CS 7641 HW 3 | Author: Michael Pecorino | GT ID: mpecorino3

1 Introduction

Dataset 1: sports analytics is becoming increasingly interesting for two main reasons. The first has to do with technology. Professional sport leagues are consistently researching ways to bring technology into the game that will capture data in new and exciting ways. For example, Major League Baseball (MLB) is using the Hawk-Eye [1] camera system to track ball and player movement. In the most recent upgrade, the system can even track player's limbs which can allow for an entirely different set of questions to be answered. The National Basketball Association (NBA) and Division I NCAA Men's College Basketball (NCAAMBB) use similar motion-capture technology called STATS [2]. The second driver behind sports analytics is the increase in the legalization of sportsbooks, which are companies that facilitate gambling on sporting events within certain states. The sports betting industry is expected to be an $8 billion-dollar industry by 2025 [3]. That figure will only grow as sports betting inevitably becomes legal in all 50 states. For this project, I am focusing on NCAAMBB school win/loss classification.

Dataset 2: Similar to the technology used to track every play in sporting events, smartphones have built in sensors that describe the orientation and movement of the device. While the phone is in our pocket, it can determine if we are standing, sitting, laying down, walking, walking downstairs, and walking upstairs. Therefore, the classification problem is of the multi-class type. This information can provide stakeholders (smartphone users and perhaps one day hospitals and insurance companies) with the data to study how movement impacts physical and mental health.

2 Data

Data for the Division I NCAA Men's College Basketball classification problem were taken from Sports Reference [4] and will be referred to as NCAAMBB in the remainder of the analysis. Data for the smartphone sensor classification problem were taken from Kaggle [5] and will be referred to as SENSOR in the remainder of the analysis. Table 1 shows the dimensions and description of the response variable for each dataset. The NCAAMBB response is nearly perfectly balanced, and the SENSOR response is reasonably balanced. Therefore, accuracy will be used to evaluate the models throughout the analysis.

Table 1: Dimensions and response variable

Data	**Observations**	**Variables**	**Response (Numerical Encoding, % of data)**
NCAA Men's College basketball (NCAAMBB)	51,347	41	Win (1, 50.1%) /Loss (0, 49.9%)
Smartphone sensor (SENSOR)	10,299	567	Laying (0, 18.9%), Sitting (1, 17.3%), Standing (2, 18.5%), Walking (3, 16.7%), Walking Downstairs (4, 13.7%), Walking Upstairs (5, 15.0%)

NCAAMBB only uses team level statistics as data from player movement sensors is not readily available to the average person. Thus the number of variables to describe win/loss is reasonable. Relative to NCAAMBB, the engineering behind the SENSOR data requires many more variables in order to capture the orientation and movement of the phone. Data processing steps were identical for both datasets, which entirely consists of normalization for the KNN algorithm so that variables with large magnitude do not dominate the distance calculation.

Since the NCAAMBB dataset describes human decision making in the game of basketball, in which noisy signal is inevitable, it is expected that generalization error for this dataset will be much lower than the more scientific SENSOR data. This hypothesis will tested visually throughout the analysis.

3 Algorithm descriptions

3.1 Clustering

Clustering techniques are part of the unsupervised learning family of algorithms in which a correct answer, or response variable, is not used. Clustering methods are useful tools for exploratory data analysis and extracting information from the data, independent from the response. If done well, the output from the clustering algorithms can serve as inputs to a supervised modeling problem, like new features for a classification problem.

k-Means Clustering

k-Means Clustering (KMC) is a procedure that groups observations based on their similarity. The concept is similar to the K-Nearest Neighbors (KNN) algorithm from supervised learning. With KNN, we compute the similarity between a query point and observations in the training data using Euclidean distance. With KMC, we compute the similarity between observations in the data and cluster points. After each iteration of similarity calculations, the observations are assigned to their most similar cluster point. The cluster centers are then moved to the mean of the data assigned to the cluster. Since the clusters can change location at the end of each iteration, the procedure continues until all observations have stabilized their cluster assignment or until we reach some maximum number of iterations. The objective is that by the end of the algorithm, each cluster will represent a distinct group of observations.

KMC algorithm performance can be determined by a few different metrics. One popular one used in this analysis is the sum of squared error from the cluster centers, also known as the within-cluster sum of squares. It is up to the modeler to determine the optimal number of clusters using business knowledge about the problem and deciding on a tolerable sum of squared error at each cluster. Often it helps to visualize how performance changes for various cluster amounts and that plot will be presented in a subsequent section. If the number of clusters matches the number of observations, then the within-cluster sum of squares would be 0. If the number of clusters is 1, then the within-cluster sum of squares would be maximal since all of the data belongs to the same group. We will see that as the number of clusters increases, the within-cluster sum of squares improves monotonically, but not at the same rate. Depending on the problem, most of the improvement in the performance metric can be in the first few clusters, followed by a flatter improvement for the large quantities of clusters. This pattern usually works well for the interpretability of the results. The whole point of this exercise is to understand the data by sorting the observations into groups. There needs to be a reasonable number of groups in order to be able to properly understand the results.

There are a variety of methods for choosing the location of the initial cluster points. Among the methods are to initialize them as actual observations in the data or to choose their locations randomly. The outcome is dependent on the initial location of the clusters, so the algorithm has functionality to perform the entire operation n number of times, as determined by the modeler. The best result from the n iterations is kept.

Expectation Maximization

Expectation Maximization (EM) is a two step density estimation or clustering technique. The first step is an expectation step that evaluates posterior probabilities in the presence of possible latent, or missing, variables. Since this is an unsupervised learning approach, one obvious latent variable is the response variable. It estimates cluster probabilities that best fit the data. The second step maximizes the model parameters through maximum likelihood estimation and determines the class based on the estimated probabilities from the prior step. A key difference compared to KMC is that in the former algorithm, observation belongs to one cluster while in EM, observations can belong to multiple clusters. This technique then provides a probabilistic soft-clustering that gives us more flexible information to understand the data or to use as features in a model.

Like KMC, the number of clusters is an input to the algorithm and the modeler can optimize the trade-off between computation time, performance, and interpretability. EM is more computationally expensive than KMC, and for this reason it is common to choose the number of clusters through KMC first and then use that same cluster amount in the EM algorithm.

3.2 Dimensionality Reduction

Dimensionality Reduction techniques are important to the Machine Learning toolbox. Some complex problems, like those in the fields of Natural Language Processing or Computer Vision, are very high dimensional with tens of thousands of variables. Not every variable is important, and using too many variables can necessarily slow down the modeling process. It is good practice for the modeler to determine the most important variables before modeling to

prevent wasting time working with variables that are not useful. Three of the techniques discussed in this section include dimensionality reduction techniques that transform the data into a new space that allows us to see the underlying structure of the data. Those methods include, Principal Component Analysis, Independent Component Analysis, and Random Projection. We will also consider a feature selection method using a tree based model, Random Forest.

Principal Component Analysis

Principal Component Analysis (PCA) is a method for transforming the data in such a way where the new set of variables, called the principal components, are uncorrelated and ordered by the amount of variance they explain. The new set of variables is the same quantity as the original set of variables. If the original set of variables have high correlations, then PCA will be able to transform the variables in such a way where the first few principal components explain most of the variance in the data. If the original data does not have high correlations, then the principal components will more uniformly be responsible for the variance in the data. Therefore, the results of PCA leads naturally into dimensionality reduction since most of the information in the data is, hopefully, in the first few principal components. However, the actual dimension reduction task is left up to the modeler. The modeler must determine a threshold for the variance explained. As a hypothetical example, if the raw data has 12 variables and the first 5 principal components provide 80% of the variance, this may be acceptable to the modeler and the dimensions can be reduced from 12 to 5. PCA requires the data to be scaled before implementation of the algorithm. Each variable was centered by subtracting its mean value from every observation and given unit variance by dividing by its standard deviation.

Independent Component Analysis

Independent Component Analysis (ICA) is a method for isolating underlying factors in a multivariate analysis through a data transformation. One popular application of this method is blind source separation. In blind source separation, there are multiple sources transmitting to multiple receivers (like multiple cell phones emitting radio waves to multiple cell stations). Each receiver has data from all sources and the goal is to separate the data so that each source is isolated. ICA is typically used when the data is nongaussian and we want to find statistically independent components. We will use the kurtosis metric, which measures how different the distribution is from a normal guassian. Unlike PCA, the ICA output is unordered. It is up to the user to determine the usefulness of each transformed variable. However, since the goal of ICA is to find the independent sources of information, the new, transformed variables may carry more signal and help the modeling process. Like PCA, ICA requires the data to be scaled before implementation of the algorithm. Each variable was centered by subtracting its mean value from every observation and given unit variance by dividing by its standard deviation.

Random Projection

Random Projection (RP) is all about randomly projecting the data onto a lower-dimensional space so that the final result has a smaller amount of variables compared to the original data. This procedure uses the Johnson-Lindenstrauss Lemma which states that after the random projection, the distances between observations is relatively maintained and therefore the uniqueness of the observations is relatively maintained. The fact that the uniqueness of the observations is relatively maintained allows for effective classification modeling.This is accomplished through matrix multiplication in which the original data is multiplied by a random matrix with the same number of rows (observations) as the original data and each column (variable) having unit length. Unlike PCA and ICA, the modeler must choose the number of columns of the random matrix, and therefore choose the dimensions of the resulting dataset.

In other words, the number of dimensions we want in the transformed dataset is like a hyperparameter that can be tuned. The modeler can vary the number of dimensions and measure the impact to the model performance. We would examine the tradeoff between complexity and speed and model performance and decide on a transformation size. Since this is a random process, it can often be much faster than some of the other methods like PCA and ICA, so the time requirements for these algorithms will be compared. We expect this process to perform better on the higher-dimensional SENSOR dataset because the random combinations of variables will have less variance as the number of variables increases.

Tree Based Model Selection using Random Forest

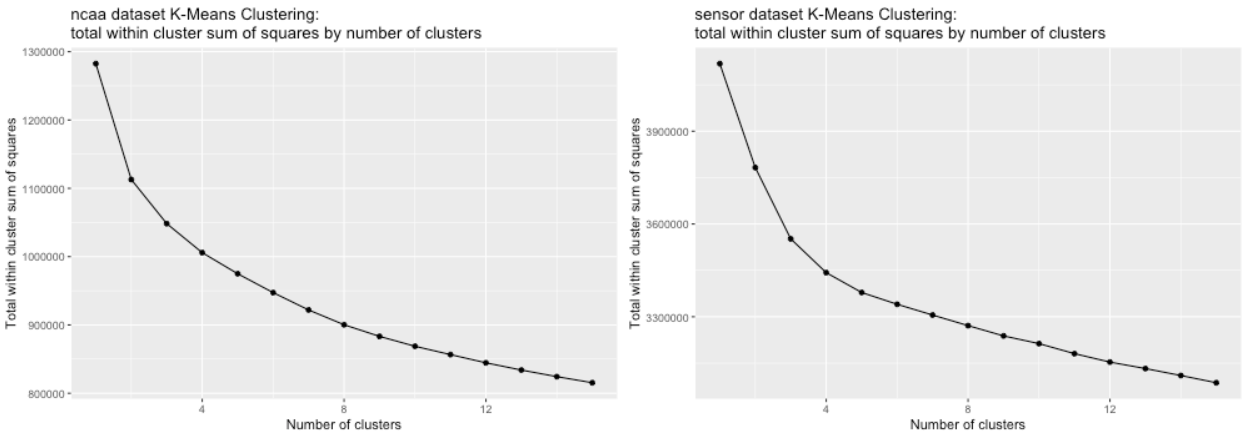
The algorithm behind decision trees, by design, will determine which variables are most important. For example, the information gain metric provides a quantitative measurement that can be used to determine a rank ordering of

variable importance For this reason, many modelers will use complex tree models like XGBoost or Random Forest to this list of variable importance. We can then determine a threshold to reduce the number of variables, e.g. select variables that provide 95% of the total information.

4 Results

The focus of the results section will highlight how k-Means Clustering (KMC) and Principal Components Analysis (PCA) work together to perform both observation grouping and dimensionality reduction and eventually achiever acceptable model performance. First, an initial KMC is run giving equal weight to every variable. Next, PCA is implemented to reduce the number of dimensions. After the dimensions are reduced, the KMC algorithm is rerun with the reduced, transformed set of variables. At the conclusion of this focused section, some results are presented for Expectation Maximization, Independent Component Analysis, Random Projection, and Random Forest feature selection.

Initial k-Means Clustering (KMC)



The plots above are sometimes referred to as "elbow" plots due to their resemblance to the bend of an arm. The elbow-shaped inflection point is a rule-of-thumb for determining the amount of clusters to use. However this can be subjective and it is good practice to determine a tolerance quantitatively. For example, if we set 15 clusters to be the maximum we will consider, then we might want to go no higher than 110% of the within cluster sum of squares of 15 clusters. This results in cluster amounts of 9 for each dataset. Note that this values are much higher than the true number of labels for the NCAA dataset (2) and slightly higher for the SENSOR dataset (6).

The confusion matrices between the labels and the clusters are provided in Tables 2 and 3. For this purpose, the KMC algorithm was implemented again with exactly the same number of clusters as there are labels. The percentages in the table use the label as the denominator, so the rows add to 100%. For the NCAA dataset, the algorithm was able to sort better the games classified as losses rather than wins. The SENSOR dataset is reasonable sorted into clusters. Each of the 6 classes have most of the data in 1 or 2 clusters. Without changing anything else, there is still room for improvement with variable selection because each variable is was given equal importance in the KMC algorithm. It is very likely that not all variables would contribute
equally to the observation groupings.

Table 2: Confusion matrix for NCAA KMC

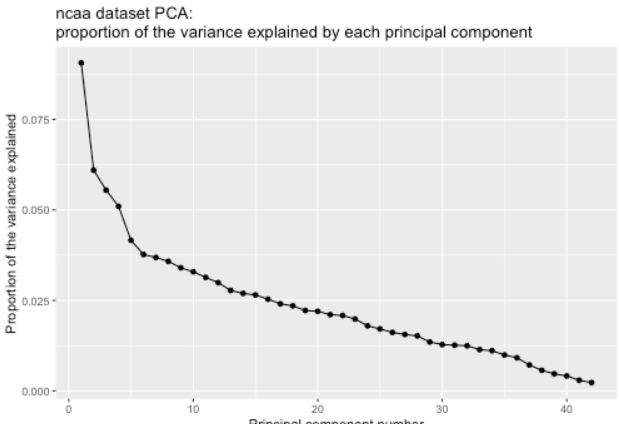
		Cluster	
		1	**2**
Label	**0 (loss)**	22.15%	77.85%
	1 **(win)**	43.57%	56.43%

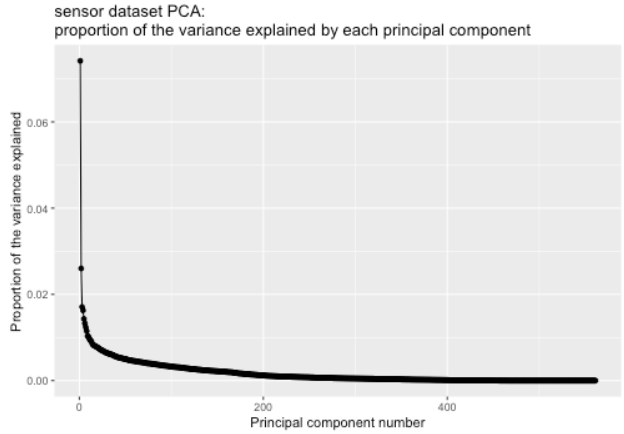
Table 3: Confusion matrix for SENSOR KMC

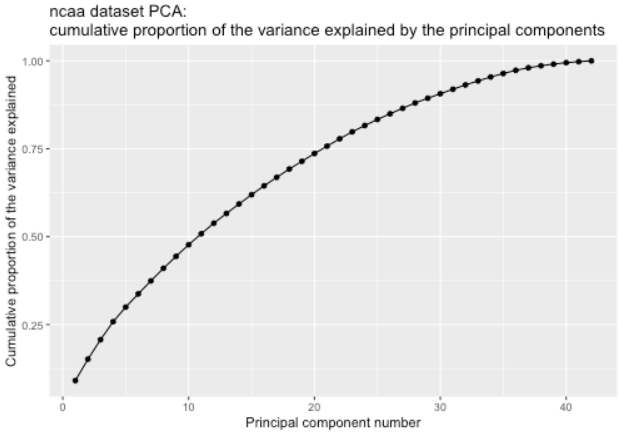
		Cluster					
		1	**2**	**3**	**4**	**5**	**6**
LABEL	**1**	2.73%	80.04%	0.26%	0.00%	0.00%	16.98%
	2	69.44%	5.12%	0.06%	0.00%	0.00%	25.38%
	3	70.30%	0.00%	0.00%	0.00%	0.00%	29.70%
	4	0.00%	0.00%	52.44%	4.47%	43.09%	0.00%
	5	0.00%	0.00%	22.83%	13.94%	63.23%	0.00%
	6	0.00%	0.00%	80.44%	0.32%	19.11%	0.13%

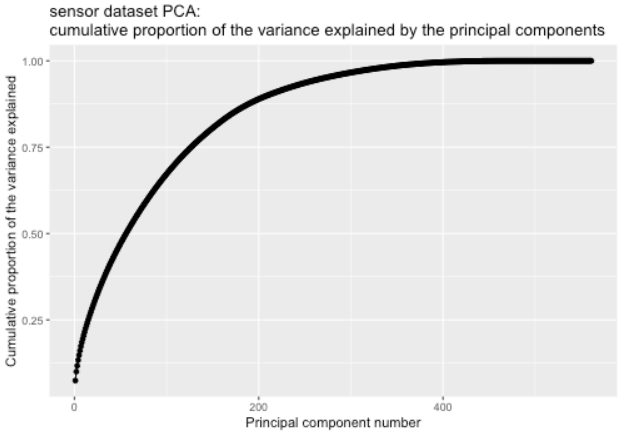
Principal Component Analysis (PCA)

We can plot the variance explained by each principal component, which also has the same shape as the plot of the eigenvalues. We can also plot the cumulative variance explained in order to visualize a threshold for the amount of transformed variables to keep. We know that if the raw variables are highly correlated, then more of the variance will be explained by the first few principal components. The variable explained by the components in the SENSOR dataset decrease much more sharply than in the NCAA dataset. In order to reduce the dimensions, we determine a threshold for the variance we want to maintain. For example, we can set the threshold at 90%. This means for the NCAA dataset we need to keep the first 30 principal components (a 29% reduction in the number of variables) and for the SENSOR dataset we need to keep the first 210 principal components (a 63% reduction in the number of variables).



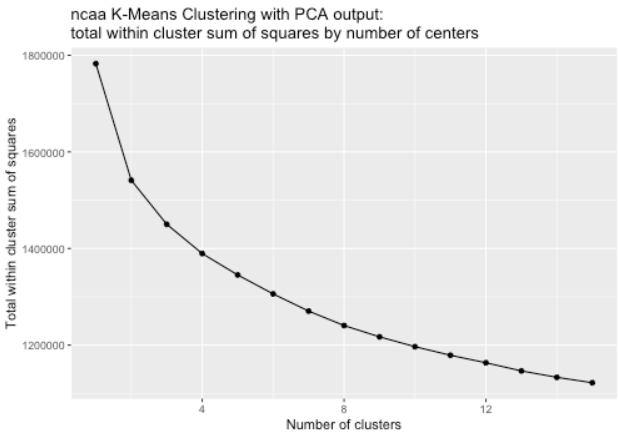


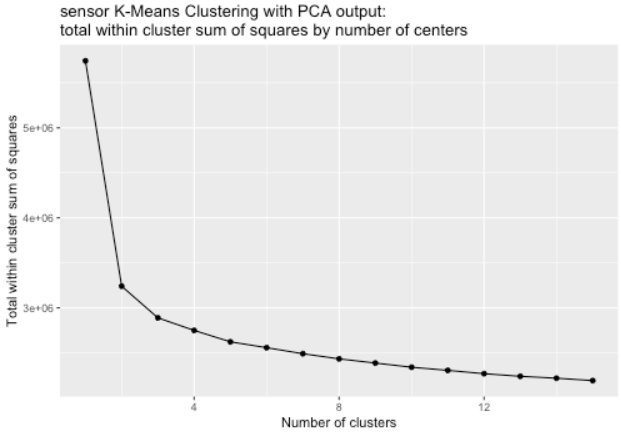




k-Means Clustering with PCA

After reducing the number of dimensions to 30 and 210 in the NCAA and SENSOR datasets, respectively, we rerun the KMC algorithm two ways. Once to determine the optimal number of cluster assignments using the same method as before, and once with the same number of clusters as the number of labels to gauge the impact of reducing the dimensions. The plots are a bit steep than before, indicating that more of the within cluster sum of squares can be captured by less clusters. However, using the 90% rule of thumb, we still require 9 clusters for each dataset. These cluster assignments will be used in the next section where Neural Network models are fit to the data.





Tables 4 and 5 provide the updated confusion matrices after performing PCA. Relative to the previous results in Tables 2 and 3, the cluster assignments are nearly identical. This shows the power of PCA. The dimensions were reduced rather significantly and we can still achieve the same cluster assignments with the transformed data.

Table 4: Confusion matrix for NCAA (PCA)

		Cluster	
		1	**2**
Label	**0 (loss)**	77.86%	22.14%
	1 **(win)**	43.55%	56.45%

Table 5: Confusion matrix for SENSOR (PCA)

		Cluster					
		1	**2**	**3**	**4**	**5**	**6**
LABEL	**1**	2.73%	80.04%	0.26%	0.00%	0.00%	16.98%
	2	69.44%	5.12%	0.06%	0.00%	0.00%	25.38%
	3	70.30%	0.00%	0.00%	0.00%	0.00%	29.70%
	4	0.00%	0.00%	52.85%	4.47%	42.68%	0.00%
	5	0.00%	0.00%	22.83%	13.66%	63.51%	0.00%
	6	0.00%	0.00%	80.57%	0.32%	18.98%	0.13%

Neural Network with PCA only

Now that the dimensions have been reduced, we can test the results in the Neural Network from assignment 1. The results are summarized in Table 6. Interestingly, performance was comparable for the NCAA dataset and better on the SENSOR dataset (although likely not statistically significantly better). There may be some overfitting occurring in the NCAA dataset that, when fixed, could improve the results of the test data. Equally important, the Neural Network was able to fit the models approximately 50% faster compared to using all of the variables.

Table 6: Neural Network results comparison after PCA

	NCAA		SENSOR	
	Train	**Test**	**Train**	**Test**
Assignment 1	70.41%	71.50%	98.93%	94.57%
Assignment 3: after PCA	72.99%	69.63%	99.93%	95.08%

Neural Network with PCA + KMC

In addition to reducing the number of dimensions with PCA, we can further reduce the dimensions by using the cluster assignments from the KMC with PCA process. We will use just the 9 cluster assignments to fit a Neural Network model. The clusters are split into dummy variables in order to capture each cluster assignment properly - keeping them in a single variable would improperly assume an order exists among the clusters. The results are in Table 7 row 3. For both datasets, the models underfit the data. The reason for this is because the cluster assignments are not properly capturing the signal in a way where a model can appropriately assign weights to each cluster feature. We can improve the modeling results here by improving the cluster assignments, which may require strategically assigning weights to variables rather than using uniform weight.

Table 7: Neural Network results after PCA and KMC

	NCAA		SENSOR	
	Train	**Test**	**Train**	**Test**
Assignment 1	70.41%	71.50%	98.93%	94.57%
Assignment 3: after PCA	72.99%	69.63%	99.93%	95.08%
Assignment 3: after PCA + KMC	61.31%	61.33%	64.53%	67.19%

Expectation Maximization (EM)

As mentioned previously, EM provides a probabilistic cluster assignment. This flexibility allows us to work with the probabilities for each cluster as well as the single cluster assignment by taking the argmax of the probabilities for each observation. Using the argmax to get a single cluster puts the output in a form similar to KMC. Here we implement the argmax method and repeat the comparison of labels and cluster assignments (Tables 8 and 9). The results for NCAA is an improvement over KMC. For the 0 label, the proportions are similar, but there is a significant improvement in the 1 label. For the SENSOR dataset, there was some mixing of the clusters compared to the KMC method.

Table 8: Confusion matrix for NCAA EM

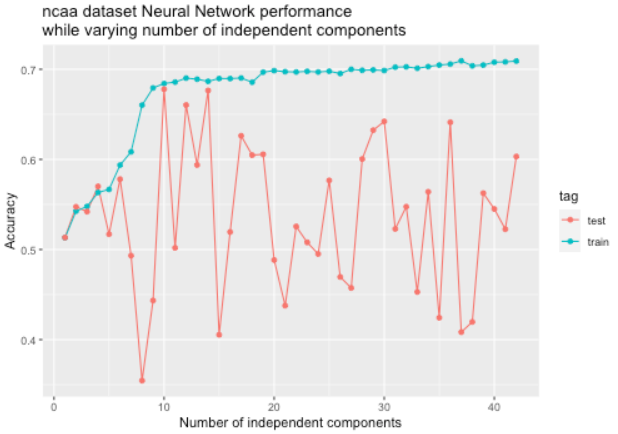
		Cluster	
		1	**2**
Label	**0 (loss)**	74.21%	25.79%
	1 **(win)**	33.27%	66.73%

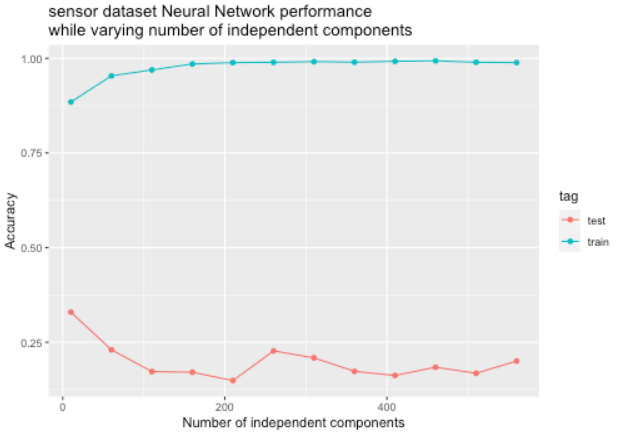
Table 9: Confusion matrix for SENSOR EM

		Cluster					
		1	**2**	**3**	**4**	**5**	**6**
LABEL	**1**	78.03%	0.62%	0.00%	0.00%	0.00%	21.35%
	2	75.97%	0.11%	0.00%	0.00%	0.00%	23.92%
	3	65.79%	0.00%	0.00%	0.00%	0.00%	34.21%
	4	0.00%	32.52%	46.69%	3.60%	17.19%	0.00%
	5	0.00%	11.17%	26.60%	11.74%	50.50%	0.00%
	6	0.00%	61.92%	31.54%	0.13%	6.41%	0.00%

Independent Component Analysis (ICA)

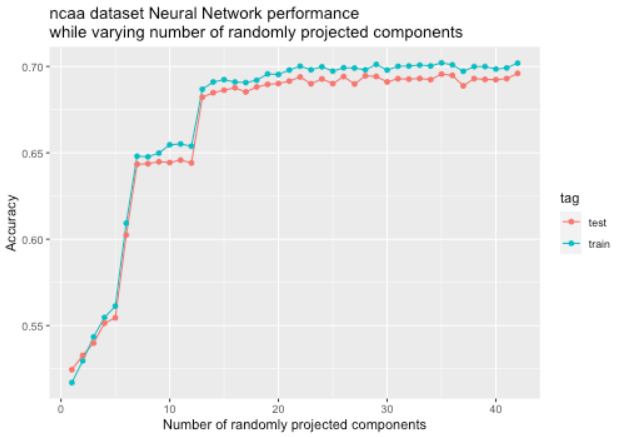
ICA transforms the variables so that they are all independent. The number of independent components was varied and then implemented in the same Neural Network model from assignment 1. No further variable selection techniques were implemented to determine the most important variables after the transformation. This is obvious in the results, especially for the SENSOR dataset for which all of the models performed very poorly at each number of independent components. For the NCAA dataset, there is much more variation in the result. The best result achieved here is just under 68%, which is significantly worse relative to the 71% best result from assignment 1. In regards to the kurtosis metric, the transformed data for the NCAA dataset was 5.38 and for the SENSOR dataset was 10.86. Further research is required here to optimize exactly which independent components are most related to the response variable.

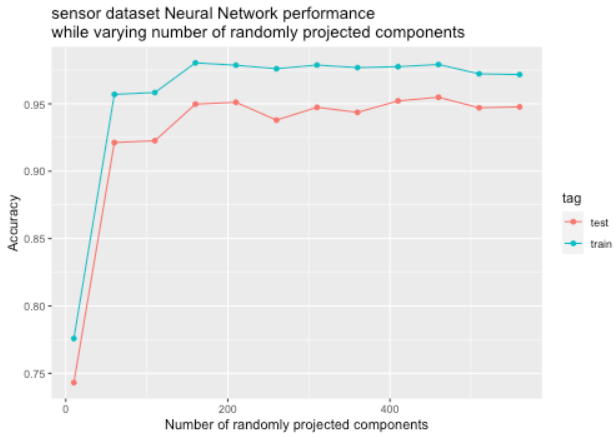


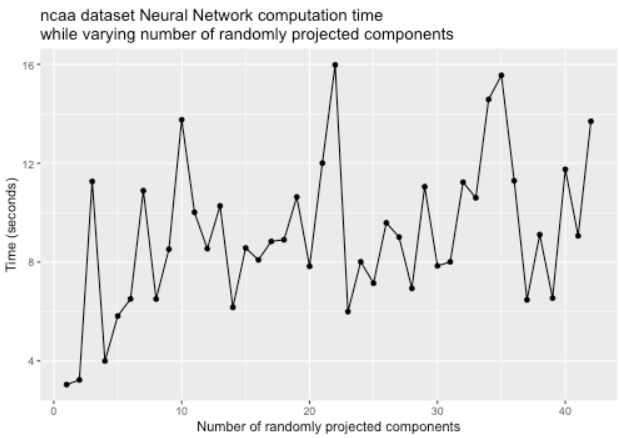


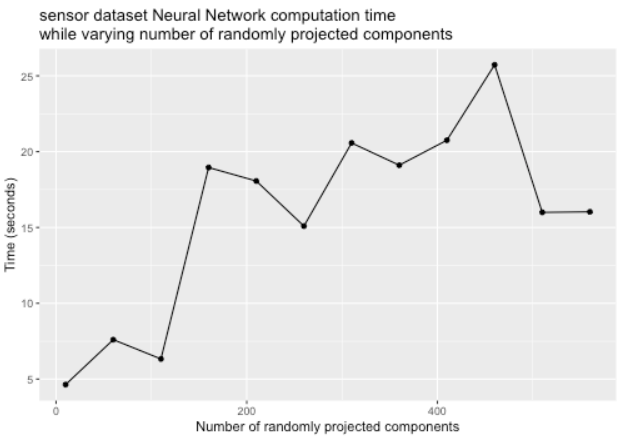
Random Projection (RP)

Here we vary the number of transformed variables and immediately measure the impact on the Neural Network model from assignment 1 in regards to performance (top charts) and computation time (bottom charts). The best Neural Network performance on the test data from assignment 1 was 71.50% accuracy and 94.57% accuracy for the NCAA and SENSOR datasets, respectively. The results below are actually an improvement for the SENSOR data, while the NCAA score is relatively close. In terms of the training time, since the NCAA dataset is a small number of variables in comparison to SENSOR, the time benefits of random projection are negligible. However, for the SENSOR dataset, the computation time is approximately 25% of the time needed in the assignment 1 model. The computation time gain, in addition to the great model performance, makes the benefit of RP obvious. The results here show that RP does better with datasets with a higher dimension of variables. Overall, we can clearly see that relatively good performance can be obtained with about 50% of the number of raw variables.



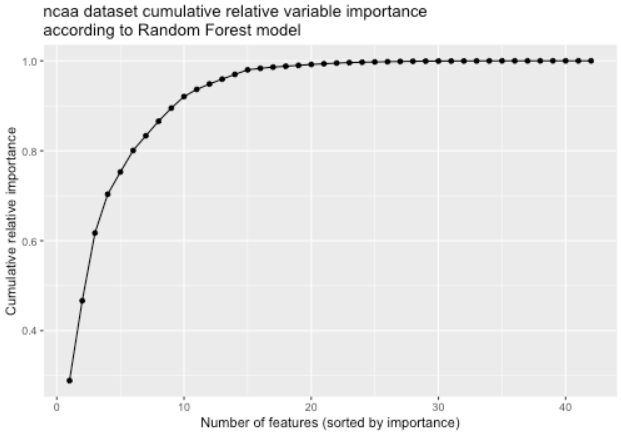






Tree Based Model Selection using Random Forest

The tree based models like Random Forest track the total information gain contributed by each variable. We can sort on this metric and then take only those variables that cumulatively provide 95% of the total information (see plots below). In doing so, we reduce the NCAA dataset from 42 to 12 features (71% reduction) and the SENSOR dataset from 561 to 127 (77% reduction). We will take the top features according to this method and refit the Neural Network model from assignment 1 comparing the performance and computation time. Table 10 provides the model performance. Relative to Assignment 1, the performance can be improved, but the model fit took only 25% of the computation time on the reduced number of dimensions. In future research, the 95% threshold for the total information gain should be tuned to determine the impact on model performance. The 95% threshold reduced the dimensions by over 70%. It's possible we can strike a balance by increasing the threshold and getting better performance. For example, if the threshold was set to 97.5% of the total information, the corresponding reductions would have been 67% in the NCAA dataset and 71% in the SENSOR dataset. These are still better than the results we got from PCA.



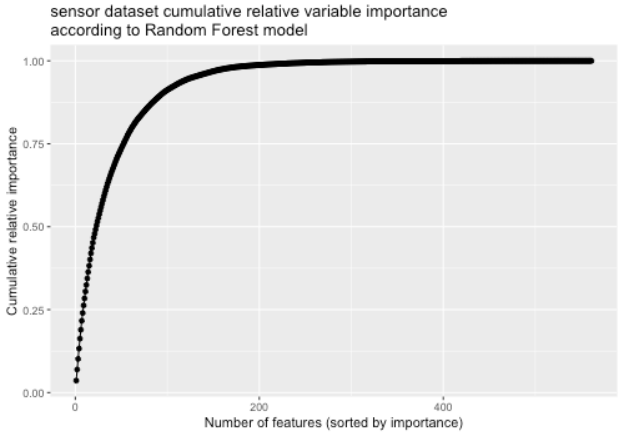


Table 10: Neural Network results after Random Forest variable selection

	NCAA		SENSOR	
	Train	**Test**	**Train**	**Test**
Assignment 1	70.41%	71.50%	98.93%	94.57%
Assignment 3: after PCA	72.99%	69.63%	99.93%	95.08%
Assignment 3: after PCA + KMC	61.31%	61.33%	64.53%	67.19%
Assignment 3: after Random Forest	69.19%	68.64%	95.32%	89.68%

5 Conclusion

This report explores various clustering and dimensionality techniques. We first worked with k-Means Clustering and showed how to group observations into clusters. We then performed dimensionality reduction using Principal Component Analysis and reduced the number of variables in the NCAA dataset by 29% and the number of variables in the SENSOR dataset by 63%. We then used the transformed, smaller dataset to fit the same Neural Network model from Assignment 1 and achieved similar results for NCAA and better results for the SENSOR dataset. We then repeated the k-means Clustering algorithm on the smaller dataset and saw that the cluster assignments were nearly identical to using the entire dataset. This shows that an algorithm like k-Means clustering can benefit from using a variable transformation or selection technique prior to running the algorithm. We further reduced the number of dimensions by using the cluster assignments as variables in the Neural Network model, but the models were very underfit. The best way to improve the performance of the process is to change from the uniform weighting applied to the variables. We can use a variable importance technique to vary the weights for each variable in the hopes of improving the cluster assignments.

We also explored another cluster algorithm, Expectation Maximization, as well as three other dimensionality reduction techniques: Independent Component Analysis, Random Projection, and tree based selection with Random Forest. Similar clustering results were obtained with Expectation Maximization, but k-Means Clustering performed slightly better. Dimensionality reduction with Random Projection worked very well in the Neural Network model from Assignment 1. We were able to reduce the number of dimensions by about 50% while still getting about the same accuracy as in the Assignment 1 model. For the SENSOR dataset, we were able to do this using only 25% of the computation time, showing that Random Projection performs better when the data has more dimensions. The NCAA dataset performance was also close to the former result with comparable computation time. The key difference was the size of the dimensions and we learned that Random Projection performs better on datasets with a large amount of variables. We then implemented a Random Forest algorithm to determine the top features that provide 95% of the total information gain. Model performance was not as good as Assignment 1, but the computation time needed was only 25% of the time needed to fit the model using all of the variables. In the future, the 95% cutoff should be tuned to see how model performance is impacted.

6 References

1. https://www.hawkeyeinnovations.com/sports/baseball

2. https://apnews.com/press-release/pr-businesswire/e3b90b0946e549e5952159a82c14e69a

3. https://www.marketwatch.com/story/firms-say-sports-betting-market-to-reach-8-billion-by-2025-2019-11-04

4. https://www.sports-reference.com/

5. https://www.kaggle.com/