# CS 7641 Machine Learning: Project-3

[Arti Chauhan: Mar-30-2018]

## ABSTRACT

This paper presents analysis for unsupervised algorithms and Dimensionality Reduction (DR) techniques used to train neural networks. Analysis was done in Scikit learn. Please refer to README.txt for more details. Graphs with better resolution can be found in CHARTS_P3.xlsx. Paper is organized into four parts.

- Part-1 explores clustering algorithms namely K-means and Expectation Maximization (EM).
- Part-2 presents Dimensionality Reduction and clustering analysis for principal component analysis (PCA), independent component analysis (ICA), random projection (RP) and random forest (RF).
- Part-3 evaluates neural network with dimension-reduced data.
- Part-4 evaluates neural network using clustering as a technique for dimensionality reduction.

## DATASETS

For this assignment I am reusing datasets from assignment-1.

- **PHISHING:** Phishing dataset entails features that have proved to be sound and effective in predicting phishing websites. It has 2456 instances and 30 discrete attributes. Goal is to predict if a website is malicious or not, given these attributes.
- **ABALONE:** this dataset has 4177 instances and 10 attributes (mix of categorical, integer real values) and 30 labels (age). The goal is to predict the age of abalone from physical measurements such as length, weight, height, gender etc. This is a not a balanced dataset.There are way less samples for age 1-7 and 12-29 when compared to 8-11. To combat this, I split the into 3 classes - 0 [<8] , 1 [8-10] and 2 [>10], which makes dataset somewhat balanced.

Both datasets are interesting for this assignment as they have many features and thus good candidates for evaluation of dimensionality reduction techniques. Also, reusing dataset provides me a baseline to compare against.

## PART 1 – CLUSTERING

K-Means and EM are two canonical approaches to clustering, i.e. dividing data points into meaningful groups. K-means works iteratively, reassigning data points to clusters and computing cluster centers based on the average of the point locations. In contrast, EM uses maximum likelihood parameters. It toggles between estimating the log-likelihood of current estimates (E step) and maximizing the likelihood based on the E step (M step). EM can be thought of as soft-clustering and K-means as hard clustering technique.

**Selection of optimal clusters (C):** Determining the optimal number of clusters in a data set is a fundamental issue in partitioning clustering.

1. For this purpose, I plotted WCSS (Within cluster sums of squares) for K-Means and Log-likelihood (LL) for GMM a function of cluster-count and used elbow method to hone into optimal C, ie point after which there is little to no improvement in WCSS or LL.
2. NN accuracy – I ran grid-search for each C to determine best NN parameters and used accuracy from best model in conjunction with step-1 to determine optimal C.

**Cluster Validation:** To validate choice of C from previous step, I used intrinsic measure 'Silhouette score' (which doesn't require knowledge of ground truth). Since in this setting I have access to true labels, I'm leveraging this info to compute completeness, homogeneity, V-Measure and Adjusted MI and accuracy to further evaluate quality of resulting clusters.

1. Silhouette score (SS) was used to understand shape and structure of selected clusters. Average silhouette determines how well each object lies within its cluster. A high average silhouette width indicates a good clustering.
    a. $S_i = b - a / \max(a, b)$
    b. *Where 'a' is average distance to all other observations within same cluster as that of observation* i *while 'b' is minimum of average distance to all other observations from all other clusters.*
2. It's worth mentioning that I used 'accuracy' as a goal to improve upon but not as sole criteria for cluster quality evaluation because it was easy to achieve high accuracy with large number of clusters as can be seen in charts below.
3. 'Homogeneity' and 'Completeness' are two most desirable properties clustering but they shouldn't be evaluated in isolation. Homogeneity is maximized when each cluster contains elements of as few different classes as possible. Completeness aims to

put all elements of each class in single clusters. One can maximize Homogeneity by having each data point as a cluster and maximize completeness by putting all data point in one cluster. Thus we want to take a holistic view of these two metrics, which is given by V-Measure.

i. V-Measure = 2 * (homogeneity * completeness) / (homogeneity + completeness)

4. MI measures the amount of information by which our knowledge about the classes increases when we are told what the clusters are. MI has same issue as purity (with K=n, MI=1). But with normalization we overcome this issue. Adjusted-MI (AMI) takes a value of 1 when the two partitions are identical and 0 when the MI between two partitions equals the value expected due to chance alone.

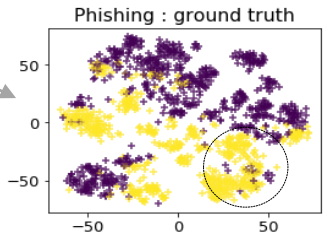
**K-Means Distance metric:** Please note that I did this assignment using Sklearn which by default uses Euclidian distance for K-Means and doesn't provide option to specify any other distance metric. To ensure that Euclidian is a reasonable choice for my dataset, I ran cluster analysis in Weka using manhattan distance. Euclidian based clustering was more closely aligned with natural clustering (based on labels) in my data. It intuitively makes sense because K-means aims to minimize within-cluster variance and variance is identical to sum of squared Euclidean distances from the center.

**Cluster initialization** – To avoid getting stuck in local optima, K-means was run with different centroids , where in each run, centroid was picked using Kmeans++, which is more likely to pick points far away from existing clusters proportional to the distance squared.

## 1.1 PHISHING

Charts below show all the afore-mentioned metrics as function of number of clusters (x-axis)

- Based on Fig-1.1a, I chose C=2 because it had best silhouette score. Based WCSS chart, one can argue C= 2 -7 is gives maximum reduction. But other metrics including accuracy favored C=2. For C>2, all metric show decline. This value makes sense as this Phishing is a binary classification dataset.
- For GMM, my results were quite poor compared to K-Means, which was counter intuitive. My expectation was that, given GMM is more flexible, it will be better or at par with K-Means. So tried different covariance settings to constrain the covariance of different classes estimated – 'tied' gave the best performance followed by 'spherical'. Based on Fig-1.1b, I chose C=2 based on LL & SS elbow. Metrics (V_meas, Accuracy etc) validated C=2 as optimal choice.
- Scatter plots in Fig 1 & b below show tSNE transformed data using cluster labeling produced by KM and GMM. Both methods clustering are close to ground truth, but misclassify points shown in dotted circle.




| | clusters | LL/WCSS | AMI | ACC | SHIL | HOMO | COMP | V_MEA |
|---|---|---|---|---|---|---|---|---|
| K-Means | 2 | 21242.3 | 0.5098 | 0.8945 | 0.1483 | 0.5100 | 0.5113 | 0.5106 |
| GMM | 2 | -8.6469 | 0.4778 | 0.8840 | 0.1352 | 0.4779 | 0.4826 | 0.4803 |

Fig 1.1.a

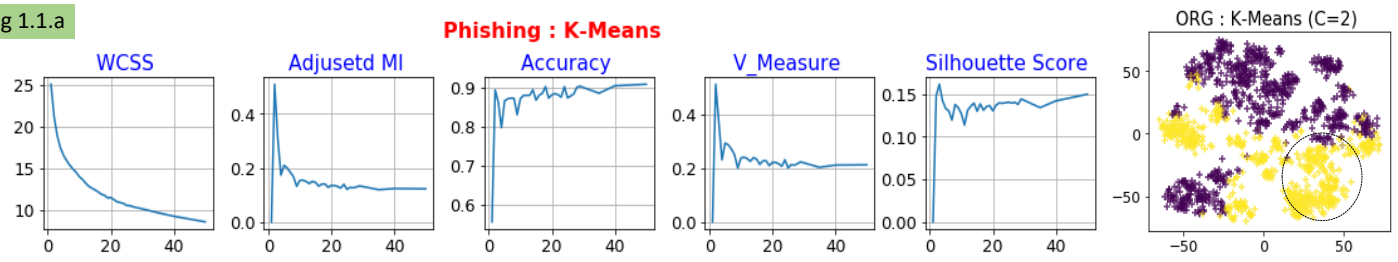
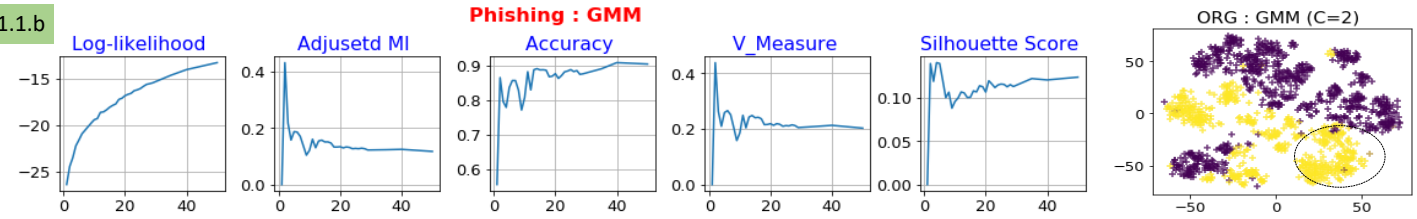


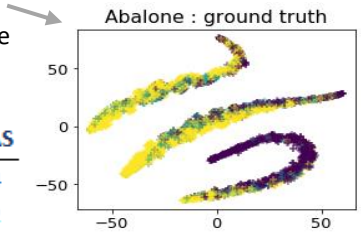

Fig 1.1.b



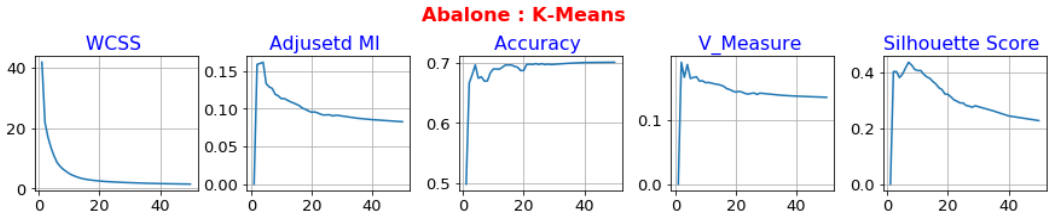

## 1.2 ABALONE

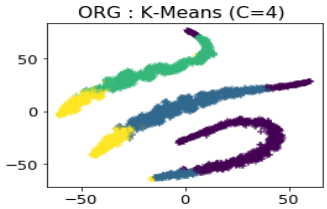
Using Elbow method, WCSS reduced the most between C= 3 to 6. After validating cluster quality based on V_Measure, accuracy and AMI, C=4 gave the best results. Whereas for GMM, best C=3. As C increases, LL and WCSS reduce but quality of clusters (silhouette-score, V_Measure) reduce significantly. We can see from table below, K-Means and GMM gave very comparable results for V_Measure and Accuracy, however number of clusters in GMM are more aligned with actual classes in the dataset, which is 3.

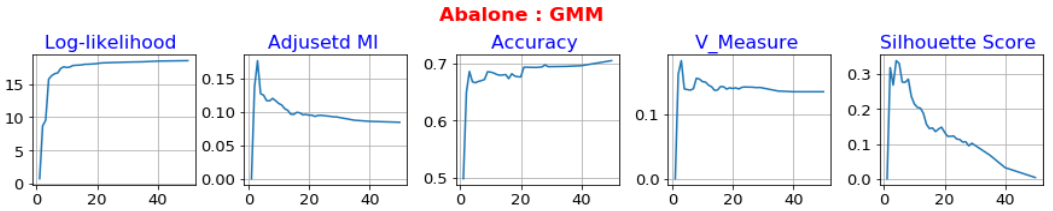
In contrast to Phishing, t-SNE transformed data for Abalone doesn't show very clean clusters. This evident from cluster Accuracy as well which runs around 70%.This tells that, given available features in the dataset, we may not be able to have perfect classification.

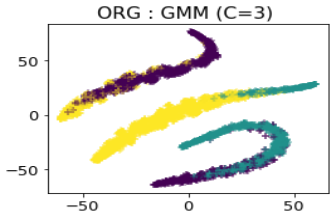



| Abalone | clusters | LL/WCSS | AMI | ACC | SHIL | HOMO | COMP | V_MEAS |
|---|---|---|---|---|---|---|---|---|
| K-Means | 4 | 13511.9 | 0.1614 | 0.6967 | 0.3818 | 0.2169 | 0.1618 | 0.1854 |
| GMM | 3 | 9.5378 | 0.1771 | 0.6864 | 0.2688 | 0.1916 | 0.1774 | 0.1842 |




# PART 2 – DIMENSIONALITY REDUCTION AND CLUSTERING

**Why we need dimensionality reduction?** Dimension reduction algorithms transform the input data to fewer dimensions, ensuring that it conveys similar information concisely. It removes multi-collinearity, redundant features and noise, there by compressing the data and reducing storage space requirement.

**Methodology**

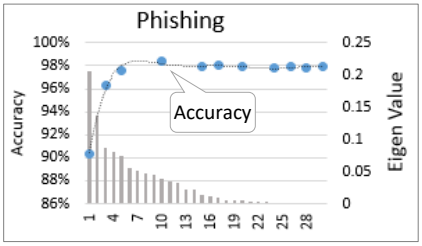
1. Apply dimensionality reduction algorithm and get the newly transformed dataset for each algorithm.
2. Perform exhaustive grid search on MLP classifier (NN) using dataset from step1 to determine optimal number of components.5 fold cross validation used. Create a new dataset using only optimal components.
3. Apply K-means and EM clustering analysis on dataset from step-2.
4. Clustering analysis - I followed same procedure as described Part-1 for determining optimal clusters and validating cluster quality.

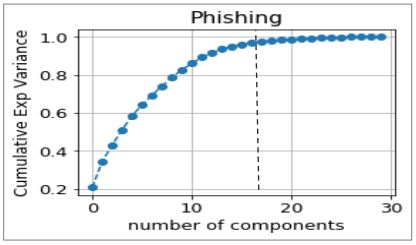
## 2.1 PRINCIPAL COMPONENT ANALYSIS (PCA)

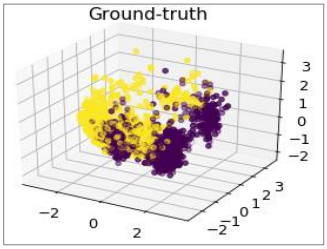
PCA finds the orthogonal eigenvectors that best captures the maximum amount of variance. It replaces original variables with new variables, called principal components, which are orthogonal and have variances (called eigenvalues) in decreasing order.

### 2.1.1 Dimensionality Reduction Analysis

**Phishing –** Leftmost chart below show normalized eigenvalues (gray bar) vs. accuracy (blue dots) for each principal component. Middle chart shows cumulative variance. First 10 PCs explain 82.5% variance and 1st 17 PCs 96.7% variance, yielding ~98% accuracy, after which is there is no gain in explained variance or accuracy. Hence PC=17 is a reasonable choice for this dataset.
Scatter plot below shows distribution of data along first 3 PCs – we can see resonable separation of labels except in the middle where purple points are mixed with yellow.




**Abalone**

**a)** Fig-d below shows cumulative explained variance. For this dataset, we see that 1st PC captures a huge amount (~70%) of variance and 1st 7 components explain 99.6% variance, yielding test-accuracy of 72.32%. PC=7 is a reasonable choice for this dataset.
**b)** In each PC, features that have a greater absolute weight "pull" the principle component more to that feature's direction. Fig-b shows weight of each feature for first 6 PCs. It is interesting to note that PC1 which explains most variance, gives equal weight to all features, except for gender=F,M, which means 8 out 10 features are equally important in explaining the variance of the data.
**c)** Fig-f shows distribution of data along PC-1. We can see that there is good separation for class-0 (purple) and class-2(yellow) not for class1.This can be related to data, where most data points lie in class-0 and 2 and less samples in class-1.

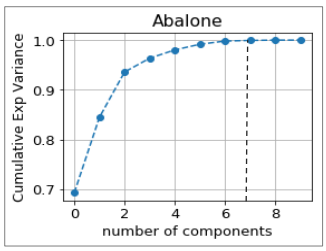



Fig-2.1.1d

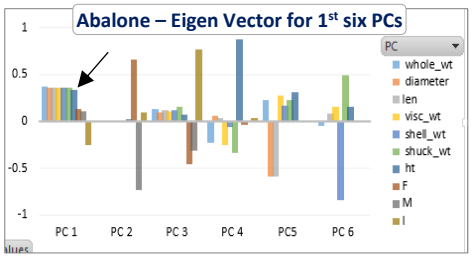



Fig-2.1.1e

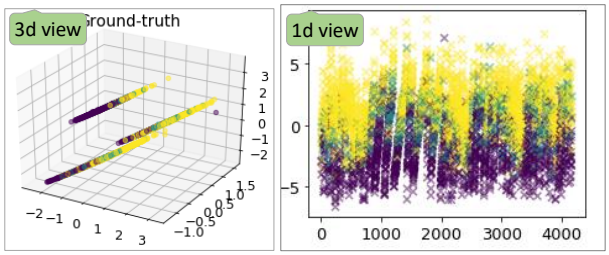
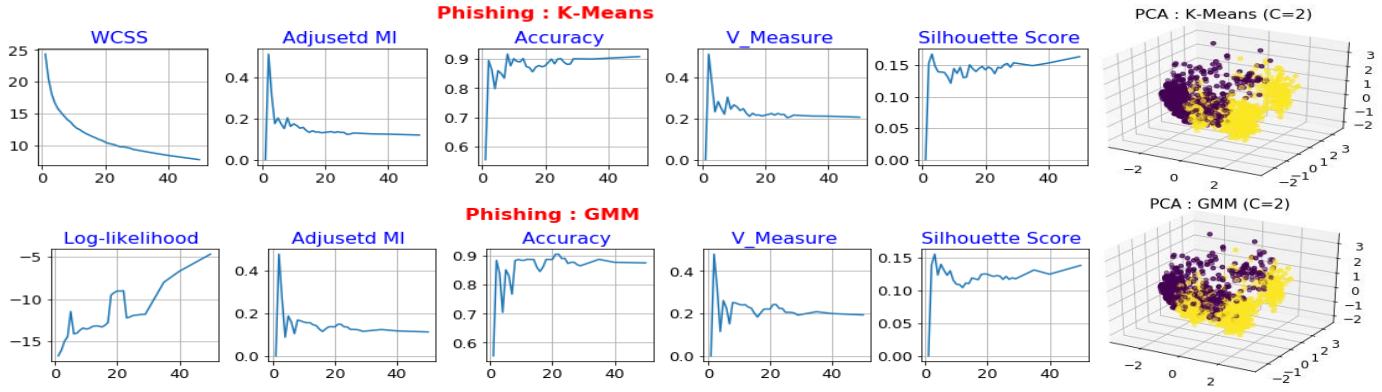



Fig-2.1.1f

## 2.1.2 Clustering Analysis

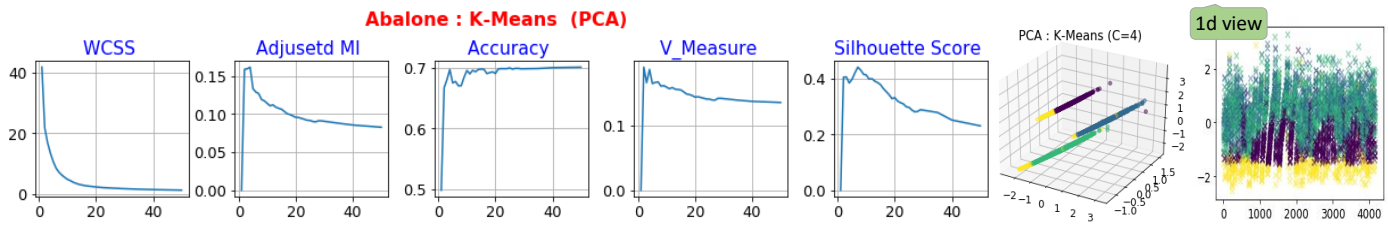
**Phishing**: With PCA reduced data, natural clustering was in range 2-6.However, based on cluster-quality metric, C=2 was the most optimal choice. These results are very close to clustering with original data, however here we worked with 17 (46% less) features. This suggest PCA did a very good job capturing important information and discarding irrelevance and noise. Visually too, we can see in 3d plots that clusters align closely with ground truth (shown in 2.1.1)

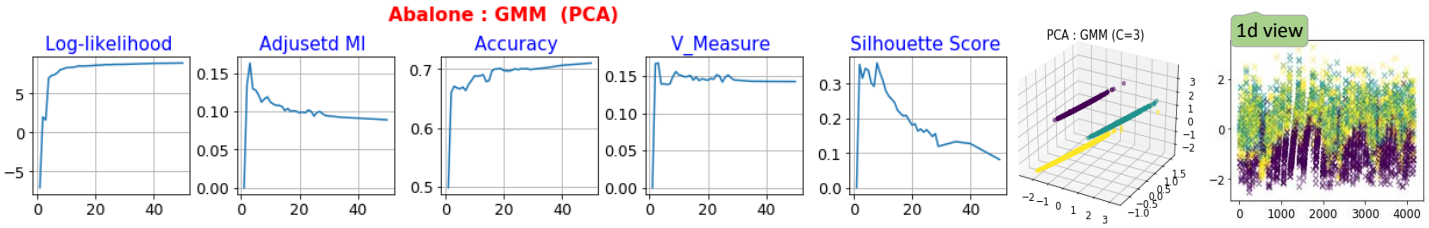
| PCA | Phishing | clusters | LL/WCSS | AMI | ACC | SHIL | HOMO | COMP | V_MEAS |
|---|---|---|---|---|---|---|---|---|---|
| | K-Means | 2 | 20459.8 | 0.5111 | 0.8950 | 0.1528 | 0.5112 | 0.5125 | 0.5119 |
| | GMM | 2 | -16.0396 | 0.4766 | 0.8836 | 0.1391 | 0.4767 | 0.4815 | 0.4791 |




**Abalone**: Similar to original data, here too natural clusters came out to be in 3-7 range but given accuracy and other intrinsic cluster quality measures, C=4 for K-means and C=3 for GMM was best choice. Compared to original data, V_Measure for GMM was slightly lower (0.167 vs. 0.184) but not too far apart. Since maximum variance is captured by 1st PC, I am showing below 1d view in addition to 3d view. Comparing to ground truth (Fig-2.1.1f), GMM made quite a few mistakes, thus lowering V_Measure and Accuracy, compared to K-Means.

| PCA | Abalone | clusters | LL/WCSS | AMI | ACC | SHIL | HOMO | COMP | V_MEAS |
|---|---|---|---|---|---|---|---|---|---|
| | K-Means | 4 | 13431.5 | 0.1614 | 0.6967 | 0.3838 | 0.2169 | 0.1618 | 0.1854 |
| | GMM | 3 | 1.6076 | 0.1630 | 0.6706 | 0.3161 | 0.1716 | 0.1634 | 0.1674 |






## 2.2 INDEPENDT COMPONENT ANALYSIS (ICA)

While PCA aims to find a set of uncorrelated components, ICA recovers a set of independent components. It uses Kurtosis and neg-entropy as surrogates for non-gaussianity. Since ICA separates sources by maximizing their non-Gaussianity, perfect Gaussian sources cannot be separated. However, in absence of complete independence between sources, ICA is still able to find a space where they are maximally independents.

### 2.2.1 Dimensionality Reduction Analysis

I am using Fast-ICA for this analysis. It rotates data (unitary transform) so that each axis looks as non-Gaussian as possible. Data was centered and whitened. Why? By bringing the mean to zero (centering) and normalizing the variance in all directions (whitening), algorithm gets freedom to rotate in all directions, otherwise algorithm is restricted rotate whole block to one axis.

**Phishing:** To find optimal independent components (ICs), I ran ICA with different K (number of components) and fed transformed data to ANN to determine accuracy and plotted it against mean kurtosis for each K (Fig-a below). Even though mean kurtosis (gray bars) keeps increasing with K, Accuracy flattens out after K > 17, suggesting optimal range of ICs is around 17 or so.

Next, I evaluated Kurtosis for each IC when K=17, 20, 22, 26. As can be seen from Fig-b & c that several ICs show Kurtosis! = 3, suggesting non Gaussian distribution. K=20 is deemed optimal as it provided a middle ground for reasonable accuracy and kurtosis.

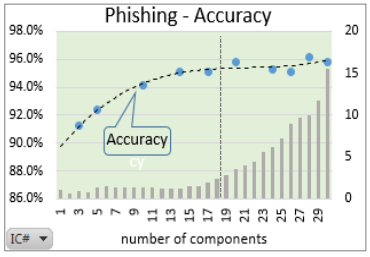



Fig 2.2.1a

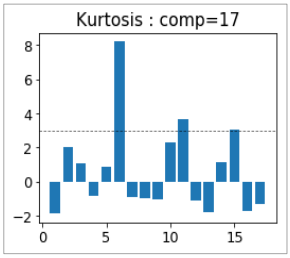



Fig 2.2.1b

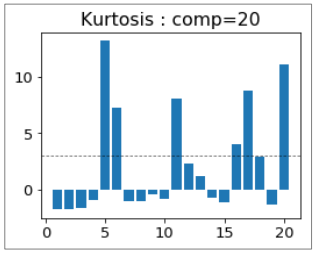
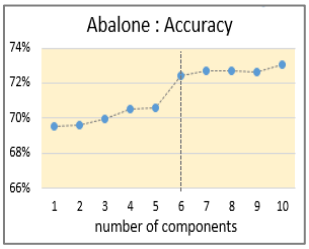
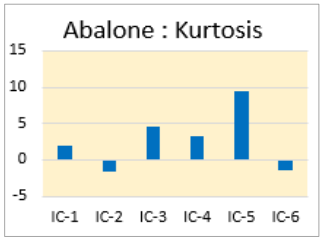



Fig 2.2.1c

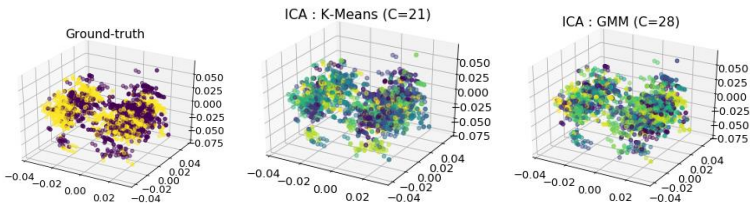
**Abalone:** Similar analysis done for Abalone. Accuracy chart suggests optimal IC to be 6, beyond which accuracy doesn't improve much. When I plotted Kurtosis (K=6), IC-1 was very high (1028), which raised a red flag for me. After some digging, I found that for 4 (out 4176) samples had transformed values were very high. Removing those 4 samples brought kurtosis down to 2.02.My data to ICA was centered and whitened. Hence not sure why those outliers turned up. But it taught me a lesson not take output of these APIs at face value and sensitivity of kurtosis to outliers.




### 2.2.2 Clustering Analysis

Optimal number for ICs determined in previous section were fed into K-Means and GMM models. Best clusters and corresponding quality metrics are shown in table below. For Abalone, clustering results on ICA-reduced data produced slightly lower but comparable scores to original datasets.

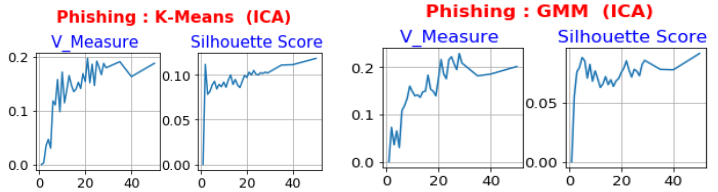



Phishing

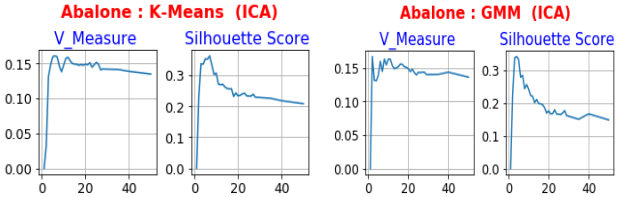
For phising datasets, cluster quality metric dipped and it took lot more clusters to get comparable accuracy. So tried tweaking number of ICs in range 10-25 range, but it didn't make a discernable change.ICA assumes that sources linearly mixed. May be for this dataset this assumption doesn't hold true or sources are gaussians. Scatter plot on left shows that data projectd on first 3 ICs doesn't reveal clean clusters

*Note: Due to lack of space, in subsequent sections I will be showing charts for important cluster quality metrics (V_Measure and Silhouette Score) but charts for all other metrics can be found in P3_charts.xlsx.*

| :A | Phishing | cluste | LL/WC | AMI | ACC | SHIL | HOM | COMP | V_ME |
|---|---|---|---|---|---|---|---|---|---|
| | K-Means | 21 | 8.7 | 0.120 | 0.859 | 0.103 | 0.528 | 0.121 | 0.197 |
| | GMM | 28 | 41.3 | 0.138 | 0.907 | 0.082 | 0.646 | 0.139 | 0.229 |




| Abalone | clu | LL/W | AMI | ACC | SHIL | HOM | COMP | V_ME |
|---|---|---|---|---|---|---|---|---|
| K-Means | 6 | 2.7 | 0.134 | 0.672 | 0.351 | 0.199 | 0.135 | 0.161 |
| GMM | 8 | 23.3 | 0.129 | 0.684 | 0.243 | 0.217 | 0.130 | 0.163 |




## 2.3 RANDOM PROJECTION (RP)

Random projection achieves dimensionality reduction by projecting the original input space on a randomly generated matrix. What it brings to the table is speed in computation time. For a data of size n∗k, PCA takes $O(k^2 n + k^3)$ whereas RP takes O(nkd) where d is reduced subspace. There can be slight loss of accuracy depending dataset as it doesn't strive to find best projections like PCA. I used Sparse RP for this analysis - it uses a sparse random matrix for projection, which is more memory-efficient and allows faster computation of the projected data, compared to alternative Gaussian random projection matrix.

### 2.3.1 Dimensionality Reduction Analysis

To determine optimal number of components, I plotted reconstruction–error against different number of random projections. This step was repeated for several seed values to account for randomness and evaluate the spread of reconstruction error. Then I trained the neural network with different number random components via exhaustive grid search to determine optimal components.

- Fig-a & c below show the spread of Reconstruction-Error for different seed values (orange dots). Phishing dataset had less variance in error than Abalone with Std_dev of 0.01 and 0.1 respectively. Error decreases with increased number of projections.
- Accuracy plots (Fig b & d) guide us to determine point of diminishing returns. Projection# >6 for Abalone and Projection# > 15 show minimal gain in accuracy, while giving reasonable reduction in reconstruction-error and hence were chose as optimal components.

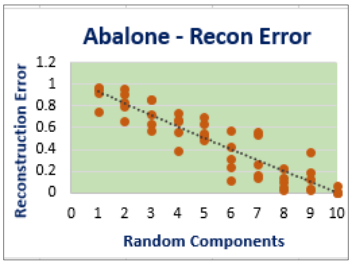



Fig 2.3.1a

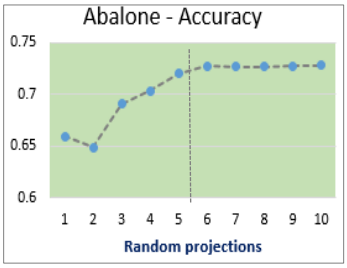



Fig 2.3.1b

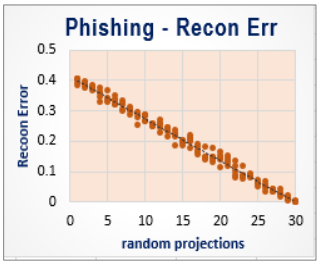



Fig 2.3.1c

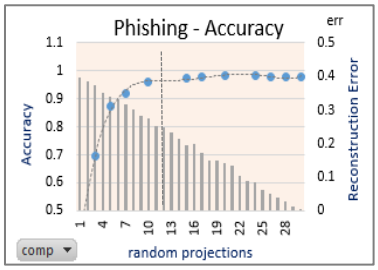



Fig 2.3.1d

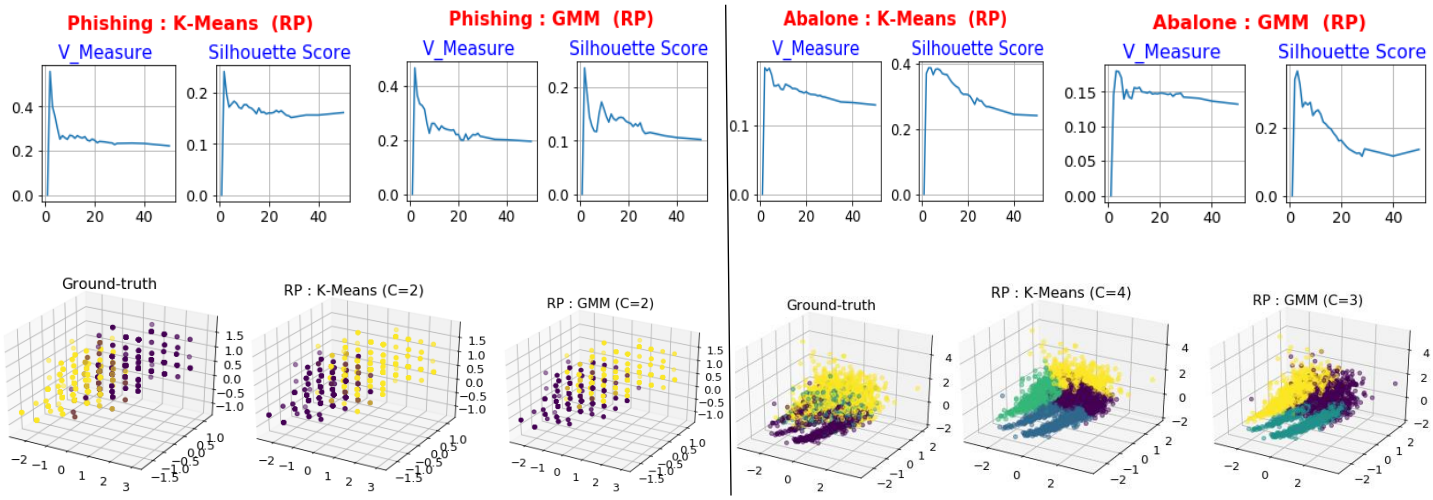
### 2.3.2 Clustering Analysis

Original data was transformed via RP, reduced to optimal number of components and fed to K-Means and GMM. For Phishing dataset, both algorithms had very clear peak in cluster-quality metrics at C=2 and results are very close to original dataset despite 50% less features. This suggests RP successfully managed to project this high dimensional dataset into a suitable lower-dimensional space in a way which preserves critical information about the data.

Similarly for Abalone dataset, RP's performance is similar to original dataset, aligning very closely to natural clusters (labels) of the dataset.

| RP | Phishing | clus | LL/WC | AMI | ACC | SHIL | HOMO | COMP | V_MEA |
|---|---|---|---|---|---|---|---|---|---|
| | K-Means | 2 | 20130.1 | 0.556 | 0.909 | 0.241 | 0.556 | 0.560 | 0.558 |
| | GMM | 2 | -11.5 | 0.464 | 0.879 | 0.236 | 0.464 | 0.470 | 0.467 |

| Abalone | cluster | LL/WC | AMI | ACC | SHIL | HOMO | COMP | V_MEA |
|---|---|---|---|---|---|---|---|---|
| K-Means | 4 | 6054.7 | 0.160 | 0.697 | 0.386 | 0.213 | 0.160 | 0.183 |
| GMM | 3 | 3.4 | 0.172 | 0.696 | 0.368 | 0.187 | 0.172 | 0.179 |




Scatter plots above show data along first 3 random-projections. For Phishing dataset, we see clusters produced by KM and GMM are close to ground truth (Accuracy=90%).But for Aabalone, model makes quite a few mistakes. That's probably because there aren't that many samples for medium ring size (green dots in ground truth plot).That's the limitation with the available data, not the algorithm.Probably using SMOTE to make this class more balanced will help imporve accuracy.

## 2.4 RANDOM FOREST (RF)

For this section I am using Random-Forest, an ensemble of Decision Trees (DT), for dimensionality reduction .During training, each DT classifier computes how much each feature decreases the weighted impurity in a tree. For a forest, the impurity decrease from each feature is averaged and features are ranked according to this measure. This ranking can be used to select features that account for most decrease in impurity. I used Gini as impurity criteria and 300 DT estimators were used in ensemble.

### 2.4.1 Dimensionality Reduction Analysis

**Phishing** – Fig-a show features-importance for Phishing dataset in descending order(blue bars) , along with accuracy of neural network for model trained with corresponding features.Though there are 30 features in all , top-5 features explain 75.7% of the data and top-10, 90.5%. Most discriminating features are SSL state, url of anchor and site's populatrity (web_traffic) -these features intuitvely makes sense why they were ranked higher. I removed 14 features from the dataset that explained mere 3.5% of data.

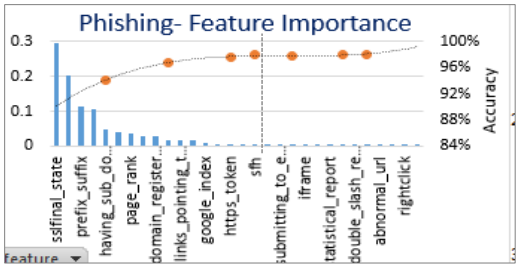



Fig-2.4.1a

**Abalone** – In contrast, we see almost all features (except for gender) have comparable importance. This aligns with PCA output where PC-1 had almost equal weight for all features but gender. Top 5 features explain 59% of data. I selected top-7 features, which explained 95% of data.

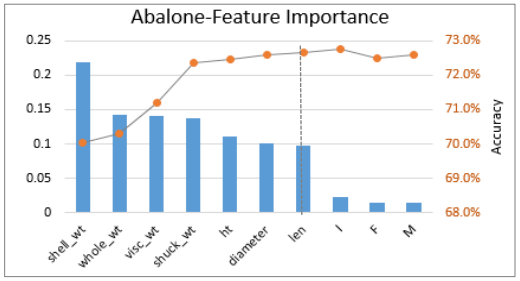


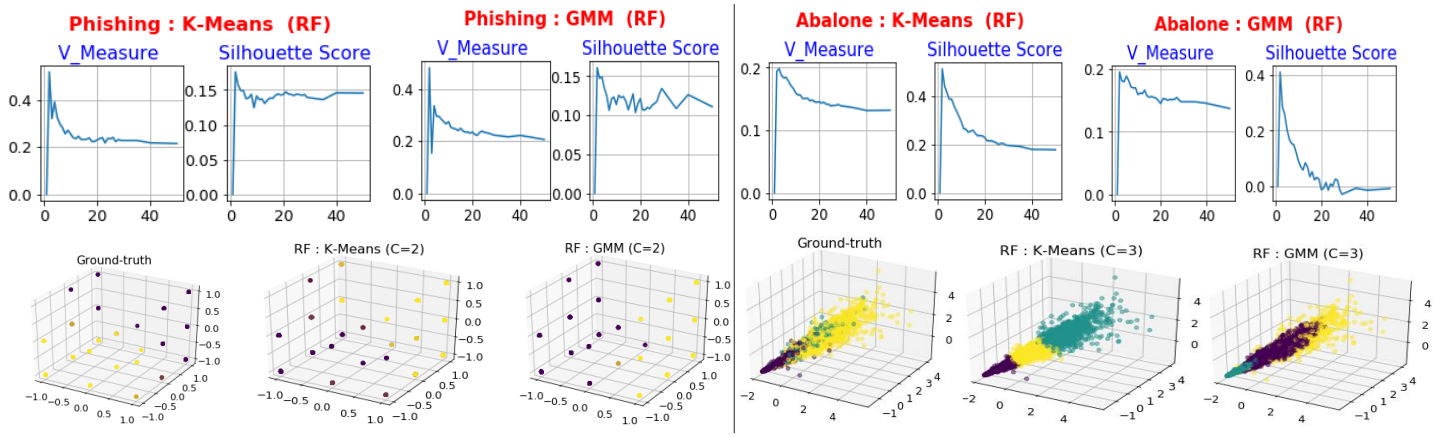

Fig-2.4.1b

### 2.4.2 Clustering Analysis

Table below show clustering results for Phishing with 16 features and for Abalone with 7 features. Results are at par with clustering performed with original dataset and slightly better than other three DR techniques. That's probably because RF has one advantage over other DR techniques and that is it made use of ground-truth (actual labels) when determining which features are important. For Phishing, optimal clusters C=2 and for Abalone C=3, which aligns with classes in these datasets.

| RF | Phishing | clust | LL/WCSS | AMI | ACC | SHIL | HOMO | COMP | V_MEA |
|---|---|---|---|---|---|---|---|---|---|
| | K-Means | 2 | 16560.6 | 0.516 | 0.897 | 0.176 | 0.516 | 0.518 | 0.517 |
| | GMM | 2 | -14.0 | 0.478 | 0.884 | 0.161 | 0.478 | 0.483 | 0.480 |

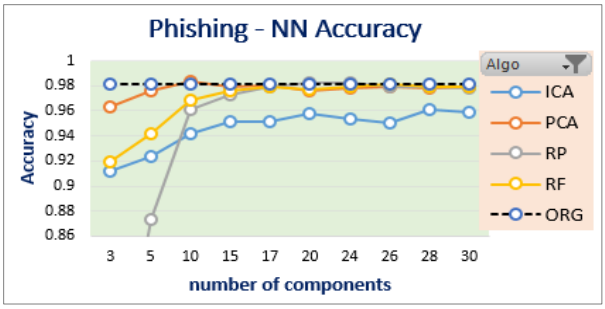
| Abalone | cluster | LL/WCS | AMI | ACC | SHIL | HOMO | COMP | V_MEA |
|---|---|---|---|---|---|---|---|---|
| K-Means | 3 | 7024.2 | 0.191 | 0.678 | 0.442 | 0.205 | 0.192 | 0.198 |
| GMM | 3 | 0.159 | 0.173 | 0.671 | 0.284 | 0.173 | 0.189 | 0.181 |




Scatter plots above show data along first 3 random-projections. For Phishing dataset, data looks sparse because features are binary and hence datapoint are overlapping.We see clusters produced by KM and GMM are close to ground truth (Accuracy=90%).For Abalone, we don't see that clear demarkation. That's because 1st three features explain only 50% of data.Accuracy for Abalone dipped slightly (67.8 vs 69.7) when compared to other DR. This was expected as almost all features matter for this datatset and I got rid of gender that accounted for 5% of data.

## PART 3 – DIMENSIONALITY REDUCTION AND NEURAL NETWORK (NN)

For this part of the assignment I picked **Phishing** dataset to train a neural network with four dimensionality-reduced datasets as input. For each DR algorithm, I created datasets with components = [3,5,10,15,17,20,24,26,28,30] and fed it to neural network. Exhaustive search was performed to derive best set of NN parameters (alpha, hidden layers / neurons, activation and solver) for each set, using 5 fold cross validation. Solver L-BFGS solver gave much better performance than default 'Adam'. Phishing is a small dataset and L_BFGS is known to perform better for small datasets. Results with 'ReLu' were slightly better than that with 'Sigmoid' activation. Results of this experiment are shown in graph below.




Recall this dataset has 30 features.

- PCA did exceptionally well when compared to other 3 algorithms for low component range (5-15).
- Beyond n=17, RP and RF were comparable to PCA and original dataset.
- ICA's performance was reasonable but not at par with other 3 algorithms.

Table on right summarizes performance of different DR algorithm for component count =10. Test accuracy and F1-score/AUC are quite close. That's probably because Phishing is a quite balanced dataset.

PHISHING WEBSITES

| Component = 10 | Accuracy | | | F1 | Recall | Precisio | AUC | time | |
|---|---|---|---|---|---|---|---|---|---|
| | train | CV | test | | | | | train | test |
| PCA | 0.9937 | 0.9837 | 0.9878 | 0.9878 | 0.9878 | 0.98785 | 0.9881 | 5.68 | 0.000 |
| ICA | 0.9643 | 0.9418 | 0.9472 | 0.9472 | 0.9472 | 0.9476 | 0.9479 | 11.71 | 0.000 |
| RP | 0.9905 | 0.9609 | 0.9512 | 0.9511 | 0.9512 | 0.9515 | 0.9489 | 5.88 | 0.000 |
| RF | 0.9774 | 0.9682 | 0.9594 | 0.9594 | 0.9594 | 0.9595 | 0.9598 | 3.84 | 0.155 |
| original Data | 0.9942 | 0.9803 | 0.9810 | 0.9810 | 0.9810 | 0.9811 | 0.9814 | 2.80 | 0.001 |

Baseline

1. PCA is the winner here. It in fact had better test accuracy (.988 vs .981) and AUC than original dataset. PCA was able to capture key structure/information of underlying data using only 1/3 of feature-size. Fact that it delivered better results than original dataset suggest that PCA transformation got rid of irrelevance/noise in the data. This highlights the strength of using DR.

2. Random-Forest was runner up, followed by Random-Projection. Note that Random Forest used actual labels in determining which features are important, where as other three algorithms are truly unsupervised in their dimensionality reduction approach. It was bit unsettling to see RP projection test-accuracy deviate by as much as 1-1.5% as seed was changed. So I used average of 5 runs.

3. ICA didn't perform that well. This is probably driven by low non gaussianity (Kurtosis) of ICs determined by algorithm (shown section 2.2.1). Train time for ICA was high because it required more complex NN (60 neurons) to achieve this accuracy, where as other three algorithms used 30 neurons.

## PART 4 – CLUSTERING AS DIMENSIONALITY REDUCTION AND NN

In this section, clustering was used as a technique for dimensionality reduction, where output of K-Means (distance from the centroid) and EM (probability) were used as the only features to train Neural Net for Phishing dataset.

**Experiments** - With my initial runs I got low accuracy compared to original data (91.5 vs. 98%). This was contrary to my expectation as I was not expecting this delta to be that high.

After a hard look at my code, I realized I could try different solvers and scaling output of KM and GMM.Changing solver from Adam to Lbfgs increased the accuracy from 91.5% to 96.94% for K-means (Table 4a). Scaling the clustering output further improved the accuracy to 98.65%.However, for GMM improvement was not that dramatic. After trying numerous NN config, I experimented with different covariance type. Changing it from default ('Full') to 'Tied' gave a significant boost to accuracy (Table 4b) .

| cluster-scaled ? | Solver | K-Means | GMM |
|---|---|---|---|
| No | Adam | 91.52 | 83.26 |
| No | Lbfgs | 96.94 | 83.63 |
| yes | Lbfgs | 98.65 | 83.67 |

Table 4a

| cluster-scaled ? | Covariance Type | GMM |
|---|---|---|
| Yes | Full | 83.67 |
| Yes | Tied | 88.61 |

Table 4b

**Performance Analysis –** Fig 4c and d below show Neural Net CV-accuracy with K-Means and GMM clusters as input. This info is presented for different cluster sizes and for different NN configurations. CV-Accuracy for K-Means improved from 90% to 98% as number of clusters increased on 2 to 7. Beyond that gain in accuracy were minimal. However same phenomenon is not seen for GMM, where CV-accuracy peaked at C=2 and then declined and improved again beyond C=7 but didn't go as high as C=2.

Fig-4c

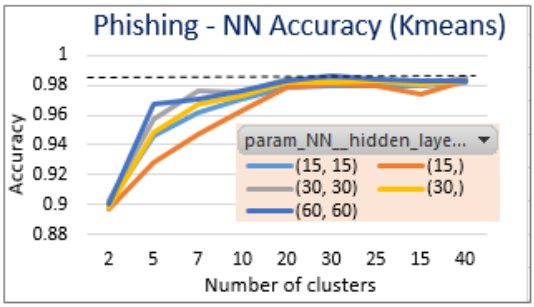
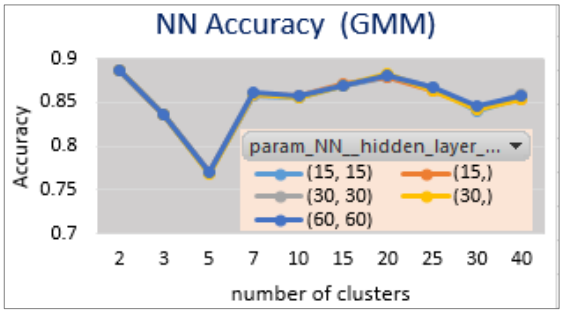



Fig-4d




To compare time and performance metrics, I selected C =2, 10, 20. Table below summarizes this info for Train/CV/Test set.

1. For C=2, K-Means and GMM give similar accuracy and F1 score but values are below baseline (98.1).However, K-means surpass GMM at C=10 (98.37 vs 85.77) and C=20.
2. For C>=10, K-means beats the baseline. With C=10 (1/3 features), test accuracy improves by 0.38% compared to original dataset and with C=20 (2/3 features) by 0.80%.
3. However this gain in accuracy came at the cost of increase in model complexity (right most column) and increased training time (which almost doubled compared to baseline)

**PHISHING WEBSITES**

| Algo | cluster/ feature# | Accuracy | | | F1 | Recall | Precisio | AUC | time | | NN alpha | NN hidden layer |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | train | CV | test | | | | | train | test | | |
| K-Means | 2 | 0.9204 | 0.9023 | 0.8577 | 0.8572 | 0.8577 | 0.8580 | 0.8531 | 5.27 | 0.0005 | 0.1 | (30, 30) |
| | 10 | 0.9937 | 0.9764 | 0.9837 | 0.9838 | 0.9837 | 0.9839 | 0.9844 | 14.81 | 0.0007 | 1 | (60, 60) |
| | 20 | 0.9937 | 0.9837 | 0.9878 | 0.9878 | 0.9878 | 0.9878 | 0.9872 | 6.08 | 0.0009 | 0.1 | (30, 30) |
| GMM | 2 | 0.8891 | 0.8868 | 0.8659 | 0.8651 | 0.8659 | 0.8668 | 0.8604 | 0.13 | 0.0007 | 0.001 | (15,) |
| | 10 | 0.8738 | 0.8571 | 0.8577 | 0.8577 | 0.8577 | 0.8576 | 0.8557 | 1.97 | 0.0013 | 0.1 | (30, 30) |
| | 20 | 0.8701 | 0.8827 | 0.8577 | 0.8562 | 0.8577 | 0.8614 | 0.8496 | 1.22 | 0.0017 | 0.001 | (30,) |
| original Dat | 30 | 0.9942 | 0.9803 | 0.9810 | 0.9810 | 0.9810 | 0.9811 | 0.9814 | 2.80 | 0.001 | 0.1 | (30,) |

Fig-4e

Baseline

**Cluster label as an additional feature:** I also tried using cluster labels as a new feature, added to original dataset, to train NN. Result improve significantly with this approach as expected because clustering by itself was able to achieve 88-89% accuracy (section 1.1 above). Comparison shown in table on right. With Cluster=2, KM improves from 90% to 98% and GMM accuracy improves from 88% to 98%. However, this doesn't reducing the dimensions. Experiment was done for my understanding.

**Accuracy**

| Algo | clusters | Using clustering output -> NN | | Original Data + cluster labels -> NN | |
|---|---|---|---|---|---|
| | | train | CV | train | CV |
| K-Means | 2 | 0.9204 | 0.9023 | 0.9873 | 0.9805 |
| | 10 | 0.9937 | 0.9764 | 0.9880 | 0.9796 |
| | 20 | 0.9937 | 0.9837 | 0.9952 | 0.9772 |
| GMM | 2 | 0.8891 | 0.8868 | 0.9884 | 0.9805 |
| | 10 | 0.8738 | 0.8571 | 0.9902 | 0.9809 |
| | 20 | 0.8701 | 0.8827 | 0.9912 | 0.9800 |

## CONCLUSION

Above experiments highlight the effectiveness of dimensionality reduction algorithms in dealing with curse of dimensionality, which mars many datasets. In addition, these algorithm provide a way to visualize high dimensional data in some meaningful way. These are useful tools to gain some intuition behind dataset we are working with.

In table below I have summarized all important metrics – Log likelihood (GMM) , WCSS(K-Means), Cluster Accuracy ( comparison against ground truth), Silhouette score, cluster Homogeneity, Completeness and V_Measure for all DR techniques for easy comparison. All DR algorithms except for ICA, performed better or at par with original dataset for both Phishing and Abalone. This is really powerful because by reducing number of features, we save storage as well as execution time, without comprising on model's predictive power.

**Phishing**

| | Phishing | cluster | LL/WCSS | AMI | ACC | SHIL | HOMO | COMP | V_MEA |
|---|---|---|---|---|---|---|---|---|---|
| ORG | K-Means | 2 | 21242.3 | 0.510 | 0.895 | 0.148 | 0.510 | 0.511 | 0.511 |
| | GMM | 2 | -8.6469 | 0.478 | 0.884 | 0.135 | 0.478 | 0.483 | 0.480 |
| PCA | K-Means | 2 | 20459.8 | 0.511 | 0.895 | 0.153 | 0.511 | 0.513 | 0.512 |
| | GMM | 2 | -16.0396 | 0.477 | 0.884 | 0.139 | 0.477 | 0.482 | 0.479 |
| ICA | K-Means | 21 | 8.7 | 0.120 | 0.859 | 0.103 | 0.528 | 0.121 | 0.197 |
| | GMM | 28 | 41.3 | 0.138 | 0.907 | 0.082 | 0.646 | 0.139 | 0.229 |
| R-Projecti | K-Means | 2 | 20130.1 | 0.556 | 0.909 | 0.241 | 0.556 | 0.560 | 0.558 |
| | GMM | 2 | -11.5 | 0.464 | 0.879 | 0.236 | 0.464 | 0.470 | 0.467 |
| R-Forest | K-Means | 2 | 16560.6 | 0.516 | 0.897 | 0.176 | 0.516 | 0.518 | 0.517 |
| | GMM | 2 | -14.0 | 0.478 | 0.884 | 0.161 | 0.478 | 0.483 | 0.480 |

**Abalone**

| Abalone | clust | LL/WCS | AMI | ACC | SHIL | HOMO | COMP | V_MEA |
|---|---|---|---|---|---|---|---|---|
| K-Means | 4 | 13511.9 | 0.161 | 0.697 | 0.382 | 0.217 | 0.162 | 0.185 |
| GMM | 3 | 9.5378 | 0.177 | 0.686 | 0.269 | 0.192 | 0.177 | 0.184 |
| K-Means | 4 | 13431.5 | 0.161 | 0.697 | 0.384 | 0.217 | 0.162 | 0.185 |
| GMM | 3 | 1.6076 | 0.163 | 0.671 | 0.316 | 0.172 | 0.163 | 0.167 |
| K-Means | 6 | 2.7 | 0.134 | 0.672 | 0.351 | 0.199 | 0.135 | 0.161 |
| GMM | 8 | 23.3 | 0.129 | 0.684 | 0.243 | 0.217 | 0.130 | 0.163 |
| K-Means | 4 | 6054.7 | 0.160 | 0.697 | 0.386 | 0.213 | 0.160 | 0.183 |
| GMM | 3 | 3.4 | 0.172 | 0.696 | 0.368 | 0.187 | 0.172 | 0.179 |
| K-Means | 3 | 7024.2 | 0.191 | 0.678 | 0.442 | 0.205 | 0.192 | 0.198 |
| GMM | 3 | 0.159 | 0.173 | 0.671 | 0.284 | 0.173 | 0.189 | 0.181 |

- I had my doubts about how random projections can capture meaningful info about data but I was proven wrong. Random-Projection gave best performance for Phishing and Random-Forest for Abalone, improving both homogeneity and completeness and AMI of resulting clusters. PCA was not very far off either.
- ICA didn't perform that well. ICA is rooted in BSS (Blind Source separation) and assumes that sources are independent and non-Gaussian. Low scores suggests that my datasets didn't fully conform to these assumptions.
- Clustering as dimensionality reduction technique was quite effective as well and gave performance close to original dataset despite using 1/3 of features (Fig-4e)
- This project also helped me develop good intuition about my datasets as I was able to visualize the data and was able to determine which features are important. Eg: For Abalone, almost all features are important in determining age (label).In case of phishing, SSL state, url of anchor and site's populatrity (web_traffic) are much more important than others.