the spread of our data. We got these clusters due to the scaled nature of the data.

1b

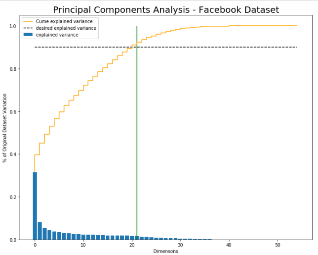
1.  Feature Selection – We used our final decision tree from homework 3 (depth of 6) to help us select our features. We choose all features which had an importance of > 0. This resulted in reducing our features from 55 to 25. We used entropy as our splitting criterion.

2a

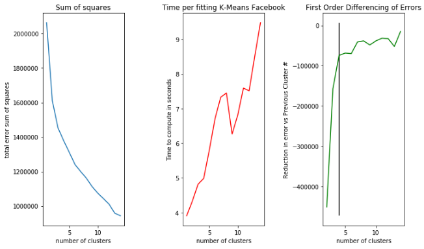
* 1. To select our k, we performed a grid search between 1 and 15, k. We looked at the sum of square and plotted that against cluster number. We employed first differencing as way to find the optimal cluster via “elbow” method. This returned an optimal value of 4. We also must be cognizant of time too, but this is a quick fit as it takes 4-5 seconds to fit.

2b

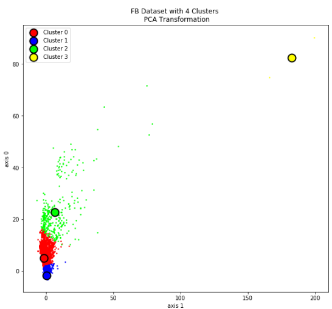
* 1. I believe we got the types of clusters we did was based on the data density. This was very similar to what 1b returned. Data close to the origin are higher density, whereas, as you go away from origin they tend to be spread out and a different cluster is capturing that.

1. PCA – We are using 90% of the variance explained from PCA as the number of features. This reduces our features from 55 to 22. We choose 90% because going from 80% to 90% only increased the dimensions by a small amount.

3

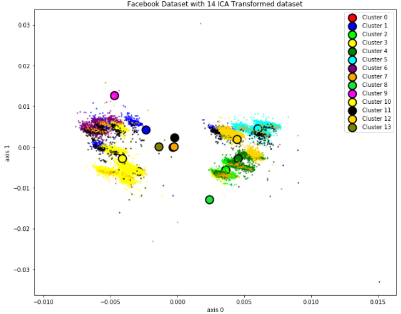
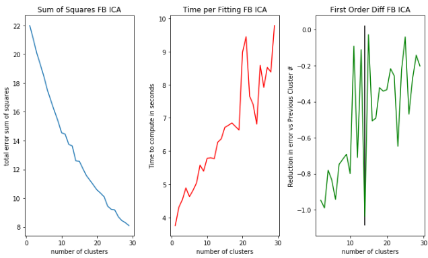
* 1. To select our k, we performed a grid search between 1 and 15, k. We looked at the sum of square and plotted that against cluster number. We employed first differencing as way to find the optimal cluster via “elbow” method. This returned an optimal value of 4. Generally, we need to factor in time but this only too 4 seconds to fit.

3a

* 1.  The cluster types we are getting back are different from 1b & 2b. These are trapezoidal in shape. This doesn’t surprise us as PCA is forcing our data to be mutually orthogonal. We can see this in how the cluster is cut from another. Most of our data density is down at the origin which explains why most of cluster are there. As the variability increases our cluster centers become further out.

1. ICA – Finding the correct number of components with ICA is tough. ICA is trying to parse out the underlying signals from the data, i.e maximizing independence. This is done by keeping mutual information as high as possible with original features and keeping the new features independent. In the end, we ended going with the same number of dimensions as PCA, 22. This was done to keep things similar when comparing between feature transformation techniques.

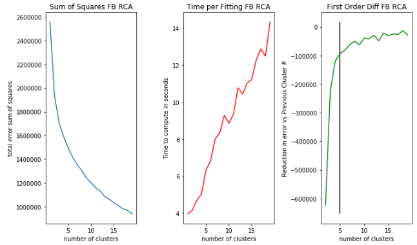
3b

* 1. To select our k, we performed a grid search between 1 and 30, k. We looked at the sum of square and plotted that against cluster number. We employed first differencing as way to find the optimal cluster via “elbow” method. The error sum of square is a very gradual decline and appears to elbow at 14. We didn’t want to choose a higher number given how much longer it would take. Our error sum of square numbers appear very small but those numbers are deciving as our underlying values are very small.

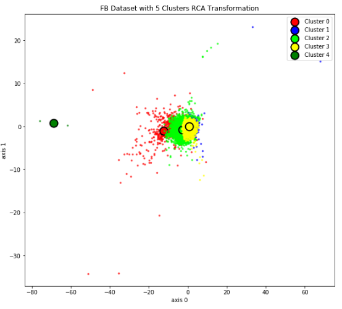
4a

* 1. Since ICA is about independence, we can see how this projects our data into rather distinct clusters which is different than what we saw in 1b. ICA has taken our data and given us 2 distinct separations, within these separations we see natural clustering of data. It’s no surprise we are seeing this given the underlying independence between features of ICA. While one would our clustering algorithm would be able to identify these natural clusters, it is not. The reason being k-means can get stuck and can stick natural clusters together when it should be its own group. We employed 20 random restarts to help with this, but out clusters are still getting stuck in the middle of the separation. Overall, it’s ok as it’s finding some unique groupings, but it is missing some natural clusters.

4b

1. Random Projections – Finding the correct number of components with random projections can be tough because it’s a bit arbitrary. Random projections are taking random directions and projecting the data onto these random directions. The benefit of this RP being able to still pick up on correlation between features. Since our lecture notes state that RP features need to be slightly higher than PCA or ICA, we went with 25 features.
   1. To select our k, we performed a grid search between 1 and 20, k. We looked at the sum of square and plotted that against cluster number. We employed first differencing as way to find the optimal cluster via “elbow” method. This returned an optimal value of 5. We also must be cognizant of time because as we increase clusters, we increase time. This only took 5 seconds to fit, though.

5a

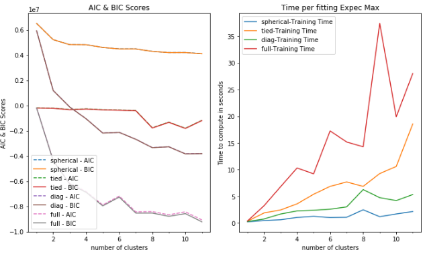
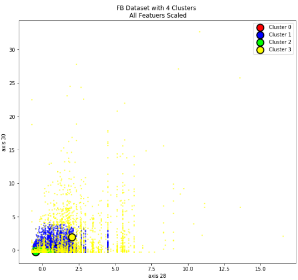
* 1. These clusters are very different than what was returned by 1b. This makes sense, we are projecting the data into a multivariate gaussian distribution and the clusters we are returning appear to be gaussian. This doesn’t surprise us as our data projection is gaussian. The underlying package we used, is putting it in a gaussian space. The 5 clusters which we deemed as optimal by the elbow method show data points on top of each other. There is one outlier to us which is baffling but could be the result of randomness. We tried 20 restarts to get rid of this but that didn’t suffice.

5b

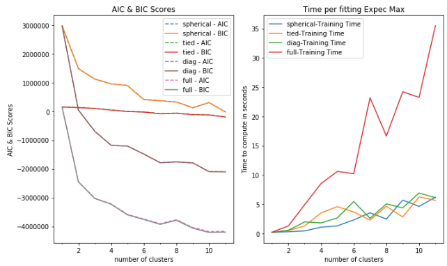
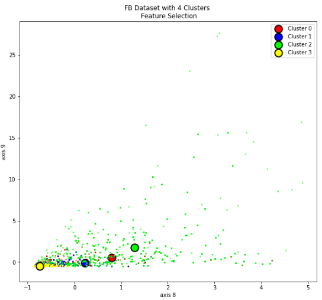
1. Comparing and Contrasting K-Means
   1. When we compare charts 1b, 2b, 3b, 4b, & 5b we see similarities and differences between Feature selection, PCA, ICA and RP. 1b(all scaled features), 2b(feature selection), 3b(PCA) & 5b cluster similar. It’s finding the data densities and centering the clusters over them. 4b(ICA) seems to do ok. It can pick up on some unique groupings but then we see it get stuck, as clusters are in the middle of our separation. We see that across different transformations we are getting different clusters and cluster numbers, except for 1b & 2b which are returning similar cluster types and numbers. 1b & 2b results shouldn’t surprise us given our data is in the same space just with a few less features. The other feature transformation being performed are giving different clusters as it’s projecting our data in a different feature projection space.

## Expectation Maximization Dimensionality Reductions

1a

1. Clustered on Scaled dataset All Features (Task 1) –
   1. To select our k in expectation maximization we performed a grid search between 1 and 12 k and over 4 different types of covariances spherical, tied, diagonal and full. We looked at the AIC & BIC values of each covariance type and plotted against cluster number. We are looking for the “elbow” with the lowest AIC/BIC score. This returned an optimal value of 4 with full covariance matrix. We need to be cognizant of time as different covariance matrix take longer than others. In this case, it did take at least 5 seconds longer for our optimal value vs the others at our optimal value but our better aic/bic scores warrant the extra time.
   2. Looking at the clusters we got back we are seeing similar results to what k-means returned. Data points close to the origin tend to cluster together while this further away and spread out tend to cluster together. We are seeing similar results between k-means and expectation maximization here. The only difference, clusters more towards the origin of the graph due to the higher densities of data points. These densities will have a greater effect of pulling the clusters centers towards it.

1b

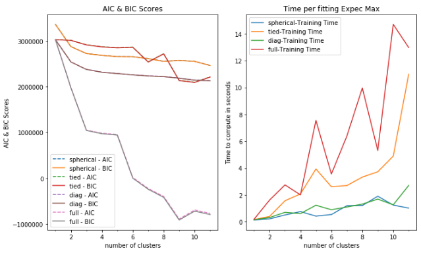
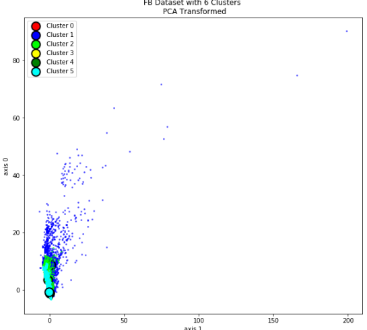
1. Feature Selection – We used the same feature selection as we did in K-means.
   1. To select our k in expectation maximization we performed a grid search between 1 and 12 k and over 4 different types of covariances spherical, tied, diagonal and full. We looked at the AIC & BIC values of each covariance type and plotted against cluster number. We are looking for the “elbow” with the lowest AIC/BIC score. This returned an optimal value of 4 with full covariance matrix. We need to be cognizant of time, in this case, it did take at least 5 seconds longer for our optimal value but our better aic/bic scores warrant the extra time.

2a

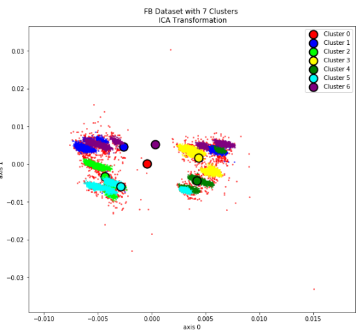
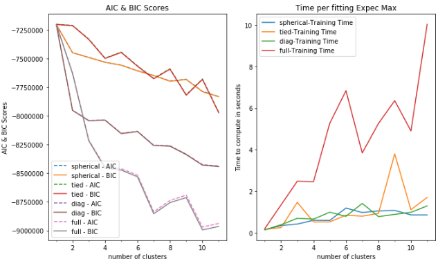
2b

* 1. Looking at the types clusters we got back, they are very similar to K-means. The closer the data is towards x-axis the more compact our clusters. The only small difference is our clusters are closer to 0 in the axis8 space vs k-means. I think we got the cluster we did because the edges have more density towards them, so expectation maximization is having a larger pull towards the axis8 than K-means.

3a

1. PCA – Used same dimensions as we did in k-means, 22.
   1. To select our k in expectation maximization we performed a grid search between 1 and 12 k and over 4 different types of covariances spherical, tied, diagonal and full. We looked at the AIC & BIC values of each covariance type against plotted against cluster number. We are looking for the “elbow” with the lowest AIC/BIC score. This returned an optimal value of 6 with full covariance matrix. We need to be cognizant of time as different covariance matrix take longer than others. In this case, the fit times at our optimal value were relatively comparable therefore picking full covariance matrix makes sense as aic/bic values are lowest.
   2. Visually, the plots look similar vs k-means. The plotting difference is the clusters are closer to our origin vs k-means. The number of clusters between expec max and k-means are different. Expec max returned 6 clusters while k-means only had 4. While the clustering displayed visually might be similar, underneath expec max is telling us that the addition of two clusters is warranted because the data within those clusters are different than if we combined them with other clusters. The types of cluster we are getting isn’t surprising as some of these clusters are rectangular which makes sense given the orthogonal assumptions of PCA.

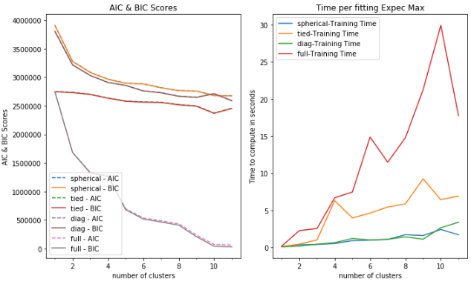
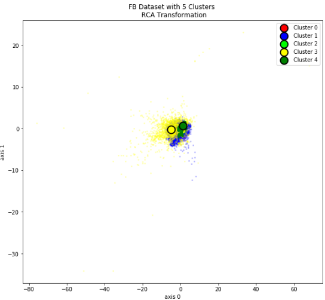
3b

1. ICA – Used same dimensions as we did in k-means, 22.
   1. To select our k in expectation maximization we performed a grid search between 1 and 12 k and over 4 different types of covariances spherical, tied, diagonal and full. We looked at the AIC & BIC values of each covariance type against plotted against cluster number. We are looking for the “elbow” with the lowest AIC/BIC score. This returned an optimal value of 7 with full covariance matrix. We need to be cognizant of time as different covariance matrix taking longer than others. In this case, the fit times for our choosing was fairly quick, <5 seconds.

4a

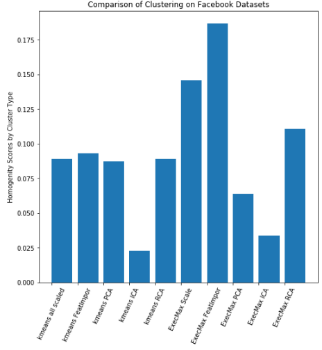
* 1. Since ICA is about independence, we can see how this projects our data into rather distinct clusters which is different than what we saw in 1b. ICA has taken our data and given us 2 distinct separations, within these separations we see natural clustering of data. It’s no surprise we are seeing this given the underlying independence between features of ICA. While one would our clustering algorithm would be able to identify these natural clusters, it is not. k-means & expec max can get stuck and stick natural clusters together when it should be its own group. This what we saw with both k-means expec max. We employed 10 random restarts to help with this, but our clusters are stuck in the middle of the separation. While the optimal clusters are different the results are similar, both have a hard time finding the independent groupings.

4b

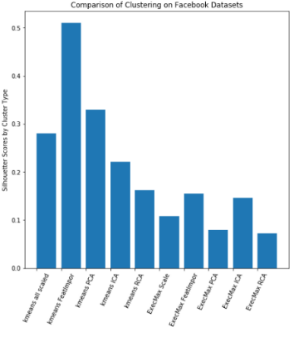
1. Random Projections – Used same dimensions as we did in k-means, 25.
   1. To select our k in expectation maximization we performed a grid search between 1 and 12 k and over 4 different types of covariances spherical, tied, diagonal and full. We looked at the AIC & BIC values of each covariance type against plotted against cluster number. We are looking for the “elbow” with the lowest AIC/BIC score. This returned an optimal value of 5 with full covariance matrix. We need to be cognizant of time as different covariance matrix taking longer than others. In this case, the fit times for our choosing our optimal values was quick, < 7 seconds.

5a

5b

* 1. These clusters are very different than what was returned by 1b but are similar to what RP for k-means returned. The location of our cluster centers between k-means and expec max for RP are very similar. This should not be surprising as our random projection and expectation maximization package are both using a gaussian process underneath to project and cluster. There is one clustering difference between k-means and expec max, k-means has one outlier cluster while expec max doesn’t have any, they are all stacked on top of each other. The type of clusters returned appear to be grouped with each other and on top of each other.

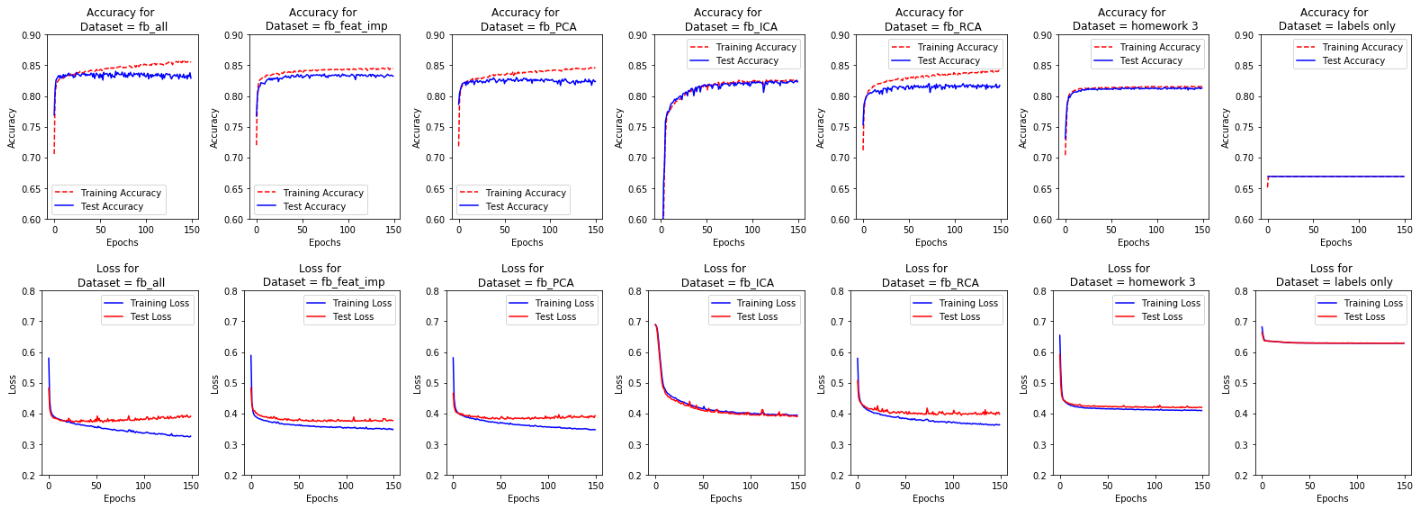
6a

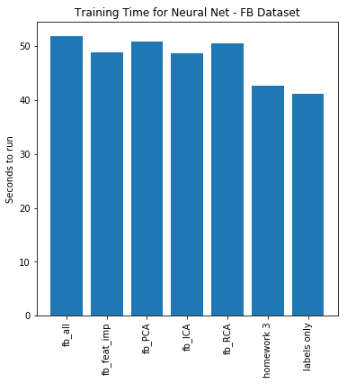
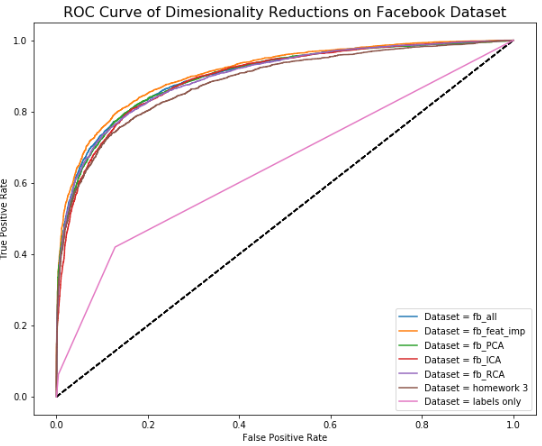
1. Comparing/Contrasting K-Means & Expectation Maximization
   1. When comparing all these feature transformations and cluster algorithms we want to see how they line up with our class labels and how compact the cluster are. In order to do this, we employed homogeneity score and a silhouette score. A cluster is homogenous if all class labels in the cluster are the same. We want our score to be as high as possible. Here the two best dimensionality reductions are feature selection and scaling all our data using clustering via expectation maximization. It’s not really a surprise that within each clustering algorithm, k-means & expec max, we see feature selection as the highest. We used a greedy algorithm, decision tree, to split our data based on the class labels. We also see the benefits of the probabilistic nature of expectation maximization here with our feature selection as it’s assigning groups based of the mean value of the cluster and the probability that it belongs in that cluster, whereas-means is finding the average of all points in that cluster. It doesn’t surprise us that ICA is the worst performing clustering technique with respect to homogeneity as both clustering algorithms had a difficult time picking up the independent clusters.

6b

* 1. We also need to see the compactness of our clusters. We used a measure called Silhouette score to tell us this. Silhouette score tells us how similar an object is to its own cluster compared to other clusters. Values close to 1 represent values identified well for its cluster where as a value of -1 would say its poorly matched and it belongs in a different cluster. All clusters algorithms have a value 0 which indicate that our clusters are not overlapping or are relatively compact. We see feature importance, PCA and all scaled variables using k-means clustering algorithm are producing the best clusters based on silhouette score. This tells us we are producing good clusters which are similar to each other and are compact. While they may be middle of the road in terms of homogeneity, they are producing clusters which are most similar.

## Facebook Neural Net

1. Run your neural network learner from assignment 3 on the data after dimensionality reduction (from task 2). Explain and plot your observations (error rates, etc.)
   1. Looking at both Accuracy and Loss we see that all types of dimensionality reduction techniques are producing similar accuracy and loss results. The worst performing neural net is the labels only, which, is using the cluster labels as the inputs. Both ICA and labels only from task 1 had the highest loss and lowest accuracy. Also, all seem to take the same amount of time to run, 40-50 seconds. This variation can be explained by other things running simultaneously only my computer.
   2. Looking at the ROC/ACU curves the following are showing a similar AUC score of 90%,all features scaled (fb\_all), feature selection (fb\_feat\_imp, PCA, ICA & RCA. Our results form homework 3 follow these at 88% and last is the labels only which see a roc/auc score of 65%.



1b

1a

1a