Better Segmentation Results With Deep Learning: Dimensionality Reduction Using Auto-Encoders

Joseph Retzer, ACT-Solutions

"Unsupervised learning is primarily concerned with uncovering latent, non-random structure in data. By uncovering this structure, a deeper understanding of the data, and potentially how it was produced, is possible."

Philip Waggoner

Abstract

This paper investigates the comparative performance of dimensionality reduction techniques in the context of unsupervised learning, with particular emphasis on cluster analysis. The traditional method of Principal Component Analysis (PCA) Jolliffe (2002) is contrasted with deep learning-based Auto-encoders (AEs) Hinton and Salakhutdinov (2006), using a dataset of 28 numeric features. Both PCA and AE compress the feature space to seven dimensions before clustering is performed using the Partitioning Around Medoids (PAM) algorithm Kaufman and Rousseeuw (1990). Cluster quality is assessed using the Caliński-Harabasz Index (CHI) Caliński and Harabasz (1974), Davies-Bouldin Index (DBI) Davies and Bouldin (1979), and Silhouette plots/scores Rousseeuw (1987). Results show that auto-encoders substantially outperform both raw data and PCA-transformed data, achieving a silhouette score of 0.32, which exceeds the commonly accepted threshold for interpretability.

In addition to improved clustering results, the paper highlights the interpretability and flexibility of auto-encoder components using correlation heatmaps, dependence plots, and feature permutation importance. While AEs require careful model tuning, their ability to capture nonlinear relationships and preserve more information makes them a superior choice for dimensionality reduction in high-dimensional data. The paper also emphasizes the importance of reproducible feature engineering pipelines and introduces the `tidymodels` ecosystem in `R`, illustrating its use for tasks such as normalization, imputation, and interaction term creation. Overall, the study demonstrates that deep learning methods like auto-encoders can offer meaningful advantages in segmentation and unsupervised modeling tasks.

Preparing Data for Modeling: Feature Engineering:

As most researchers are aware, data analysis typically begins with the time-intensive process of transforming raw data into meaningful features that improve model performance. This critical stage, typically the most time-consuming in the modeling workflow, often involves a series of complex and iterative steps.

In the machine learning field, this process is known as "feature engineering". Basic feature engineering encompasses a number of familiar steps outlined below:

- Basic Feature Engineering (data cleaning)
  - Imputing missing data,
  - remove outliers,
  - correcting inconsistencies (spelling errors, duplicate records, formatting inconsistencies, etc.),
  - converting data types (e.g. dummy / one-hot encoding), etc.

While basic data cleaning is critical for pre-processing data, feature engineering also involves numerous additional transformations, some of which are outlined below:

- Going Beyond Data Cleaning
  - Feature transformation (e.g., normalization, log transformation),
  - feature creation (creating interaction terms, splines, etc.),
  - collapsing infrequent levels in high cardinality categorical variables,
  - dimensionality reduction, etc.

Dimensionality reduction, often accomplished through Principal Component Analysis (PCA), is arguably the most critical step when preparing data for cluster analysis. High dimensional data can negatively impact cluster quality and should be dealt with at the outset, before the clustering algorithm is run.

High Dimensional Data: Whats the Problem?

In order to illustrate the impact of high dimensional data on cluster analysis results, we may consider the following:

> **Example**: Assume we have 10 variables with an average number of levels equal to 5.
>
> This results in $5^{10} = 9,765,625$ possible combinations.

The feature space in which the data reside is notably large, even for this relatively simple dataset. The resulting data sparsity causes observations to appear nearly equidistant from one another, making it difficult to discern meaningful patterns. While various commonly used clustering algorithms will still produce the requested number of clusters under these conditions, the resulting clusters typically exhibit poor quality due to the lack of distinct groupings.

Moreover, a frequently requested deliverable in applied clustering—namely, a scoring algorithm—is also adversely affected. Specifically, the predictive performance of such algorithms tends to degrade as the distinctiveness of clusters diminishes. This decline in both clustering effectiveness and predictive accuracy, with increasing dimensionality, is a manifestation of the well-known "Curse of Dimensionality." A widely used strategy to mitigate this problem is "Dimensionality Reduction," which may involve reducing the number of variables in a dataset while preserving as much relevant information as possible (e.g., through Principal Component Analysis).

Two approaches to dimensionality reduction, Principal Component Analysis (PCA) and Deep Learning-based Auto-encoders (AE), will be described and illustrated. Cluster analysis results will be compared using data created from each approach along with the raw data.

An overview of the methodology to accomplish this, is described below.

- Describe and illustrate PCA and Deep Learning AE's.
- Employ a data set comprised of 28 features, all numeric.
- Use PCA to find optimal number of PC's.
  (The same number of Auto-Encoders (AE's) will also be constructed)
- Perform cluster analysis using "Partitioning Around Medoids" (PAM) on
  (1) Raw Data,
  (2) Principal Component Data and
  (3) Auto-Encoder Data using the following metrics:

Resulting partitions will be compared based on:

- The Caliński-Harabasz Index (CHI),
- Davies-Bouldin Index (DBI) and,
- Silhouette Value / Plot

Principal Component Analysis (PCA):

Principal Component's (PC's) are *linear combinations* of the original variables. Weights used for averaging the variables are taken from the eigenvectors of the decomposition of the data covariance matrix. It is straightforward to show that using the eigenvectors as weights results in the maximum variance of the average.

PC's are then ordered by the amount of variance captured from data. Only the first few, most informative, PC's are retained. The process therefore reduces the number of dimensions while attempting to retain as much of the data's variability as possible.

Note: Dimensionality reduction using PCA "requires" loss of information since some PC's must be dropped.

PCA Output

A graphical depiction of both the percentage of information and its proxy, eigenvalues associated with each PC, is shown in Figure 1 below.

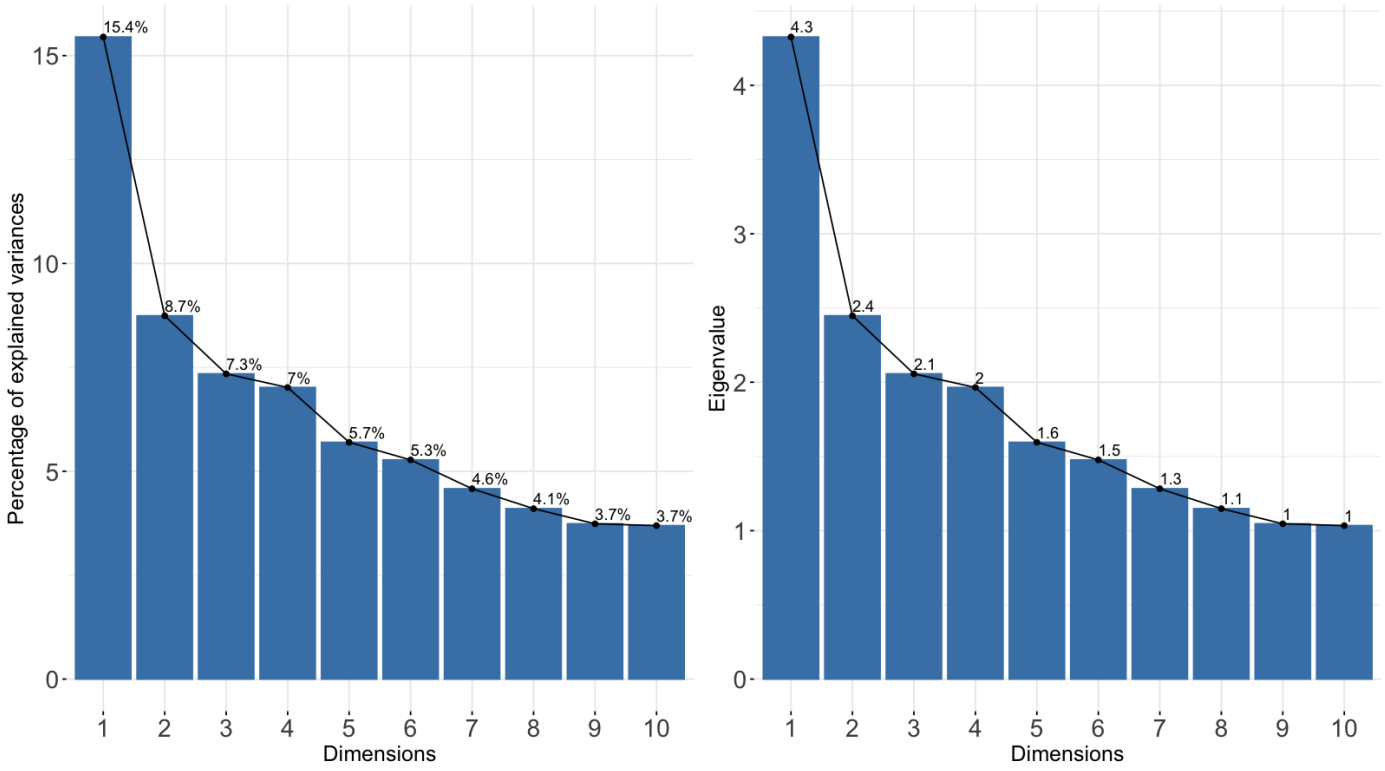


Figure 1: PCA Output

Based on the PCA output, the first 7 PC's were chosen to replace the data used to construct them. The same number of Auto-Encoders will also be selected.

Neural Network Based Auto-Encoders

A graphical depiction of a simple auto-encoder is shown in Figure 2 below.

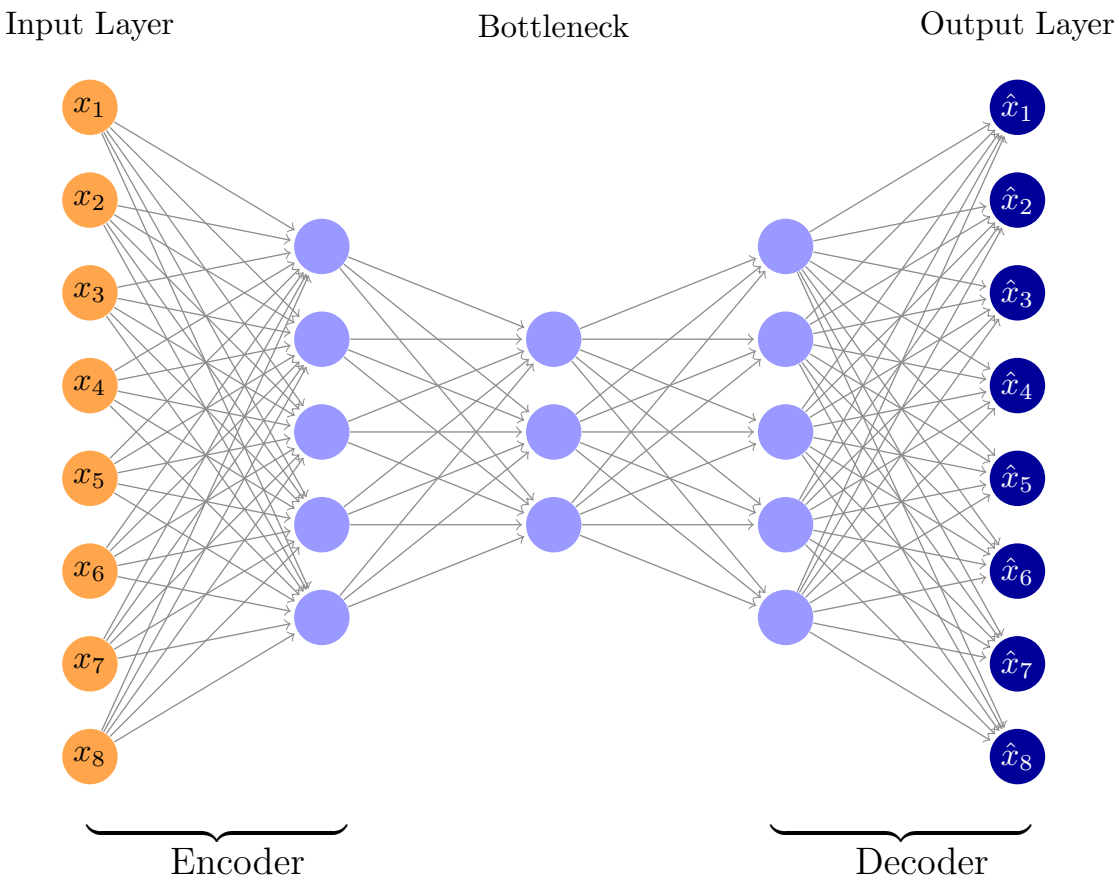


Figure 2: Simple Auto-Encoder

Note that each column of circles is referred to as a "layer" and each circle in a column is referred to as a "node".

The auto-encoder depicted in Figure 2 illustrates a **fully connected, symmetric auto-encoder** with the following structure:

- **Input Layer:** Consists of 8 input features (nodes), denoted as $x_1, x_2, \ldots, x_8$. This is the raw data.
- **Bottleneck Layer:** A central latent space where the input data is compressed into a lower-dimensional representation (3 auto-encoders in this example) known as the bottleneck layer. This layer captures the most salient features of the input.
- **Output Layer:** Comprises 8 output nodes, labeled $\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_8$, which aim to reconstruct the original data.

This auto-encoder is designed to learn a compact encoding of 8-dimensional input data via the bottleneck layer. The encoder compresses the input into a lower-dimensional latent representation, while the decoder reconstructs the input from this compressed form.

The network is trained to minimize reconstruction error—in this case it would be the mean squared error (MSE) loss function involving the difference between the actual and reconstructed features—thereby enabling the discovery of efficient, information-preserving representations of the original data.

Comparing Raw Data, PC's and AE using Cluster Analysis

As noted earlier, partition quality metrics **Caliński-Harabasz Index (CHI)**, **Davies–Bouldin index (DBI)** and the **Silhouette Score / Plot** will be used to evaluate cluster solutions resulting from "Partitioning Around Medoids" (PAM) cluster analysis. The CHI and DBI are both internal cluster validity indices used to evaluate clustering quality however CHI is often preferred for evaluating well-separated and compact clusters, particularly in high-dimensional settings.

The **Silhouette Score** measures how well each data point fits within its assigned cluster. It also provides a graphical depiction of quality by cluster.

As evidenced in the graphics shown in Figure 3, both DBI and CHI strongly indicate a higher quality solution provided using auto-encoders.

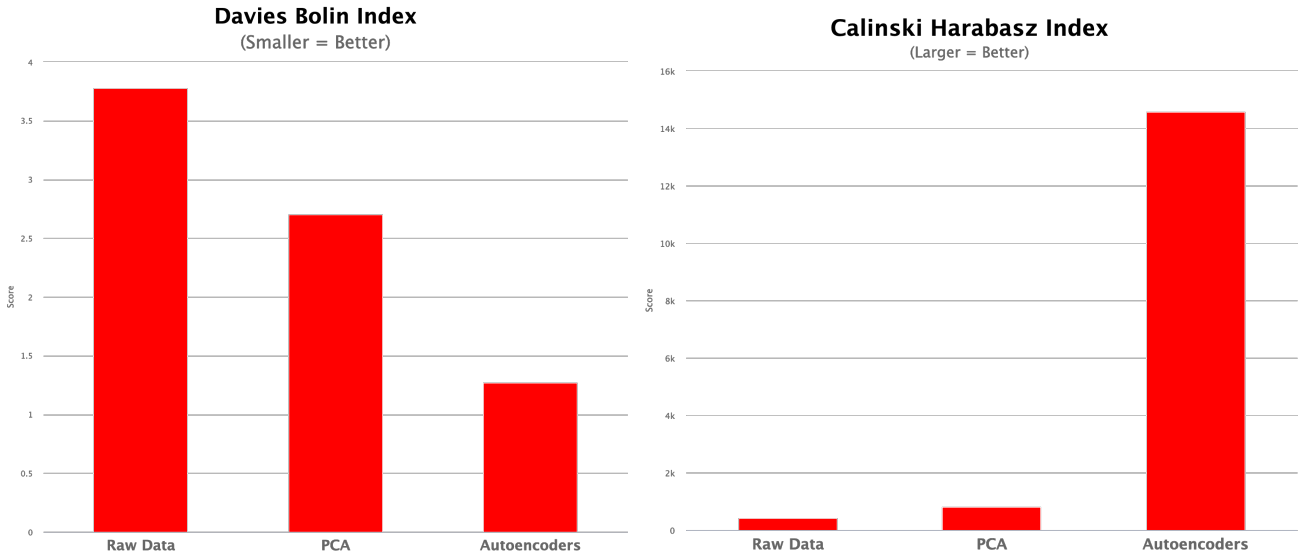


Figure 3: DBI & CHI Metrics

Next, the silhouette plots (and average silhouette scores) will be compared again using partitions derived from (1) raw data, (2) PC's and (3) auto-encoders.

As shown in the silhouette plots in Figure 4, both the clustering partitions based on the raw data and those derived from the principal components exhibit unacceptably low average silhouette values. In particular, neither partition achieves the commonly accepted threshold of 0.2, which is typically considered the minimum for an interpretable and reliable clustering solution.

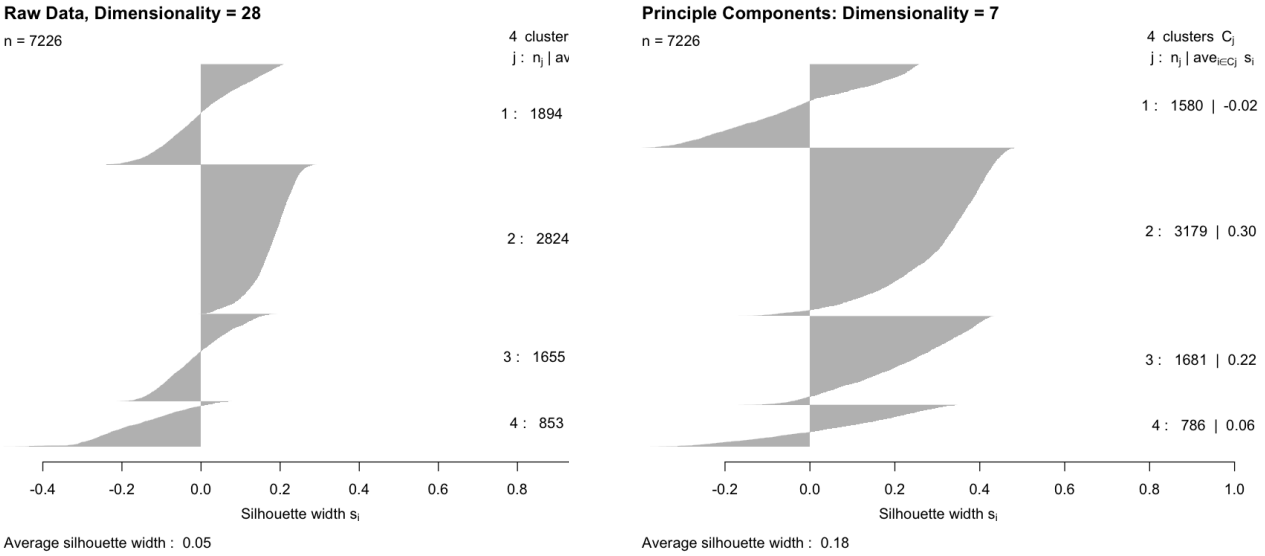


Figure 4: Silhouette Plots: Raw Data & PC's

Figure 5 reflects the silhouette plot when using auto-encoders as data. This plot clearly indicates a greatly improved partition with average silhouette calculated as .32, well above the minimal threshold of .2.

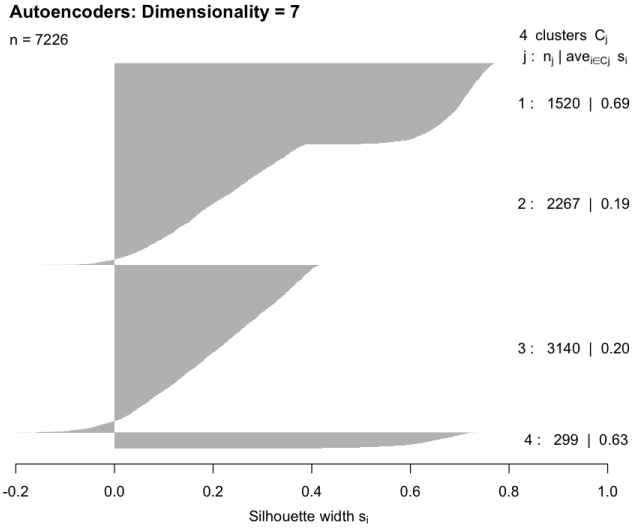
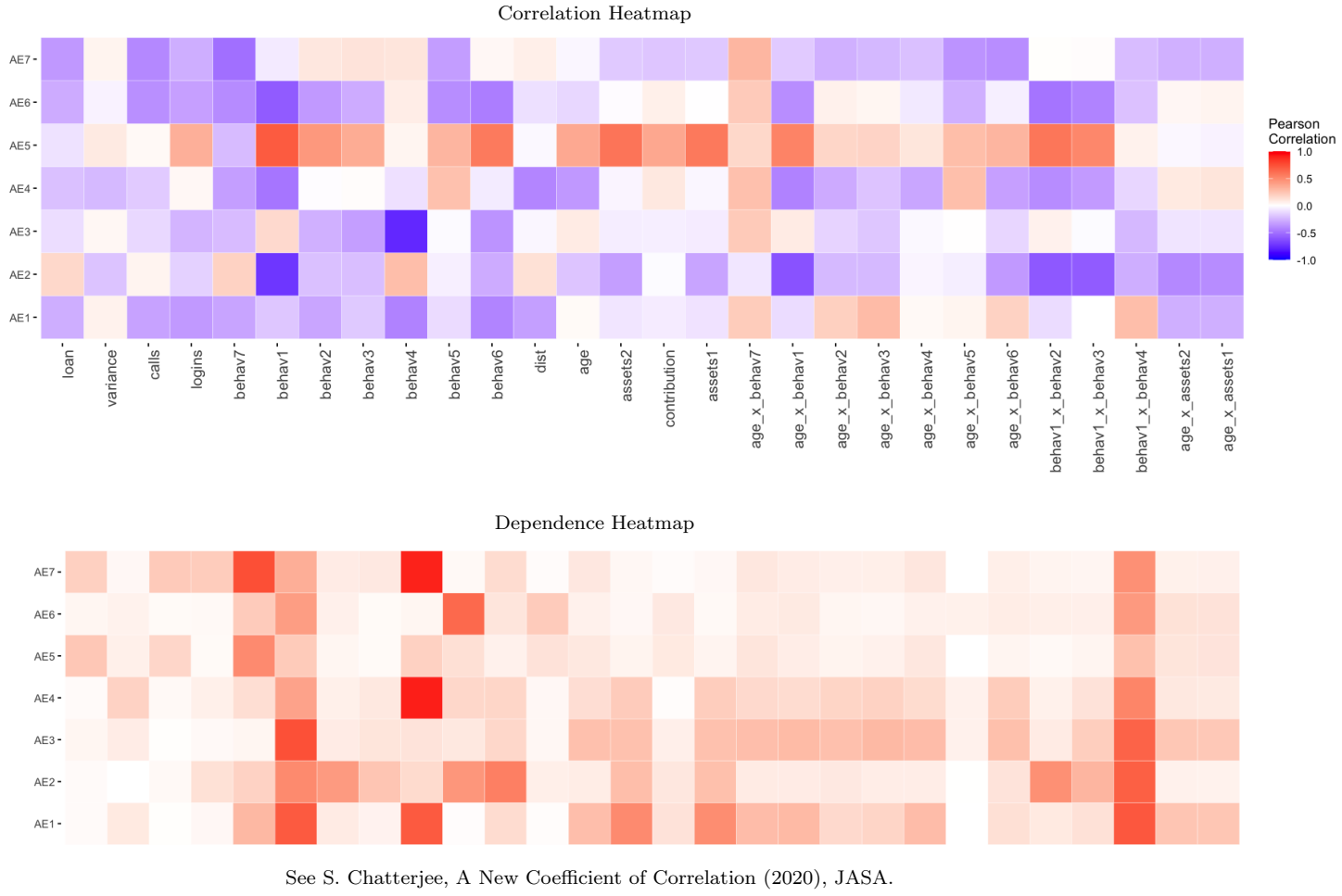


Figure 5: Auto-Encoder Silhouette: Average Silhouette = .32

Given that the auto-encoders yield higher-quality partitions, it is important to interpret the structure and meaning of each resulting component. To facilitate this understanding, a range of visualization techniques may be employed. Among them, a correlation heatmap and a more general dependence measure heatmap—such as the one proposed by Chatterjee Chatterjee (2021)—are particularly useful. These visualizations are presented in Figures 6 and 7. In these plots, the vertical axis corresponds to the auto-encoder components, while the horizontal axis represents the original input features.



See S. Chatterjee, A New Coefficient of Correlation (2020), JASA.

Figure 6: Correlation & Dependence Heat Maps

Another potentially useful visualization pertaining to auto-encoders is "Feature Permutation Importance Bar Chart". This involves making comparisons between approximated feature values and the original data before and after permuting a specific feature of interest. The greater the difference in actual vs. estimated features due to the permutation, the more important is the feature in the construction of the auto-encoders. A feature permutation bar chart is illustrated in Figure 7.

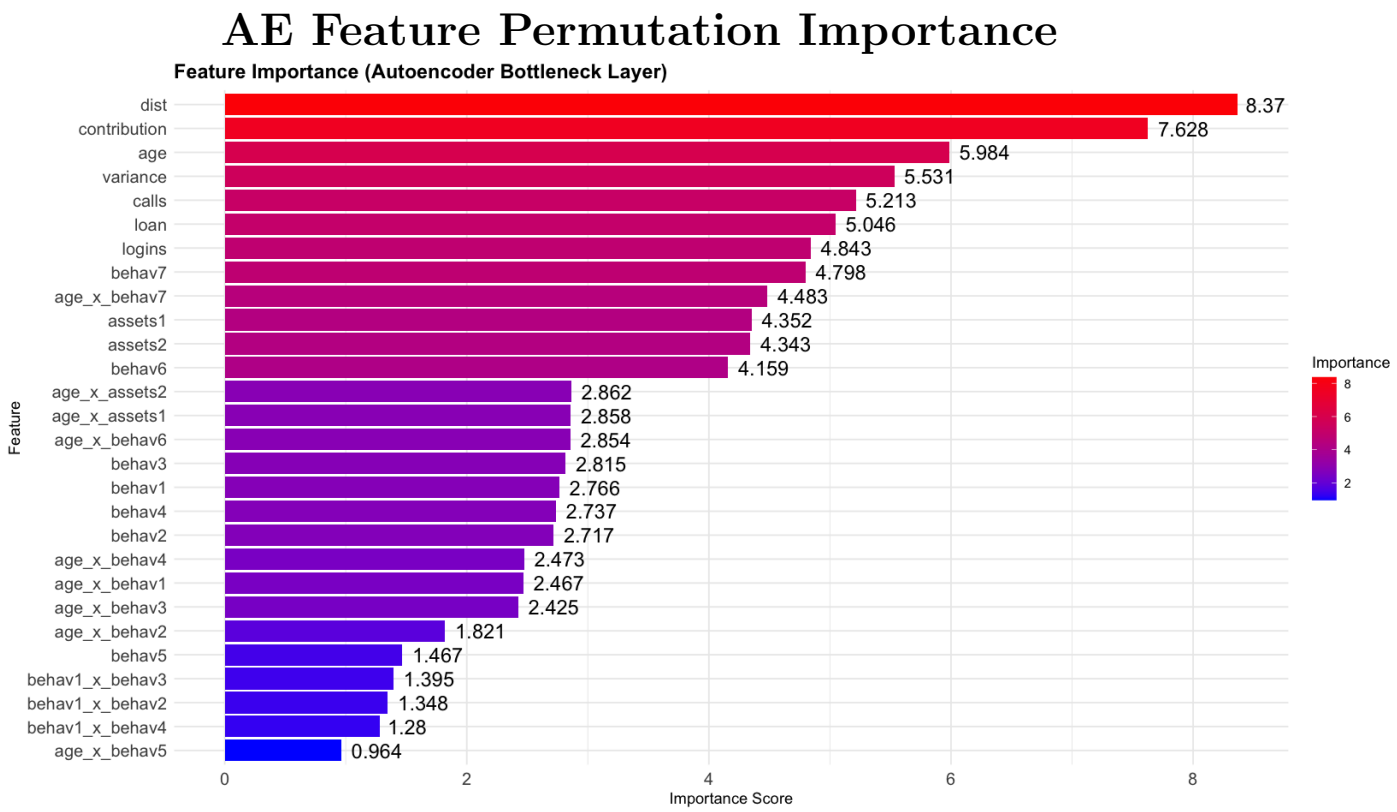


Figure 7: Feature Permutation Importance

Based on this example, several conclusions can be drawn regarding the comparative effectiveness of auto-encoders and principal component analysis (PCA) as techniques for dimensionality reduction:

- **Dimensionality Reduction**
  - AE's learn non-linear relationships and preserve information. This may result in higher quality cluster solutions.
  - It is also noteworthy that, due to their superior ability to retain information, auto-encoders may offer more effective two- or three-dimensional graphical representations of the data compared to principal components.

- AE's may also be used to detect data anomalies, as well as, de-noising data for subsequent analysis.

- **Drawback of AE** Unlike PCA, auto-encoder construction may involve significant model tuning. Numerous measures may be tuned including, but not limited to: gradient descent learning rate, optimal number of nodes, layers, epochs, batch size (number of observations used in a single forward/backward propagation. Default is usually 32), activation function(s) choice, etc.

Feature Engineering with `tidymodels`

While dimensionality reduction is often the most critical feature engineering step in cluster analysis, it is typically accompanied by a range of additional data cleaning and transformation tasks. An effective suite of tools for carrying out these operations is available within the `tidymodels` ecosystem in R. This collection of packages not only facilitates the easy implementation of various feature engineering techniques, but also enables the construction of pipe-able / reproducible workflows that can be applied consistently across multiple datasets.

More specifically, `tidymodels` is a unified framework in R that provides a coherent, modular, and reproducible approach to data preprocessing, model development, evaluation, and tuning. Its consistent syntax and design philosophy promote transparency and scalability in the modeling process.

Three functions for data pre-processing are particularly useful for this purpose. They are:

- `recipe`
- `prep`
- `bake`

These functions, along with descriptions, are illustrated in the diagram in Figure 8 below:

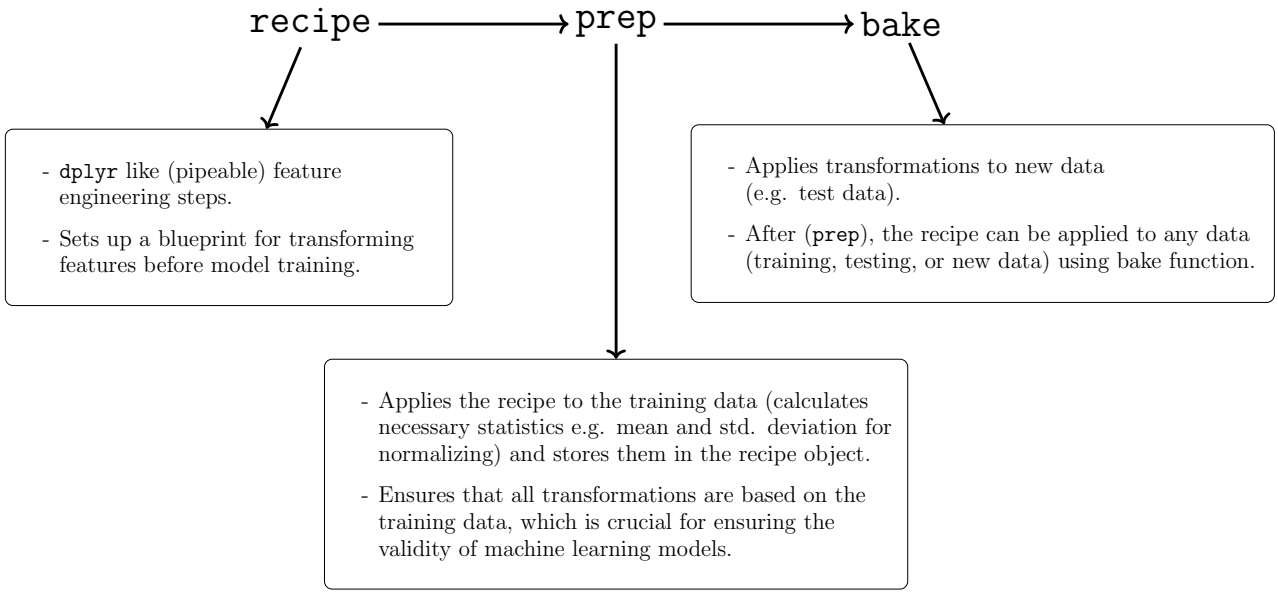


Figure 8: `tidymodels: recipe, prep, bake`

An example of a recipe is shown below. This recipe takes a dataset called `myDat` and performs the transformations noted in each step:

`recipe` Example

```
myData recipe <- recipe(id ~ ., data=myDat) %>%

# Impute missing values with KNN
  step_impute_knn(all_predictors(),neighbors=5) %>%

# Create interaction terms
  step_interact(terms =~ age:starts_with("behav")) %>%

# Group infrequent levels: "other"
  step_other(myCatVar, threshold = 0.001) %>%

# Approximates normalizing transformation
  step_orderNorm(all_numeric_predictors()) %>%

# PCA
  step_pca(all_numeric_predictors(), num_comp = 7)
```

Many other steps are available including

- `step_downsample`: balance a model response variable for predictive modeling,
- `step_spline_natural`: create splines for non-linear feature modeling,
- `step_zv`: identify and remove features with zero variance, etc.

as just a few additional examples.

Conclusions

For the data examined, this work demonstrated how auto-encoders outperformed both PCA and raw data in terms of producing high quality clusters evidenced by all three metrics used: DBI, CHI and Silhouette Scores. Specifically, AE-based data produced a significantly higher average silhouette score (0.32), exceeding the minimum acceptable threshold (0.2), unlike PCA and raw data partitions.

Visualization tools—including correlation and dependence heatmaps and feature permutation importance plots—provided interpretability of the AE components.

A number of clear advantages of auto-encoders were identified / noted:

- AE's can learn non-linear relationships and preserve more information than PCA.
- They enable more informative low-dimensional visualizations.
- AE's can also be used for anomaly detection and de-noising.

One caution regarding AE implementation involves model tuning (e.g., learning rate, architecture, activation functions), making it more complex than PCA. However, it seems the advantages far outweigh the drawbacks of using auto-encoders for dimensionality reduction.

In addition to using auto-encoders for dimensionality reduction, tools for implementing addition feature engineering steps were also reviewed. Specifically, the `tidymodels` framework in R is highlighted as a powerful and reproducible approach to feature engineering and model development. Key components of this ecosystem include functions: `recipe()`, `prep()`, and `bake()` which streamline transformation pipelines, supporting tasks like imputation, interaction term creation, normalization, and PCA to name but a few.

This work demonstrates that deep learning methods, specifically auto-encoders, offer a powerful enhancement to traditional dimensionality reduction techniques and can lead to significantly better segmentation and modeling outcomes in unsupervised learning applications.

References

Caliński, T. and Harabasz, J. (1974). A dendrite method for cluster analysis. *Communications in Statistics*, 3(1):1–27.

Chatterjee, S. (2021). A new coefficient of correlation. *Journal of the American Statistical Association*, 116(536):2009–2022.

Davies, D. L. and Bouldin, D. W. (1979). A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):224–227.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. In *Science*, volume 313, pages 504–507. American Association for the Advancement of Science.

Jolliffe, I. T. (2002). *Principal Component Analysis*. Springer Series in Statistics. Springer, 2nd edition.

Kaufman, L. and Rousseeuw, P. J. (1990). *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley Series in Probability and Statistics. John Wiley & Sons.

Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65.