1. INTRODUCTION

This report aims to build a model that can discriminate between several classes of data, with particular importance on classification tasks. A classification algorithm learns from the given set of training data, to predict the type of the new data, which is called test data.

It is crucial to divide the available data into two sets, training, and testing, so that a machine learning model performs well not only on the data it was trained on but also on fresh, untested data. The testing set, which acts as a fresh set of data for the model, is used to assess the model's performance after the training.

The MNIST dataset consists of 70,000 pictures of handwritten numbers. The pictures are split into two sections, 10,000 for testing and 60,000 for training. Every image is grayscale 28x28 pixel representation of a number. The MNIST dataset, which requires classifying each image into one of ten groups (digits 0-9), serves as an example of the classification problem. The separation of the data into training and testing sets serves as an example of the fundamental machine learning procedure of training a model on a subset and testing it on another subset to assess how well a particular model will perform against the new, untested data.

2. DATA PREPARATION AND PCA DIMENSIONALITY REDUCTION

2.1 Preparation of dataset:

Data preparation is a necessary initial step before working with any machine learning model. For the MNIST dataset, there are several preparation steps to complete.

1. Setting the Working Directory: Specifies the directory where the MNIST data files are stored.

2. Used the load_mnist() function to load the image and label files for the training and testing data, thereby loading the MNIST digit recognition dataset.

3. Used the any(is.na()) function to check the dataset for missing values to guarantee data integrity.

4. To scale the features between 0 and 1, the pixel values in the dataset were normalized by dividing them by 255, the maximum pixel value.

2.2 Applying PCA dimensionality reduction and visualizing different classes in two dimensions:

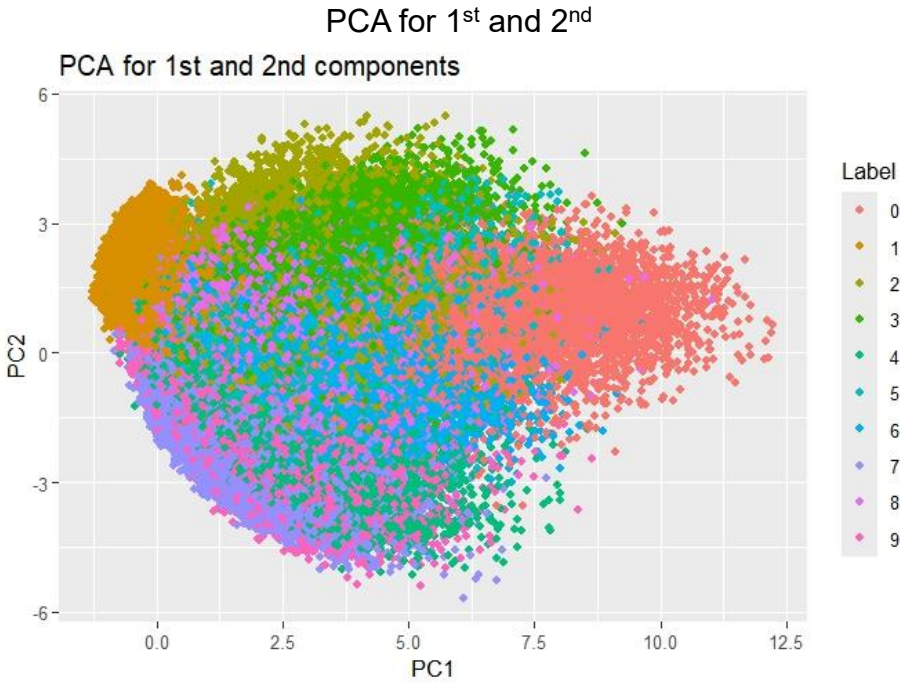

Fig.1

The plot(Fig.1) illustrates how data variation is captured in two dimensions for the first two principal components of the MNIST dataset, after applying PCA,

- With different colors denoting different digit labels (0-9), each point in the plot represents an image of digits.

- Most of the variance is captured by the X-axis(PC1), which may represent variations in digit size or orientation, The second most variance is captured by the Y-axis (PC2), which most likely represents style variations or other subtle features.

- The way the colors overlap and spread out shows how unique each digit is in this smaller area. Similar colored clusters indicate that PCA is effective at reducing dimensionality and have good class separability.

- High classifier performance is shown by well-defined clusters, but substantial overlaps could compromise classification accuracy.

PCA for 3rd and 4th

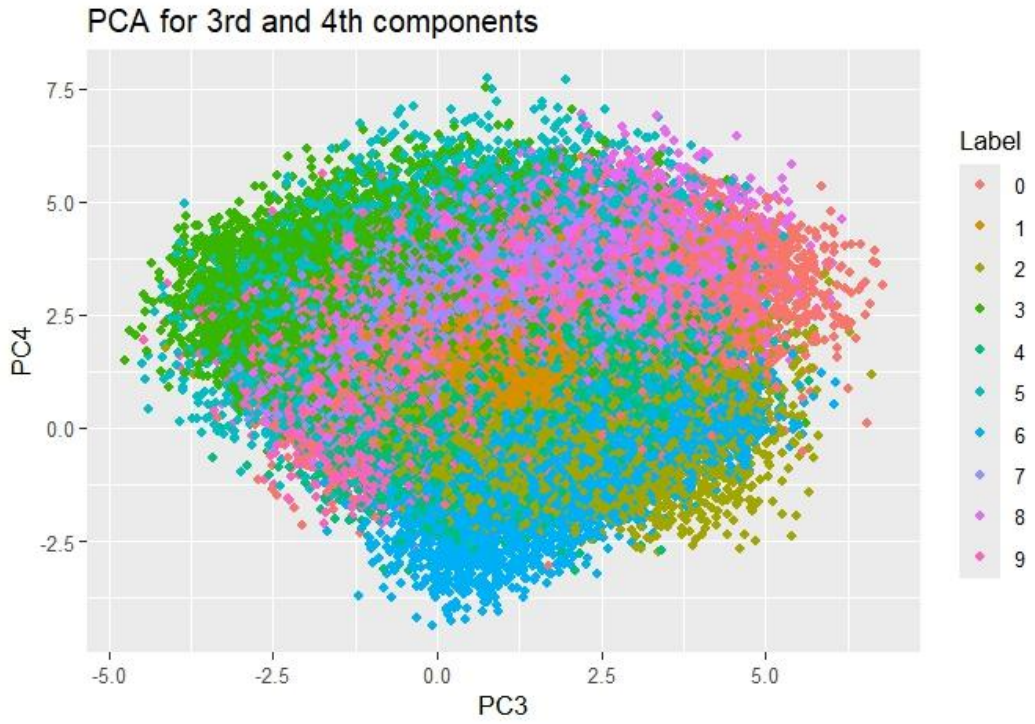

Fig.2

- This plot(Fig.2) shows how the MNIST dataset is distributed across the third and fourth principal components (PC3 and PC4).

- Even though PC3 and PC4 capture less variance than the first two components, they still provide important insights into the data structure that may be related to less noticeable features of the digits like variations in stroke thickness or small angular differences.

- These are not as discriminative as the first two components in terms of clearly separating all digit classes. This suggests that additional components are required to obtain more details.

2.3 Investigate how many principal components are needed to encode the dataset and give an example of reconstruction:

To determine the number of principal components to retain, I have calculated the cumulative variance explained by the components. This is done by summing up the variance explained by each principal component consecutively until a significant portion of the total variance is accounted for, this is called the Heuristic Rule in PCA.

```
> num_components <- which(cum_var_explained >= 0.90)[1]  # for the number of co
> cat("Number of principal components: ", num_components, "\n")
Number of principal components:  87
```

Fig.3

As per the Fig.3 the number of principle components that explain at least 90% variance are 87.

Reconstruction:

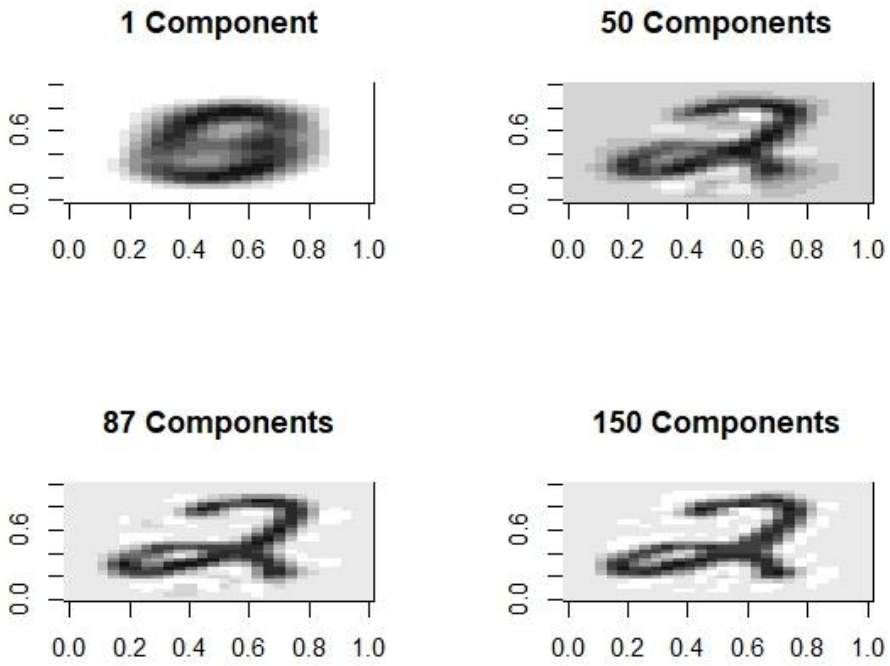

Fig.4

From the reconstruction image (Fig.4) of the original data using the selected principal components 1, 50, 87, 150, we can notice that with just 1 or 50 components, the image is not clearly recognizable. However, with the 87 components, the number in the image is noticeable, which indicates that 87 components capture the essential features of the structure of the data. This is also backed up by Fig.3.

3. K-NEAREST-NEIGHBOUR CLASSIFICATION AND ERROR RATE EVALUATION

3.1 Evaluating the kNN classifier and determining the value of k:

To begin with, I have created a subset of the training data which has only 3% of the data, this also helps in speeding up the training process.

Based on the square root of the total number of observations in the training dataset, a range of k values, ranging from 1 to 41, was selected. Only the odd values were considered to avoid ties.

The kNN algorithm was applied twice for each k value to check the training error and the test error.

- Train Error: To assess how well the model fits the training data, the classifier was tested and trained on the same subset (train_kx).

- Test Error: The classifier was trained on the subset but tested against the entire test dataset (x_test). This measures how well the model reacts to unseen data.

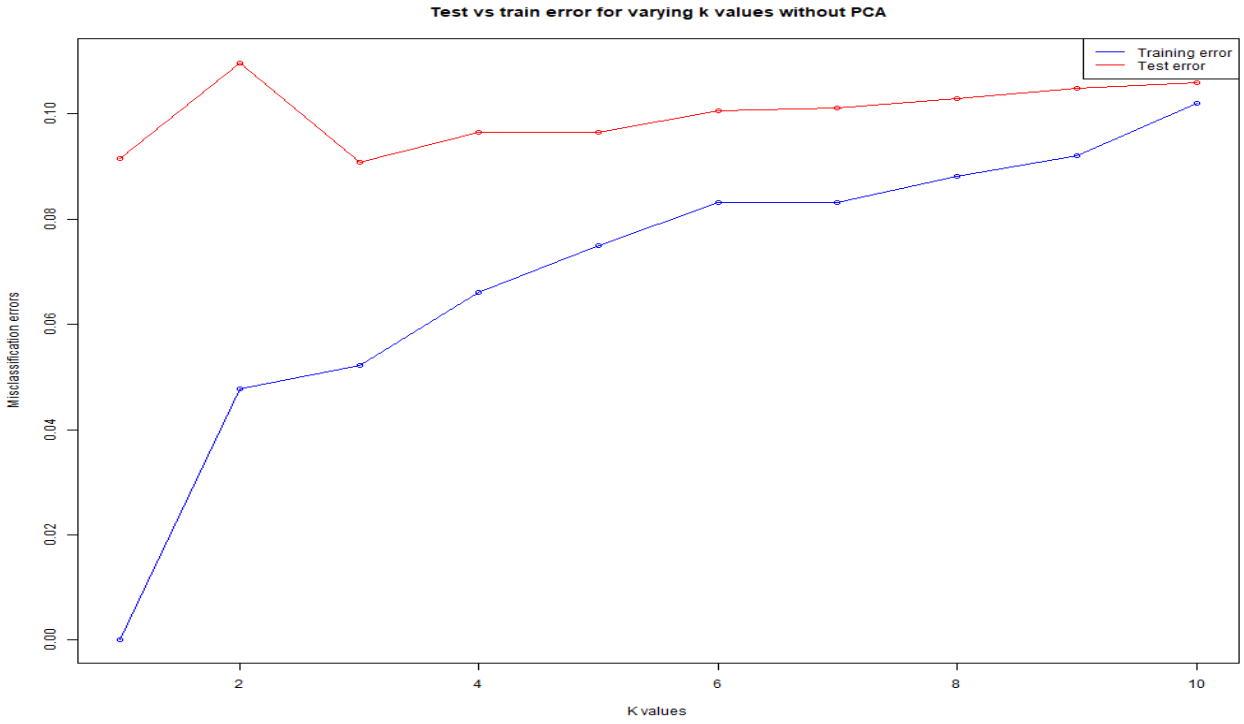

Fig.5

From Fig.5,

Training Error: This line in blue (training error) generally increases as k increases, which is typical behavior. Although a higher k value tends to reduce noise in the data, it can also cause underfitting, which lessens the model's sensitivity to changes in the training set.

Testing Error: The red line initially decreases as k increases from 1 to 3, suggesting that a k of 1 overfits the data. After reaching a low point at k=3, the test error increases slightly with stability, which indicates that 3 is the optimal value of k from the above plot.

Confusion matrix for **k=3** (as the error plot indicates from Fig.5)

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1    2    3    4    5    6    7    8    9
         0  954    0   26    3    3    5   13    1   16    7
         1    1 1129   40   11   29   13    7   46   18   13
         2    5    2  886    7    0    2    0    4    6    3
         3    1    1   14  918    0   25    0    2   35    7
         4    1    0    5    1  805   10    2   14    9   23
         5    5    0    2   26    0  775    6    0   61    6
         6   11    3    8    5   15   22  930    0    9    4
         7    1    0   31   13    8    8    0  926   14   25
         8    0    0   13   15    3   11    0    0  774    3
         9    1    0    7   11  119   21    0   35   32  918

Overall Statistics

               Accuracy : 0.9015
                 95% CI : (0.8955, 0.9073)
    No Information Rate : 0.1135
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8905

 Mcnemar's Test P-Value : NA
```

Fig.6

Confusion matrix for **k=5** (A possible k value as we can see a slight elbow pattern at around k=5 in the error plot)

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1    2    3    4    5    6    7    8    9
         0  954    0   25    2    1    7   14    1   11   10
         1    1 1130   61   13   38   15    8   57   23   14
         2    0    2  872    6    1    3    0    4    9    3
         3    1    1    9  911    0   32    0    0   34    4
         4    1    0    5    0  790    5    2   10    9   21
         5    5    0    1   34    0  768    3    1   47    5
         6   16    2    8    7   16   19  930    0   14    2
         7    1    0   31   13    6   11    0  915   15   23
         8    0    0   16   12    3   10    0    0  784    1
         9    1    0    4   12  127   22    1   40   28  926

Overall Statistics

               Accuracy : 0.898
                 95% CI : (0.8919, 0.9039)
    No Information Rate : 0.1135
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8866

 Mcnemar's Test P-Value : NA
```

Fig.7

Comparing the performance of the k-nearest neighbours (kNN) classifier for k = 3 (Fig.6) and k = 5 (Fig.7) we can make perceptive observations regarding the model's sensitivity to the selection of k.

The confusion matrix shows that the model achieved an accuracy of about 90.15% for k = 3. With a Kappa statistic of 0.8905, this model appears to have much higher agreement between the actual and predicted classifications.

For k=5, the accuracy slightly decreased to about 89.8%. With a Kappa value of 0.8866, this model is likewise high but slightly less than the k = 3 model. The findings indicate that a smaller k value(k=3) may be more effective for this dataset, possibly because there is less noise influencing the classification decision at k=3.

A p-value significantly less than 0.05 is obtained for both configurations, indicating statistically significant model accuracy.

3.2 Impact of PCA on kNN.

Confusion matrix of the PCA on kNN:

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1    2    3    4    5    6    7    8    9
         0  974    0    6    0    1    3    3    0    4    5
         1    1 1130    3    1    5    0    3   14    0    4
         2    1    3 1002    4    1    0    0    5    3    1
         3    0    0    2  976    0    9    0    0   12    7
         4    0    1    1    1  952    2    3    2    4    9
         5    1    0    0   12    0  865    3    0   14    5
         6    2    1    0    0    4    6  946    0    4    1
         7    1    0   15    7    2    1    0 1001    3    4
         8    0    0    3    5    0    2    0    0  924    3
         9    0    0    0    4   17    4    0    6    6  970

Overall Statistics

               Accuracy : 0.974
                 95% CI : (0.9707, 0.977)
    No Information Rate : 0.1135
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9711

 Mcnemar's Test P-Value : NA
```

Fig.8

- The accuracy of the PCA-based kNN model (Fig.8) is 0.974, with a 97.07% to 97.7% confidence interval. This high accuracy suggests that the model exhibits great performance in every class.

- A kappa statistic of 0.9711 that the model is very dependable in terms of accuracy and consistency over a number of runs.

- The model's accuracy is demonstrated by the significant p-value (< 2.2e-16), which confirms that the accuracy of the model is statistically significant.

Comparison with the non-PCA kNN model (Fig.6):

- The accuracy of the PCA-based kNN model is significantly higher (0.974) than that of the non-PCA kNN model (0.9015), with k = 3. This shows that classifications that are more accurate on kNN with PCA.

- The higher kappa value (0.9711) in the PCA-based kNN model compared to the lower value in non-PCA based kNN model (0.8905) suggests that the high accuracy is not simply due to a dataset imbalance, but it truly reflects improved classification across all categories.

- In conclusion, the PCA-based approach outperforms the k=3 kNN model.

3.3 Scree Plot:

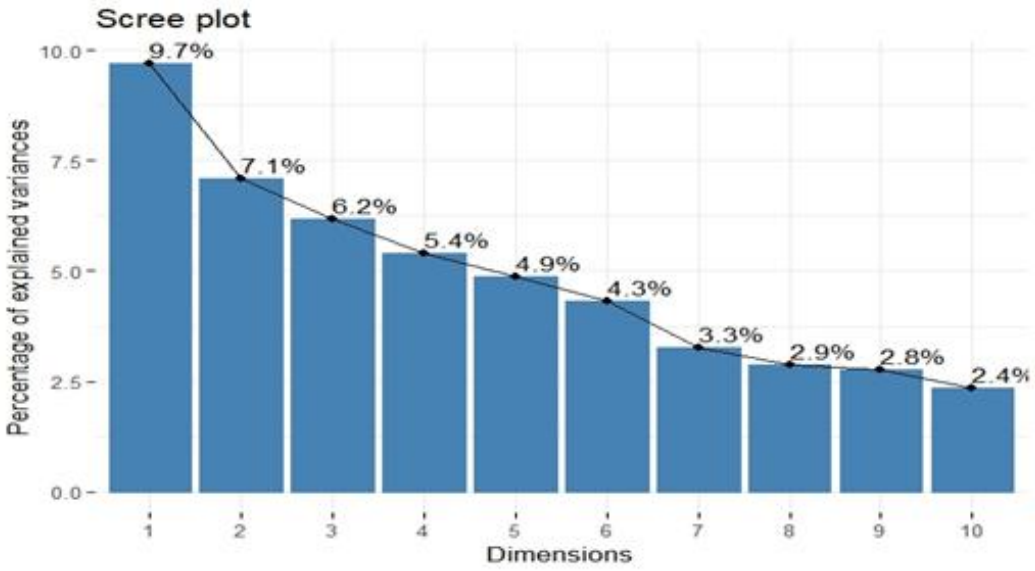

Fig.9

- In the Fig.9, The first principal component (PC1) explains the most variance of any of the principal components, accounting for approximately 9.7%. This suggests that this component detects the most important patterns or trends in the data.

- Each subsequent principal component explains less variance than the previous one. For example, PC2 explains 7.1%, and this figure decreases further with each subsequent component, reaching 2.4% by PC10.

- In the scree plot, the elbow point appears to be at the third dimension. This is where the line's slope changes the most dramatically, from a steep drop after the first and second components. Beyond this point, each additional component contributes progressively less to the total variance, which suggests that most of the useful information is captured within the first three components.

4. SECOND ML TECHNIQUE - SUPPORT VECTOR MACHINE (SVM)

The SVM classifier uses pixel intensity to convert each image into a 784-dimensional vector in order to predict the digit for the handwritten digits in the MNIST dataset. In order to effectively divide the classes and distinguish between various digits, it builds hyperplanes in this high-dimensional space.

4.1 Running the SVM on the reduced dataset:

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1    2    3    4    5    6    7    8    9
         0  926    0   15    5    1    5   12    2    9    9
         1    0 1117   23    6   12   12    5   41   28   13
         2    2    2  868   18    4    4    9   19    9    7
         3    1    4   14  875    0   37    0    0   33   12
         4    0    0   16    1  811    7    7   12    5   25
         5   27    3    4   48    0  760   19    0   49   11
         6   20    4   29    7   14   28  906    0   19    3
         7    2    0   16   16    3    6    0  896    8   21
         8    2    5   36   20    6   10    0    7  778    3
         9    0    0   11   14  131   23    0   51   36  905

Overall Statistics

               Accuracy : 0.8842
                 95% CI : (0.8778, 0.8904)
    No Information Rate : 0.1135
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8713

 Mcnemar's Test P-Value : NA
```

Fig.10

Comparing the confusion matrix and the statistics of the SVM (Fig.10) with the kNN (Fig.6, where k=3):

- The SVM model has the accuracy of 88.42, with a 95% confidence interval, whereas, kNN has the accuracy of 90.15% with a 95% confidence interval.
- SVM has the Kappa statistic of 0.8713, while the kNN has the Kappa value of 0.8905 which is slightly higher than the SVM, suggesting a marginally better agreement between the predicted and actual labels.
- P-values for both models is significant.

This suggests that kNN model is slightly more accurate and consistent in classification.

4.2 PCA on SVM

The confusion matrix and the statistics after applying the PCA on SVM is shown below

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1    2    3    4    5    6    7    8    9
         0  971    0    3    0    0    2    3    0    4    4
         1    0 1125    1    0    0    0    2    3    0    1
         2    1    3 1011    5    6    0    0   10    1    2
         3    0    1    2  991    0    7    0    1    2    4
         4    0    1    1    1  962    0    2    2    2   11
         5    2    1    0    2    0  878    3    1    4    3
         6    3    1    1    0    1    4  947    0    1    0
         7    1    1    4    4    0    0    0 1003    2    8
         8    2    2    8    5    3    1    1    1  954    1
         9    0    0    1    2   10    0    0    7    4  975

Overall Statistics

               Accuracy : 0.9817
                 95% CI : (0.9789, 0.9842)
    No Information Rate : 0.1135
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9797

 Mcnemar's Test P-Value : NA
```

Fig.11

4.3 Comparison of PCA on SVM vs PCA on kNN (Fig.8):

- The application of PCA on both SVM and kNN has shown an improvement in accuracy of the models.
- With an accuracy of 98.17%, the PCA on SVM outperformed the PCA on KNN, which had a 97.4% accuracy.
- High Kappa statistics are shown by both techniques (0.9711 for KNN and 0.9797 for SVM), suggesting exceptional agreement that goes beyond chance.
- The error rate of PCA on kNN (2.6) is greater than PCA on SVM(1.83).

In conclusion, PCA on KNN is also very effective and offers a simpler and possibly faster alternative depending on the phase of model application (training vs. testing), even though PCA on SVM exhibits a slight edge in overall accuracy and per-class precision.

5. Conclusion:

In the comparison of the analysis of the KNN and SVM algorithms on the MNIST dataset, our methods show better effectiveness. According to the MNIST Wikipedia page, advanced neural network techniques on the dataset, set a high standard with rates as low as 0.18% error rates. Although not reaching these lows, our less complex PCA-based KNN and SVM methods exhibit strong performance with error rates of 2.6% and 1.83% comparable to those of the simpler methods mentioned on the same page.

The SVM classifier enhanced with PCA showed the highest accuracy (98.17%) and is the most appropriate method for classifying the MNIST dataset in my research. This method effectively manages the large number of dimensions in the data while preserving good classification performance.

A method's suitability is determined by several factors, including model interpretability, training time, and computational complexity, in addition to classification accuracy. Even though SVM is typically more accurate, it can require a lot of processing power, particularly when dealing with big datasets and complex parameter tuning.

We can consider using advanced feature engineering techniques like LDA, optimizing hyperparameters through cross-validation, using regularization to prevent overfitting, and experimenting with deep learning techniques like CNNs for better feature extraction and generalization on image data in order to improve the k-NN and SVM models on MNIST.