Assignment 3 : Unsupervised Learning and Dimensionality Reduction

Stella L. Soh (lsoh3)

Abstract

This report explores clustering algorithms and dimensionality reduction techniques prior to applying supervised learning to datasets.

1. Introduction

From Assignments 1 and 2, we have built up the concepts and understanding of supervised learning. In this assignment, we explore 2 clustering - k-Means and Expectation Maximization (EM) and 4 dimensionality reduction algorithms - Principal Component Analysis (PCA), Independent Component Analysis (ICA), Random Projection (RP) and Random Forest Classifier (algorithm of choice) and observe their behavior prior to supervised learning.

This report is structured in 3 parts as follows: **PART 1** deals with clustering, **Parts 2** and **3** pertains to application of dimensionality reduction and re-clustering of the 2 datasets, and **Parts 4** and **5** discusses dimensionality reduction and re-clustering with neural networks.

1. 1 Dataset 1: Wine Quality

The wine quality dataset is determined by 11 physical attributes and contains 2 classes **-** 'good' (1) and 'bad' (0). From a machine learning perspective, most of the physical attributes in Dataset 1 are numeric features, and the equally occurring classes lend themselves well to using clustering and dimensionality reduction to train neural network.

1.2 Dataset 2: Star3642 Balanced

This dataset contains 2 classes - Giants (1) vs Dwarfs(0). Primarily, B-V color and the Absolute Magnitude are used to identify Giant or Dwarf stars. Being a 6 attributes by 3642 examples - this dataset offers me only a sparse sampling of space. However, this represents the reality of some datasets, and may offer a contrast to the behavior of clustering and dimensionality reduction against Dataset 1.

PART 1

2. Clustering Algorithms

Clustering is an unsupervised task whose aim is to find natural groups or clusters within the feature space of input data. We will explore 2 clustering algorithms - k-Means and Expectation-Maximization (EM) on Dataset 1 and Dataset 2.

2.1 k-Means and Expectation-Maximization (EM)

k-Means algorithm divides a set of **N** samples into **k** disjoint clusters **C,** each described by the means of the samples in the cluster. The means are commonly called the cluster "**centroids**". For Part 1, I set kclusters (or the number of clusters) to span from 2 to 100 in steps of 2 in my code, and used sklearn.cluster's KMeans(), whereby I obtained Silhouette scores, Homogeneity scores, and F1-scores.

Expectation-Maximization (**EM**) is a classic algorithm that aims to iteratively find **k** distributions of data such that the log-likelihood of data (given the distribution) is maximized. It alternates between the **E**-step (Expectation) and the **M**-step (Maximization) until there is convergence.

Dataset 1 (Wine Quality)

For Dataset 1, when I ran **k-Means**, I found that F-1 scores and Homogeneity scores peak at n_clusters around 90 and 95 respectively. However, at these cluster values, the Silhouette scores hover around 0.25 and 0.30 for Dataset 1. This represents very low Silhouette scores. So how do I choose the value of k, the optimal number of clusters? The k-Means algorithm "clusters data by separating samples into k groups of equal variance, minimizing a criterion

known as inertia or within-cluster sum of squares" [1]. The inertia_attribute of k-Means offers an elbow method to choose k. The elbow in Fig. 1 is the location whereby the inertia of k-Means stops decreasing. In Fig. 1, I observe that the approximate position could be at k=20 for Dataset 1. So when I next run evaluate_kmeans(), I used k=20 for cluster size.

How about **EM**? Again, I obtained the Silhouette scores, Homogeneity scores, and F-1 scores charts just like in the case of k-Means. Of interest to note is: In addition to those charts, I used Scikit-Learn's GMM estimator to compute Akaike Information Criterion (AIC)[2] and Bayesian Information Criterion (BIC)[3]. Fig. 2 illustrates the model complexity of Dataset 1 using AIC and BIC.

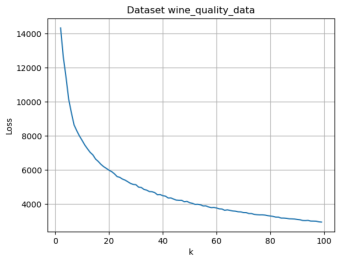


Fig. 1 - Loss vs k for k-Means: Dataset 1

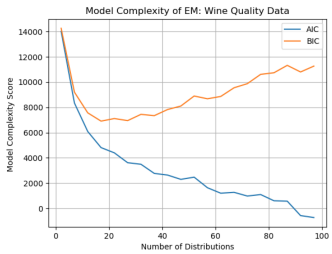


Fig. 2 - Model Complexity of EM: Dataset 1

The optimal number of clusters is the value that minimizes AIC or BIC. I am opting to use the AIC because per Machine Learning Mastery, the AIC statistic "penalizes complex models less". The optimal number of clusters (if opting to use AIC) appears to be 98, as shown in Fig. 2 above.

Dataset 2 (Star3642 Balanced)

For Dataset 2, again, I obtained corresponding Silhouette scores, Homogeneity scores, and F-1 scores charts. The loss vs k for k-Means and the model complexity charts are as shown in Figs. 3 and 4:

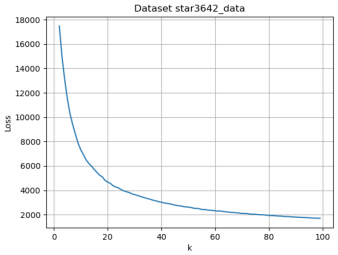


Fig. 3 -- Loss vs k: Dataset 2

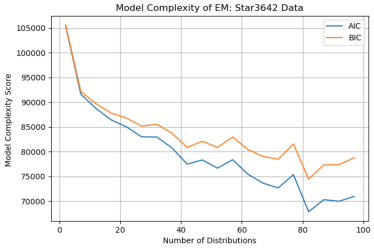


Fig. 4 -- Model Complexity of EM: Dataset 2

In this case, as shown in Fig. 3. k=18 could be the approximate elbow location for k-Means. Again, opting to use AIC, the minimum value of AIC occurs at around 82 as shown in Fig. 4 for the case of EM.

Based on the respective optimal cluster and component values obtained, I have generated confusion matrices and evaluation metrics. The results of evaluation metrics are tabulated in Tables 1 and 2 below:

K-Means

	Training Time	F-1	Accuracy	Precision	AUC	Recall
Wine Quality	0.3613 s	0.7127	0.7061	0.7465	0.7079	0.6819

[1] Refer to *Finding The Optimal Number of Clusters*
[2] *Akaike Information Criterion*
[3] *Bayesian Information Criterion*

Star3642	1.2741 s	0.8828	0.8825	0.8804	0.8825	0.8852

Table 1 - Training time, F-1, Accuracy, Precision, AUC and Recall - k-Means

EM

	Training Time	F-1	Accuracy	Precision	AUC	Recall
Wine Quality	1.0322 s	0.7464	0.7386	0.7755	0.7400	0.7193
Star3642	0.8552 s	0.8807	0.8828	0.8965	0.8828	0.8655

Table 2 - Training time, F-1, Accuracy, Precision, AUC and Recall - EM

CONCLUSION

Across the board, Wine Quality dataset shows an improvement in F-1, Accuracy, Precision, Recall and AUC under EM compared to k-Means. This could be at the expense of longer (1.0322 s vs 0.3613 s) training time. On the other hand, Star3642 datasets fares slightly better in F-1 and recall under k-Means. Star3642 dataset shows a shorter training time under k-Means compared to EM. This came as a surprise to me as I understand that EM needs to iteratively calculate the log-likelihood for current parameters and maximizing them, and hence the time taken to run EM would be significantly higher compared to k-Means.

To address the question of whether the clustered datasets align with the labels, I tapped into Seaborn's pairplot class to explore the pairwise relationship between k-Means cluster labels and target labels, and between EM cluster labels and target labels for both Wine Quality and Star3642 datasets.

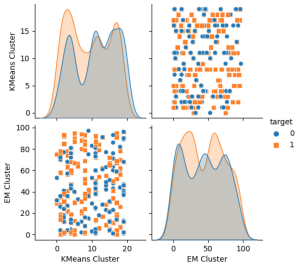


Fig. 5 - Pairplot: Wine Quality

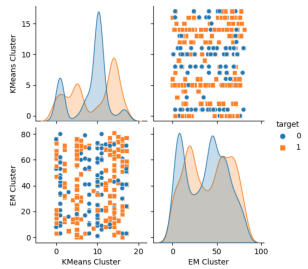


Fig. 6 - Pairplot: Star3642

In Fig 5 above, let us consider the top right quadrant. Each of the blue circles (class 0) and orange squares (class 1) are colored by target labels y. For example, a particular class 1 target label could be made up of a sample from k-Means cluster 7 and EM Cluster 12; a particular class 0 target label could be made up of a sample from k-Means cluster 9 and EM Cluster 22. To consider the relationship between k-Means cluster label and target label, temporarily assume the absence of EM Cluster in the top right quadrant. Each row of horizontal line made up of orange squares and blue circles corresponds to one k-Means Cluster. Most of these horizontal lines are of almost uniform blue or orange color.

In the bottom left quadrant, we could see the same behavior exhibited by the EM Cluster. Only now in this round of analysis, consider k-Means out of the picture as we are considering the relationship between EM and target label. This shows that the newly-clustered Wine Quality dataset's k-Means and EM cluster labels align well with the original, unclustered dataset's target label.

Same analysis goes for the pairplot of Star3642 in Fig.6 if we were to consider the top right and bottom left quadrants. We could, therefore, conclude that the newly-clustered Star3642 dataset's k-Means and EM cluster labels align well with the original, unclustered dataset's target label.

PARTS 2 and 3

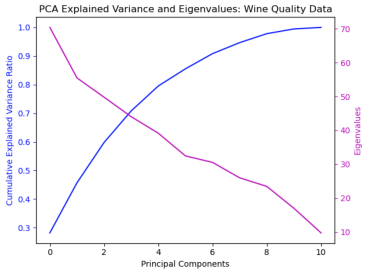
3. Dimensionality Reduction (DR) Algorithms

Dimensionality reduction (DR) algorithms select features based on feature importance, and extract features and hence compress information (in the case of PCA) or separate information (in the case of ICA) such that the input space is transformed into a space of lower dimension prior to modeling.

In part 3, I re-clustered the datasets after running DR algorithms. Out of the slew of Silhouette, Homogeneity, F-1 scores and training time charts generated, it'd be most interesting to look at the 16 F-1 scores charts as we could correlate them with the data shown in the tables. To me, in evaluating the clustering after the application of DR algorithms, knowing the accuracies such as F-1 scores, Accuracy, Precision, Recall and AUC gives me better insight. I have, therefore, continued to tabulate the results of evaluate_kmeans() and evalute_EM() in Tables 3 -12.

3.1 Principal Component Analysis (PCA)

I used PCA class of sklearn.decomposition and plotted eigenvalues and cumulative explained variance ratio as shown in Figs. 7 and 8 below:



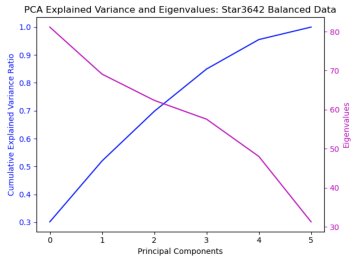


Fig. 7 - Dataset 1: PCA's Explained Variance and Eigenvalues

Fig. 8- Dataset 2: PCA's Explained Variance and Eigenvalues

Cumulative explained variance enables us to visualize the variance explained by each principal component. The eigenvalues correspond to the proportion of cumulative variance explained by each component. The question is: how many principal components are sufficient to represent the dataset? A good rule of thumb is to select the number of components such that the chosen subset of components could explain 80 - 90% of the variance. In this case, for Dataset 1, I shall choose n_components=10, as the explained variance is highest, and eigenvalues=10 at the corresponding principal components of 10. For Dataset 2, I shall choose n_components=5 as a principal components of 5 explained 90% of the variance in Fig. 8.

Based on n_components=10 for Dataset 1 and n_components=5 for Dataset 2, I first run PCA on them, and then followed by k-Means and EM. Results of evaluate_kmeans() and evalue_EM() are tabulated in Tables 3 and 4:

K-Means

	Training Time	F-1	Accuracy	Precision	AUC	Recall
Wine Quality	0.3303 s	0.6880	0.7011	0.7784	0.7074	0.6164
Star3642	0.5792 s	0.8813	0.8786	0.8623	0.8786	0.9012

Table 3 - PCA: Training time, F-1, Accuracy, Precision, AUC and Recall - k-Means

EM

	Training Time	F-1	Accuracy	Precision	AUC	Recall
Wine Quality	0.0480	0.6745	0.6698	0.7132	0.6720	0.6398
Star3642	0.0614 s	0.7854	0.7946	0.8222	0.7946	0.7518

Table 4 - PCA: Training time, F-1, Accuracy, Precision, AUC and Recall - EM

From Tables 3 and 4 above, for both datasets, there appears to be a decline in the F-1, Accuracy, Precision, AUC and recall values. This is a little surprising to me as I expected there to be an improvement in all these areas under EM.

Figs. 9 - 12 show F-1 scores for kMeans and EM improve with increasing number of clusters and distributions:

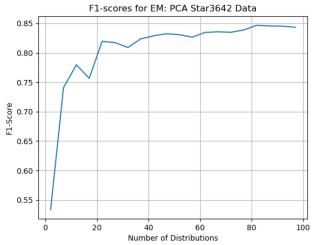


Fig. 9 -- F1-scores for kMeans: PCA Wine Quality

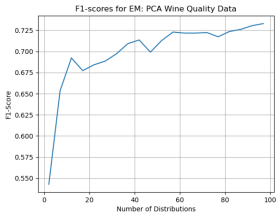


Fig. 10 -- F1-scores for EM: PCA Wine Quality

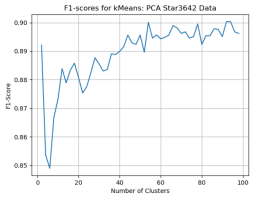


Fig. 11 - F-1 scores for KMeans: PCA Star3642

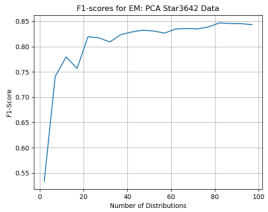


Fig. 12 - F-1 scores for EM: PCA Star3642

CONCLUSION

Comparing the results of Tables 3 and 4 with Tables 1 and 2 above, I see that for k-Means after applying PCA, Wine Quality's F-1, Accuracy, and Recall seem to have declined. For the Star3642 dataset, under k-Means after applying PCA, the F-1, Accuracy and AUC have declined, while Recall has improved. Under EM after applying PCA, for the Wine Quality and Star3642 datasets, while the training times have shortened dramatically, the precision, F-1, Accuracy, AUC and Recall have also fallen.

Cluster accuracies (in terms of precision, F-1, and recall) and AUC have all declined for both datasets under k-Means and EM after applying PCA. Below are my hypotheses of why this is happening.

Dimensionality reduction reduces the number of attributes while keeping as much as possible the variation in the original dataset. However, in datasets such as my Wine Quality and Star3642 dataset, where the number of attributes is already comparatively small, those attributes that have been removed may still have contributed in a meaningful way to the target label. As such, we may have lost some percentage of variability in the original data,

In PCA, reconstruction error or loss is the sum of eigenvalues of the ignored subspace. Let's say, there are 10 dimensional samples (or data points) , and we select 4 principal components, our principal subspace has 4 dimensions and corresponds to 4 eigenvalues and thus 4 respective vectors. In this case, the reconstruction error would be the sum of all the 6 remaining eigenvalues in the ignored subspace. [4]

Minimizing the reconstruction error translates to having to minimize the contribution of the ignored eigenvalues. This often depends on the distribution of the data and how many components we are selecting. I hypothesize that under EM after applying PCA,there is a larger reconstruction error i.e. a larger average squared distance between the original data and the respective projections onto the principal subspace. This could explain why cluster accuracies declined under EM after applying PCA.

3.2 Independent Component Analysis (ICA)

Independent Component Analysis (ICA) allows a mixture of signals to be separated into their different sources. The ICA finds the independent components by maximizing the statistical independence of estimated components. For example, it could: (1) minimize the mutual information or (2) maximize the non-Gaussianity.

The average kurtosis vs independent components charts of Figs. 13 and 14 (below) would help in selecting the n_components. As is shown in Fig. 13, for Dataset 1, the maximum kurtosis is reached when independent components is 8. For Dataset 2, I will select n_components=5, as Dataset 2 has only 6 features.

[4] Meaning of "reconstruction error" on PCA

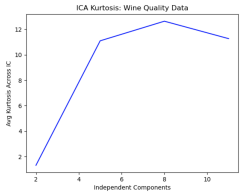


Fig. 13 - Dataset 1: Average Kurtosis vs independent components

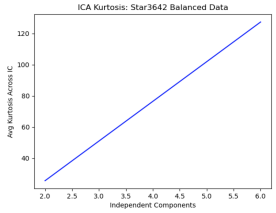


Fig. 14 - Dataset 2: Average Kurtosis vs independent components

Similar to the analyses of the other algorithms above, although I have run k-Means and EM after applying ICA and have myriads of Silhouette, Homogeneity, and F-1 charts, it would be more meaningful to look at the evaluation metrics which I have tabulated in Tables 5 and 6 below:

K-Means

	Training Time	F-1	Accuracy	Precision	AUC	Recall
Wine Quality	0.3150 s	0.7015	0.6923	0.7289	0.6935	0.6760
Star3642	0.5728 s	0.8806	0.8767	0.8540	0.8767	0.9088

Table 5 - ICA: Training time, F-1, Accuracy, Precision, AUC and Recall: k-Means - k-Means

EM

	Training Time	F-1	Accuracy	Precision	AUC	Recall
Wine Quality	0.0607 s	0.6160	0.6585	0.7725	0.6694	0.5123
Star3642	0.0805 s	0.8236	0.8094	0.7667	0.8094	0.8896

Table 6 - ICA: Training time, F-1, Accuracy, Precision, AUC and Recall: k-Means - EM

Figs. 15 - 18 show the F1-scores for k-Means and EM run after ICA has been applied on both datasets.

Fig. 15 -- F1-scores for k-Means: ICA Wine Quality

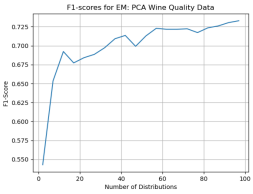


Fig. 16 - F1-scores for EM: ICA Wine Quality

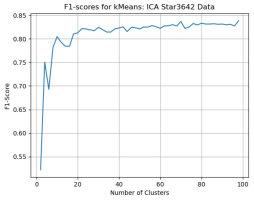


Fig. 17 - F1-scores for k-Means: ICA Star3642

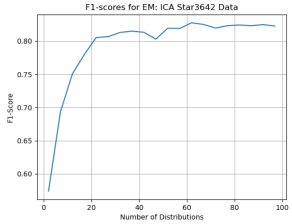


Fig. 18 - F1-scores for EM: ICA Star3642

CONCLUSION

Comparing Tables 5 and 6 with Tables 1 and 2, we see that under k-Means after applying ICA, Wine Quality's precision, accuracy, F-1 and recall all declined. For the Star3642 dataset, under k-Means after applying ICA, the precision, accuracy, F-1 did not improve either. Only the recall value has improved for the Star3642 dataset. Having high values for both precision and recall could be a good thing as this indicates that the ground-truth true positives and the model's true positives are pretty close. Under EM after applying ICA, both dataset's precision, accuracy, F-1 and AUC values declined. Only the recall value for the Star3642 dataset saw an increase of 2.7%. This decline in evaluation metrics could be due to the loss in variability of the original data as explained in **Section 3.1's Conclusion** above.

3.3 Randomized Projections (RCA)

I used Scikit-learn's SparseRandomProjection class for implementing this step. In Fig. 19 below, for Dataset 1, the mean reconstruction correlation is highest at 0.90 when the number of random components = 10. I chose n_components=10 for running RCA. For Dataset 2, Fig. 20 shows the mean reconstruction correlation of 0.90 is reached when the number of random components = 5.5. I used n_components=5 for running RCA on Dataset 2.

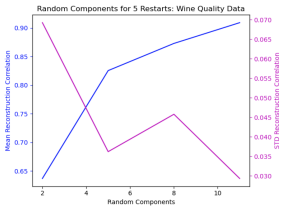


Fig. 19 - Dataset 1: Random Components for 5 restarts

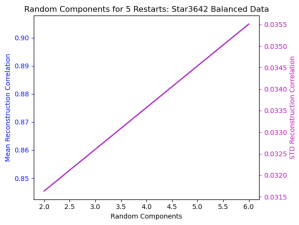


Fig. 20 - Dataset 2: Random Components for 5 restarts

Tables 7 and 8 below tabulate the results of running evaluate_kmeans() and evaluate_EM() after re-clustering algorithms have been run on both datasets.

K-Means

	Training Time	F-1	Accuracy	Precision	AUC	Recall
Wine Quality	0.3509 s	0.6491	0.6673	0.7443	0.6741	0.5754
Star3642	0.4725 s	0.8633	0.8677	0.8927	0.8677	0.8358

Table 7 - RCA: Training time, F-1, Accuracy, Precision, AUC and Recall: k-Means

EM

	Training Time	F-1	Accuracy	Precision	AUC	Recall
Wine Quality	0.0821 s	0.6675	0.6760	0.7397	0.6811	0.6082
Star3642	0.0576 s	0.8347	0.8163	0.7588	0.8163	0.9275

Table 8 - RCA: Training time, F-1, Accuracy, Precision, AUC and Recall: EM

Fig. 21 - F1-scores for k-Means: RCA Wine Quality

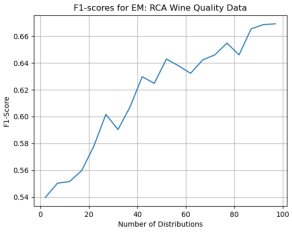


Fig. 22 - F1-scores for EM: RCA Wine Quality

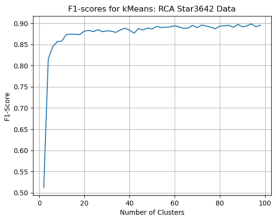


Fig. 23 - F1-scores for k-Means: RCA Star3642

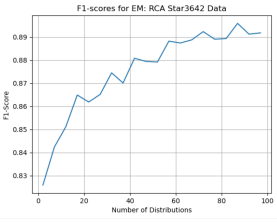


Fig. 24 - F1-scores for EM: RCA Sta3642

CONCLUSION

Comparing Tables 7 and 8 with Tables 1 and 2, we see that under k-Means after applying RCA, Wine Quality's F-1, accuracy, precision, recall and AUCl all declined. For the Star3642 dataset, except for precision, all the other metrics - accuracy, F-1, recall and AUC declined under k-Means after applying RCA. Under EM after applying RCA, Wine Quality's F-1, accuracy, precision, AUC and recall all declined. For the Star3642 dataset, except for recall, all the other metrics - accuracy, F-1, precision and AUC declined. This decline in evaluation metrics could be due to the loss in variability of the original data as explained in **Section 3.1's Conclusion** above.

3.4 Random Forest Classifier (algorithm of choice)

Using sklearn.ensemble's RandomForestClassifier and through run_RFC(), I singled out the **feature_importances_** attributes of my dataset, sorted in descending order the values of these feature importances, and removed any features whose cumulative sum is less than 0.95. This function returns the important features (in descending order) and their corresponding columns.

Tables 9 and 10 below tabulate the results of running evaluate_kmeans() and evaluate_EM() after re-clustering algorithms have been run on both datasets.

K-Means

	Training Time	F-1	Accuracy	Precision	AUC	Recall
Wine Quality	0.2411 s	0.7181	0.6229	0.5981	0.6023	0.8982
Star3642	1.2665 s	0.8236	0.8218	0.8154	0.8218	0.8320

Table 9 - RFC: Training time, F-1, Accuracy, Precision, AUC and Recall: k-Means

EM

	Training Time	F-1	Accuracy	Precision	AUC	Recall
Wine Quality	0.6577 s	0.7098	0.7086	0.7590	0.7117	0.6667
Star3642	0.9436 s	0.8783	0.8729	0.8422	0.8729	0.9176

Table 10 - RFC: Training time, F-1, Accuracy, Precision, AUC and Recall: EM

Fig. 25 - F1-scores for k-Means: RFC Wine Quality

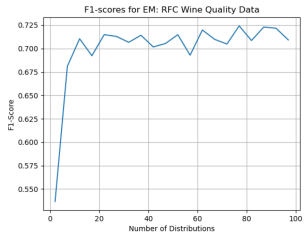


Fig. 26 - F1-scores for EM: RFC Wine Quality

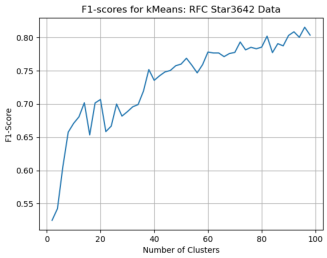


Fig. 27 - F1-scores for k-Means: RFC Star3642

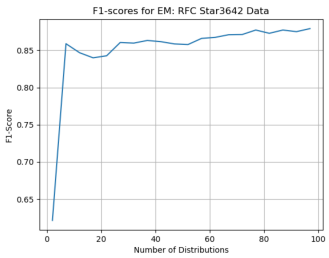


Fig. 28 - F1-scores for EM: RFC Star3642

CONCLUSION

Comparing Tables 9 and 10 with Tables 1 and 2, we see that under k-Means after applying RFC, Wine Quality's F-1, accuracy, precision, AUC and recall all declined. For the Star3642 dataset, under k-Means after applying RFC, all these metrics declined as well. What's alarming is that: for Wine Quality dataset, under k-Means after applying RFC, the AUC declined by as much as 17.5%. This means the model's ability to distinguish between the 2 classes of wines could be worsening. Under EM after applying RFC, Wine Quality's F-1, accuracy, precision and recall all declined. Similarly, across the board, we see a decline for Star3642 dataset under EM after applying RFC.

As we can see, running clustering algorithms, whether it be k-Means or EM, after applying RFC appears to cause BOTH datasets' accuracy, F-1, precision, recall and AUC to decline. It could be that for random forest classifier, in trying to achieve a balanced tree, the overall error rate [5]increased. This higher overall error rate thus contributes to the evaluation metrics (accuracy, F-1, precision, recall and AUC) declining.

PARTS 4 and 5

4. Dimensionality Reduction (DR) and Neural Networks

[5] Refer to "Balancing Prediction Errors" in *Random Forests*

This section calls for running dimensionality reduction on one dataset and observing the behavior under neural networks. I have chosen Dataset 1 - Wine Quality since this is richer in features, and probably lends itself to more interesting observation of the effects of dimensionality reduction and re-clustering.

In this part of the experiment, before running dimensionality reduction algorithms (PCA, ICA, RCA, RFC) on Wine Quality dataset, I ran the original dataset through a neural network with hidden_layer_sizes=(5, 2). This gives me a baseline when I ran a final_classifier_evaluation(). Following this, I then applied the 4 dimensionality reduction algorithms to Wine Quality dataset, and re-run the neural network learner on this newly projected data.

Learning curves and charts of model training times were generated. However, what I find more illuminating are the evaluation metrics such as F-1, accuracy, precision, recall and AUC of the learners after dimensionality reduction (using each of the 4 algorithms) and running them through a neural network. I have tabulated the results in Table 11 below:

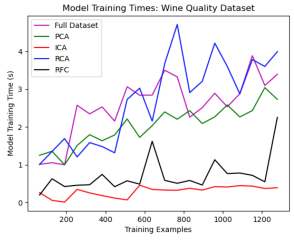
	Training Time	F-1	Accuracy	Precision	AUC	Recall
(Original) Wine Quality Dataset	2.07322 s	0.7896	0.7719	0.7874	0.7701	0.7919
PCA-reduced	0.25935 s	0.6749	0.5094	0.5094	0.500	1.000
ICA-reduced	0.25770 s	0.6965	0.5344	0.5344	0.500	1.000
RCA-reduced	4.38406 s	0.7449	0.7281	0.7299	0.7266	0.7605
RFC-reduced	0.45821 s	0.7173	0.6281	0.6138	0.6038	0.8629

Table 11 - RFC: Training time, F-1, Accuracy, Precision, AUC and Recall: DR -> NN

From Table 11, I observed that apart from RCA-reduced data, the training time for all the other dimensionality-reduced (PCA, ICA, RFC) datasets have training times which are at least 8 times shorter. The F-1, accuracy, and precision have fallen somewhat from the original Wine Quality dataset.

In the case of PCA and ICA, the recall score was 1.0. This translates to: for any sample, whether it be class 1 ('good' wines) or class 0 ('bad' wines), once that sample is fed into the learner, we get a prediction or output telling us that it's class 1 - a good wine. The corresponding precision value of 0.5094 and 0.5344 respectively for PCA- and ICA-reduced datasets, however, tells us that the ground-truth reporting of true good wines is only ~50%. These two metrics, **recall** and **precision**, taken into consideration concurrently means that the learner doesn't give a good prediction at all of good wines in the dataset.

Graphically, we could see the model's training time, prediction time and learning rates as shown below in Figs. 29 - 31:



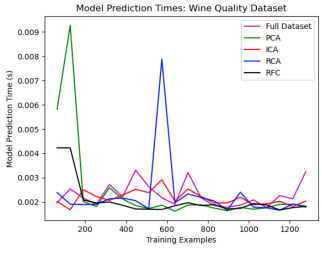


Fig. 29 - Model's training time **Fig. 30 - Model's prediction time** **Fig. 31 - Model's Accuracy**

Fig. 29 reaffirms what Table 11 has illustrated. ICA-reduced dataset takes the crown for the shortest training time, followed by RFC-reduced, and then PCA-reduced. RCA-reduced dataset, as shown in Fig. 29 takes the longest time to train. In Fig. 30, there appears to be a spike at 100 training examples for PCA-reduced, and another spike at 600

training examples for RCA-reduced. I looked at Fig. 31, and saw that at around 100 training examples, PCA-reduced dataset starts increasing in F-1 score, and at around 600 training examples, RCA-reduced dataset also increases in F-1 score. My hypothesis is that at 100 and 600 training examples, PCA- and RCA-reduced datasets are most accurate in their predictions. A longer time taken in prediction could explain why these PCA- and RCA-reduced datasets have higher F-1 scores. In Fig. 31, for most of the training examples from 200 -1200, the full dataset running through the neural network gives a higher F-1 score. All except for training examples between 900 - 1100, the PCA-reduced dataset shows higher F-1 score than the full dataset.

5. Applying Clustering Algorithms to DR-datasets and Running Through Neural Networks

This part of the experiment calls for running clustering algorithms to dimensionally-reduced datasets and then running them through neural networks.

	Training Time	F-1	Accuracy	Precision	AUC	Recall
(Original) Wine Quality Dataset	1.59992 s	0.7213	0.6813	0.6633	0.6763	0.7904
PCA-reduced	0.42530 s	0.6965	0.5344	0.5344	0.500	1.000
ICA-reduced	0.49930 s	0.7018	0.5406	0.5406	0.500	1.000
RCA-reduced	1.99284 s	0.6839	0.6562	0.6879	0.6538	0.6800
RFC-reduced	1.65957 s	0.6978	0.65156	0.5844	0.6092	0.8659

Table 12 - RFC: Training time, F-1, Accuracy, Precision, AUC and Recall: DR -> Clustering -> NN

From Table 12 above, I observed that except for RCA- and RFC-reduced datasets which take a much longer training time, PCA- and ICA-reduced datasets take a much shorter time to train. Looking at the F-1, accuracy, precision, recall and AUC scores of these datasets reconstructed by DR-algorithms, they struck me as falling below that of the original wine quality dataset. In the case of PCA- and ICA-reduced datasets, I again saw that the recall is 1.000. This means the clustering accuracy (i.e. clustering into class 1 - 'good' wines and class 0 - 'bad' wines) for Wine Quality dataset is very high if one were to use PCA- or ICA- reduced datasets. However, that may not necessarily be a good sign as the precision showed only around 53 - 54% for those PCA- and ICA-reduced datasets.

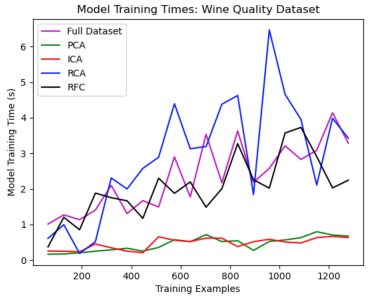


Fig. 32 - Model's training time

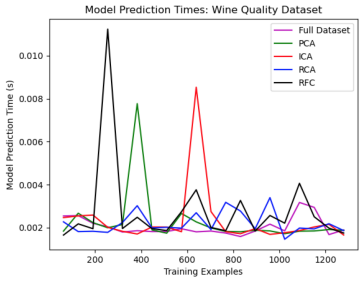


Fig. 33 - Model's prediction time

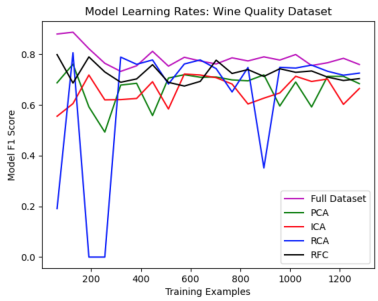


Fig. 34 - Model's Accuracy

Fig. 32 above affirms that PCA- and ICA-reduced datasets take the shortest time to train, while RCA-reduced dataset takes the longest time. In Fig. 33, there appears to be a spike at around 250 training examples for RFC-reduced datasets; a spike at 400 training examples for PCA-reduced datasets and a spike at 650 training examples for ICA-reduced datasets. We can see that in Fig. 16, correspondingly at 250 training examples, RCA-reduced dataset starts its ascent in F-1 score. This could explain why we would see a spike at 250 training examples for RFC-reduced datasets in Fig. 33. However, in Fig. 34, at 400 training examples, PCA-reduced dataset actually shows a decline in F-1 score. At 650 training examples, ICA-reduced dataset also shows a gradual decline in F-1 score. This behavior or PCA- and ICA-reduced datasets in Fig. 16 does not quite align with the spikes for these 2 datasets in Fig. 33. Currently, I do not have a good explanation of why this is the case.