Indonesian Journal of Electrical Engineering and Computer Science
Vol. 33, No. 1, January 2024, pp. 463∼475
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v33.i1.pp463-475

A hybrid model for data visualization using linear algebra methods and machine learning algorithm

Mohsin Ali[1], Jitendra Choudhary[1], Tanmay Kasbe[2]
[1]Department of Computer Science, Medi-Caps University, Indore, India
[2]Department of Computer Science, Shri Vaishnav Vidyapeeth Vishwavidyalaya, Indore, India

Article Info

Article history:

Received Nov 1, 2023
Revised Nov 12, 2023
Accepted Nov 14, 2023

Keywords:

Data visualization
Drosophila melanogaster
Principal component analysis
QR decomposition
t-SNE

ABSTRACT

The t-distributed stochastic neighbor embedding (t-SNE) is a powerful technique for visualizing high-dimensional datasets. By reducing the dimensionality of the data, t-SNE transforms it into a format that can be more easily understood and analyzed. The existing approach is to visualize high-dimensional data but not deeply visualize. This paper proposes a model that enhances visualization and improves the accuracy. The proposed model combines the non-linear embedding technique t-SNE, the linear dimensionality reduction method principal component analysis (PCA), and the QR decomposition algorithm for discovering eigenvalues and eigenvectors. In Addition, we quantitatively compare the proposed model QRPCA-t-SNE with PCA-t-SNE using the following criteria: data visualization with different perplexity and different principal components, confusion matrix, model score, mean square error (MSE), training, testing accuracy, receiver operating characteristic curve (ROC) score, and AUC score.

Corresponding Author:

Mohsin Ali
Department of Computer Science, Medi-Caps University
A.B Road Rao, Indore, India
Email: coolbuddy.next.door@gmail.com

1. INTRODUCTION

In recent years, there has been an exponential increase in the amount of digital information being generated across various fields, which has led to a significant surge in the size, complexity, diversity, and dimensions of data [1], which has given rise to a new type of data known as high dimensional data (HDD) [2], [3] HDD has been widely utilized across various industries, including healthcare, the Internet, education, commerce, and social networking [4], to name a few. The ever-increasing availability of new high-dimensional data can take on various formats, such as text [5], digital images [6], speech signals [7], and videos [8], among others. As such, developing new tools and techniques to manage and analyze such data effectively has become increasingly important to extract meaningful insights and drive innovation.

Regarding machine learning (ML) models, dealing with high-dimensional data can present many difficulties for precise classification visualization [9]. Due to the high computational complexity, learning in high-dimensional spaces or with many features can be challenging. The curse of dimensionality is used to characterize the complications of working with large datasets. Due to sparsity, increased complexity, the need for more data, confusion in distance measurements, and irrelevant features make data analysis and modeling harder. Including more data for each additional dimension [10] can be beneficial in avoiding this problem and improving accuracy. Dimensionality reduction (DR) is a data analysis technique that involves reducing the number of dimensions in a dataset while maintaining the structural integrity of the data. This technique has been widely used in social sciences and bioinformatics for visual data analysis [11], [12]-[15]. Among the most popular DR techniques used in the past are principal component analysis (PCA) and multidimensional scaling (MDS). However, in recent years, there has been a proliferation of new DR methods, including locally linear embedding (LLE), Isomap, maximum variance unfolding (MVU), laplacian eigenmaps, neighborhood retrieval visualizer, maximum entropy unfolding, t-distributed stochastic neighbor embedding (t-SNE), and others [16]-[28]. These new techniques have expanded the possibilities for data dimensionality reduction, giving researchers and analysts more options to optimize their data analysis processes.

t-SNE [16] is an algorithm used to reduce the dimension of high-dimensional data. It minimizes the difference between two probability distributions. The first distribution measures the similarity between the initial data objects, while the second measures the similarity between the corresponding embedding nodes. Visualizing high-dimensional data in a low-dimensional space makes interpreting and analyzing the data simpler.

t-SNE, or t-distributed stochastic neighbor embedding, has a time complexity of N^2, where N is the number of data points. However, Maaten [17] proposed a barnes-Hut-SNE algorithm optimization over this algorithm. This algorithm employs an octree structure to organize particles based on their proximity, thereby reducing the distance calculations required and conserving computational resources. The barnes-Hut-SNE algorithm has a time complexity of $O(N \log N)$, a significant advance over the time complexity of the original t-SNE algorithm. The barnes-Hut-SNE algorithm is a more practicable and efficient method for conducting t-SNE analysis. This paper tackled the following two questions that have not been attempted to t-SNE. (1) Would combining the linear dimensionality reduction method and eigenvalue, eigenvector method improve the visualization of t-SNE? (2) Does QRPCA- t-SNE perform well with various estimation parameters?

In response to the first query, it is advisable to use PCA as a pre-processing step before t-SNE because it can reduce computational cost and data noise. Specifically, PCA helps remove redundant features, improving the efficacy of t-SNE by reducing the data dimensionality. We use the QR algorithm to discover stable eigenvalue and eigenvector.

To resolve the second question regarding model estimation, we test our proposed model on drosophila melanogaster after the spaceflight dataset. After applying the model to the drosophila dataset, the prediction result is superior to the convolutional approach PCA-t-SNE on multiple parameters, including the confusion matrix, model score, mean square error (MSE), training, testing accuracy, receiver operating characteristic curve (ROC) score, area under the curve (AUC) score. In addition, we present a model visualization based on perplexity and K distinct principal components.

The remaining parts of the paper are partitioned into their respective divisions. In the 2 section, a brief description of the experimental results, input from one component to another component, as well as the setup that was used in the study is presented. In the 3 sections, a more in-depth analysis of the experiment as well as the results of the research. The study's findings are summarized and discussed in section 4, along with viable directions for further investigation.

2. METHOD

The goal of this work is to increase the accuracy and visualization of t-SNE. The steps we will take to complete this research are described in this part and are as follows. The dataset is covered in the first section, and combining the QR decomposition algorithm and PCA are covered in the second section. The binding procedure between the previous output, t-SNE and accuracy parameters are covered in the third section. Finally, investigation tool is covered in fourth section.

2.1. Dataset description

To test the efficacy of our proposed model, we selected genes from unsupervised learning datasets in a random manner. For testing, we utilized the NASA open science data repository (OSDR) dataset of drosophila gene expression levels, made available by the NASA OSDR program open science for life in space. This dataset consisted of 15,997 rows of data and 112 columns of features for testing and evaluating the effectiveness of our proposed model. By using this dataset, we were able to conduct a thorough and detailed assessment of our model's performance and obtain a more accurate understanding of its capabilities.

2.2. Proposed model component

Figure 1 depicts the QRPCA-t-SNE model, which utilizes a comprehensive machine-learning framework that analyzes datasets, identifies significant features, and develops accurate classifiers. It involves several components, including the QR decomposition algorithm, PCA, and t-SNE. The datasets undergo dimensionality reduction and standardization before being processed by the PCA and t-SNE algorithms. The resulting output matrix is then used to train a classifier, which is tested on a separate test set to evaluate its accuracy using precision, recall, and F1-score.

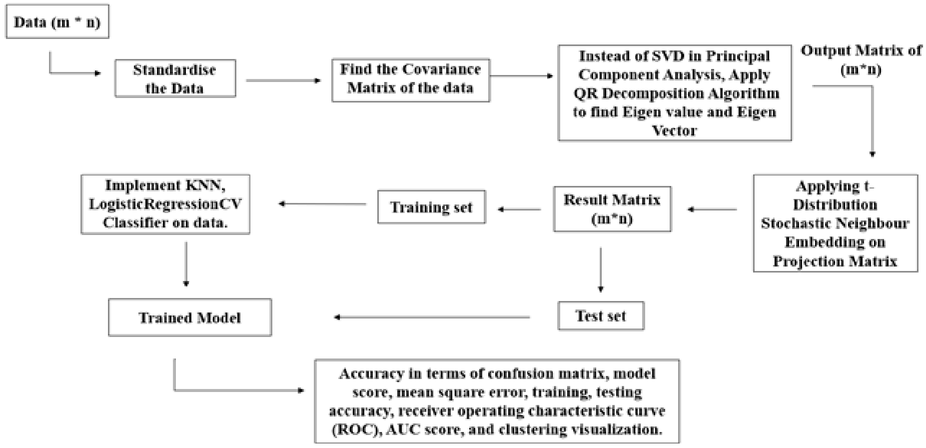

Figure 1. QRPCA-t-SNE (Self made)

2.3. Combining the QR decomposition algorithm and PCA

We standardize the data, which refers to a pre-processing step in which the features or variables of a dataset are transformed to have zero mean and unit variance. $Z:=\frac{X-\mu}{\sigma}\sim\mathcal{N}(0,1)$. After pre-processing the data, the next step is to find the covariance matrix of the data Z. Furthermore, we apply PCA, which is often used as a pre-processing step before t-SNE because it can help to reduce the computational cost and noise in the data. Internally, PCA uses eigenvalue and eigenvector for further process, but we explicitly supply eigenvalue and eigenvector via QR decomposition algorithm to find eigenvalue and eigenvector. QR algorithm [29] steps are as:
start with an initial square matrix Z,

$$Z=\left[z_1\,|z_2|\cdots|z_n\right]\ . \tag{1}$$

find the orthogonal projection of the first column vector and the second column z_2 is subtracted by the previous projection on the column vector:

$$z_1.v_1=z_1,e_1=\frac{v_1}{v_1} \tag{2}$$

$$v_2=z_2-\text{proj}_{v_1}\left(z_2\right)=z_2-\left(z_2\cdot e_1\right)e_1,e_2=\frac{v_2}{v_2}.$$

this process continues up to the n column vectors, where each incremental step $k+1$ is computed as,

$$v_{k+1}=z_{k+1}-\left(z_{k+1}\cdot e_1\right)e_1-\cdots-\left(z_{k+1}\cdot e_k\right)e_k,e_{k+1}=\frac{u_{k+1}}{u_{k+1}}$$

this $\|\cdot\|$ is the L_2 norm, which is defined by:

$$\sqrt{\sum_{j=1}^{m}v_k^2} \tag{3}$$

thus, the matrix A can be factorized into the QR matrix as the following:

$$Z=\left[z_1\,|z_2|\cdots|z_n\right]=\left[e_1\,|e_2|\cdots|e_n\right]$$

$$\begin{bmatrix} z_1{\cdot}e_1 & z_2{\cdot}e_1 & \cdots & z_n{\cdot}e_1 \\ 0 & z_2{\cdot}e_2 & \cdots & z_n{\cdot}e_2 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & a_n{\cdot}e_n \end{bmatrix} = QR$$

2.4. PCA output, t-SNE and model accuracy parameters

Following the discovery of the eigenvalue and the eigenvector, the values obtained are then supplied to the projection phase of the PCA to get output. The output is carried forward and delivered to t-SNE to visualize the data; in addition, this is employed for hypothesis testing. In addition, the suggested model accuracy checks through several essential criteria, including training accuracy, testing accuracy, MSE, confusion matrix, and operational characteristic curve, all of which are further detailed in the result and discussion phase.

2.5. Investigation tool

This investigation was conducted using a hybrid model that employed machine learning algorithms. The system was run on a desktop computer with a powerful Intel Core i7 processor clocked at 2.60 GHz and a substantial 32 GB RAM. The computer was installed with the latest Windows 11 operating system version and was equipped with an advanced NVIDIA RTX 3080 Graphics card. The machine learning algorithms were implemented using two popular libraries: tensor-flow and sci-kit-learn. The hybrid model was designed to harness the computer's processing power and the libraries' advanced algorithms to achieve optimal results.

3. RESULTS AND DISCUSSION

In order to surpass the results of the previous approach, the proposed model underwent rigorous testing with various parameters. The model was also verified through hypothesis testing, ensuring its validity and reliability. As a result of this process, several crucial parameters such as training accuracy, testing accuracy, MSE, confusion matrix, and operating characteristic curve have seen significant improvement. This means that the model's overall performance and effectiveness have been greatly enhanced, providing more accurate and reliable results.

3.1. QRPCA-t-SNE parameter having perplexity =50 and principal component (N=5)

The following steps were applied to implement the QRPCA-t-SNE model. Figure 2 shows the dataset, considered an input in the whole analysis. Figure 3 explains the preprocessing step. Figure 4 illustrate the QR decomposition algorithm. Figure 5 exemplifies the two-dimensional t-SNE dense visualization of the drosophila dataset with setting t-SNE with perplexity =50 and n_component =2, yielding a better visualization. Figure 6 shows model score of QRPCA-t-SNE using logistic regression CV (Cs=[0.001, 0.01, 0.1], cv =3, max_iter =5000, multiclass='ovr'). Figure 7 depicts the proposed model MSE. Figure 8 depicts the model accuracy of QRPCA-t-SNE during the training and testing phase. Figure 9 shows the plot of the QRPCA-t-SNE ROC, where the blue dotted line shows the perfect classifier and the green line shows the random classifier. Figure 10 shows the AUC score of the proposed model. Figure 11 represents the confusion matrix of four different classes. Figure 12 depicts the accuracy of the confusion matrix in terms of precision-score, recall-score, F1-score, and accuracy score of the QRPCA- t-SNE proposed model.

```
from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive

[3] import pandas as pd
    path = "/content/drive/MyDrive/kaju/Celniker_Drosophila_Annotation_20120616_1428_allsamps_MEAN_gene_expression.csv"
    df_bonus = pd.read_csv(path)
```

Figure 2. Datasets input stage (Self made)

```
import pandas as pd
import numpy as np
import math
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.manifold import MDS
from sklearn.manifold import TSNE
from sklearn.cluster import KMeans
from yellowbrick.style.colors import resolve_colors
from sklearn.metrics import silhouette_score
from yellowbrick.cluster import SilhouetteVisualizer
from scipy.cluster.hierarchy import dendrogram, linkage, cophenet, fcluster
from scipy.spatial.distance import pdist
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression, LogisticRegressionCV
from __future__ import division
from itertools import chain, combinations
import warnings
from itertools import combinations_with_replacement as combinations_w_r
from distutils.version import LooseVersion
import numpy as np
from scipy import sparse
from scipy import stats
from scipy import optimize
from sklearn.utils import check_array

from sklearn.utils.extmath import row_norms
from sklearn.utils.extmath import _incremental_mean_and_var

from sklearn.utils.sparsefuncs_fast import (inplace_csr_row_normalize_l1,
                                            inplace_csr_row_normalize_l2)
from sklearn.utils.sparsefuncs import (inplace_column_scale,
                                       mean_variance_axis, incr_mean_variance_axis,
                                       min_max_axis)
from sklearn.utils.validation import (check_is_fitted, check_random_state,
                                      FLOAT_DTYPES)
def scale(X, axis=0, with_mean=True, with_std=True, copy=True):
```

Figure 3. Prepossessing stage (Self made)

```
import random
from sklearn import datasets

#finally
class PCA1:


    Z=0

    def fit_transform(self, X, n_components=2):
        # get number of samples and components
        self.n_samples = X.shape[0]
        self.n_components = n_components

        self.A=X
        # standardize data
        # calculate covariance matrix
        covariance_matrix = self.get_covariance_matrix()
        Z=covariance_matrix
        return Z

    def get_covariance_matrix(self, ddof=0):
        # calculate covariance matrix with standardized matrix A
        C = np.dot(self.A.T, self.A) / (self.n_samples - ddof)
        return C

    def find_eig_qr(self,A):
      pQ = np.eye(A.shape[0])
      X=A.copy()
      for i in range(100):
        Q,R = np.linalg.qr(X)
        pQ = pQ @ Q;
        X = R @ Q;
      return np.diag(X), pQ

    def get_eigenvectors(self,n_components):
      # sort eigenvalues descending and select columns based on n_components

      n_cols = np.argsort(e_v)[::-1][:n_components]
      selected_vectors = e_vec[:,n_cols]

      return selected_vectors

    def project_matrix(self,eigenvectors):
```

Figure 4. QR decomposition and PCA step (Self made)

```python
import time
perplexity = 50

X_transformed_tsne = TSNE(n_components=2, perplexity=perplexity).fit_transform(projected_matrix1)

plt.scatter(X_transformed_tsne[:,0], X_transformed_tsne[:,1])
plt.xlabel('t-SNE Component 1')
plt.ylabel('t-SNE Component 2')
plt.title('Drosophila, QR-PCA-t-SNE (perplexity={})'.format(perplexity), size=15)
plt.show()
```

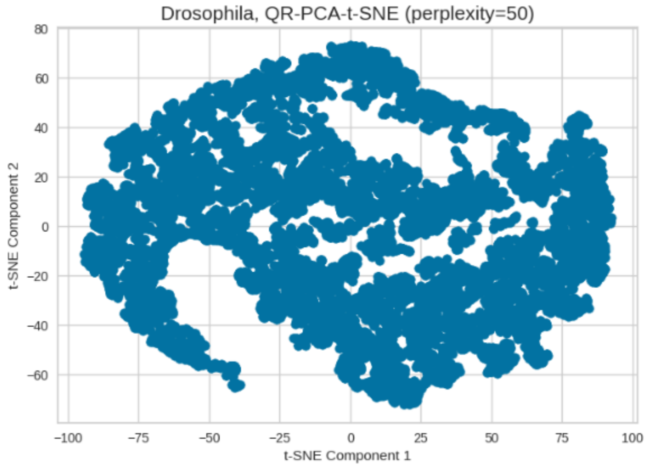

Figure 5. Demonstration of data visualization using a QRPCA-t-SNE (Self made)

```
LogisticRegressionCV
LogisticRegressionCV(Cs=[0.001, 0.01, 0.1], cv=3, max_iter=5000,
                     multi_class='ovr')
```

```python
score1 = model.score(X_test, y_test)
print(score1)
```

```
0.9142045454545454
```

Figure 6. Applying logistic regression CV on the proposed model (QRPCA-t-SNE) to calculate the model score (Self made)

```python
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test, y_pred)
```

```
1.3321969696969698
```

Figure 7. Illustrates the MSE of the proposed model (QRPCA-t-SNE) during the testing phase (Self made)

```python
print("Training Accuracy: {0:.3f}".format(train_accuracy))
print("Testing Accuracy: {0:.3f}".format(test_accuracy))
```

```
Training Accuracy: 0.905
Testing Accuracy: 0.914
```

Figure 8. Depicts the training and testing accuracy (QRPCA-t-SNE) of the proposed model (Self made)

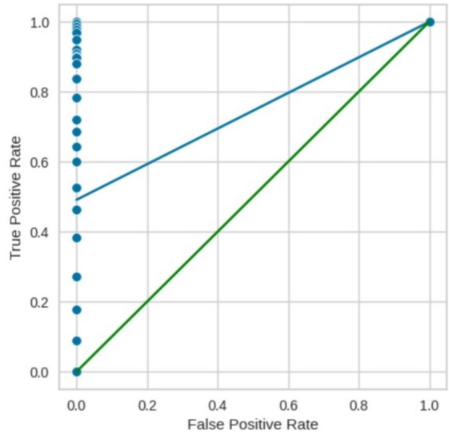

Figure 9. Utilizing the ROC to illustrate the true positive and false positive rates for the proposed model QRPCA-t-SNE (Self made)

```
from sklearn.metrics import roc_auc_score
# auc scores
auc_score1 = roc_auc_score(y_test, y_proba,multi_class='ovr')
print(auc_score1)
print(np.round(auc_score1,3))

0.9944062569291192
0.994
```

Figure 10. Displays the AUC score of the proposed model QRPCA-t-SNE (Self made)

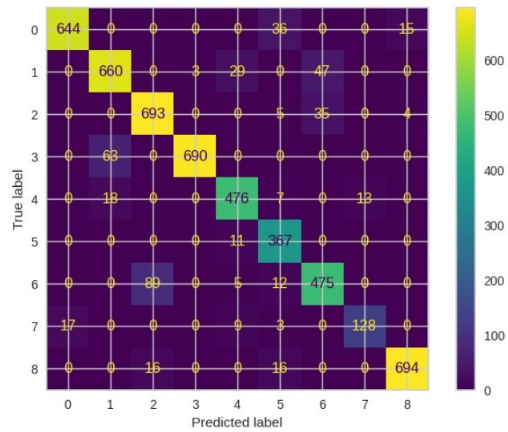

Figure 11. Demonstrates the confusion matrix for the proposed model QRPCA-t-SNE (Self made)

```
from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score
print('Precision: %.3f' % precision_score(y_test, y_pred,average='macro'))
print('Recall: %.3f' % recall_score(y_test, y_pred,average='macro'))
print('Accuracy: %.3f' % accuracy_score(y_test, y_pred))
print('F1 Score: %.3f' % f1_score(y_test, y_pred,average='macro'))

Precision: 0.909
Recall: 0.907
Accuracy: 0.914
F1 Score: 0.907
```

Figure 12. Represent the precision-score, recall-score, F1-score, accuracy-score of the proposed model QRPCA-t-SNE (Self made)

3.2. Compare the 3.1 results with PCA t-SNE on the same parameters perplexity =50 and principal component (N=5)

In the implementation of PCA-t-SNE, the following steps are taken. Figure 13 demonstrates the input and applies the PCA to it. Figure 14 illustrates the two-dimensional t-SNE visualization of the drosophila dataset with perplexity=50 and n_component=2. Figure 15 shows the model score of PCA-t-SNE using logistic regression CV (Cs =[0.001, 0.01, 0.1], cv =3, max_iter =5000, multiclass =‘ovr’). Figure 16 depicts the model PCA-t-SNE MSE. Figure 17 shows the model accuracy of PCA-t-SNE during the training and testing phase. Figure 18 shows the plot of the PCA-t-SNE ROC, where the blue dotted line shows the perfect classifier and the green line shows the random classifier. Figure 19 shows the AUC score of the PCA-t-SNE. Figure 20 represents the confusion matrix of four different classes. Figure 21 depicts the accuracy of the confusion matrix in terms of precision-score, recall-score, F1-score, and accuracy-score of the PCA-t-SNE.

```
[ ] df=np.asarray(df_bonus)
    print(df.shape)
    jason=df

    (15997, 112)

    from sklearn.decomposition import PCA

    # Assuming you want to keep 5 principal components
    pca = PCA(n_components=5)

[ ]
    data_pca1 = pca.fit_transform(jason)
```

Figure 13. Input and PCA stage (Self made)

```
import time
perplexity = 50

X_transformed_tsne1 = TSNE(n_components=2, perplexity=perplexity).fit_transform(data_pca1)

plt.scatter(X_transformed_tsne1[:,0], X_transformed_tsne1[:,1])
plt.xlabel('t-SNE Component 1')
plt.ylabel('t-SNE Component 2')
plt.title('Drosophila, PCA t-SNE (perplexity={})'.format(perplexity), size=15)

plt.show()
```

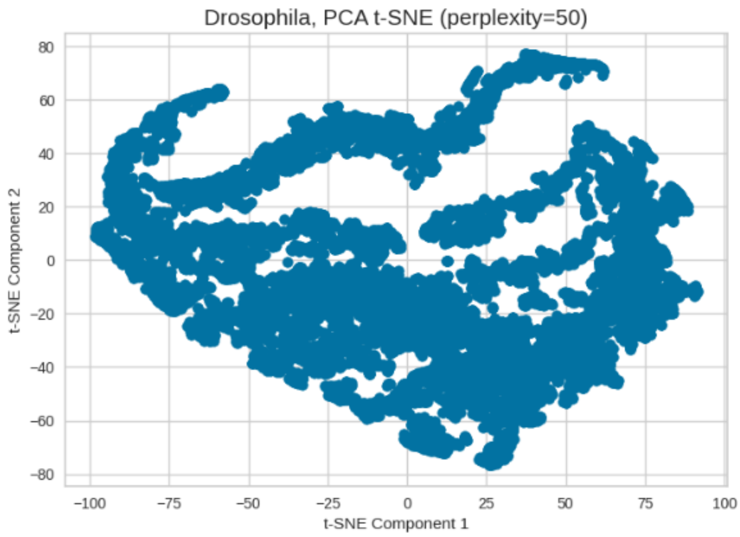

Figure 14. Demonstration of data visualization using PCA-t-SNE (Self made)

```
LogisticRegressionCV
LogisticRegressionCV(Cs=[0.001, 0.01, 0.1], cv=3, max_iter=5000,
                     multi_class='ovr')
```

```
score1 = model.score(X_test1, y_test1)
print(score1)
```

```
0.8784090909090909
```

Figure 15. Applying logistic regression CV on the PCA-t-SNE to calculate the model score (Self made)

```
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test1, y_pred1)
```

```
1.446590909090909
```

Figure 16. Illustrates the MSE of the PCA-t-SNE during the testing phase (Self made)

```
test_accuracy12 = accuracy_score(y_test1, y_test_pred1)
print("Training Accuracy: {0:.3f}".format(train_accuracy12))
print("Testing Accuracy: {0:.3f}".format(test_accuracy12))
```

```
Training Accuracy: 0.873
Testing Accuracy: 0.878
```

Figure 17. Depicts the training and testing accuracy on PCA-t-SNE (Self made)

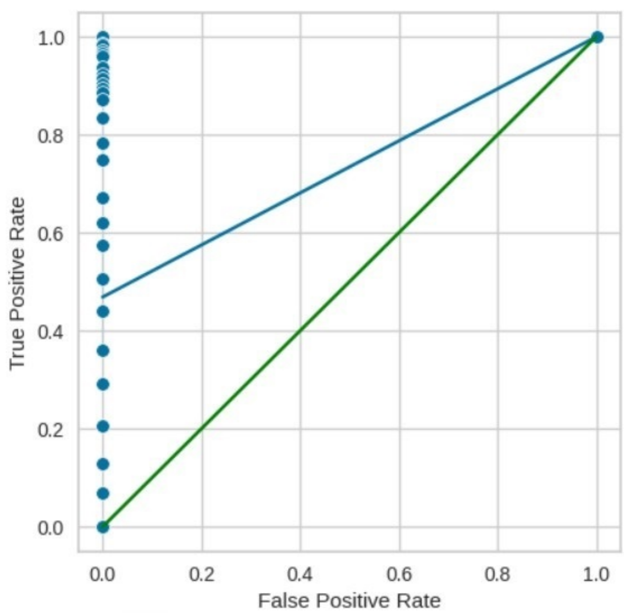

Figure 18. Utilizing the ROC to illustrate the true positive and False positive rates for the PCA-t-SNE (Self made)

```
from sklearn.metrics import roc_auc_score
# auc scores
auc_score1 = roc_auc_score(y_test1, y_proba1,multi_class='ovr')
print(auc_score1)
print(np.round(auc_score1,2))
```

```
0.9916692493206161
```

Figure 19. Displays the AUC score of the PCA-t-SNE (Self made)

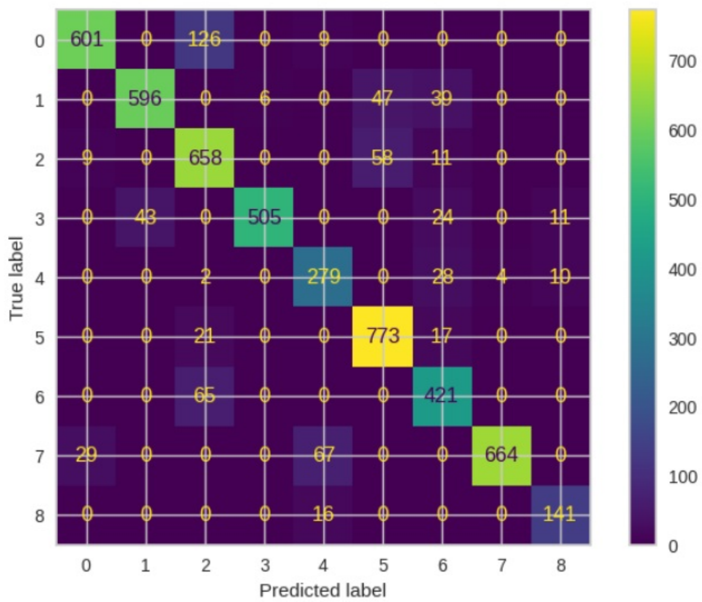

Figure 20. Demonstrates the confusion matrix for the PCA-t-SNE (Self made)

```
[ ] from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score
    print('Precision: %.3f' % precision_score(y_test1, y_pred1,average='macro'))
    print('Recall: %.3f' % recall_score(y_test1, y_pred1,average='macro'))
    print('Accuracy: %.3f' % accuracy_score(y_test1, y_pred1))
    print('F1 Score: %.3f' % f1_score(y_test1, y_pred1,average='macro'))
```

```
Precision: 0.877
Recall: 0.878
Accuracy: 0.878
F1 Score: 0.874
```

Figure 21. Represent the precision-score, recall-score, F1-score, and accuracy-score of PCA-t-SNE (Self made)

3.3. Model verification via hypothesis testing

In order to evaluate the superiority of this strategy in comparison to the usual approach, hypothesis testing is employed. In Figure 22, variables P and P1 are utilized to store the data obtained by QRPCA t-SNE and PCA-t-SNE, respectively;

H0: $p < 0.1$.
H1: $p > 0.1$.

Setting a 90% confidence interval having α=0.1 and applying two sample t-test shows the result's outcome in Figure 22. After conducting a two sample t-test, the resulting p-values are P=0.07 and P1=0.93. The initial value (P) is below the significance level α, indicating superior performance compared to the convolution approach. Therefore, our model demonstrates improvement over the previous approach.

```
[ ]  t_stat, p_value = stats.ttest_ind(P, P1)

     # Print the results
     print("T-statistic:", t_stat)
     print("P-value:", p_value)

     # Set the significance level (alpha)
     alpha = 0.1

     # Compare p-value to alpha and make a decision
     if p_value[0] < alpha:
         print("Reject the null hypothesis: There is a significant difference between the groups.")
     else:
         print("Fail to reject the null hypothesis: There is no significant difference between the groups.")

     T-statistic: [ 1.8091972 -0.0763685]
     P-value: [0.07042976 0.93912643]
     Reject the null hypothesis: There is a significant difference between the groups.
```

Figure 22. Hypothesis testing (Self made)

3.4. Comparison table: (QRPCA-t-SNE) vs (PCA-t-SNE)

Table 1 illustrates a comprehensive comparison between the outcomes of the proposed model, namely QRPCA-t-SNE, and the traditional approach, PCA-t-SNE, across various parameters. These parameters include the confusion matrix, model score, MSE, training and testing accuracy, ROC score, and AUC score. These results were derived from implementing our proposed model and the conventional approach. Row 1 proposed model has achieved a lucrative outcome, surpassing the performance of row 2, despite both rows having similar perplexity values of 50 and principal component values of 5.

Table 1. QRPCA-t-SNE vs PCA-t-SNE (Self made)

Model description	Model score	MSE	Training accuracy	Testing accuracy	AUC score	Precision	Recall	Accuracy	F1-score
Proposed model (QRPCA-t-SNE) perplexity =50 and principal component=5	0.914	1.332	0.905	0.914	0.994	0.909	0.907	0.914	0.907
PCA-t-SNE Perplexity=50 and principal component=5	0.878	1.446	0.873	0.878	0.991	0.877	0.878	0.878	0.874

4. CONCLUSION

Non-linear embedding techniques transform high-dimensional data into a lower-dimensional representation while preserving the underlying. Non-linear relationships between data points. To examine data visualization deeply, we proposed a model that combines the QR decomposition algorithm, PCA, and t-SNE, which we called QRPCA-t-SNE. We evaluated our model by applying the drosophila dataset with perplexities and principal components. Also, we have achieved our objective regarding data visualization and accuracy like the confusion matrix, model score, mean square error, training, testing accuracy, ROC score, and AUC score. Our model QRPCA-t-SNE contributes to advancing genomics, cancer research, and society by incorporating this model.

REFERENCES

[1] L. Gao, J. Song, X. Liu, J. Shao, J. Liu, and J. Shao, "Learning in high-dimensional multimedia data: the state of the art," *Multimedia Systems*, vol. 23, no. 3, pp. 303–313, 2017, doi: 10.1007/s00530-015-0494-1.

[2] D. Amaratunga, J. Cabrera, and Y. S. Lee, "Resampling-based similarity measures for high-dimensional data," *Journal of Computational Biology*, vol. 22, no. 1, pp. 54–62, Jan. 2015, doi: 10.1089/cmb.2014.0195.

[3] T. Ortner, P. Filzmoser, M. Rohm, C. Breiteneder, and S. Brodinova, "Guided projections for analyzing the structure of high-dimensional data," *Journal of Computational and Graphical Statistics*, vol. 27, no. 4, pp. 750–762, 2018, doi: 10.1080/10618600.2018.1459304.

[4] S. Bahrami and M. Shamsi, "A non-parametric approach for the activation detection of block design fMRI simulated data using self-organizing maps and support vector machine," *Journal of Medical Signals and Sensors*, vol. 7, no. 3, pp. 153–162, 2017, doi: 10.4103/jmss.JMSS-2-17.

[5] B. Tang, M. I. Heywood, and M. Shepherd, "Comparing and combining dimension reduction techniques for efficient text clustering," *Siam*, pp. 1–10, 2005.

[6] Y. Tang and R. Rose, "A study of using locality preserving projections for feature extraction in speech recognition," *in ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing-Proceedings*, Mar. 2008, pp. 1569–1572, doi: 10.1109/ICASSP.2008.4517923.

[7] C. D. Chang, C. C. Wang, and B. C. Jiang, "Singular value decomposition based feature extraction technique for physiological signal analysis," *Journal of Medical Systems*, vol. 36, no. 3, pp. 1769–1777, Jun. 2012, doi: 10.1007/s10916-010-9636-3.

[8] S. C. Ng, "Principal component analysis to reduce dimension on digital image," Procedia Computer Science, vol. 111, pp. 113–119, 2017, doi: 10.1016/j.procs.2017.06.017.

[9] P. Naik *et al.*, "Challenges and opportunities in high-dimensional choice data analyses," *Marketing Letters*, vol. 19, no. 3–4, pp. 201–213, Dec. 2008, doi: 10.1007/s11002-008-9036-3.

[10] W. K. Vong, A. T. Hendrickson, D. J. Navarro, and A. Perfors, "Do additional features help or hurt category learning? The curse of dimensionality in human learners," *Cognitive Science*, vol. 43, no. 3, Mar. 2019, doi: 10.1111/cogs.12724.

[11] T. Kohonen, "The self-organizing map, a possible model of brain maps," *Medical and Biological Engineering and Computing*, vol. 34, no. SUPPL. 1, pp. 5–8, 1996, doi: 10.1068/v970002.

[12] P. Horst, "Theory and methods of scaling . Warren S. Torgerson. Wiley, New York; Chapman and Hall, London, 1958. xiii + 460 pp. Illus. 9.50.," *Science*, vol. 129, no. 3348, pp. 560–560, Feb. 1959, doi: 10.1126/science.129.3348.560.a.

[13] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, Dec. 2000, doi: 10.1126/science.290.5500.2323.

[14] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, Dec. 2000, doi: 10.1126/science.290.5500.2319.

[15] K. Q. Weinberger and L. K. Saul, "An introduction to nonlinear dimensionality reduction by maximum variance unfolding," *Proceedings of the National Conference on Artificial Intelligence*, vol. 2, pp. 1683–1686, 2006.

[16] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of machine learning research*, vol. 9, pp. 2579–2605, 2008.

[17] L. van der Maaten, "Barnes-Hut-SNE," *arXiv*, 2013, doi: 10.48550/arXiv.1301.3342.

[18] J. Venna, S. Kaski, H. Aidos, K. Nybo, and J. Peltonen, "Information retrieval perspective to nonlinear dimensionality reduction for data visualization," *Journal of Machine Learning Research*, vol. 11, pp. 451–490, 2010, doi: 10.1145/1756006.1756019.

[19] F. Anowar, S. Sadaoui, and B. Selim, "Conceptual and empirical comparison of dimensionality reduction algorithms (PCA, KPCA, LDA, MDS, SVD, LLE, ISOMAP, LE, ICA, t-SNE)," *Computer Science Review*, vol. 40, p. 100378, May 2021, doi: 10.1016/j.cosrev.2021.100378.

[20] S. Ayesha, M. K. Hanif, and R. Talib, "Overview and comparative study of dimensionality reduction techniques for high dimensional data," *Information Fusion*, vol. 59, pp. 44–58, Jul. 2020, doi: 10.1016/j.inffus.2020.01.005.

[21] S. Sakib, M. A. Bakr Siddique, and M. A. Rahman, "Performance evaluation of t-SNE and MDS dimensionality reduction techniques with KNN, ENN and SVM classifiers," *in 2020 IEEE Region 10 Symposium, TENSYMP 2020*, 2020, pp. 815–818, doi: 10.1109/TENSYMP50017.2020.9230983.

[22] S. Velliangiri, S. Alagumuthukrishnan, and S. I. T. Joseph, "A review of dimensionality reduction techniques for efficient computation," *Procedia Computer Science*, vol. 165, pp. 104–111, 2019, doi: 10.1016/j.procs.2020.01.079.

[23] G. C. Linderman, M. Rachh, J. G. Hoskins, S. Steinerberger, and Y. Kluger, "Fast interpolation-based t-SNE for improved visualization of single-cell RNA-seq data," *Nature Methods*, vol. 16, no. 3, pp. 243–245, Mar. 2019, doi: 10.1038/s41592-018-0308-4.

[24] M. C. Su, Y. Z. Hsieh, C. H. Wang, and P. C. Wang, "A jacobian matrix-based learning machine and its applications in medical diagnosis," *IEEE Access*, vol. 5, pp. 20036–20045, 2017, doi: 10.1109/ACCESS.2017.2677458.

[25] D. Araújo, A. Dória Neto, A. Martins, and J. Melo, "Comparative study on dimension reduction techniques for cluster analysis of microarray data," *in Proceedings of the International Joint Conference on Neural Networks*, Jul. 2011, pp. 1835–1842, doi: 10.1109/IJCNN.2011.6033447.

[26] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 2, no. 1–3, pp. 37–52, Aug. 1987, doi: 10.1016/0169-7439(87)80084-9.

[27] M. Vlachos, C. Domeniconi, D. Gunopulos, G. Kollios, and N. Koudas, "Non-linear dimensionality reduction techniques for classification and visualization," *in in Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002, pp. 645–651, doi: 10.1145/775047.775143.

[28] H. Liu *et al.*, "Using t-distributed stochastic neighbor embedding (t-SNE) for cluster analysis and spatial zone delineation of groundwater geochemistry data," *Journal of Hydrology*, vol. 597, p. 126146, Jun. 2021, doi: 10.1016/j.jhydrol.2021.126146.

[29] "QR decomposition with the gram-schmidt procedure," RPubs by RStudio, https://rpubs.com/aaronsc32/qr-decomposition-gram-schmidt.

BIOGRAPHIES OF AUTHORS

Mohsin Ali is an Assistant Professor in Medi-Caps University in the Department of Computer Science. He has completed BCA (2016), and MCA (2018). Mr. Ali is a Research scholar at Medi-Caps University. His research areas are machine learning and data science. Mr. Ali has done Certain certifications from the Massachusetts Institute of Technology from edX like machine learning, introduction to probability, and fundamentals of statistics. He has 5 years of teaching experience. He can be contacted at email: coolbuddy.next.door@gmail.com.

Jitendra Choudhary was born in 1984 at Dewas, M.P. India. He received his B.Sc. degree in Computer Science from Holkar Science College, Indore in 2003, M.Sc. degree in Computer Science in 2005, and his M.Tech. degree in Computer Science (with Distinction) in 2010 from SCSIT, Devi Ahilya University Indore. He received his Ph.D. degree from Devi Ahilya University Indore in 2014. His areas are software engineering and software testing. His research area includes Extreme Programming and Software Maintenance. He has published more than 20 research paper in reputed international journal and conferences. He has received Gold Medal (AIR-1) in Software Engineering course run by IIT Kharagpur through Swayam-NPTEL. He is an Associate Professor and HOD, CS at Medi-Caps University, Indore, M.P., India. He has 17 years of teaching work experience at the UG and PG levels. He can be contacted at email: jitendra.scsit@gmail.com.

Tanmay Kasbe is an Associate Professor in the Shri Vaishnav Institute of Computer Application at Shri Vaishnav Vidyapeeth Vishwavidyalaya Indore. He has completed BCA, MCA and Ph.D. in Computer Science specialization in artificial intelligence and machine learning. He has more than 12 years of teaching and research experience. He has published many research papers in SCI/Scopus indexed international conferences and journals, and is associated with various international journals as a reviewer and editorial board member. His research areas include: artificial intelligence, machine learning, image processing, disease diagnosis, and wireless network. He has also granted patents in his area. He can be contacted at email: tanmaykasbe@svvv.edu.in.