CS7641 Machine Learning Fall 2021
Project 3 : Unsupervised Learning and Dimensionality Reduction

Ruhan Li
(rli445)

Abstract

This project aims to apply unsupervised learning algorithms to identify clusters. Because one of the original data set that I chose for Project1 and Project2 contains only categorical features (-1, 0, 1), an additional data set has been selected for experiment in this project. K-means and Expectation Maximization (EM) were applied to all data sets. On top of that, three dimensionality reduction algorithms (Principal Component Analysis (PCA), Independent Component Analysis (ICA), and Random Projections (RP)), as well as a feature selection algorithm (Random Forest Classifier (RFC)) were applied to eliminate noise. The second part of the project is to rerun Neural Network (NN) with the four dimensionality reduction and feature selection algorithms, as well as adding clustering labels as new features. The performances are compared in terms of training time, accuracy, precision. The implement of algorithms and graphs are based on python and with public libraries, including sklearn, yellowbrick, seaborn, matplotlib, and pandas.
Link of code:https://github.com/Hanlarious/ML/tree/main/A3

Brief Introduction

Basic data set information

Originally I have two data sets: a customer satisfaction questionnaire from an airline and phishing websites detection. The phishing websites data set contains only classification features, which makes K-means and PCA algorithms meaningless, since they are using euclidean distance. Therefore, I selected an additional data set: school prediction, in which all features have continuous values.

Different data pre-process work for comparison

Despite the fact that I kept an untouched test set for each data set to evaluate the performance of their Neural Network part, I did different data pre-process work for comparison based on different characteristics of each data set:

- **Phishing websites:** this data set is slightly imbalanced, and has only categorical features with three values, so I did not scale or balance the data set.
- **Satisfaction questionnaire:** this data set is also slightly imbalanced, and has both categorical and numerical features, ranging from single digits to thousands, so I did three experiments with different pre-process work: 1. imbalance but scaled; 2. balanced and scaled; 3. balanced, categorical features dropped, and scaled. To ensure convincing results, *MinMaxScaler* and training and test sets splitting were always applied as last two steps.
- **School prediction:** this data set is super imbalanced with 6 classes (Figure 1). It contains only numerical features, including teacher's yearly salary, number of students, and number of non-classroom-based support, so the values vary largely from single digits to tens of thousands. I think it would be interesting to see how an imbalanced data set performs, therefore, I kept the data set imbalanced, and did the scaling to avoid over-weighting large numbers.

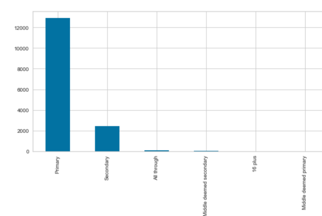


Figure 1: School:imbalanced data

Due to space limitations, in this report, I will only show interesting or meaningful findings from the comparisons.

General Implement

For all experiments, I followed the following progress to decide on the number of clusters and number of components:

1. **Number of clusters:** For K-means clustering, I plotted the elbow curve based on the sum of squared error(SSE). Ideally the inflection point would be the best clusters, but in practice, I found that sometimes the inflection point did not return the best metrics scores. Among all metrics, Silhouette Coefficient is the one that does not involve the ground truth labels, therefore, I also plotted the Silhouette Coefficient curve to help selecting the number of clusters. For EM, because it is a soft clustering algorithm, I plotted the per-sample average log-likelihood to help finding the best number for clustering. However, my log-likelihood curves have a lot of ups and downs, which makes it hard to make a decision. Therefore, I refer to Bayesian Information Criterion (BIC) and Silhouette Coefficient as well.

2. **Number of components:** For PCA, I plotted the cumulative explained variance. I feel like 90% to 95% could represent the original information well, so I set the cutoff to 90% or 95% to get an initial number of projections. But there might still be some trivial components with small amount of variance / low eigenvalue. Therefore, I set a threshold to filter out these components.
   For ICA, I plotted the Kurtosis score curve. When Kurtosis equals to 3, it suggests a normal distribution (or Gaussian distribution), therefore, I set the cutoff to the number of components where Kurtosis equals to 3. Then I further select the components with Kurtosis greater than 3. If I did not explain it clear here, I will make further explanation with the curve later.
   For RP, I set two sets of random seeds with 5 seeds and 10 seeds respectively, and ran the algorithm multiple times using the random seeds. Then I plotted the Kurtosis curve, and take the peak as the number of components. I also plotted the reconstruction error, and I would adjust my choice based on the reconstruction error if necessary.
   For RFC, I made histograms to show the top rated features with their corresponding importance. Then I set a threshold of importance to filter out trivial features.

3. **Metrics:** For K-means, I'm using a bunch of metrics to analyse the clustering, including homogeneity score, adjusted mutual info score, adjusted rand index score, completeness score, v measure score, and fowlkes mallows score. I care more about the adjusted rand index score, which measures the similarity, and the adjusted mutual info score (AMI), which measures the agreement. The homogeneity shows "each cluster contains only members of a single class", whereas the completeness shows "all members of a given class are assigned to the same cluster" [2], which are also helpful when comparing the cluster labels with the ground truth labels.
   For EM, on top of the above mentioned metrics, I feel like BIC is an useful reference too. Besides, fowlkes mallows score shows the geometric mean of pairwise precision and recall, which may convey good information when comparing the labels. Therefore, I finally decide to plot the silhouette coefficient, BIC, homogeneity score, fowlkes mallows score, and adjusted mutual info score.
   Also, in *GaussianMixture*, there are four types of covariance: full, tied, diag, and spherical. Gaussians with full covariance can independently adopt any position and shape; gaussians with a tied covariance means "the same covariance matrix is shared by all the gaussians"; gaussians with diagonal covariance matrices indicates that the axis is oriented along the coordinate axis, otherwise the eccentricity between the components may be different; with spherical gaussians, "the variance is the same along all axes and zero across-axes"[1]. I plotted the selected metrics with all four types of covariance.

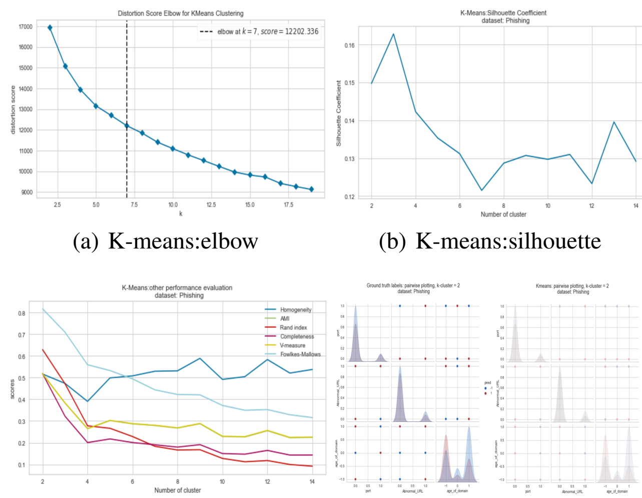
4. **Neural Network evaluation:** For the NN part of this project, I applied the four dimensionality reduction and feature selection algorithms to the customer satisfaction questionnaire data set exactly the same way as I did in the first part, and reran my NN using the same set of tuned hyper parameters as in project1.
   With the same setting, I later added the K-means and EM clustering labels to the data set as new features, and reran NN again.
   I think both accuracy and precision are important metrics to this data set, since the precision focuses on the False Positive, which can bring more harmful impact. Therefore, the final performances are evaluated based on accuracy and precision.

5. **Other implements that worth mention:** When doing comparison, I would like to minimize variables, therefore, I set the random state to 7 throughout the entire experiment, except the RP algorithm, where I intentionally gave two sets of random seeds.

Unsupervised Learning

Phishing websites

- **K-means**



(a) K-means:elbow (b) K-means:silhouette

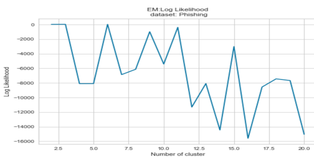
(c) K-means:metrics (d) K-means:pairwise plot

Figure 2: K-means:phishing

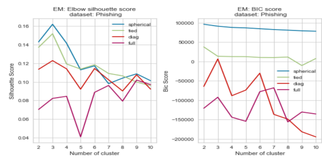
The elbow curve suggested the cluster number of 7 (Figure 2(a)), but as is shown in the Silhouette Coefficient curve (Figure 2(b)), 7 clusters has the lowest Silhouette Coefficient. Since the elbow isn't obvious, the number of clusters that suggested by Silhouette Coefficient makes more sense to me. Because originally this data set has two categories: phishing website and not phishing website. However, it is possible that there's a third cluster: not sure, need further analyse.
It is surprising that the V-measure and AMI are completely coincident in the metrics graph (Figure 2(c)).
To make the pairwise plotting, I picked three features that I believed would have the greatest impact based on the domain knowledge, including 'port', 'Abnormal_URL', and 'age_of_domain'. The pairwise plot shows the comparison of ground truth labels and the clustering labels (Figure 2(d)). The density plots show very similar distributions. I

think this is because the data set contains only categorical features.

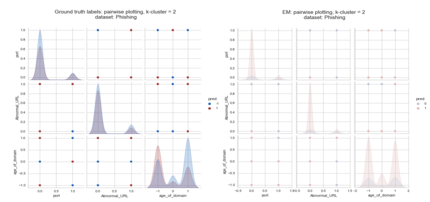
- **EM**



(a) EM:log likelihood



(b) EM:metrics



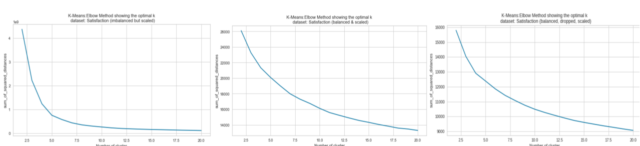
(c) EM:pairwise plot

Figure 3: EM:phishing

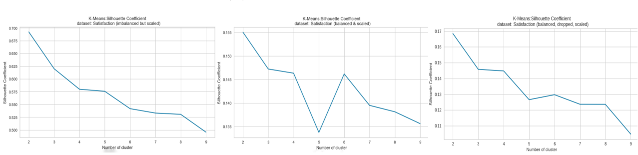
The log likelihood (Figure 3(a)) goes up and down dramatically. I think this is mainly because my data set is not a normal distribution. But it still shows the peaks at two, three, and six. In addition, the silhouette coefficient also shows peaks at 3 (Figure 3(b)). Therefore, for this data set, I feel like three is the best choice of clusters.
Different from the K-means algorithm, the pairwise plotting (Figure 3(c)) shows that with EM, the number of data points that are label "not phishing website" is significantly decreased.

Satisfaction questionnaire

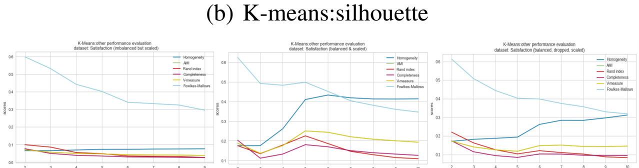
- **K-means** It is interesting to compare the different performance of the same data set with different pre-processing.



(a) K-means:elbow



(b) K-means:silhouette

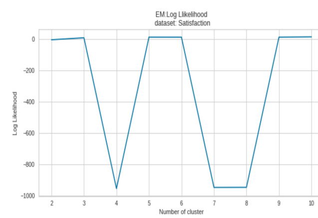


(c) K-means:metrics

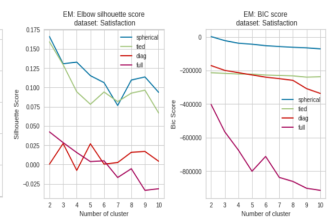
Figure 4: K-means:satisfaction

The following graphs show the curves of imbalanced but scaled (A), balanced and scaled (B), and balanced, categorical features dropped and scaled (C) data sets respectively (Figure 4). The results are not the same as expected. A has the most obvious elbow whereas B and C almost do not have an elbow (Figure 4(a)). B and C has similar Silhouette Coefficient except for 5 and 6 clusters (Figure 4(b)). A has much worse performance with metrics except the fowlkes mallows score (Figure 4(c)), which suggest that scaling and balancing has little effect on the pairwise precision and recall. Further, I expected dropping categorical features to bring about big changes, but to my surprise, it did not. But theoretically, keeping categorical features brings negative impact to the experiments. Therefore, in the rest of the report, I will only show the results of C data set if nothing is interesting about the other two data sets.
Looking at the curves of C data set, I feel like the elbow curve shows a slight inflection point at 3 or 4, which is supported by the Silhouette Coefficient curve.

- **EM**



(a) EM:log likelihood

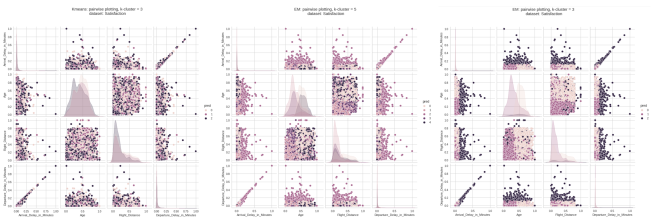


(b) EM:metrics

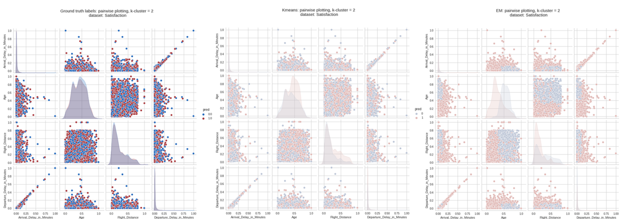
Figure 5: EM:satisfaction

For this data set, I don't have a great log likelihood curve, and my other metrics seem to be inconclusive (Figure 5). However, if I only focus on the full covariance, I will say the BIC curve suggest 5 or 7 clusters, because the smaller the BIC value, the better the performance. Combined with the log likelihood curve, 5 is one of the six peaks that it suggests. Therefore, I will probably agree to have 5 clusters. Practically, I think this number makes sense as well.
Originally, this data set has binary classifications with label "satisfied" and "neutral or dissatisfied" only. However, for the sake of the company, I think the labels should be further refined into customer satisfaction degrees, so that the company can have different marketing strategies for different groups.

- **Comparison**
Another noticeable point is the pairwise plotting. I have chosen four features to make the graph based on domain knowledge, including the arrival and departure delay information, which I believe affects customer satisfaction directly.
As I analysed before, the best number of clusters for K-means is 3, and for EM is 5. So I did the pairwise plotting to see their performance. With the best numbers that I found, neither K-means nor EM separates the clusters



(a) K-means:3 (b) EM:5 (c) EM:3



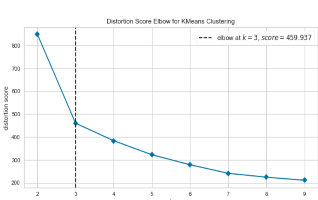
(d) ground truth label vs clustering label

Figure 6: Satisfaction:pairwise plotting

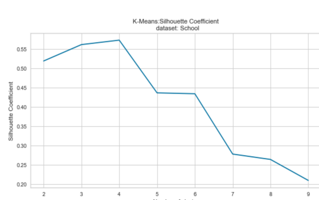
clearly (Figure 6(a & 6b)), but when I set the number of clusters to 3, EM did a better job (Figure 6(c)).
It is very obvious that the K-means algorithm could not separate the clusters clearly (Figure 6(d)), but the density plots show its similar distributions with the ground truth labels. On the contrary, even though EM algorithm did not return me good metrics, it separates the data points way better than the K-means algorithm. However, the EM clustering labels are less corresponding to the ground truth labels compared with the K-means algorithm. Another information that I could extract from the plotting is that feature “Departure_Delay_in_Minutes” and feature “Arrival_Delay_in_Minutes” are positively correlated. This is also in line with our general perception.

School prediction

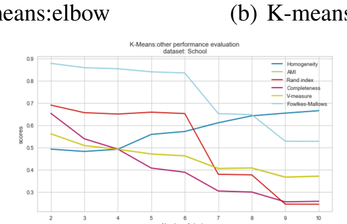
- **K-means**



(a) K-means:elbow



(b) K-means:silhouette

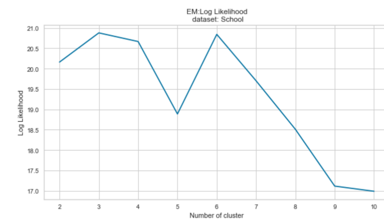


(c) K-means:metrics

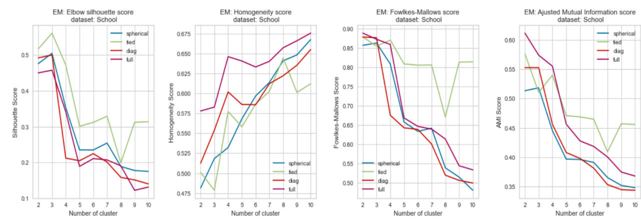
Figure 7: K-means:school

This data set shows clear elbow at 3 (Figure 7(a)), and highest Silhouette Coefficient at 4 (Figure 7(b)), and the other metrics somehow support it (Figure 7(c)).

- **EM**



(a) EM:log likelihood



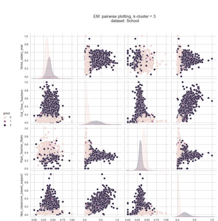
(b) EM:metrics

Figure 8: EM:school

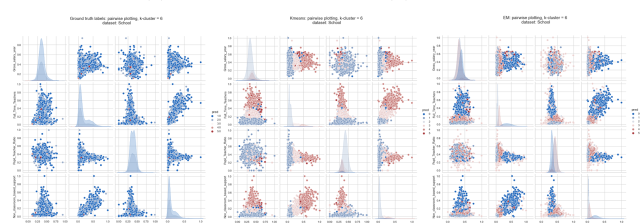
The log likelihood peaks when the number of clusters is 3 and 6 (Figure 8(a)), and the Silhouette Coefficient points to the number 3, whereas the Homogeneity points to the number 4. The rest two metrics show that the clustering perform best when the number of clusters is either 3 or 4 (Figure 8(b)). Given these clear information, I think either 3 or 4 could be a good choice of the number of clusters, and I believe that the results will not differ too much.

- **Comparison**

(a) K-means:3



(b) EM:3



(c) ground truth label vs clustering label

Figure 9: School:pairwise plotting

Based on the domain knowledge, I think the features “Gross_salary_year”,“Full_Time_Teachers”,“Pupil_Teacher_Ratio”,

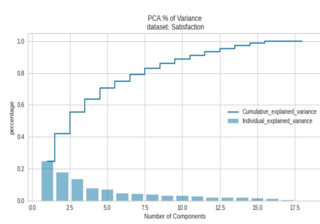
and “Non_classroom_based_support” have the greatest impact on the school phase prediction. Therefore, I made the pairwise plotting with these selected features, and I can extract some useful information from it.
I decide to set the number of clusters to 3 to both K-means and EM, so that I can compare their performance. Both algorithms have only two obvious clusters, and they have very similar boundaries for the clusters. This makes me feel that this data set may have obvious clusters.
As I mentioned before, this data set is very imbalanced (Figure 1), so the majority of ground truth labels are the first category: Primary. However, with K-means algorithm, it is easy to conclude that most of the time the data set is divided into three clusters pretty evenly (Figure 9(c)). Although I requested that it be divided into 6 clusters, there are three clusters that are basically invisible. Consequently, the distribution of the K-means clustering labels is very different from the ground truth labels. With the EM algorithm, the number of data points in one of the clusters is far more than the number of data points in other clusters, and the division boundaries of different clusters are not very clear. The density plots suggest that EM clustering label distribution is also very different from the ground truth label distribution.

Dimensionality Reduction

Due to the page limitation and the quality of the data set, I decided not to decompose the Dimensionality Reduction of the phishing websites data set in detail.

Satisfaction questionnaire

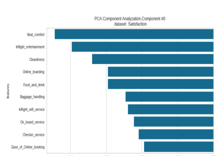
- **PCA** As is shown in the cumulative explained variance



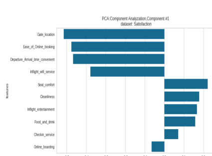
(a) PCA:variance

```
---- RUNNING PCA -------
Original PCA components
     pca_stats_df  e-variance-ratio      eigen
1        0.295041          0.245390  68.513502
2        0.212402          0.176658  58.131905
3        0.162725          0.135341  50.881842
4        0.093180          0.077499  38.503208
5        0.084707          0.070452  36.710802
6        0.053230          0.044272  29.101314
7        0.050138          0.041701  28.243613
8        0.045182          0.037579  26.811333
9        0.036328          0.030214  24.041137
10       0.034228          0.028468  23.336122
11       0.029919          0.024884  21.817716
Threshold PCA components variance >= [0.10%]
    pca_stats_df  e-variance-ratio      eigen
1       0.295041          0.245390  68.513502
2       0.212402          0.176658  58.131905
3       0.162725          0.135341  50.881842
```

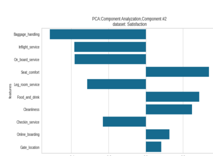
(b) PCA:threshold



(c) PCA:component 0



(d) PCA:component 1

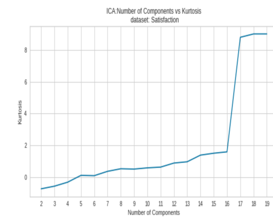


(e) PCA:component 2

Figure 10: PCA:satisfaction

graph, the variance could reach 0.9 with 11 components (Figure 10(a)), therefore, I decide to keep these 11 components as my original components. But we can tell that some components have very small variance and eigenvalue (Figure 10 (b)), therefore, I set a threshold to variance greater than or equal to 10% to filter out the trivial components. I have three components left in the end, which are composed of very different features with different wights (Figure 10(c & d & e)).

- **ICA**

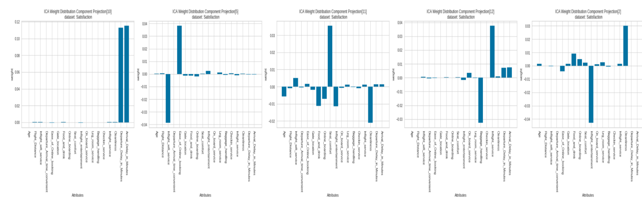


(a) ICA:kurtosis

```
Top-10 Max Kurtosis
10    124.495214
5       5.177310
11      4.233901
12      3.936066
2       3.689668
3       2.778807
6       2.324534
15      2.302554
1       2.279585
14      1.468512
```

```
Top-5 Max Kurtosis --
10    124.495214
5       5.177310
11      4.233901
12      3.936066
2       3.689668
dtype: float64
Int64Index([10, 5, 11, 12, 2]
X.shape = (15911, 18)
```

(b) ICA:components

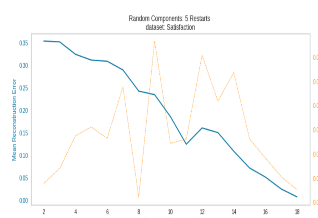


(c) ICA:top 5 component

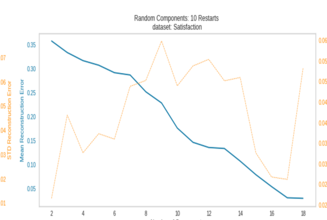
Figure 11: ICA:satisfaction

The kurtosis curve has a rapid increase at 17 (Figure 11(a)). Since the values before 17 are less than 2, and the trend after 17 tend to be flat, I decided to use 17 as my initial number of projections. The same as PCA, with so many projections, some of them might have very low kurtosis values. My understanding is that a good component need to have the kurtosis value greater than 3, therefore, I set the threshold to 3, and kept the top 5 components (Figure 11(b))
Like PCA, each component of ICA also contains different features. But the difference is that the features in PCA components share relatively even weights (Figure 10(c & d & e)), whereas the ICA components basically have two features with particularly large proportions (Figure 11(c)).

- **RP** I tried using 5 random seeds and 10 random seeds re-



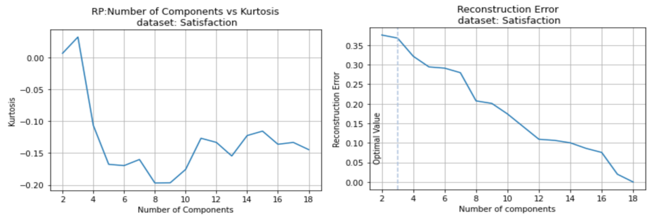
(a) 5 seeds



(b) 10 seeds

Figure 12: reconstruction error with 5 or 10 cycles

spectively, and found that with more cycles, although the reconstruction error curve becomes flattened, their overall trends are very similar (Figure 12).
To maximize the performance, I decide to use the same set of 10 random seeds for all RP experiments in this project.



(a) RP:kurtosis (b) RP:reconstruction error

Figure 13: RP:satisfaction

The kurtosis value clearly peaks at 3 (Figure 13(a)), where the reconstruction error is slightly above 0.35 (Figure 13(b)). I think this is too high. Ideally, I would want to keep the reconstruction error around 0.1. Therefore, I decided to adjust the number of components to 13.

- **RFC** Because RFC does not generate new components,

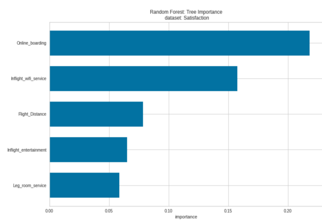
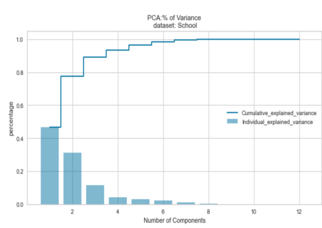


Figure 14: RFC:feature importance

instead, it sorts features by the importance, I simply set a threshold to keep only the features with great importance. In this case, I set the threshold to importance greater than 0.1, and kept the first two features (Figure 14).

School prediction

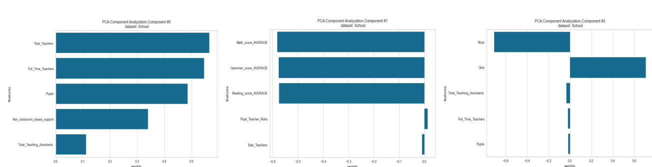
- **PCA** I followed the same procedures in determining the



(a) PCA:variance

```
---- RUNNING PCA -------
Original PCA components
   pca_stats_df  e-variance-ratio      eigen
1      0.089568          0.465541  24.881646
2      0.059796          0.310797  20.330063
3      0.021811          0.113363  12.278207
4      0.008332          0.043308   7.588991
Threshold PCA components variance >= [0.02%]
   pca_stats_df  e-variance-ratio      eigen
1      0.089568          0.465541  24.881646
2      0.059796          0.310797  20.330063
3      0.021811          0.113363  12.278207
```

(b) PCA:threshold

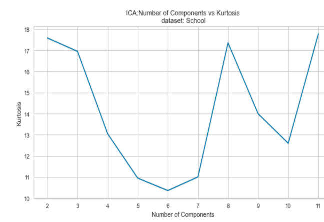


(c) PCA:component 0 (d) PCA:component 1 (e) PCA:component 2

Figure 15: PCA:school

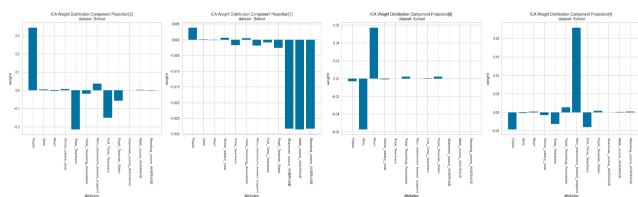
number of components as I did for the customer satisfaction questionnaire data set. So I will go directly to the result and conclusion in the analysis of this data set.
I think the distribution of the components in this data set is different. The variance is concentrated on the first few components (Figure 15(a)). I had to set a very small threshold, otherwise I would have filtered out all components (Figure 15(b)). In the end, I kept three components (Figure 15(a & b & c)). This time, as the importance decreases, the number of features that account for a large proportion in the component is also decreasing.

- **ICA**



```
Top-4 Max Kurtosis
2    38.776912
3    33.834487
6    15.669273
4    13.869148
```

(a) ICA:kurtosis (b) ICA:components

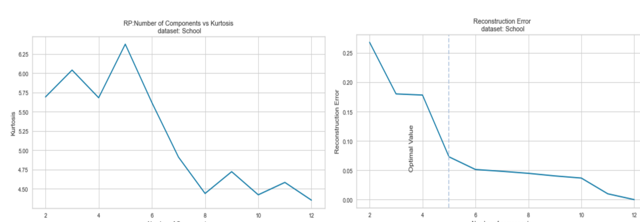


(c) ICA:top 4 component

Figure 16: ICA:satisfaction

The kurtosis curve shows a peak at 2 and 8 (Figure 16(a)). I tried to see their components with the largest kurtosis respectively. It turned out that with initially 2 projections, there's only one component that had a kurtosis greater than 3. Therefore, I set the initial projections number to 8, and filtered out the ones with small kurtosis (Figure 16(b)).
The distribution of the components here (Figure 16(c)) looks similar to the distribution of the customer satisfaction questionnaire data set (Figure 11(c)). I think this might be a characteristic of ICA algorithm: a small number of features dominate the components.

- **RP** The kurtosis peaks at 5 (Figure 17(a)), where the re-



(a) RP:kurtosis (b) RP:reconstruction error

Figure 17: RP:school

construction error is at an inflection point of a flattening curve (Figure 17(b)). And the reconstruction error is only around 0.075. I was very satisfied with this result, and decided to keep 5 components.

- **RFC**

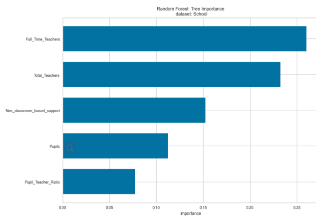


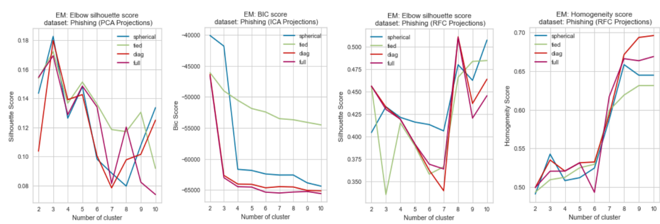
Figure 18: RFC:feature importance

With a threshold of 0.1, I was able to keep the top 4 most weighted features (Figure 18).

Reproduce Clustering Experiments

Phishing websites

- **EM** For this data set, I did not have any interesting finding with K-means clustering, so I will go directly to the EM algorithm.



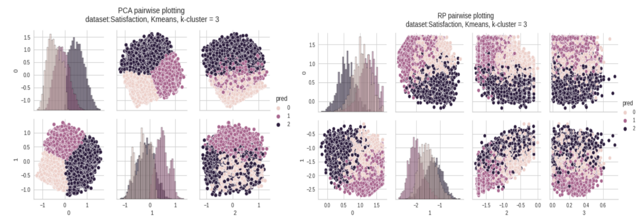
(a) PCA:sil- (b) ICA:BIC (c) RFC:sil- (d) RFC:homo-

Figure 19: EM with dimensionality reduction

If we look at the PCA and ICA algorithms, the clustering performance might be the best with 3 clusters. Especially PCA silhouette coefficient, all curves peak at the highest point simultaneously (Figure 19(a)). And the inflection point at 3 in the BIC curve of ICA also supports this idea (Figure 19(b)).
With RFC, the homogeneity ((Figure 19(c)) and silhouette coefficient (Figure 19(d)) also shows peaks at 8 clusters.
Unfortunately, I'm not able to plot the pairwise plotting for these new clustering. Since this data set has only categorical features, most of the pairwise plotting are likely to have only several data points (just like in Figure 3(c)).

Satisfaction questionnaire

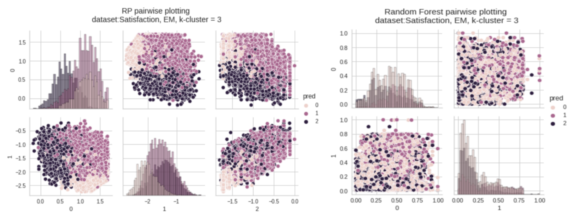
- **K-means** I was actually amazed by the PAC algorithm. I feel like with the top 2 components, the algorithm can separate the data set very well (Figure 20(a)). The RP algorithm also improves the clustering performance if you compare it (Figure 20(b)) with the original K-means clustering (Figure 6(a)).



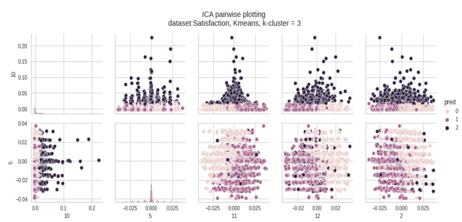
(a) PCA (b) RP (selected plots)

Figure 20: K-means with dimensionality reduction

- **EM**



(a) RP(selected plots) (b) RFC(selected plots)



(c) ICA

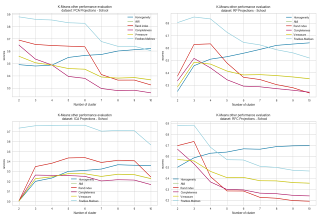
Figure 21: EM with dimensionality reduction

I think in this data set, generally speaking, the performance of EM is not as good as the performance of K-means (Figure 21). However, it is understandable that none of the dimensionality reduction algorithms has make clear boundaries of clusters since EM is a soft clustering algorithm.
Looking at the RFC (Figure 21(b)), which uses only original features, I feel like dimensionality reduction did improve the performance.
Another noticeable phenomenon is that the number of data points of ICA is always significantly smaller than that of other algorithms. I think the reason is that ICA components are likely to be dominated by fewer features compared with other algorithms, and with threshold, the majority of components has been filtered out. Therefore, fewer data points are left. I could be wrong, since I did not find any research that could explain it.

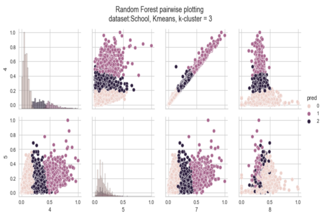
School prediction

- **K-means**

I think I can draw a conclusion from the graph of metrics for the four dimensionality reduction (Figure 22(a)) that



(a) metrics(PCA ICA RP RFC)



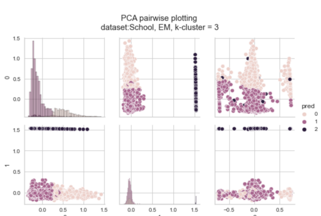
(b) RFC

(c) ICA

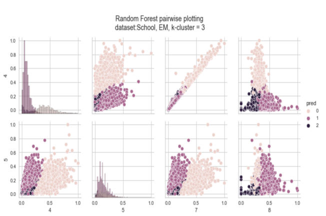
Figure 22: K-means with dimensionality reduction

K-means clustering is likely to perform best with 3 clusters. Because most of the metrics reached their peaks with 3 clusters, regardless of the types of covariance.
When looking at the scatter plots of RFC and ICA, I feel like they can separate the clusters quite well. Especially when we compare it with the K-means clustering without any dimensionality reduction (Figure 9(a)). In the original K-means experiment, the algorithm separated the data set into two parts, even thought I required for three clusters. With dimensionality reduction or feature selection, the performance is significantly improved.

- **EM**



(a) PCA)



(b) RFC

(c) ICA

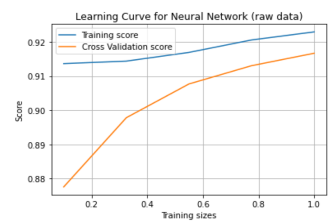
Figure 23: EM with dimensionality reduction

I'm not able to extract useful information from the metrics curves, so I will go to the scatter plots directly. It looks like EM clustering can separate two clusters quite well with any of the dimensionality reduction algorithms. However the third cluster is still mixed inside, especially with the RFC algorithm (Figure 23). Compared with the original EM algorithm(Figure 9(b)), it performs way better with the help of dimensionality reduction algorithms. But compared with the K-means algorithm, with the same type of dimensionality reduction or feature selection, like RFC (Figure 23(b) & 22(b)) and ICA (Figure 23(c) & 22(c)), EM clustering performs worse.

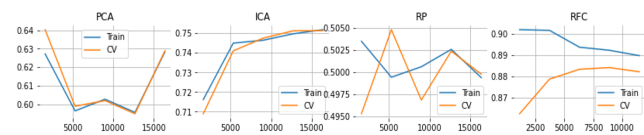
NN with Dimensionality Reduction

I chose the customer satisfaction questionnaire data set in this part. As I explained before, I used the same Neural Network hyper parameters as in Project1, and the same dimensionality reduction procedure as in the previous part.

NN, dimensionality reduction, and original features



(a) PCA)



(b) ICA

Figure 24: learning curves

Originally, NN algorithm has a high accuracy, low bias and low variance learning curve (Figure 24(a)). Although it is not quite convergent, it still has great performance.
With the four dimensionality reduction algorithms, the performance has declined. As we can see, none of the four learning curves is showing convergency. Among the four algorithms, RFC has the best accuracy score, as well as the most potential to converge.

Algo	RAW	PCA	ICA	RP	RFC
Train-Time(s)	3.235	2.714	7.776	2.627	42.16
Pred-Time(s)	0.011	0.022	0.007	0.02	0.009

Table 1: Summary of Time

RFC also has the longest training time, which is 14 to 15 times longer than others (Table 1). ICA has the second longest training time, but is still significantly shorter than the training time of RFC. For the prediction time, the performance of each algorithm is similarly short.

Clustering labels added

I added the clustering labels given by K-means and EM algorithms to the data set as new features, and ran the NN algorithm again to observe the performance. As a result of previous experiments, I'm using 3 clusters. 10 experiments have been done in this part.

	run_name	train_time	precision_train	precision_test	Description
0	Kmeans: RAW	64.853	0.8965	0.889	NN, adding Km labels as a new feature
1	EM: RAW	54.886	0.9056	0.915	NN, adding EM labels as a new feature
2	Kmeans: PCA	29.748	0.7854	0.806	NN with PCA, adding Km labels as a new feature
3	EM: PCA	26.704	0.8386	0.672	NN with PCA, adding EM labels as a new feature
4	Kmeans: ICA	16.463	0.6728	0.513	NN with ICA, adding Km labels as a new feature
5	EM: ICA	18.198	0.6679	0.340	NN with ICA, adding EM labels as a new feature
6	Kmeans: RP	36.466	0.8784	0.859	NN with RP, adding Km labels as a new feature
7	EM: RP	51.446	0.8775	0.816	NN with RP, adding EM labels as a new feature
8	Kmeans: RFC	35.538	0.8504	0.792	NN with RFC, adding Km labels as a new feature
9	EM: RFC	19.746	0.8429	0.677	NN with RFC, adding EM labels as a new feature

Figure 25: RFC:feature importance

This graph (Figure 25) contains all the results.
As I mentioned before, I think precision is an important metric based on the domain knowledge, and I was using it in the Project 1 to evaluate my model performance. Therefore, I'm using it as the metric to evaluate performance again.
It is obvious that both clustering algorithms have the longest training time with original features, which makes sense. The dimensionality reduction algorithms filter out some features or components according to the thresholds, therefore, they have less contents to train. Based on this logic, I found that ICA algorithm has filtered out the most features, so it has the shortest training time with both clustering algorithms. However the consequence is that ICA has the lowest test precision scores, especially with the EM algorithm, its precision score is even worse than random guess.
With the original labels, the EM algorithm has slightly better performance in both training and test data set, but this is not true with dimensionality reduction algorithms.
There are some good performance, including RP projections with K-means and EM clustering. All training and test precision scores are higher than 0.8, and the training precision are almost 0.9. But it also has the second longest training time.

Conclusion

I think my phishing website data set is not the best choice of this project, but it contributes some interesting points. It actually gave me the most obvious peaks, whereas the curves generated for the other two data sets were often inconclusive.
Whether to balance the data set in the data pre-processing part is not crucial, but scaling makes huge difference if the values of features are in a large range. In fact, my super imbalanced school prediction data set has the best performance in clustering.
Generally speaking, I think K-means has slightly better performance than EM algorithm, in that it separates clusters more clear. The dimensionality reduction algorithms definitely helps with clustering algorithms, because when we compare the scatter plots, it is very obvious that the clustering experiments on the data sets projected by the dimensionality reduction algorithms have way better performance. I think this is because 1) dimensionality reduction has reduced noises; and 2) some features may not be able to make contribution directly when doing clustering (take the RFC of the customer satisfaction questionnaire using EM clustering as an example), therefore, it needs PCA, ICA or RP to create new projections to make clustering easier.
As for the Neural Network part, I feel like the training time, number of features or components, and the accuracy/precision are directly proportional. It is interesting that NN with only ICA has a relatively long training time, whereas NN with ICA and clustering labels has the shortest training time. I think clustering has helped ICA to filter out more components. On the contrary, NN with only RFC has significantly longer training time, whereas NN with RFC and clustering labels has shorter training time. But the precision is not as good as NN with only RFC. In general, adding clustering labels as new features helped improving the performance of PCA, ICA, and RP, and it also helped reducing the training time of RFC.

References

[1] Jonathon Hare. "Enum GaussianMixtureModelEM.CovarianceType". In: (). URL: `http://openimaj.org/apidocs/org/openimaj/ml/gmm/GaussianMixtureModelEM.CovarianceType.html`.

[2] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.