DATA604 - Data Representation and Modeling
Final Project

Nischal Chandur

May 31, 2024

UID: 120116650

1 Original Data

The Fruit-360 Dataset is a comprehensive collection of images encompassing fruits, vegetables, and nuts.

- **Primary Link**: Kaggle - Fruit-360 Dataset
- **Secondary Link**: GitHub - Fruit-360 Dataset

Dataset Properties:

- **Total Images**: 90,483
- **Training Set Size**: 67,692 images
- **Test Set Size**: 22,688 images
- **Total Classes**: 131
- **Image Size**: 100×100 pixels

2 Pre-processing

For this project, I refined the Fruit-360 dataset to a more manageable and balanced subset.

1. **Class Reduction:** Manually trimmed the dataset from 131 to 16 visually distinct fruits.

Figure 1: Fruit classes selected for this project

2. **Sample Reduction:** Decreased the samples per class from over 900 to 625.

3. **Color Conversion:** Converted images from RGB to grayscale.

4. **Image Resizing:** Resized images from $100 \times 100 \rightarrow 32 \times 32$ pixels.

5. **Data Organization:** Structured the dataset into a three-dimensional matrix format resembling the USPS Handwritten dataset: $16 \times 625 \times 1024$.

    - **First Dimension:** Represents fruit classes.
    - **Second Dimension:** Indicates the number of samples per class.
    - **Third Dimension:** Denotes the pixel values of individual samples.

In the future, *Original Dataset* refers to this pre-processed dataset stored in matrix form.

3 Dimension Reduction Methods

In this section, we'll dive into different methods of Dimension Reduction to see how well they capture the essence of our dataset in fewer dimensions and to speculate on how a classification algorithm might perform

with the transformed data. Four distinct methods have been selected for this study: PCA, kPCA with Gaussian and polynomial kernels, t-SNE, and Autoencoder embedding. To perform these transformations, I utilized the Matlab Toolbox for Dimensionality Reduction, developed by Laurens van der Maaten.

3.1 Original Dataset

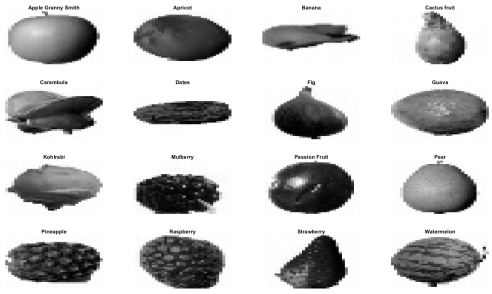

Figure 2: Samples of Original Dataset

3.2 Principal Component Analysis (PCA)

I applied PCA to the dataset, retaining all principal components sorted by the decreasing values of the eigenvalues. Here's how the samples from the previous image are represented after undergoing PCA transformation.

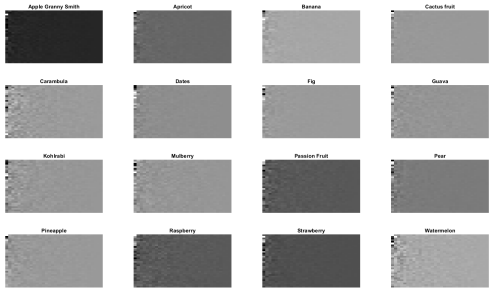

Figure 3: Data after PCA Transformation

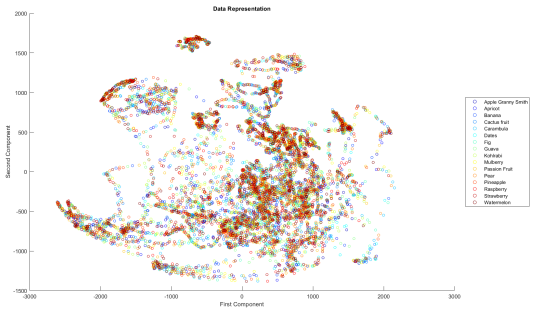

Figure 4: 2D Representation of transformed data

The 2D representation of the data following PCA reveals a well-spread distribution of samples across the space, spanning a range from approximately -3000 to 3000 on the x-axis and -1500 to 2000 on the y-axis. This suggests that in the lower dimension (2D), a significant amount of information from the higher dimension is preserved. Consequently, the classifier is expected to perform effectively on the data after PCA mapping.

3.3 Kernel Principal Component Analysis (kPCA)

I utilized kPCA on the dataset, preserving all principal components. This section explores the application of both Gaussian and polynomial kernels in the kPCA transformation process.

3.3.1 Gaussian kPCA

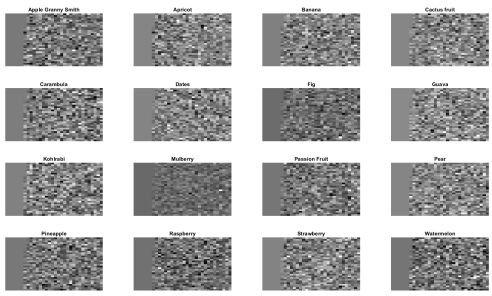

Figure 5: Data after Gaussian kPCA Transformation

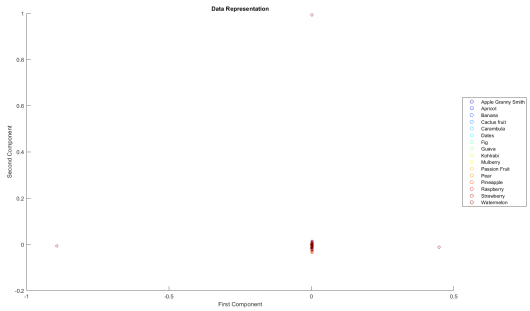

Figure 6: 2D Representation of transformed data

The 2D representation of the data post kPCA transformation with a Gaussian kernel indicates that most samples are clustered around 0, with minimal variation along the x-axis and slight dispersion along the y-axis, except for some outliers. This suggests that the lower dimensions may not adequately represent the data, and the information from the higher dimensions might not be effectively captured. Consequently, classification algorithms could encounter challenges with this representation. Additionally, it's worth noting that kPCA requires considerably more computational time compared to PCA, as highlighted in the table 2.

3.3.2 Polynomial kPCA

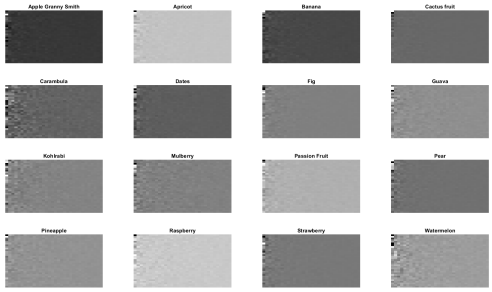

Figure 7: Data after Polynomial kPCA Transformation

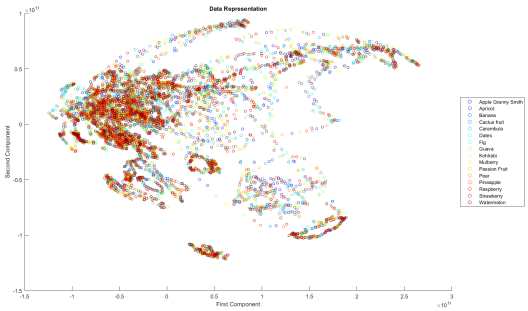

Figure 8: 2D Representation of transformed data

In contrast to the Gaussian kernel, the polynomial kernel presents the data with a well-distributed spread of samples across the 2D space. This indicates that the data is effectively represented in the lower dimension, with the information being preserved adequately. Consequently, the classifier is expected to perform well when kPCA is conducted using the polynomial kernel.

3.4 t-distributed Stochastic Neighbor (t-SNE) Embedding

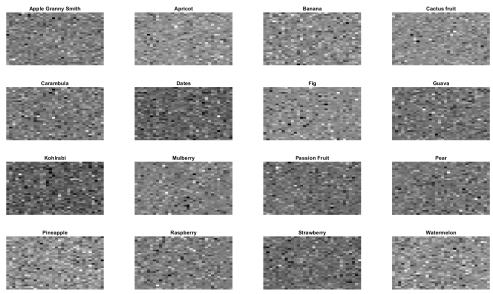

Figure 9: Data after t-SNE embedding

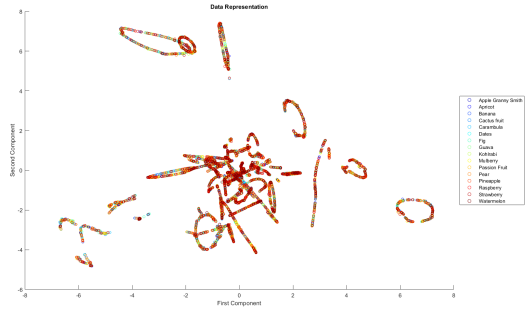

Figure 10: 2D Representation of transformed data

The 2D t-SNE representation displays a degree of dispersion among the samples, with various pockets and clusters scattered across the 2D space. This suggests that some information from the higher dimensions is reasonably well-captured in the lower dimensions, as evidenced by the decent spread of samples. Consequently, the classifier is anticipated to perform well on the data in this representation.

3.5 Autoencoder Embedding

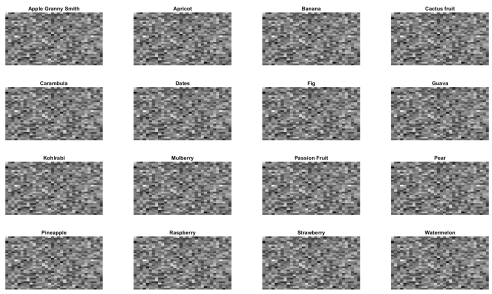

Figure 11: Data after Autoencoder embedding

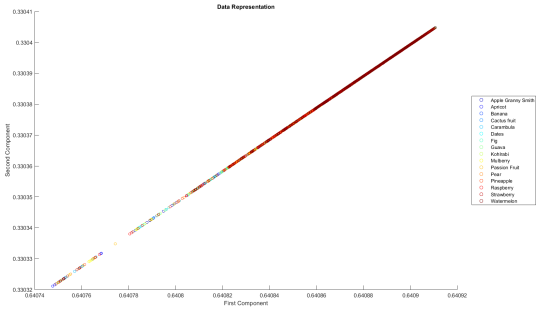

Figure 12: 2D Representation of transformed data

In the 2D autoencoder representation, the samples are depicted along an approximately straight line. However, this linear distribution indicates that the 2D space is not fully utilized. Moreover, unlike the spread observed in Polynomial kPCA and t-SNE embedding, there is a lack of dispersion across the 2D space, indicating that the information from higher dimensions may not be effectively captured in the lower dimensions. Consequently, the classifier may encounter difficulties in performing well on the dataset represented using autoencoder embedding.

4 Classification Algorithms

In this section, we'll explore into two classification algorithms: KNN Classification and kSVM classification. These models will be compared based on three metrics: test accuracy, validation accuracy, and classification time. Then, we'll apply the best-performing algorithm from this comparison to our transformed datasets to evaluate their performance. We'll determine the number of principal components required to achieve high accuracy, providing insights into which method from the previous section most efficiently represents our dataset.

To compare the KNN Classifier and kSVM Classifier, I'll follow a standardized procedure using the original dataset. Initially, I'll shuffle samples within each class, a method proven to enhance classifier robustness, as learned from project 1. Subsequently, I'll partition the dataset with 400 samples per class for training, 200 samples per class for testing, and reserving 25 samples per class for the validation dataset.

Algorithm	**Test Accuracy**	**Validation Accuracy**	**Time Taken (s)**
KNN Classification	0.981875	0.990000	0.577916
kSVM Classification	0.999687	1.000000	354.926349

Table 1: Comparison of performance of KNN and kSVM Classifiers

From the table, it's evident that the Kernel SVM classifier marginally outperforms the KNN classifier, achieving higher test and validation accuracy. However, the Kernel SVM classifier also exhibits a substantially longer learning time compared to the KNN classifier. Moreover, the 100% validation accuracy of the Kernel SVM classifier raises concerns about potential overfitting. Therefore, based on these observations, I prefer proceeding with the KNN classifier for further exercises. It offers a high accuracy while requiring significantly less time for training compared to the Kernel SVM classifier.

In the next phase of this exercise, we'll investigate how varying the number of retained principal components in the transformed data impacts testing accuracy. This analysis aims to identify the optimal number of principal components that yield the highest testing accuracy.

4.1 Principal Component Analysis (PCA)

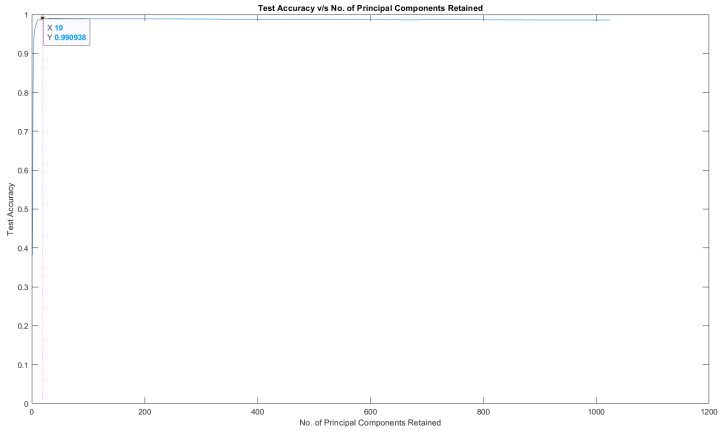

Figure 13: Test Accuracy v/s Number of Principal Components Used for Training

- Maximum accuracy: 0.9909, achieved using 19 principal components.
- Lower dimensions retain high-dimensional information effectively.
- The classifier is expected to perform well given the retained information.

4.2 Kernel Principal Component Analysis (kPCA)

4.2.1 Gaussian kPCA

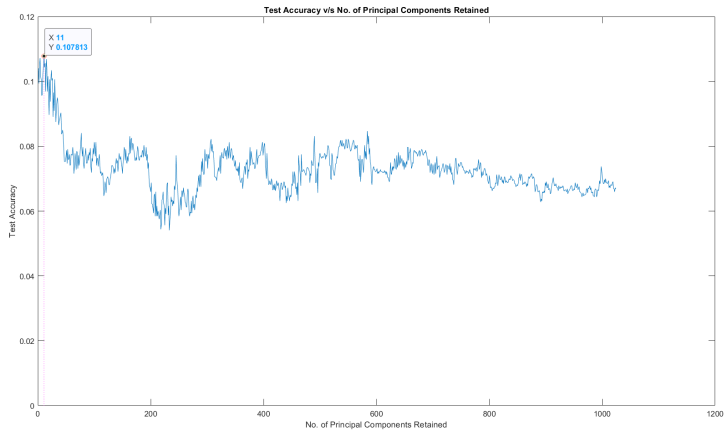

Figure 14: Test Accuracy v/s Number of Principal Components Used for Training

- Maximum accuracy: 0.1078, achieved using 11 principal components.

- Lower dimensions does not retain high-dimensional information effectively.
- The classifier is not expected to perform well given the retained information.

4.2.2 Polynomial kPCA

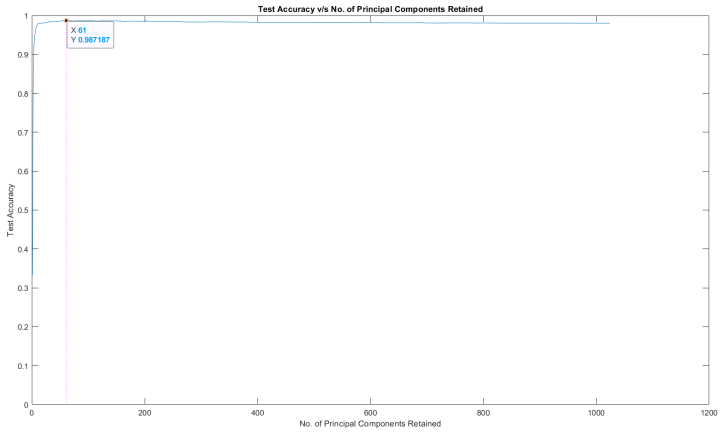

Figure 15: Test Accuracy v/s Number of Principal Components Used for Training

- Maximum accuracy: 0.9872, achieved using 61 principal components.
- Lower dimensions retain high-dimensional information effectively.
- The classifier is expected to perform well given the retained information.

4.3 t-distributed Stochastic Neighbor (t-SNE) Embedding

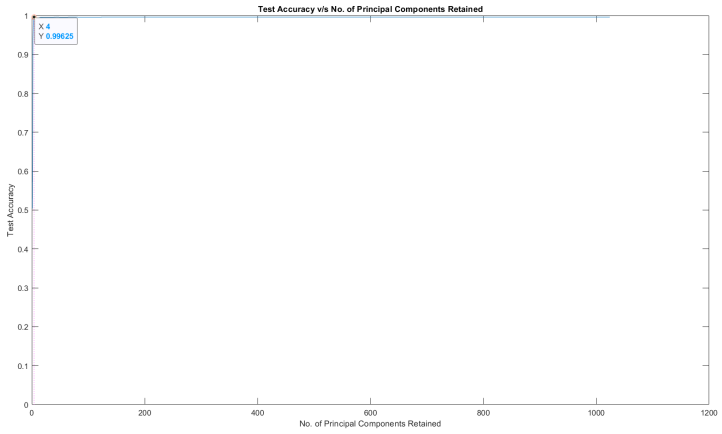

Figure 16: Test Accuracy v/s Number of Principal Components Used for Training

- Maximum accuracy: 0.99625, achieved using 4 principal components.
- Lower dimensions retain high-dimensional information effectively.
- The classifier is expected to perform well given the retained information.

4.4 Autoencoder Embedding

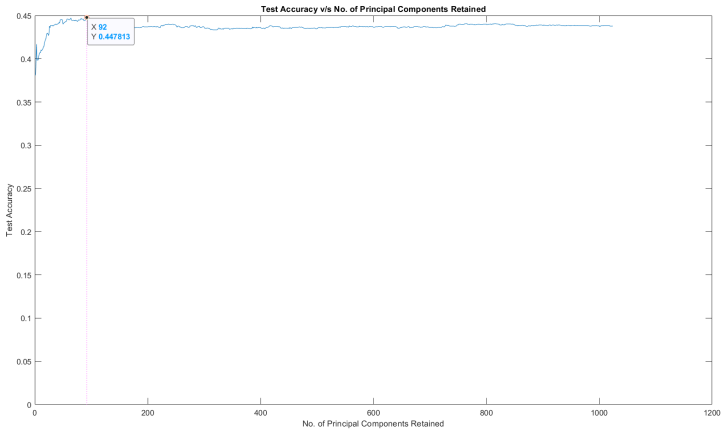

Figure 17: Test Accuracy v/s Number of Principal Components Used for Training

- Maximum accuracy: 0.4478, achieved using 92 principal components.
- Lower dimensions does not retain high-dimensional information effectively.
- The classifier is not expected to perform well given the retained information.

5 Results and Conclusions

5.1 Time Taken for performing Dimension Reduction

In this section, we will examine the time required to map the original dataset into the various transformations discussed in the previous sections. This analysis aims to provide insights into the computational complexity of each of these algorithms.

DR Method	Time Taken for DR (s)
PCA	0.78
kPCA (Gaussian)	352.45
kPCA (Polynomial)	344.81
t-SNE	892.39
Autoencoder	24660.55

Table 2: Time Taken for Dimensionality Reduction (DR) Methods

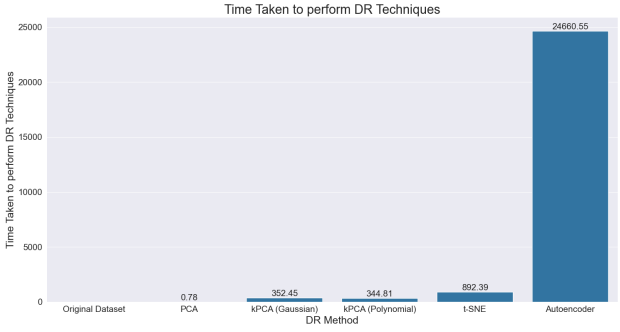

Figure 18: Time Taken for performing DR techniques

1. **Principal Component Analysis (PCA)**: Among the techniques considered, PCA is the fastest method, completing the transformation in 0.78 seconds. Its efficiency stems from directly computing the principal components of the data matrix, resulting in a low computational burden. Its complexity is generally $O(n^2 \cdot d + d \cdot n^2)$, where n is the number of samples and d is the number of features.

2. **Kernel Principal Component Analysis (kPCA)**:

   - **Gaussian kPCA**: This method demands considerably more time, taking 352.45 seconds for transformation. The computation of pairwise distances necessary for the Gaussian kernel contributes to its higher computational complexity, which is $O(n^2 \cdot d + n^3)$, where n is the number of samples and d is the number of features.

   - **Polynomial kPCA**: Similar to Gaussian kPCA, the polynomial kernel completes transformation in 344.81 seconds. Its computational complexity is $O(n^2 \cdot d + d^3)$, where n is the number of samples and d is the number of features.

3. **t-distributed Stochastic Neighbor Embedding (t-SNE)**: t-SNE consumes 892.39 seconds for transformation. Its computational intensity primarily arises from the computation of probabilities of

pairwise similarities, leading to a complexity of $O(n^2)$, where n is the number of samples.

4. **Autoencoder Embedding**: Autoencoder embedding is the slowest technique of transformation taking 24660.55 seconds (approximately 6.85 hours). The training of a neural network for autoencoder inherently demands more time compared to linear or kernel-based methods due to its complex architecture and iterative optimization process. Its complexity depends on the architecture and complexity of the neural network, typically $O(n \cdot e)$, where n is the number of samples and e is the number of epochs.

5.2 Optimal Number of Principal Components

In this section, we will study the outcomes of the exercise conducted in 4.

DR Method	Principal Components Used
Original Dataset	N/A
PCA	19
kPCA (Gaussian)	11
kPCA (Polynomial)	61
t-SNE	4
Autoencoder	92

Table 3: Principal Components Used by DR Methods

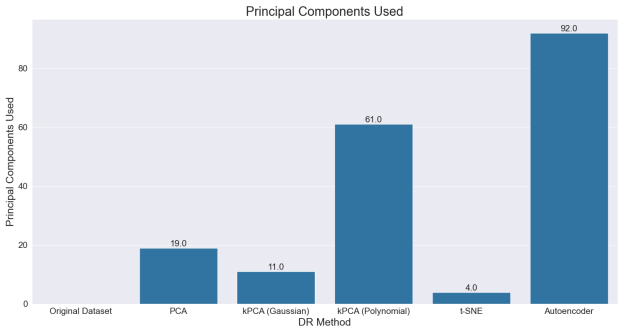

Figure 19: Principal Components used by DR Methods

From the table and graph, it's evident that t-SNE requires the fewest number of components to effectively capture the information from higher dimensions in lower dimensions.

5.3 Analyzing Accuracy and Time Taken for Classification

After determining the optimal number of principal components required for each of the different embedding methods, I conducted KNN classification by retaining these optimal components and compared the classifier's performance. To mitigate the affect of the initial position of centers, I executed the classifier 10 times and averaged the metrics. The graph below illustrates the testing and validation accuracy of the KNN classifier after retaining the optimal number of principal components for each DR method. For all DR methods, the parameters of the KNN classifier is as follows:

- Training Samples: 400 per class
- Testing Samples: 200 per class
- Validation Samples: 25 per class
- $K = 16$
- Distance metric: Euclidean

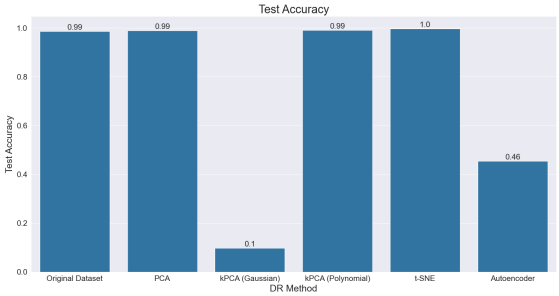

Figure 20: Test Accuracy of DR Methods

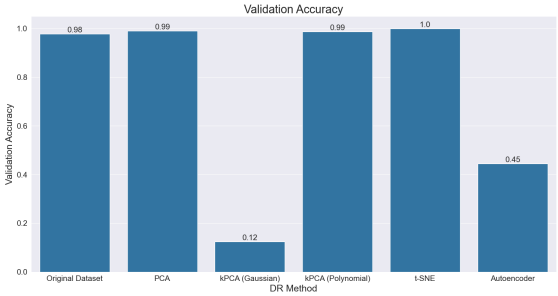

Figure 21: Validation Accuracy of DR Methods

Figure 22: Accuracy of DR Methods after performing KNN Classification

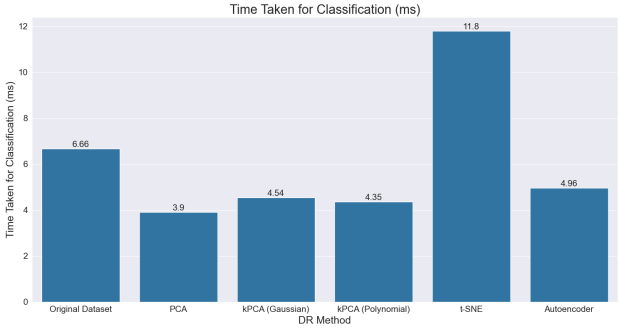

Figure 23: Time Taken for Classification (milliseconds)

Based on the graphs in the Figure 22, the Gaussian kernel Principal Component Analysis (kPCA) and Autoencoder techniques yielded sub-optimal results, with average testing accuracies of 0.1 and 0.46 respectively across 10 simulations. Consequently, these methods are deemed unsuitable for further consideration due to their inadequate performance metrics.

The original dataset, linear PCA, polynomial kPCA, and t-SNE transformations exhibit impressive average testing accuracy, ranging between 0.98 and 1.00. However, to identify the most optimal DR technique, we refer to the Figure 23. Analysis of classification time highlights the superior performance of linear PCA and polynomial kPCA, with classification times of 3.9 ms and 4.35 ms respectively, outperforming other transformations.

Further refinement involves discarding the original representation and t-SNE transformation due to their prolonged classification times. Consequently, the focus narrows down to the choice between linear PCA and polynomial kPCA, which exhibit comparable accuracies and classification times.

Referring to the analysis conducted in section 5.1, it is observed that linear PCA offers significantly faster data transformation, requiring 0.78 seconds compared to polynomial kPCA's notably longer duration of 344.81 seconds (5.75 minutes). This substantial discrepancy in transformation time renders polynomial kPCA impractical for the dataset at hand.

Thus, considering both accuracy and computational efficiency, linear PCA emerges as the preferred DR technique for transforming the dataset. Its superior performance in terms of both accuracy and computational speed makes linear PCA as the optimal choice.

6 Conclusion

Throughout this project, I explored various data representation techniques, namely linear PCA, kernel PCA, t-SNE, and Autoencoder. Focusing on the KNN classifier due to its superior accuracy and efficiency, I determined the optimal number of principal components required for each transformation, aiming for efficient representation without compromising accuracy.

While preserving the identified optimal principal components, I compared the performances of the KNN classifier across various transformations. This evaluation encompassed testing accuracy, validation accuracy, classification time, and transformation time, prioritized in that sequence.

From my analysis, I conclude that the linear PCA is the most viable data transformation technique. This conclusion is substantiated by several reasons:

1. Exceptionally high testing and validation accuracy, both at 0.99.

2. Minimal time required for classification, with a processing time of 3.9 milliseconds.

3. Transformation of data completed in 0.78 seconds.

The outcomes of this project closely align with the findings layed out in the article by L.J.P. van der Maaten, E.O. Postma, and H.J. van den Herik, titled "Dimensionality Reduction: A Comparative Review," Tilburg University Technical Report, TiCC-TR 2009-005, 2009. In that article, the authors found that despite their large variance, nonlinear techniques for dimensionality reduction often fail to outperform traditional linear techniques such as PCA.