Unsupervised Learning (April 2017)

William Z. Ma, *Student, Georgia Institute of Technology*

Abstract – This document provides detailed analyses of six unsupervised learning algorithms of two types: clustering and dimensionality reduction. For clustering, the algorithms of concern are k-means clustering and expectation maximization. As for dimensionality reduction, the algorithms of concern are principal component analysis, independent component analysis, randomized projections, and an information gain feature selection algorithm chosen from WEKA. For each algorithm, we will tune its hyperparameters, and then run a neural network learner on the newly projected data and evaluate the learner's performance relative to its previous performance as given in my supervised learning and randomized optimization papers.

1 Introduction and Overview

1.1 Familiar Datasets and Analysis Invariants

For this paper, we will be making use of two datasets that we've used before: one of them in the randomized optimization paper, and both in the supervised learning paper. In particular, the datasets we will be looking at are:

- **2016 New Coder Survey (CS):** This dataset represents the results of Free Code Camp and CodeNewbie's 2016 collaborative survey, intended to reach out to people who are actively learning to code. 15,655 people responded to the survey's 48 questions; this dataset represents all 15,655 people's responses. We originally used age, income, whether someone is an ethnic minority, and whether someone is a software developer to predict their gender. This dataset is provided under the copyleft Open Database License, and was sourced for this paper from kaggle.com.
- **Aviation Accident Database & Synopses (AA):** This dataset is the aggregation of the National Transportation Safety Board (NTSB)'s data on civil aviation accidents and incidents in the United States since 1962. It includes data on over 70,000 accidents and incidents. We originally used latitude and the extent of damage to the aircraft to predict a longitude class, divided into 18 classes of 20 degrees each, and consequently the approximate area of the crash. The database is available under a Creative Commons, CC0: Public Domain License, and was sourced from kaggle.com.

These datasets will be used to make insights about the various unsupervised learning algorithms, which may in the process provide further insights about the data itself.

Furthermore, there will be several invariants throughout the analysis so that we may isolate the hyperparameters of concern and thereby be reasonably confident about the values that we do find:

- All unsupervised learning algorithms are those provided in the Waikato Environment for Knowledge Analysis (WEKA), available for free from the University of Waikato.
- All the data was pre-normalized to a Gaussian distribution using WEKA's provided normal filter
- Jitter was used to improve the visibility of clusters for clustering algorithms
- The neural network learner will use the same optimized hyperparameters as we found in the supervised learning paper, namely that the:

1. Maximum number of nodes in a hidden layer: **18**
2. Maximum number of hidden layers: **14**

2 Clustering

2.1 Introduction to Clustering

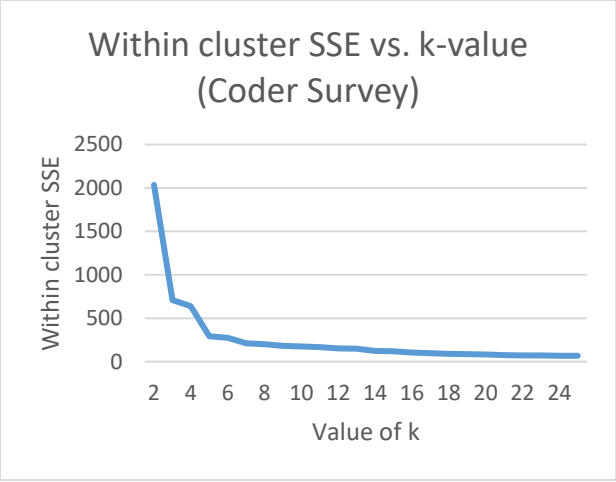
In the context of unsupervised learning, clustering refers to the general task of taking a set of data consisting of points or objects, and grouping the points or objects into clusters such that, for each cluster, the objects within that cluster are closer or more similar to one another than to objects in other clusters. This definition of closeness or similarity is given by a metric, examples of which include Euclidean distance and Manhattan distance.

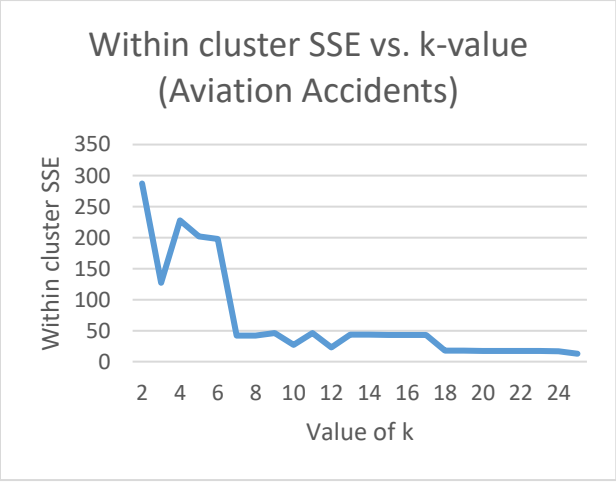
The similarity metric we have chosen to use for the analysis of clustering algorithms is the default one given by WEKA, which is the Euclidean distance, which uses a standard distance formula on attributes. This was found to be better than other available distance metrics such as the Manhattan distance or the Minkowski distance, both of which were deemed unsuitable for our dataset, as it was pre-normalized and made continuous, disqualifying the Manhattan distance, and we are not dealing with more generalized higher dimensions with our number of attributes/classes as would make the Minkowski distance suitable.

2.2 k-means Clustering (KM)

The first clustering algorithm we will look at is one of the simplest, yet with more power than the simplest we know of, which would be single linkage clustering. K-means clustering partitions the data into clusters in a more effective manner and with more intuitive end results than single linkage clustering would be able to. K-means, being less naïve about the data, begins by picking k centers at random, which do not necessarily have to be points of data, but it typically works better when they are. Each center then claims all the points of data that are closer to it than to other centers. For each of these clusters the center point is then recomputed based on the averaged or mean location of all cluster points. The algorithm then begins again by claiming points with these newly recomputed center points. This process continues until convergence, or when the center points no longer move.

With k-means clustering, there is only one hyperparameter that we can attempt to optimize on our dataset, which is the value of k, corresponding to the target number of clusters. We then attempt to find this optimal number of clusters for our data:





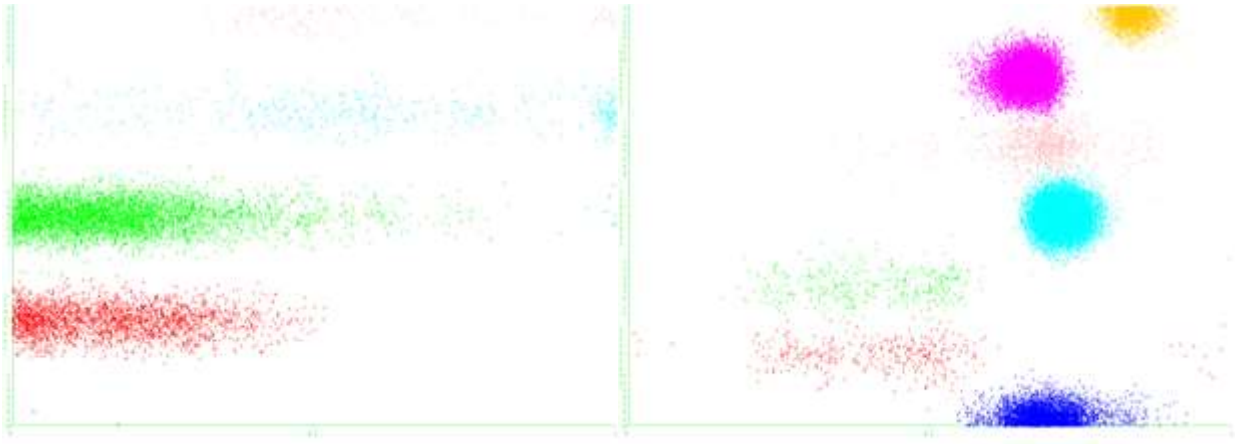
What is the within cluster SSE? It refers to the sum of squared errors within each cluster that the algorithm assigns; i.e. the actual Euclidean distance from each data point in a cluster to that cluster's centroid, averaged.

To find the appropriate number of clusters for our datasets, we made use of the elbow method, originally used by Robert L. Thorndike, a method which both graphs seem to support well, given the trends of the respective curves. We then ideally avoid overfitting. Unlike with some datasets, we can distinguish a clear elbow point at which the error no longer decreases significantly and instead begins a fairly uniform trend of asymptotically slow decrease towards an error of 0, at which the value of k would be equal to the number of data points. For the two datasets, this elbow point was determined to be k = 5 for the Coder Survey, and k = 7 for the Aviation Accidents dataset.

2.3 Expectation Maximization (EM)

Unlike k-means, EM is a soft clustering algorithm, which effectively allows a point to be in potentially as many clusters as possible (as opposed to hard clustering, where a point is in a cluster or not). Then, for expectation maximization, instead of a data point being considered to be entirely in one cluster or not, for some k number of clusters, we assign to each point k probabilities, which signify the probability of that point being in a certain cluster, which in effect actually corresponds to a Gaussian distribution for each cluster, and the probability represents the probability that that distribution could've generated such a point. We then try to maximize the likelihood that the data could've come from these distributions.

On this basis, we ran WEKA's expectation maximization clustering algorithm on both datasets, first allowing WEKA to automatically determine what it believed to be the optimal number of clusters, before then trying the optimal number of clusters that we found for the k-means algorithm. For our coder survey dataset, WEKA selected k = 2, with a log likelihood of 5.65917, while our k-means value of k = 5 gave us a log likelihood of 7.84706. A naïve way to select this value of k for expectation maximization may be to simply pick the value with the higher log likelihood. However, this would be a flawed way to select this value, as while the log likelihood for k = 5 is in fact higher than that for k = 2, the value at k = 5 is far more likely to be a result of overfitting, while the value for k = 2 was selected by WEKA using cross-validation. In addition, if we look at the actual clusters, as shown in the charts below, we see that the k = 5 parameter generates 3 clusters with an order of magnitude fewer members than a primary two, corresponding to the selected k = 2 value. This would further imply that the k = 5 value achieved its higher log likelihood value by overfitting. We thereby pick k = 2 as the optimal value of k for the coding survey dataset.



On the left is a visual representation of the k = 5 clusters for the coder survey dataset, the right the k = 7 for aviation accidents.

We see similar reasoning justified for the aviation accidents dataset, which saw WEKA finding a value of k = 2 again with a log likelihood of 11.26397, while our k-means algorithm value of k = 7 gave a log likelihood of 4.88964. We again pick WEKA's value of k = 2, for the same reasons as to avoid overfitting, the highly uneven distribution of members among the k-means value of k = 7 clusters, and also simply because the log likelihood of the k = 2 value is notably higher than the value that k = 7 gave us.

On the other hand, if we take a look at the k = 2 clusters for the two datasets, as demonstrated below, we seem to see that the algorithm seems to cluster around two discrete attribute values (adding further force behind this being the optimal); this will become important when we get to discussing the dimensionality reduction algorithms.

We can see above that the data was well clustered around the attribute values for isEthnicMinority in the coder survey dataset.

2.4 Conclusion

To see a summary of all the optimized hyperparameter values we found, see the table below:

Clustering Algorithm	Coding Survey Hyperparameters	Aviation Hyperparameters
k-means (KM)	**k:** 5	**k:** 7
Expectation Maximization (EM)	**k:** 2	**k:** 2

Furthermore, given the nature of the algorithms and the clusterings the algorithms seemed to find, I would place substantially more trust in the EM algorithm, in particular its optimal k-value finding. This has resulted in arguably better k-values than the elbow method would otherwise indicate.

3 Dimensionality Reduction

3.1 Introduction to Dimensionality Reduction

While clustering attempts to reveal some hidden structure of the data by grouping the data into clusters, as is its namesake, dimensionality reduction revolves around exactly what it sounds like; attempting to mitigate the curse of dimensionality, and in so doing reduce the amount of features or attributes required by a learning algorithm, thereby improving its performance (in both runtime and in accuracy). There are multiple ways of doing this, which are typically split into two general ideas: feature selection and feature transformation.

Feature selection is the process of finding some smaller subset of actual features which effectively preserve and potentially increase the accuracy of a learning algorithm upon them.

Feature transformation on the other hand, is the construction of derived features that are transformed versions of existing features, yet, again, preserve and potentially increase accuracy. This can be done by determining a transformation (in linear algebra terms) of the features into some smaller feature space (in dimensions), projections onto which allow us to achieve similar if not better accuracy than on the original dataset. This transformation typically reduces the number of features, thereby reducing the complexity of the problem and potentially making the data linearly separable, and is akin to the transformations done with kernels in support vector machines (SVMs). In this sense, feature transformation is more advanced than feature selection, being able to deal with insufficient indicators, features that generate false positives or false negatives. This gives feature

transformation potentially more power than feature selection. In fact, three of the four algorithms we will be looking at are feature transformation algorithms.

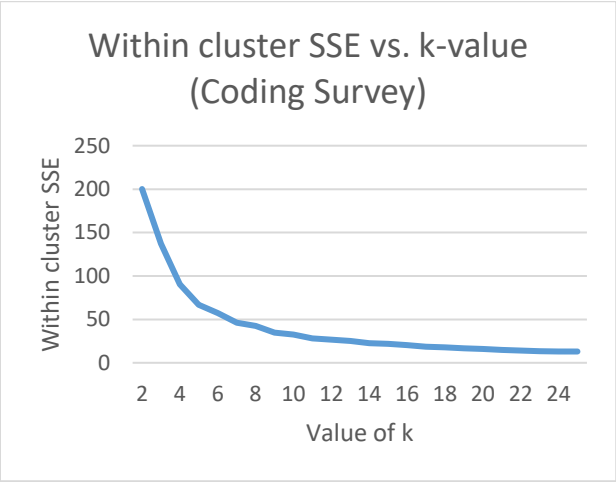
3.2 Principal Component Analysis (PCA)

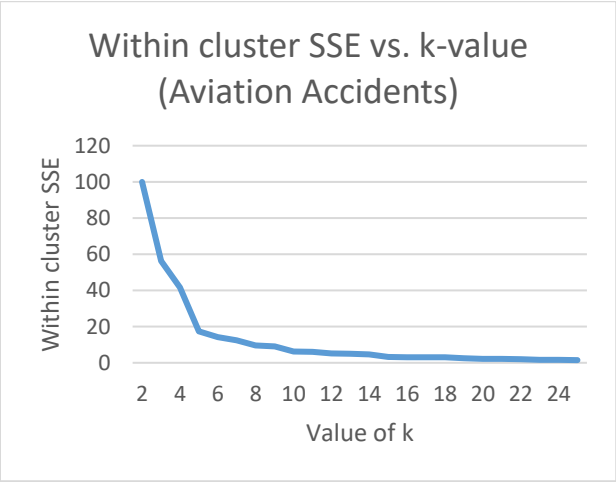
We begin with principal component analysis (PCA), a feature transformation algorithm that interprets the problem as an eigenproblem, and attempts to find a linear transformation of the features, interpreted as axes of a plot of the data, such that the projections of the data onto the transformed axes have maximum variance. These transformed axes are known as the principal component vectors. After finding a first principal component vector, the result of a linear transformation onto the original vector of features, we then look for orthogonal principal component vectors to add more dimensions with which the data is modelled. Each of these vectors have an eigenvalue which decreases the more principal component vectors that are found.

The question then, is how do we model the performance of the algorithm? Suppose in a dataset with 8 features, PCA can reduce it down to 4 suitable principal component vectors, half the original space. We would consider that to be good performance of the algorithm. Then the question is, what makes a principal component vector suitable? From factor analysis, we use the older albeit still relevant Kaiser criterion, originally proposed by Kaiser (1960), that keeps only principal component vectors with associated eigenvalues greater than 1.

Using WEKA's PCA algorithm, for the coding survey, we found only two principal component vectors that could pass the Kaiser criterion, and for the aviation accidents dataset, only one (albeit weakly); thus, we consider this a successful use of PCA to reduce the feature space in half, the original feature space consisting of four features.

We then looked to analyze the performance of clustering algorithms with principal component analysis, and look for the optimal values of k to use for k-means and for EM. The optimal number of clusters proved to be quite different than we had found originally, shown below for k-means (plotted against within cluster SSE):





We again make use of the elbow method, and find that we ended up with different values of k for our k-means clustering algorithm. While the k-value for aviation accidents is clear, at k = 5, the k-value for the coding survey is less obvious. The elbow, although in this case not sharp by any means, would arguably be the point after the inflection point in the rounded elbow, which in this case would be a value of k = 7.

As for expectation maximization, since we found that WEKA's method of finding the optimal number of clusters automatically was rather reliable, avoiding what we perceived as overfitting, while producing reasonable log likelihood, we made use of it again to determine the optimal k-values, finding a value of k = 13 for the coding survey and a value of k = 6 for the aviation accidents dataset, with log likelihoods of -2.18318 and -1.1, respectively. One may note that these log likelihoods are negative and in fact substantially less than what we

found originally. This may potentially imply a flaw in the use of the Kaiser criterion, but will ultimately be evaluated using the neural network learner. This log likelihood seems acceptable given the visualization of the clusters:

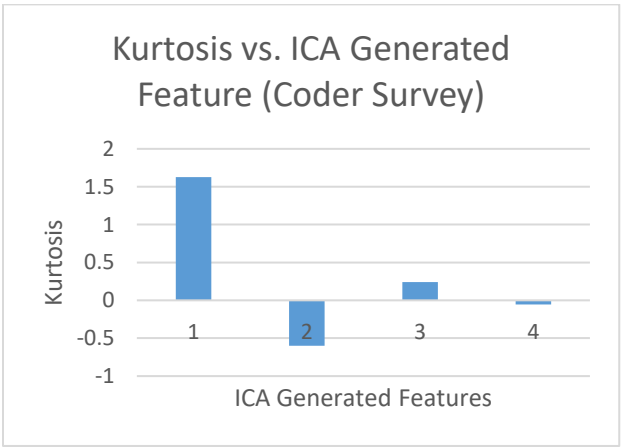
Coder Survey with k = 13 on the left, and Aviation Accidents with k = 6 on the right. Good distribution of values for clusters.

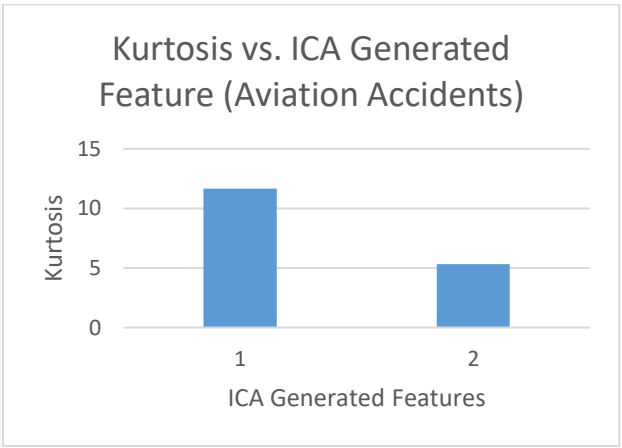
We can see above that the log likelihood is acceptable given that the visualization seems to imply that the data is not being overfit (the clusters are decently spaced and are evenly distributed in terms of objects per cluster, a major improvement in that sense over the non-dimensionally reduced runs).

3.3 Independent Component Analysis (ICA)

Independent component analysis (ICA) is another algorithm for feature transformation that has largely the same goals as PCA, but attempts to achieve it in a different way. ICA works by assuming there are hidden independent features underlying a dataset's actually described features. It attempts to find, from such features, derived independent components, that provide no mutual information to each other, but when considered as a whole, maximize mutual information with the original dataset, allowing us to potentially reconstruct it.

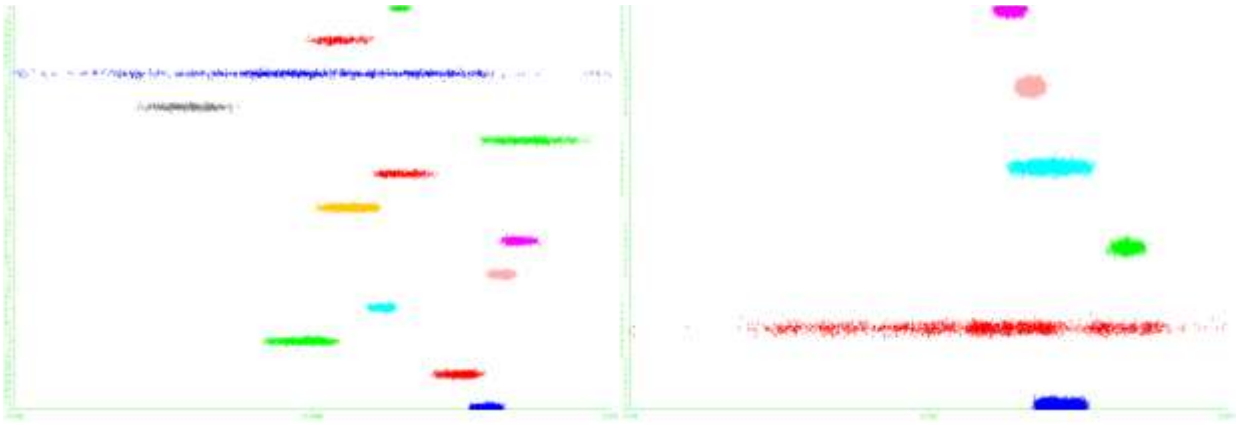
We then ran the ICA algorithm on each dataset, computing the independent components for each dataset; we then looked at the kurtosis of each independent component, attempting to determine its similarity to a normal distribution; given that ICA is assuming that each independent component is highly non-normal (given that the components should be mutually independent), we can determine how successful it was at accomplishing this by looking at how far the respective kurtoses were from the kurtosis of an ideal normal distribution, a kurtosis of 3:





As we can see above, none of the ICA generated features have kurtoses near a value of 3. This implies a high degree of mutual independence between the generated features, implying a largely successful run.

We then tested the ICA generated data with the clustering algorithms, KM and EM; we used the same values for k that we used for PCA, and found similar results; in fact, WEKA automatically determined the ideal number of clusters as being the same as the results we found for ICA (actually, we saw it being unreliable for the first time, having determined k = 38 as the ideal number of clusters for the coding survey, but this result was dismissed as only a marginal improvement in log likelihood while being far more likely to have overfit the data):



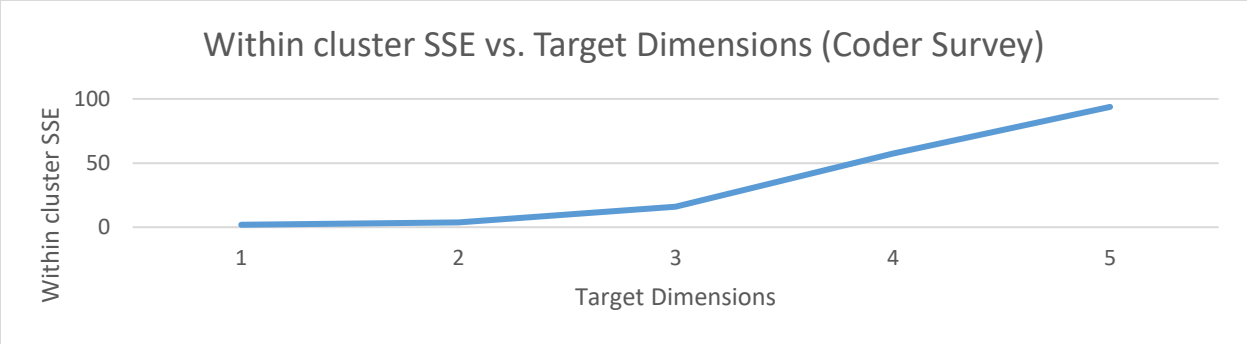
Coder Survey, k = 13 on left, and Aviation Accidents, k = 6 on the right. Good, albeit different distribution of values for clusters.

One thing to note with the above visualization of the clusters is that the use of ICA (as opposed to PCA) resulted in different choices for clusters by the EM algorithm (affecting the coder survey more so than the aviation accidents). With aviation accidents we see largely the same distribution as we saw with PCA. This may be a result of the rare case where the orthogonal components found by the PCA algorithm were in fact the most mutually independent components found by the ICA algorithm, resulting in largely similar, if not the same, transformation on the features. It could also be a result of the smaller number of features used with the aviation accidents dataset; with only two features, it may have had more trouble determining the independent components when there were so few to begin with: the independent components may very well be the original components themselves, with little change necessary (a concern that would affect PCA as well). This seems to be supported by the very much different distribution of clusters given by the algorithm on the coder survey dataset (compared to PCA), where much more freedom was given to the algorithm given a more reasonable number of features (4 in this case instead of 2).

3.4 Randomized Projections (RP)

While PCA and ICA are quite powerful algorithms in their own right, RP runs significantly faster and works well enough in practice. True to its name, the algorithm takes a specified target number of dimensions, and uses that information to generate a randomized matrix that is then used to transform and project the features into another feature vector in a smaller feature space of the specified number of dimensions.

This target number of dimensions is in fact the hyperparameter of concern when it comes to randomized projections, as in fact we can not only project into fewer dimensions but could even do so into higher dimensions (the goal being that the features may be codependent with features unknown to us, and a higher projection may thereby actually capture some of this correlation). The graph below illustrates the relationship I found between target number of dimensions and clustering error (we use the k-values that we found for PCA on k-means, results averaged over 10 trials with randomized seeds; also note the result is HIGHLY sensitive to the initial seed value, as could be expected for a randomized algorithm):



The data seems to imply that the number of target dimensions we ought to pick is 2, which is remarkably similar to the number of dimensions that we were given using PCA with Kaiser's criterion. For this reason, we pick 2 as the number of target dimensions for randomized projections with the coder survey. Similarly to ICA, attempts to reduce the number of dimensions for the aviation accidents dataset using random projections were largely unsuccessful. For this reason, the optimal number of target dimensions I picked for aviation accidents was also 2.

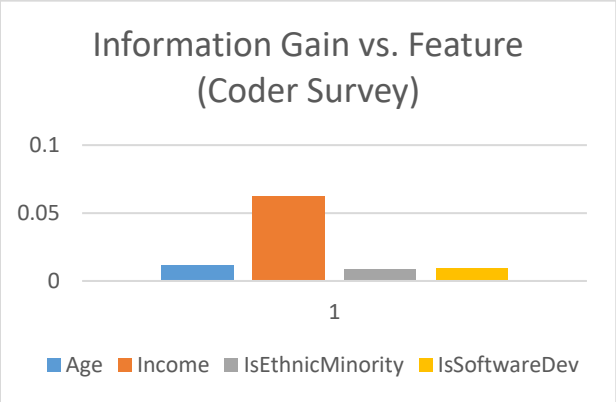
We then took a look at EM with randomized projection (using the same k-values), hoping to see some random distribution of clusters. For the coder survey, this random distribution held true, as we could not get it to 13 clusters no matter what. The algorithm always settled at 2 or 3 clusters, with 2 having the higher log likelihood. The aviation accidents dataset was more well-behaved, actually showing similar distribution to the ICA and PCA clusters, again potentially a result of the small dimensionality of the dataset to start with (neither plots of the clusters were shown, as the aviation set is familiar and the coder survey being polarized around an attribute value).

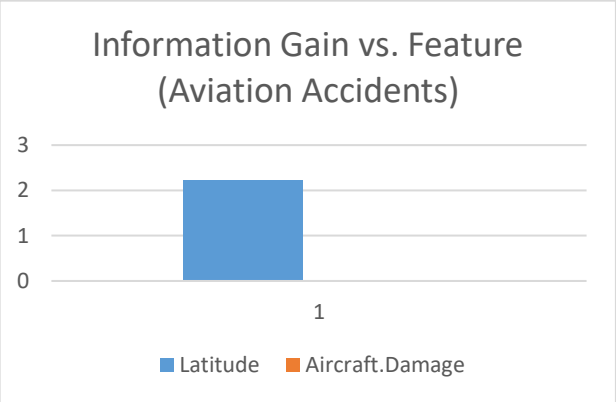
3.5 Information Gain (IG)

While PCA, ICA, and RP are all feature transformation algorithms, information gain is a feature selection algorithm. The name may be very familiar if you remember or are familiar with decision trees in the supervised learning paper. ID3, the decision tree algorithm that we used, when selecting attributes for nodes in the tree, selected attributes by maximizing information gain. Information gain feature selection works exactly the same way.

Information gain comes from information theory, and may be better known as the Shannon entropy. It is a statistical measure making use of conditional probability that ranks our features, by how impactful they are in determining the class. That is to say, the greater this value for information gain, the more important that attribute is in predicting the outcome. In this way, in order to use WEKA's information gain feature selection, we had to inform the algorithm of the class, being gender in the coder survey and longitude class in the aviation accidents. In this sense, information gain feature selection is not an entirely unsupervised learning algorithm.

There are no hyperparameters. However, see below graphs ranking the various features of our datasets:





As we can see above, certain features contribute significantly more to the result than any other feature. As a result, for coder survey we pick age and income as two features to reduce the dimensionality, and with aviation accidents we pick solely latitude as a feature to predict with. This algorithm showed similar performance to the other dimensionality reduction algorithms as far as clustering and as a result the data has been omitted.

3.6 Conclusion

We will now move on to analyzing the clustering and dimensionality reduction algorithms with a neural network learner. However, note that all dimensionality reduction algorithms were successful in reducing the number of features, or at least identifying which features were more important. Here is a summary of our findings:

DR Algorithm	Findings/Hyperparameters (CS)	(AA)
PCA	**kMeans k:** 7**, EM k:** 13	**kMeans k:** 5**, EM k:** 6
ICA	**kMeans k:** 7**, EM k:** 13	**kMeans k:** 5**, EM k:** 6
RP	**kMeans k:** 7**, EM k:** 2, **Dim:** 2	**kMeans k:** 5**, EM k:** 6, **Dim:** 2
IG	**kMeans k:** 7**, EM k:** 13	**kMeans k:** 5**, EM k:** 6

4 Neural Network Learner Results

4.1 Results without Clustering Feature

Find below a table of our neural network results, using all our tuned hyperparameters and findings from above:

DR Algorithm	CS Training Accuracy (% classified correctly)
None	80.4266
PCA	80.3569
ICA	80.3848
RP	80.3708
IG	80.3987

Oddly enough, it didn't seem like any of the dimensionality reduction algorithms had any impact on the accuracy of our neural network learner. Perhaps by adding the clustering, we may get better performance.

4.2 Results with Clustering Feature

DR Algorithm	CS Training Accuracy (KM)	CS Training Accuracy (EM)
None	80.3708	80.3987
PCA	80.3708	80.3708
ICA	80.3708	80.3708
RP	80.3708	80.3708
IG	80.3708	80.3708

It seems that even with clustering, our accuracy (and runtime) stay the same. How can we interpret these results?

4.3 Conclusion

Actually, as we saw throughout the paper, there were a variety of subtle hints that potentially would've tipped us off to the fact that neither clustering nor dimensionality reduction would seem to have an effect on the performance and accuracy of our neural network learner. For example, the similarity of clusterings despite the use

of different feature transformation algorithms. However, there are some insights to take away here, in particular about the data.

In fact the problem that we've run into there is exactly the problem as has come up previously in the supervised learning paper and in the randomized optimization paper. To recognize what I'm talking about, see the below confusion matrix, which immediately makes it clear:

0 True Positives	**1408** False Negatives
0 False Positives	**5765** True Negatives

The question that we had been asking was whether a prospective coder in the dataset was female, given information about their ethnicity, their income, their age, and their status as a developer. As a matter of fact, our neural network learner seems to have learned the most important fact about predicting from that dataset: that in fact 80.3708% of the dataset was male, and not female! Because of the heavy skewing in the dataset, this knowledge proved to be the maximum information gain if it were an attribute, and the algorithm determined that simply guessing no every time provided greater accuracy than attempting to predict the gender based off of any of the actual features that were provided.

So we have come to the same conclusion as we have before, that the dataset is so heavily skewed towards one class value, that the features are not even important when it comes to prediction. You get a very good accuracy simply from that distribution of genders! If that's the case, then what have we learned really?

What we have learned is that despite all the features that we had available in the coder survey dataset, the most telling feature of all was simply the distribution of classes, and none of the statistical tricks that we applied, be they clustering, dimensionality reduction, information gain, or any statistical manipulation, really, seem to have any impact on that. The distribution tells all, and in this case the distribution was simply the most telling of all when it came to prediction. I would surmise that if information gain and dimensionality reduction had the ability to see the distribution, they would both reduce the number of features down to a feature space of 0, simply because of how little predictive power any of the features held relative to the distribution.

However, that is not to say that none of the algorithms that we applied have any merit whatsoever, simply that we have dealt them the worst case (a highly pathological dataset when it comes to prediction). However, we can conclude, based on our experience with this dataset, the following things:

- The elbow method, albeit simple, may not be the best way to pick an ideal number of clusters: datasets may be pathological or too complex for the elbow method to be reliable
- The Kaiser criterion is not a particularly good way to pick principal components to use: while it was able to discriminate, using eigenvalues, the principal components that very much were more important than the other features, the criterion of being >1 is too simple (as we saw in this case, despite that, none of them had the predictive power that we wanted them to have)
- A more reliable metric may be necessary for determining similarity or dissimilarity to a normal distribution; in the case of ICA, we need a better, more comprehensive way of determining the best mutually independent derived features
- Random projection, for datasets where we can't uncover the underlying structure (like here), will potentially perform as well as the other algorithms for a fraction of the cost
- In this instance, information gain performed as well, if not better, than PCA or ICA, relying on simple probability and entropy factors instead of transformations

Works Cited

Hyvarinen, A. What is Independent Component Analysis? Department of Computer Science at the University of Helsinki.

Kaiser, H.F. (1960). The application of electronic computers to factor analysis. Educational and Psychological Measurement, 20, 141-151.

Sawasthi. Principal Components and Factor Analysis. Department of Mathematical Statistics at the University of Texas, Arlington.