Dataset Discussion:

The dataset I chose to use is a representation of AirBnB rental data from Austin, Texas. In projects 1 and 2 I looked into two main supervised learning problems. The first aimed to answer, “Can we forecast the price of a listing given certain attributes about that listing.” The second question aimed to answer, “Given certain attributes can we predict if a property will be vacant in 90 days.” For project 3, I have added in two different problems utilizing the same data to make them more interesting for a clustering task. I have binned the AirBnB rental prices into terciles. These terciles are labeled, “low” “medium”, and “high” and properties whose prices are [17.999, 100.0], (100.0, 198.0], (198.0, 5550.0]. Due to the dataset preprocessing and feature engineering that has taken place. This dataset has a relatively high cardinality with 38 features. Some of these are boolean and some are continuous. Data was mean centered such that the magnitude of each feature is equivalent. My first task is looking at the quality of these price bins leveraging the bins as the labels relative to the 38 original features. The interested reader may remember that one of the fields in the dataset was specific amenities that had correlation with price. This was a simple heuristic which was used to make use of the feature itself. For the second problem, I expand upon this idea by trying to create a word vector matrix over the amenities in order to find groupings of amenities that may provide some predictive power. I will be calling this problem 2. As with problem one, this data was again mean centered.

Clustering:

Given that the price of a rental should be related through some function as the combination of specific attributes related to that property we would expect to see some clusters of properties which elicit the heterogeneity of the variables that describe them. Similarly, one would expect properties that have fewer vacancies depending on the value that the property has. It should be explicitly stated that value, in this case, is not synonymous with price. For example, a lower priced property may have more value than a high priced property due to location, “bang for the buck”, unseen/unobserved differences, or random chance. We would also assume that there are specific amenities that lend themselves to higher priced rentals, pools, for example. All of the above should make for an interesting discussion around clustering techniques. The first step for my clustering analysis was to find the appropriate number of clusters for both EM and Kmeans. To answer this question I chose 6 different metrics that aim to explain the validity for a choice of k. The metrics I chose to use are as follows:

Name	**Equation**	**Definition**
Homogeneity Score	$h = 1 - \frac{H(Y_{true} \mid Y_{pred})}{H(Y_{true})}$	Measures and algorithms ability to assign all samples to a single class
Completeness Score	$c = 1 - \frac{H(Y_{pred} \mid Y_{true})}{H(Y_{pred})}$	Measures an algorithms ability to assign all samples with the same true label to the same cluster
V Measure	$V_\beta = (1+\beta)\frac{h \cdot c}{\beta \cdot h + c}$	Measures the harmonic mean between homogeneity and completeness.
Adjusted Mutual information	$AMI(U,V) = \frac{MI(U,V) - E\{MI(U,V)\}}{\max\{H(U),H(V)\} - E\{MI(U,V)\}}$	Measure of the similarity between two labels of the same data adjusted for chance.
Cluster Accuracy	$J(A,B) = \frac{\lvert A \cap B \rvert}{\lvert A \cup B \rvert} = \frac{\lvert A \cap B \rvert}{\lvert A \rvert + \lvert B \rvert - \lvert A \cap B \rvert}$	Measures the ability for an algorithm to predict an exact match between the assigned label and the actual label.
Silhouette Score	$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$	Measures how similar an object is to its own cluster compared to other clusters.

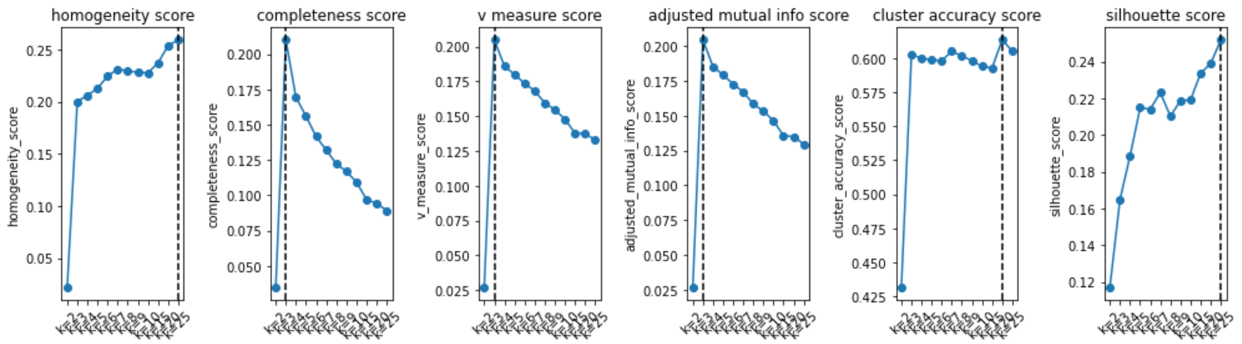


Figure 1. Results for different metrics k-means (problem 1)

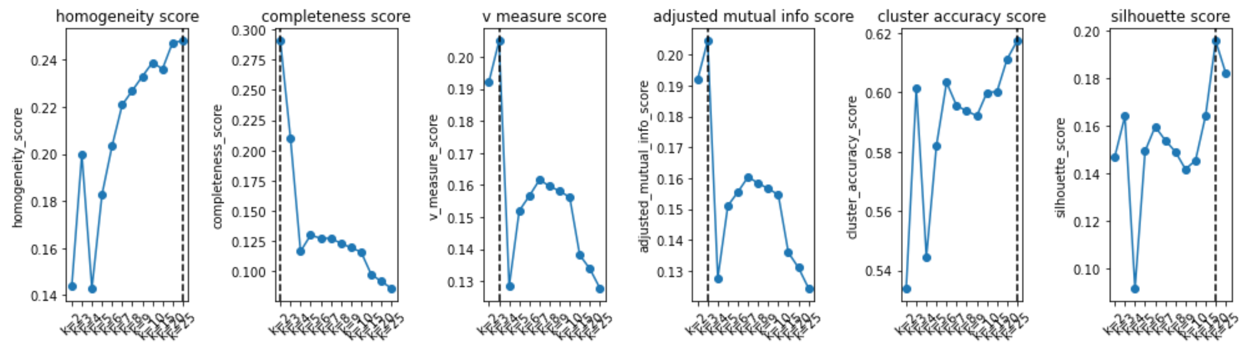


Figure 2. Results for different metrics EM (problem 1)

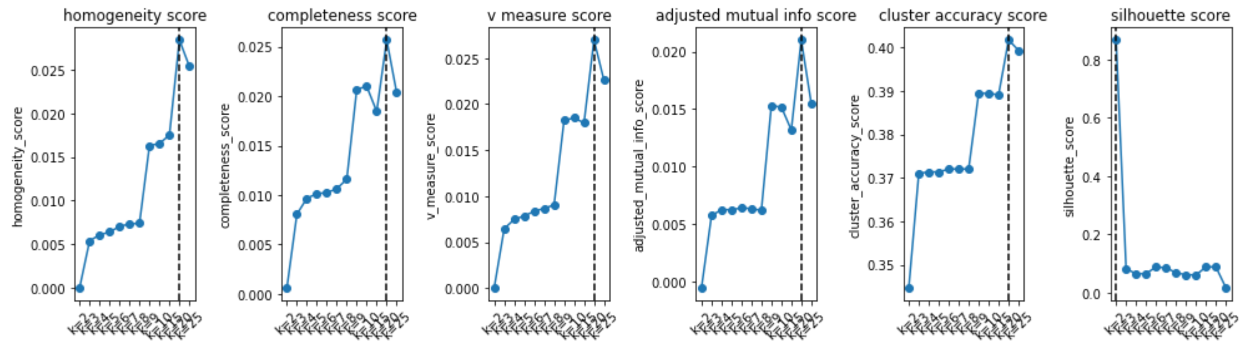


Figure 3. Results for different metrics k-means (problem 2)

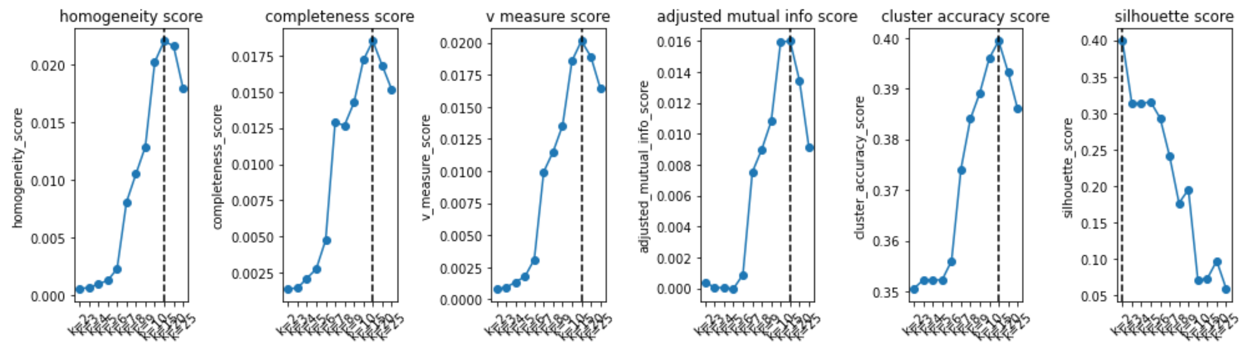


Figure 4.. Results for different metrics EM (problem 2)

All of our metrics aside from adjusted mutual information scores are dependent on the labels I have already assigned them. If this manual assignment captured enough variability in the properties we would expect our metrics have their highest values at k=3. As one can, three is never the value which has the highest point which suggests that three is the incorrect number of clusters. It also shows that there is more to cluster associations than just price.

For each of these metrics we want to find the maximum. However, as one can see, there are different results depending on both the metric and the algorithm (EM vs k-means). Given that we have different maximum values for different metrics it makes sense to look at the frequency of values. For problem one, k-means k=4 has peaks for v measure and for adjusted mutual information. Whereas EM has peaks at k=2 for adjusted mutual information, v measure, and completeness. For problem two, almost all of the values peak at 20 for kmeans where EM values peak at 10. We can also see how the dimensionality of the problem changes the number of clusters. For problem 2, there are ~300 different amenities that are used and the “optimal” clusters are >= 20 for both kmeans and EM. For problem 1, there are less features and most “optima” metrics sit below 10.

Once we have run both clustering algorithms there are several techniques we can use to inspect their quality. For problem one and two we can inspect both the cluster magnitude and cluster cardinality. For problem one, we can also do visual cluster inspection. For problem two, It would be difficult to analyze each cluster as there are >20 clusters and manually inspecting each is difficult because there are potentially >300 features per cluster, where the features themselves aren’t that interpretable (they are normalized counts). However, for problem one we can find the points that are the closest to the cluster centroid and visually inspect. Given the high cardinality of the data, I normalized the data inputs and looked at which features had the highest variance across features.

Closest Rentals to Centroid Kmeans (problem one)				
Cluster Number:	**0**	**1**	**2**	**3**
Price	116	490	116	122
Bedroom	0	3	1	2
Beds	1	6	2	2
Review Rating	99	99	96	98

K Means Labels Averages (problem one)				
Cluster Number:	**0**	**1**	**2**	**3**

Price	122	493	115	194
Bedroom	.5	3.65	1	2
Beds	1.44	5.2	1.3	2.6
Review Rating	97	96	95	96

The obvious main difference of these rentals is the number of bedrooms each listing has. Cluster 0 seems to be studios at a moderate price with one bed and one bedroom. Cluster 1 seems to be high priced rentals with a large number of bedrooms and beds. Cluster 2 is made up of 1 bedroom homes which seem to have a lower rating and cluster 3 is 2 bedrooms. As one can see, the closest rentals to the centroid have very similar attributes to the k-means label averages. This is a by-product of the distance metric (euclidean). As stated in the lectures, this is yet another representation of regression to the mean.

Given that with the expectation maximization algorithm we are looking at fitting distributions as centroids we cannot use euclidean distance for finding the centroids like in k-means. Instead, we can calculate the centroids of EM by looking at its covariance structure. We can measure the density of a data point with respect to this covariance matrix and we can find which points have the most density. These points will be our centroids. We can also use a distance metric known as Mahalanobis to reconstruct the points. It should be noted that this is what the GMM library within scikit-learn uses anyways.

Closest Rentals to Centroid		
Cluster Number:	**0**	**1**
Price	48	144
Bedroom	1	4
90 Day Availability	15	34
Is Apartment	1	0

K Means Labels Averages		
Cluster Number:	**0**	**1**
Price	122	370
Bedroom	1.46	4
90 Day Availability	29	34
Is Apartment	.36	.02

Utilizing EM, it looks like prices between clusters are hugely different. This makes sense given that we have a lot more variability to model with 2 instead of 4 clusters (EM vs k-means respectively) Cluster zero consists of mainly cheaper 1 bedroom rentals while cluster one consists of mainly 4 bedroom rentals that are pricier. Interestingly enough, there are zero one bedroom rentals that made it into the cluster one.

We can't do as thorough a visual analysis for problem 2 because the results aren't as interpretable or interesting because they are vector counts. That being said, we can still look at what features make up the most variability within the clusters.

Feature Variance by Algorithm (problem 2)	
Algorithm	**Important Features (most variance amongst clusters)**
K Means	Tv, dryer, shampoo, bathtub, oven, stove, private entrance, wifi, room-darkening shades, pack 'n play/travel crib, lockbox, patio or balcony
EM	Tv, dryer, wifi, heating, air conditioning, shampoo refrigerator coffee maker, oven, stove

While not as satisfying as problem 1, we still see that for problem 2 EM and K means both seem to have a large amount of similarities amongst their clusters. We see several of the same terms showing up amongst the clusters. This means that amenities themselves show up in pairs together. For example, for a rental we may see that if they have a TV they also have a dryer (which *could* define a cluster).

As stated, we can look at the magnitude and cardinality of each cluster. Distance in this case was the total distance to the centroid for each observation. This gives a value for how dispersed the clusters are. For kmeans, its euclidean distance and for EM it is Mahalanobis distance. These can be seen in Figure 5.

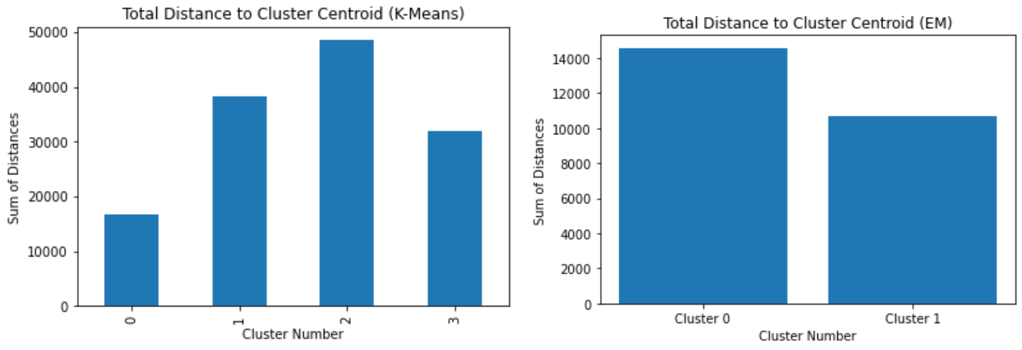


Figure 5. Total distance to cluster centroid per label for k-means and EM (Problem one)

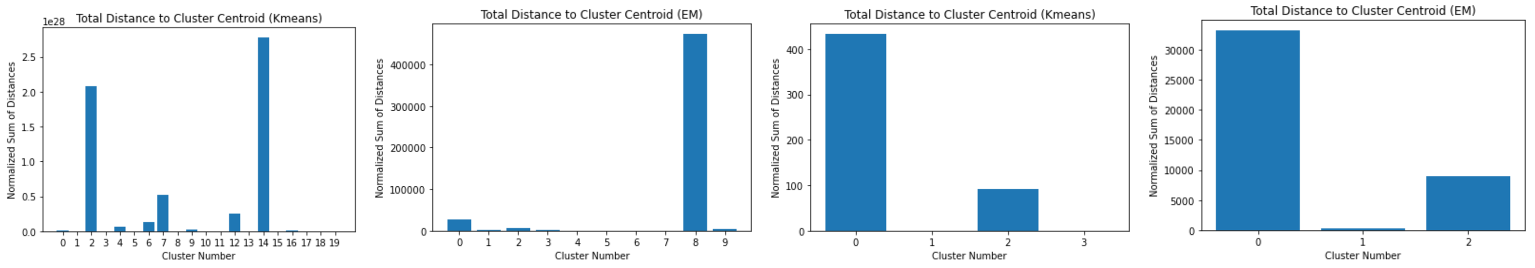


Figure 6. Magnitude for k-means and EM by number of clusters (Problem two) (high magnitude, low cardinality)

For these graphs we are looking at the total sum distance between points and their centroids divided by the number of points within a cluster (magnitude). This gives us a normalized view at the average distance between a centroid and its points. The higher the numbers the more spread out the cluster is. For kmeans we are looking at the euclidean distance between points and for EM we are looking at the number of standard deviations from the mean between a point and the specified distribution (centroid).

For problem one, looking at k means, we see that cluster two has more total distance between points compared to the others. We also see that cluster zero has less distance between the points so both of these clusters are dispersed and tight respectively. For EM, we see that cluster zero has more dispersion compared to cluster one. Intuitively (and by looking at the data) we know that there are more of certain types of rentals. For example, for cluster 0 for kmeans, we see that its mainly made up of cheap rentals with one bedroom. We know that these are less common than moderately priced two bedrooms (cluster 1). For problem two, we see that there is a lot more variability between clusters. This occurs due to the relative sparsity of the count vectors as well as the co occurrence of terms within the amenities. For kmeans, Cluster 14 and cluster 2 have a lot more distance between them and the centroids. Manually inspecting those clusters we find that cluster 2 actually lacks a lot of amenities that we would expect rentals to have: no tv, dyer, oven or coffee makers. Similarly, cluster 14 has even more sparsity with respect to amenities. This makes perfect sense as sparse vectors would be farther from other vectors in this euclidean space. Cluster 8 for EM is very similar to the other mentioned. Cluster 8 has no wifi, tv, or coffee maker showing the same sparsity as before. Note how we change the number of clusters; we don't necessarily change the cardinality of the clusters.

Dimensionality Reduction:

For dimensionality reduction we are asked to look at four different algorithms which are all very different. As such, I wanted to find a set of metrics which can measure the efficacy of each algorithm and be comparable across them. In service of this goal, I used two ways of validating all three algorithms. One of the ways was leveraging the Johnson-Lindenstrauss lemma which is a result concerning low-distortion embeddings of points from high-dimensional into low-dimensional Euclidean space. The lemma states that a small set of points in a high-dimensional space can be embedded into a space of much lower dimension in such a way that distances between the points are nearly preserved. The map used for the embedding is at least Lipschitz, and can even be taken to be an orthogonal projection. [1] This lemma is incredibly powerful for comparing these three algorithms because they are all mapping from some high dimensional space to a low dimensional space where the goal of this mapping is to retain as much information as possible. Using this concept, we can measure the pairwise euclidean distance for our original data and our projected data. Once we have this, we can divide them to get a ratio between the two. If we plot a histogram of this data we get distributions where a wide distribution with distorted ratios shows a poor ability to retain the information from the original data. Conversely, a tight distribution shows a strong ability to retain the data. As we increase the number of components we expect to see tighter distributions. The second measure I wanted to use is one we learned from the lectures called wrapping. For this, I utilized a simple logistic regression with cross-validation to measure the accuracy of the learning algorithm on the project data for a different number of components and compared the accuracy across from each component vs the accuracy considering all the data. Below we will see that the distributions get more and more centered as the number of components increases.

For both problem one and two, for PCA when the number of components is equivalent to the number of features we get a perfect reconstruction of the data. In the same vein, one can see how the mean continuously shifts closer and closer to one as the number of components increases. For random projection it acts in a similar way but due to the random nature of random projections it isn't a full reconstruction like PCA. Also, random projections start more skewed than PCA does and it takes longer to approach a true approximation of the original data. However, unlike PCA, RP never does fully reconstruct the data. For problem one, ICA starts with an incredibly close approximation of the data compared to both PCA and RP. This makes perfect sense given the underlying data.It is incredibly important to note that for both problem one and problem two we have very few normally distributed features. For problem one, there are a lot of engineered features that were computed in order to create some linearity within the data. These are generally in the form of dummy features. For problem two, we have count vectors which are most certainly not normally distributed. For a low number of components, PCA fails to capture the majority of the variance in the data because there is a lot of variance in features along a specific axis and PCA will find this and then look at a vector orthogonal to that for the next principal component. Since PCA only uses covariances and there is not enough information in these covariances it struggles (figure 9). From figure 9, we see that there is almost no covariance for problem two. I chose to look at factor analysis as well due to the fact that FA is very similar to PCA but it includes specific factors (noise)[6]. For problem one, FA will struggle on both datasets because it requires maximum likelihood estimates to estimate the loading matrix. For problem one, ICA performs much better with a lower number of components which shows that the data doesn't have a structure which is easily captured by one vector when the data should actually be separated into its independent components first. For problem one, this is intuitive due to the fact that the data can be thought of in two separate ways, the first is features which are going to be obviously correlated (price and number of bedrooms for example). These features are also going to be independent of another set of features in the data (number of bedrooms vs the distance to downtown). For problem two, there exists a similar interpretation because we expect groupings of specific amenities amongst certain rentals. For example, we would expect more (higher quality) amenities for larger (more expensive) rentals. Things such as pools, firepits, and hot tubs. Conversely, we expect cheaper rentals to have sparse vectors. As such, for problem one, ICA will have an easier time separating these distinct signals which in turn would lead to distinct linear combinations allowing us to reconstruct the original signals. Additionally, ICA thrives with non gaussian features [2] because it finds a linear decomposition which maximizes this nongaussianity conversely PCA thrives with gaussian variables [3]. Since the gaussian distribution's structure is dependent on covariances if there are little to no covariances PCA will struggle to find eigenvectors and eigenvalues that do a decent job of explaining the higher dimensional data. ICA is almost the inverse of this and it utilizes non-gaussian data (which has higher order moments and structure beyond covariance). For problem two, we see very similar results to problem one but with one exception. As the number of components increases up to 14, PCA eventually picks up a lot more of the variability in the data compared to the other two methods. We can picture our word count vectors as cooccurrences of amenities. We would expect there are a relatively lower number of unique combinations of amenities. So, the first PC isnt projected along an actual dimension which captures the variance, however, the 14th component manages to capture almost all of the variance.

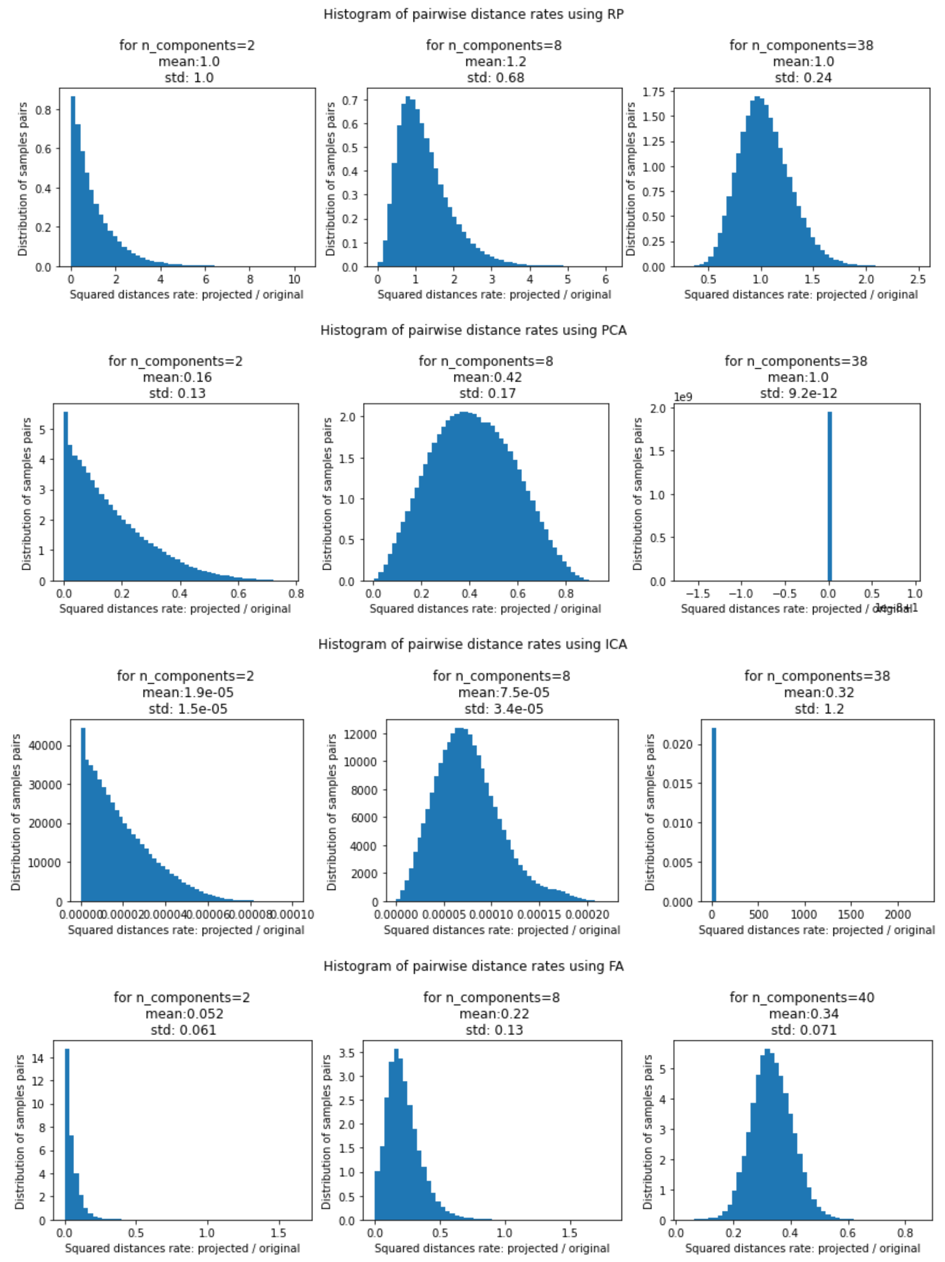
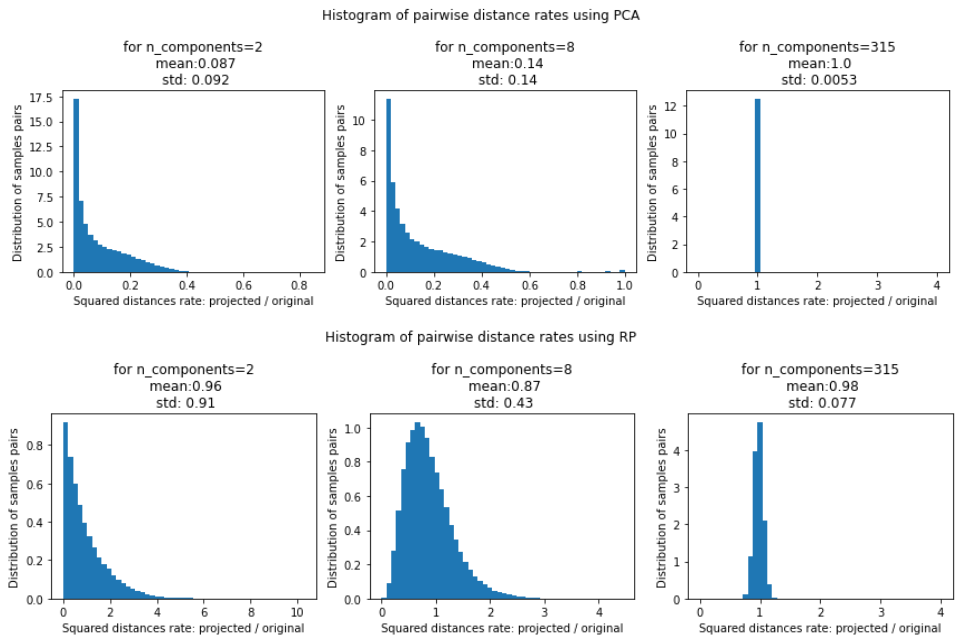


Figure 7. Problem one pairwise distance rates for PCA, ICA, and RP, FA



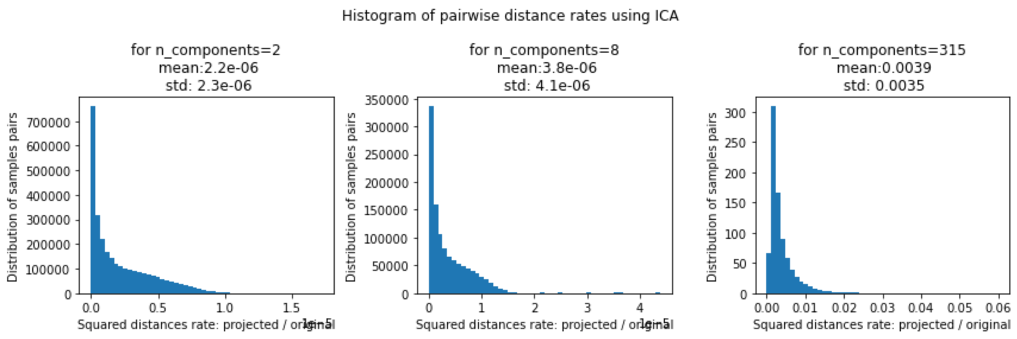


Figure 8. Problem two pairwise distance rates for PCA, ICA, and RP, FA

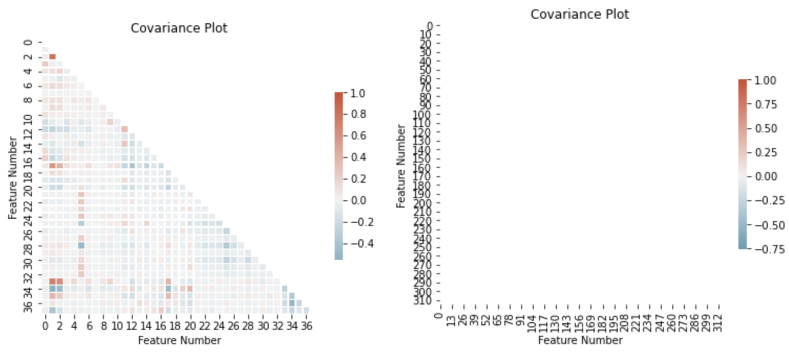


Figure 9. Covariance plots (problem one and two)

PCA also struggles because it functions over the covariance matrix. If there is little to no covariance (or correlation depending) it becomes difficult for PCA to absorb the variance of the data with minimal components. Random projects are typically useful when we don't want to go through the trouble of calculating the covariance structure. That being said, it utilizes the results from [4] which state that if our dimensionality is large enough we will end up with orthogonal vectors. We are now in the same place as before with PCA which depends on leveraging the orthogonality of the vectors to decompose the space effectively. For problem one, our data doesn't have a strong covariance structure to begin with (which is essentially what RP wants to capture) and therefore it does not work well. For problem two, random projections are able to take advantage of [4] and due to the high cardinality it produces consistent results for low number of features. Looking at Figure 8, we see the mean of the distance rates is .96 for random projections! This means that we see a large similarity between RP with 2 components and the original data. Also for problem two, ICA always has relatively skewed results for our pairwise distance plots. Looking at both problems we see that factor analysis performs poorly across most of the different values of components. Again, this isn't surprising as FA looks to leverage second order information about gaussian variables.

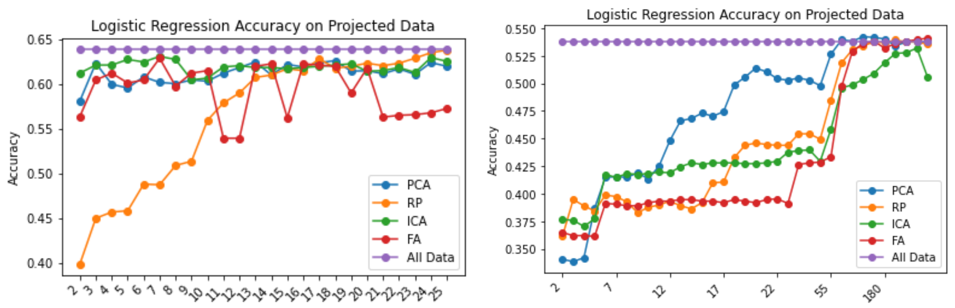


Figure 10. Wrapping accuracy for problem one (a) and problem two(b)

Above we can see the accuracy using the aforementioned wrapping method. The purple line represents the prediction we get while leveraging all of the data. ICA tends to have the highest accuracy with PCA following and RP taking more components to reach similar results. For problem two, we see volatile results. We see that PCA picks up a lot of variance by components 14. We see this again in figure 12 (b). As we see in Figure 10 (b) the accuracy is actually better using PCA or RP vs using the original data. This is the case because for high-dimensional problems PCA is functioning as a sort of regularization. From [5] The implicit assumption is that the response will tend to vary most in the directions of high variance of the inputs which explains the increased accuracy.

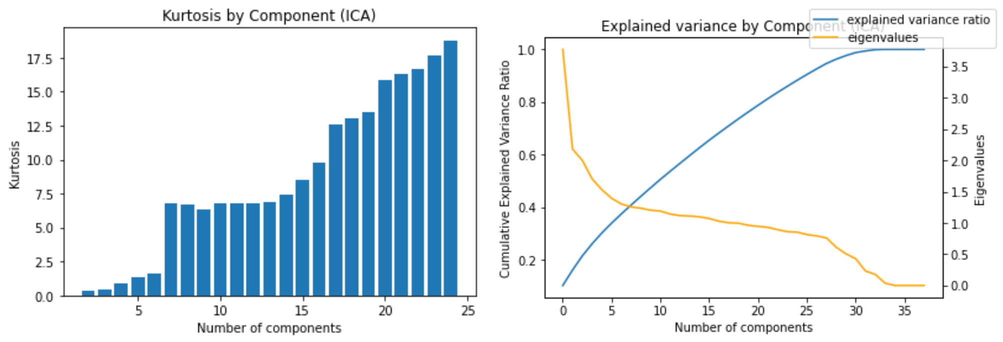


Figure 11. Kurtosis (a) and Explained variance (b) for problems one

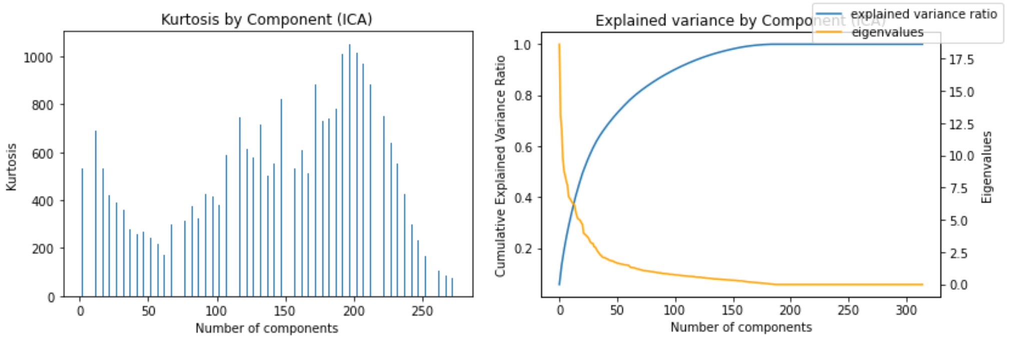


Figure 12. Kurtosis (a) and Explained variance (b) for problems two

Two things we can look at that are algorithm dependent are the kurtosis my component and the explained variance by component for ICA and PCA respectively. For problem one, we see a large spike in kurtosis around 8 components for ICA. For PCA we see that we don't capture a majority of the variance until 20+ components which is very steep considering only 38 to begin with. For problem two, we see the explained variance ends up peaking around 150 components, which suggests that we can reduce the space by about half that being said, our "elbow" is around 75 components. Looking at ICA, we see somewhat volatile measurements in kurtosis. The average value for kurtosis is much larger for problem 2 than problem 1 because of the underlying data (word count vectors vs rental features). We see a tremendous amount of kurtosis for lower components which suggests a very non-normal distribution in the data which allows ICA to perform well with limited amounts of components. Comparing Figure 11 and Figure 12, we see a huge change in how both of these algorithms perform. Figure 11 shows consistent increases in kurtosis while Figure 12 has much more erratic behavior. Figure 11 (b) shows a peak around 30 components which shows the poor performance of PCA while Figure 12 (b) shows a peak around 150 components which is great.

Clustering Reduced Data:

To select the number of components which have the largest "jump" in accuracy using our wrapping technique and which have the closest mean to 1 for the pairwise distances. For problem one, this number is 8 for all of our algorithms (RP, PCA, and ICA). We will set use the value of 150 for problem 2 because this is where the accuracy increases past the original data (Figure 10) and where we see peaks in Figure 12.

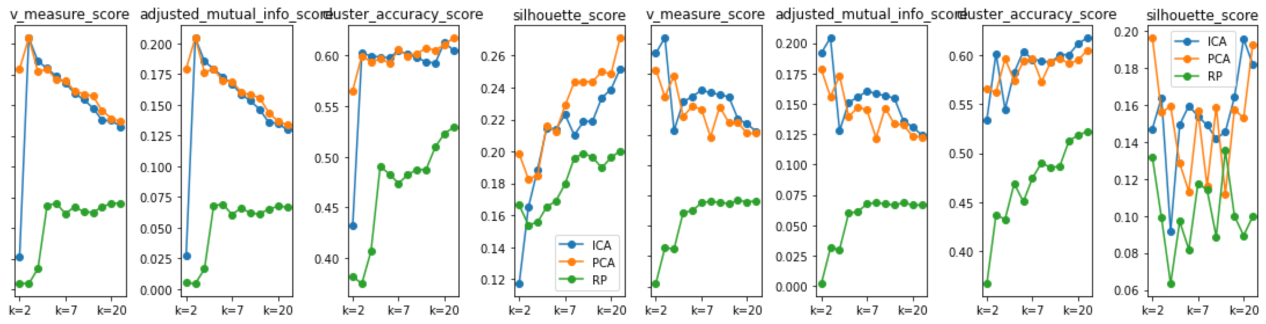


Figure 13. Different Metric Accuracy for Reduce Data: K Means(a) EM(b) (Problem One)

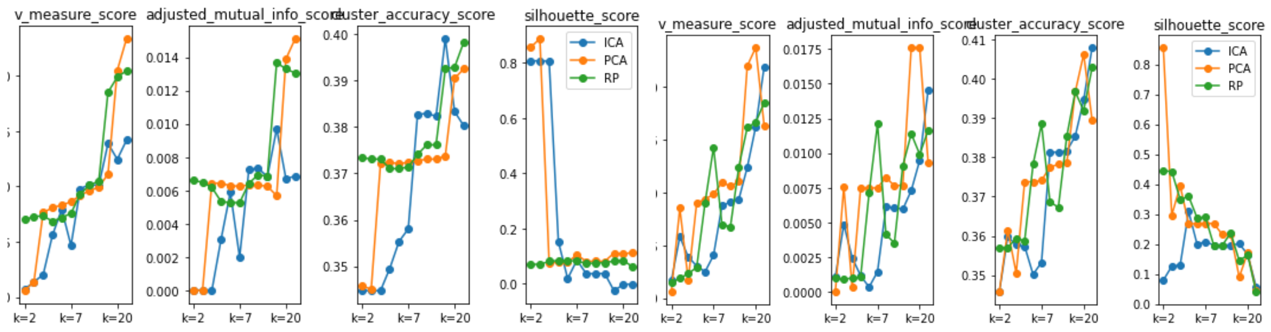


Figure 14. Different Metric Accuracy for Reduce Data: K means(a) EM(b) (Problem two)

In the above plots we see that again we have differing "optimal" values depending on which value we select for k. In figure 13 I removed homogeneity score and completeness because we look at both equally in the v measure. Again, we must look at the most frequent k within all of these plots which is now 3 instead of 4. All dimensionality reduction algorithms showing similar maximal values at 3 show that they all collect a similar amount of information when the number of dimensions kept was 8. It should also be noted (as it was above) that RP has lower scores compared to ICA and PCA. For problem two, we see that PCA tends to outperform RP and ICA regardless of the number of clusters. Both k means and EM want a higher number of clusters than before at K=20. Though, this result is surprisingly it isn't always the case that PCA will reduce the number of clusters. The first few PC's do not necessarily capture most of the cluster structure [7]. Another reason for this is fact that I didn't clean the text data. For example I could have just pulled out the nouns of the amenities so that TVs were all just TV instead of Samsung 50" TV.

For kmeans, PCA and ICA have very similar results across all metrics whereas. For EM, ICA is almost always better than PCA .This is talked about in the dimensionality reduction section but as a refresher we see how ICA can actually help capture the cluster structure. To show this we can visually inspect the cluster centroids with the decomposed ICA data with the number of dimensions set to 8.

GMM Labels Averages			
Cluster Number:	**0**	**1**	**2**
90 Day Availability	29.7	30.1	40.4
price	114.6	249.9	540.5
Greater than 3 bedrooms	0	0.00060	0.66
Is condominium	0.13	0.12	0.03

Kmeans Labels Averages			
Cluster Number:	**0**	**1**	**2**
90 Day Availability	29.40	30.16	40.40
price	115.49	243.53	710.25
beds	1.31	3.03	7.026
Is house	0.085	0.54	0.88

We see that the cluster structure is very apparent. Cluster 0 is composed of smaller cheaper rentals with less bedrooms that aren't houses. Cluster 1 is composed of slightly more expensive rentals which tend to have 3 bedrooms (or not greater than 3 bedrooms). Cluster 2 is composed of really expensive rentals. As stated above in the dimensionality reduction setting, we get features like the ones shown above which have an intuitive relationship to one another. We know from experience that price, beds, and whether a rental is a house are correlated. PCA loves this but what we also know is that some of the other features are going to be not as correlated and are in a different "direction" all together which is why ICA works so well. What is also important to note is that our initial concept of three buckets of prices is now matched by the number of clusters that both EM and Kmeans suggests that our original label concept is actually the correct one.

Feature Variance by Algorithm (problem 2)	
Algorithm	**Important Features (most variance amongst clusters)**
K Means	tv, dryer, wifi, air conditioning, heating, free parking, shampoo, refrigerator, oven, stove
EM	dryer, tv, air conditioning, heating, wifi, shampoo, free parking, refrigerator, oven, stove

The result of the above table is fascinating. We end up getting the most variance in our structure from the exact same amenities. Practically this means that even though we have > 300 different amenities our algorithms can find similar structures in this much reduced form (20 clusters from 150 components). Notice how in Figure 14 20 clusters was the peak for a lot of the metrics but it produces clusters that have high magnitude and low cardinality (Figure 15) (a). .Instead of going for the maximum value as in the original clustering section, I selected the elbow which produces clusters with more sain cardinality and magnitude. (Figure 15 (b)) . This ends up being 2 clusters for kmeans and 3 clusters for EM.

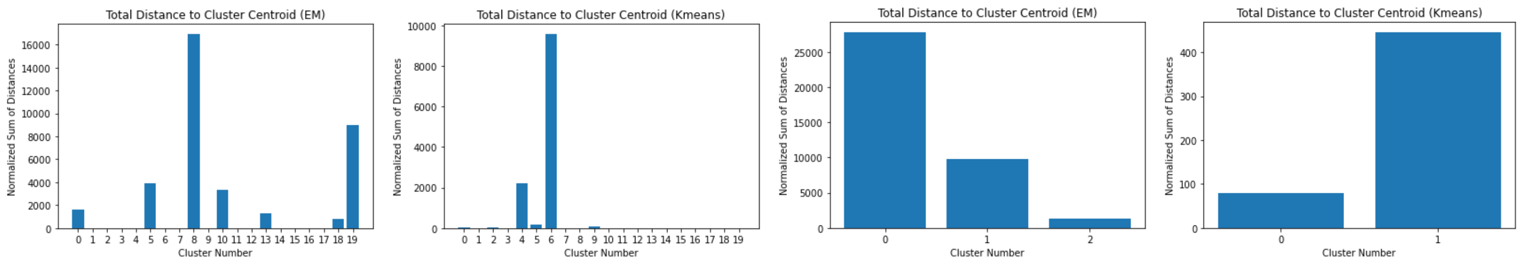


Figure 15. Different magnitude for varying number of clusters

Notice how compared with the not reduced data we end up getting clusters that have much better cardinality and magnitude than the previous clusters in the clustering section. I believe that this shows a very important thing about clustering. By not visually inspecting clusters and just going by summary statistics you can possibly ignore the fact that your "cluster" is really just an estranged point!

NN on Reduced Data:

We were only supposed to run the NN on one of the datasets. That being said, I decided to run another comparison. In problem one we have dummy variables for specifically selected amenities that I thought could provide value to the prediction task. I believe that by using the cluster results from problem 2 and adding those features to the dataset from problem one we may have an even better prediction. For my NN I kept the hyperparameters consistent from my other assignments. The learning rate is .01 and I'm using one hidden layer with a ReLu activation function. The results could be changed if I tried changing the activation function or changing the learning rate. However, I wanted my results to be comparable.

In figure 16 we can see how PCA and ICA allow us to get higher accuracy than without using any form of dimensionality reduction.As stated, this is the case for high-dimensional problems because PCA is functioning as a sort of regularization. From [5] The implicit assumption is that the response will tend to vary most in the directions of high variance of the inputs which explains the increased accuracy. This also illustrates that depending on the goal of the decomposition we may choose either ICA or PCA as our "best" metric. ICA ends up helping the NN more as long as we allow more components. However, if we care more about having lower dimensional data we may go with PCA instead of ICA because the convergence time for our NN. Overall, PCA is faster than ICA.

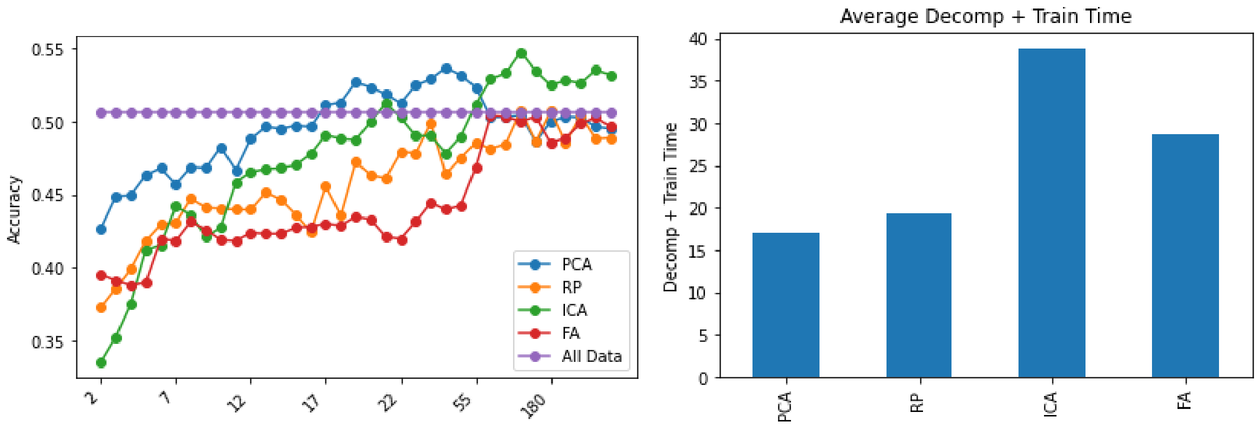


Figure 16. NN Accuracy for different number of components (a) Decomposition + Training Time(b)

Comparing figure 10 to figure 16 we see that the NN was able to perform better than the logistic regression leveraging the decomposed data. Another interesting point is that the NN was able to take advantage of a lower number of components compared to the logistic regression. This suggests that the NN was able to exploit the data better (something NNs are known for).

NN on Clustered Data:

For this experiment I took the whole dataset and fit both kmeans and EM on it for varying number of clusters. I think I took the cluster labels and turned them into dummy variables (cluster labels are categorical variables). I then split this data into both training and testing sets. I trained the same NN from above on the training set and tested the accuracy on the testing set.The labels seem to be somewhat informative for predicting a label class but the accuracy is not great (it never reaches above 41%). K Means as a method for dimensionality reduction seems to increase with values of K, which makes sense because as k increases we would eventually get k equal to the number of labels in the data which would be equivalent to the same dataset (each datapoint is its own cluster).

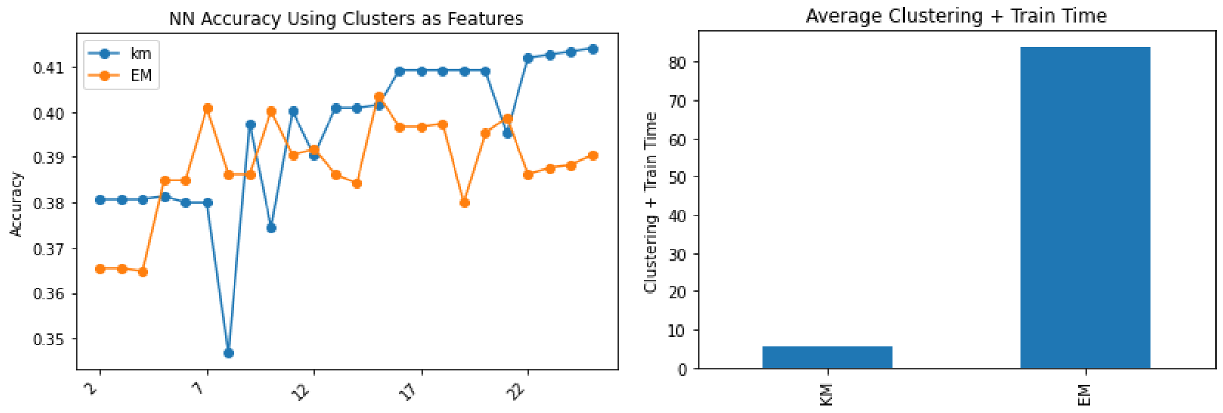


Figure 17. NN Accuracy using clustering labels as features (a) average clustering and training time (b) (problem 2)

As we know, the distance metric utilized in kmeans can have a large impact and for a next step I would try a different metric instead of euclidean. Perhaps manhattan distance or edit distance would give better results.

From here, we can combine all of our data such that we have decomposed features, clusters of our decomposed features, and our original variables (without the encoding amenities features) to predict the value bin of an airbnb rental.

	Amenities (SME)	PCA on amenities	PCA on amenities + Kmeans on PCA
Accuracy	95%	89%	88%

Looking at the above table one can see that my manual effort was able to find the most informative features. This is probably due to my own subject matter expertise (as any human knows, hot tubs at airbnbs are a must). That being said, with little SME leveraging the PCA as a feature is incredibly powerful. I still believe that the results of this would have changed if I did some more preprocessing on the amenities. For example, part of speech detection such that I only keep nouns would be a good second step.

Discussion:

There are several interesting things I learned by doing this project. First and foremost, an understanding of the data is key. Most of our dimensionality reduction algorithms have very important dependencies on the data. The main ones are around the data being either normally (or not normally) distributed. For clustering, its not always as easy as just looking at your metric plots. Sometimes it's very important to manually inspect the clusters by looking at their magnitude and their cardinality to make sure they are actually clusters and not just single points. On top of that, its important to look at the attributes of each cluster. Without doing so may lead to getting clusters that really aren't too dissimilar after all. I also think that utilizing wrapping is an awesome way to compare and contrast multiple different algorithms which may have different inner-workings but are trying to be used for the same goal. As seen in the last section, it isn't always easy for computers to beat human intuition. Starting with simple human encoded features may help in measuring the true benefit in computer generated ones.

Citations:

1. "Extensions of Lipschitz mappings into a Hilbert space". In Beals, Richard; Beck, Anatole; Bellow, Alexandra; et al. (eds.). Conference in modern analysis and probability (New Haven, Conn., 1982)
2. Hyvarinen, Aapo. "A Short Introduction to Independent Component Analysis with Some Recent Advances." https://wwwf.imperial.ac.uk/~nsjones/TalkSlides/HyvarinenSlides.pdf.
3. Shlens, Jon "A TUTORIAL ON PRINCIPAL COMPONENT ANALYSIS" 2003. PDF file.
4. Hecht-Nielsen, R. (1994). Context vectors: general purpose approximate meaning representations self-organized from raw data. Computational intelligence: Imitating life, 43–56.
5. Hastie, Trevor, Trevor Hastie, Robert Tibshirani, and J H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York: Springer, 2001. Print.
6. Rummel, R. J. (1970). Applied factor analysis. Evanston, IL: Northwestern University Press.
7. Yeung, Ka Yee, and Walter L. Ruzzo. An empirical study on principal component analysis for clustering gene expression data. Technical report, Department of Computer Science and Engineering, University of Washington, 2000