A Simple Introduction to Facial Recognition (with Python codes)

Understanding how Face Recognition works

Computer vision encompasses tasks such as image classification, object detection, segmentation, face detection and recognition, motion analysis, scene reconstruction, image restoration, and pattern recognition. These tasks enable computers to process and interpret visual data, much like humans. For example, computers can categorize objects in images, identify and track objects within videos, reconstruct three-dimensional scenes, restore image quality, and recognize patterns like handwritten text. This technology is fundamental in applications ranging from autonomous vehicles to medical imaging.

We will first understand the inner workings of face recognition, and then take a simple case study and implement it in Python.

To understand how face recognition works, it's essential to grasp the concept of a feature vector. A feature vector is a numerical representation that captures the distinctive attributes of a face. When identifying a face in a dataset, the system examines patterns such as height, width, and the proportions of facial components like the eyes, nose, and mouth. It also considers skin color. These physical dimensions and attributes vary from person to person, allowing the system to convert these unique facial features into a series of numbers that form the feature vector. This vector effectively transforms a face into a form that a computer can process and recognize. A machine learning algorithm match a new image with the set of feature vectors present in the dataset.

Implementation in Python

IGP-sep.2023/A Simple Introduction to Facial Recognition (with Python codes).ipynb at main · hh2-radwan/IGP-sep.2023 (github.com)

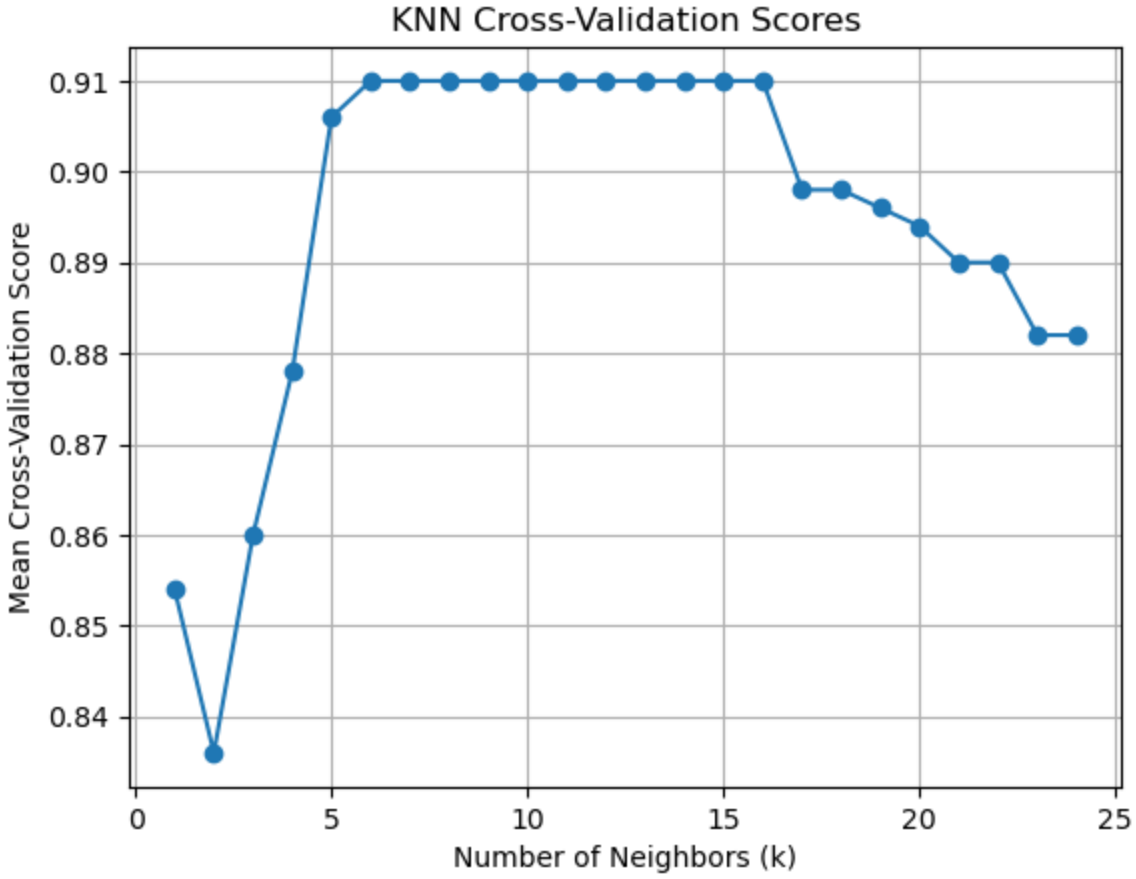
We loading, resizing, and labeling images from different directories for a face recognition task using Using OpenCV we load dataset from. Index of /afs/cs.cmu.edu/project/theo-8/ml94faces/faces . It constructs training datasets by converting image lists to numpy arrays and assigning unique labels to each individual's images. The images are then merged into a single dataset. Finally, it splits this dataset into training and testing sets using scikit-learn's train_test_split function. This setup is likely aimed at training a K-nearest neighbor, LDA, PCA and Support Vector Machine classifiers for image classification.

KNN

We extends the face recognition task by evaluating different configurations for a K-Nearest Neighbors (KNN) classifier using cross-validation. It systematically tests k values from 1 to 24 to find the optimal number of neighbors for classification. We compute the mean cross-validation score for each k using a 5-fold cross-validation on the training data. After determining the best k, it visualizes the relationship between k values and their

corresponding scores with a plot (figure 1), and finally, it retrains the KNN classifier with the optimal k value on the reshaped training dataset. The model's performance is assessed on the test data, demonstrating its accuracy. This procedure helps ensure the robustness of the classifier by selecting the most effective hyperparameter (k) based on validation scores.

```
Best value of k: 6
```




```
Accuracy: 0.92
```

Figure 1: k values vs. mean cross-validation scores

The code evaluates a K-Nearest Neighbors classifier's performance on face recognition data by predicting test labels, generating a confusion matrix(figure2), and computing precision, recall, and F1-scores. It visualizes the confusion matrix as a heatmap and outputs a detailed classification report, providing insights into the model's accuracy and classification capabilities.

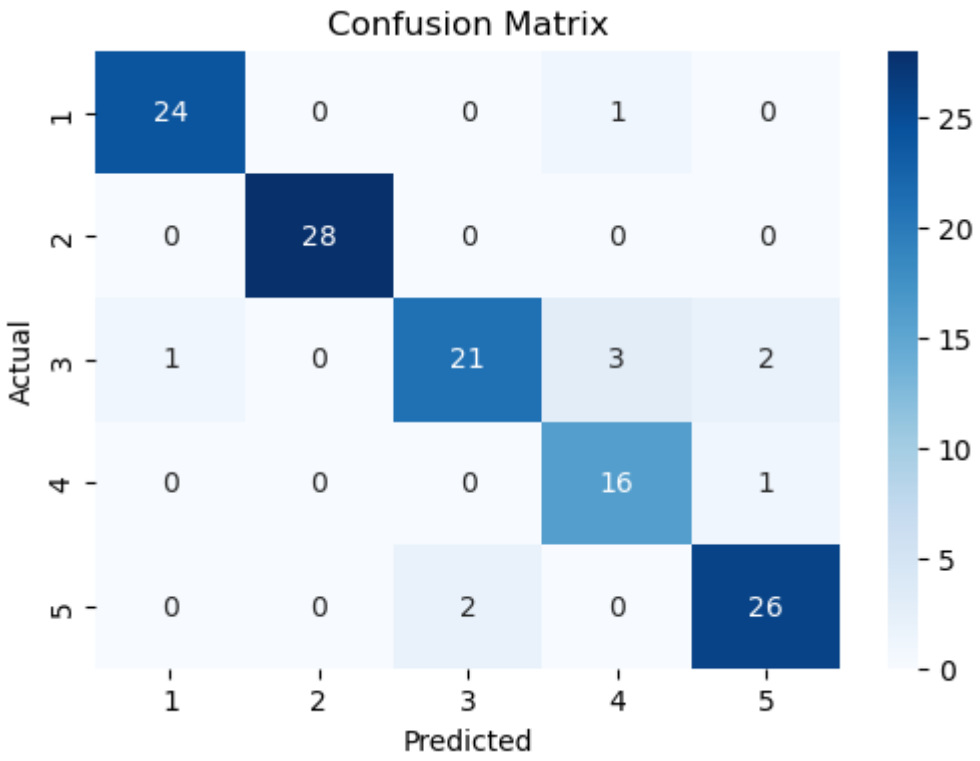


Figure 2: confusion matrix as a heatmap for KNN

The classification report (Figure 3) provides detailed metrics on the performance of a K-Nearest Neighbors (KNN) model used for face recognition across 5 distinct classes. Each class likely represents a different individual. Here's an interpretation of the metrics:

Classification Report:

	precision	recall	f1-score	support
1	0.96	0.96	0.96	25
2	1.00	1.00	1.00	28
3	0.91	0.78	0.84	27
4	0.80	0.94	0.86	17
5	0.90	0.93	0.91	28

Figure 3: Classification Report for KNN

Precision:

Class 1: The model correctly identifies 96% of all predictions for this class as true positives. This indicates high precision.

Class 2: Perfect precision (100%) means every prediction made for this class by the model is correct.

Class 3: Lower precision at 91% suggests that 9% of predictions for this class were false positives (other classes wrongly predicted as Class 3).

Class 4: Precision is 80%, indicating 20% false positives.

Class 5: Precision is 90%, with 10% false positives.

Recall (Sensitivity or True Positive Rate):

Class 1: Recall of 96% indicates that the model identifies 96% of all actual Class 1 instances.

Class 2: Perfect recall (100%) indicates that all actual Class 2 instances were identified by the model.

Class 3: Lower recall at 78% suggests some Class 3 instances were missed (22% false negatives).

Class 4: High recall of 94% means the model effectively identified most of the Class 4 instances.

Class 5: Recall of 93% is also high, showing effective identification of Class 5 instances.

F1-Score:

Combines precision and recall into a single metric. An F1-score reaches its best value at 1 (perfect precision and recall) and worst at 0.

Class 1: An F1-score of 0.96 reflects both high precision and recall.

Class 2: A perfect F1-score of 1.00 indicates excellent performance for Class 2.

Class 3: An F1-score of 0.84, influenced by lower recall.

Class 4: An F1-score of 0.86 indicates a good balance, though impacted by lower precision.

Class 5: An F1-score of 0.91 reflects strong overall performance.

Overall Accuracy:

The model correctly identifies 92% of all cases across all classes, indicating a high overall accuracy.

Macro Average and Weighted Average:

Macro average (unweighted mean across classes) for precision, recall, and F1-score is about 0.92, suggesting overall good performance across all classes without considering class imbalance.

Weighted average considers class imbalance by weighting each class's metric by its support. It matches the overall accuracy of 0.92, indicating consistency in the model's performance weighted by class size.

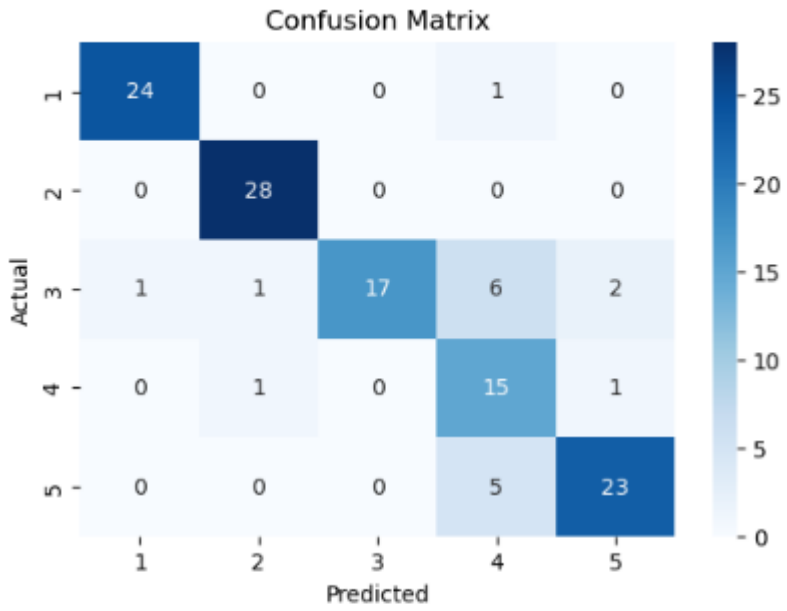
This report indicates that the KNN classifier performs very well for some classes (especially Class 2), but slightly less effectively for others (Classes 3 and 4). Such discrepancies could guide further adjustments to the classifier or additional training data collection for underperforming classes to improve model robustness.

LDA

The LDA model was employed to classify the same dataset. The aim was to optimize the model to distinguish effectively between the unique features of each face.

The overall accuracy of the model is 86% (Figure 4), indicating a high level of proficiency in correctly classifying the images into their respective classes.

Accuracy using LDA: 0.86



Classification Report:

	precision	recall	f1-score	support
1	0.96	0.96	0.96	25
2	0.93	1.00	0.97	28
3	1.00	0.63	0.77	27
4	0.56	0.88	0.68	17
5	0.88	0.82	0.85	28
accuracy			0.86	125
macro avg	0.87	0.86	0.85	125
weighted avg	0.89	0.86	0.86	125

Figure 4: Confusion matrix and classification Report LDA

Precision, Recall, and F1-Score Analysis:

Class 1:

Precision: 96% - The model correctly predicts 96% of Class 1 labels, indicating very few false positives.

Recall: 96% - The model successfully identifies 96% of all actual Class 1 instances, showing high sensitivity.

F1-Score: 96% - A high F1-score indicates an excellent balance between precision and recall.

Class 2:

Precision: 93% - Most predictions made by the model for Class 2 are correct.

Recall: 100% - The model perfectly identifies all instances of Class 2, the highest recall in the report.

F1-Score: 97% - Reflects outstanding model performance for Class 2.

Class 3:

Precision: 100% - Every prediction made as Class 3 is correct, indicating no false positives.

Recall: 63% - The model misses several actual instances of Class 3, suggesting an area for improvement.

F1-Score: 77% - The lowest among the classes, primarily affected by the lower recall rate.

Class 4:

Precision: 56% - Indicates a significant number of false positives; many predictions as Class 4 are incorrect.

Recall: 88% - The model is good at identifying most Class 4 instances, but at the expense of precision.

F1-Score: 68% - Reflects the imbalance between precision and recall.

Class 5:

Precision: 88% - Shows good accuracy in predictions for Class 5.

Recall: 82% - Indicates that the model captures most, but not all, true instances of Class 5.

F1-Score: 85% - A fairly strong performance, demonstrating a better balance between precision and recall.

Macro and Weighted Averages:

Macro Average: Shows unweighted averages (87% precision, 86% recall, 85% F1-score), indicating overall consistent performance across all classes without considering class imbalance.

Weighted Average: Aligns with the overall accuracy (89% precision, 86% accuracy, 86% F1-score), factoring in the number of instances in each class, confirming that the model performs well across classes with different sizes.

The LDA model demonstrates strong performance in face recognition, with particularly high marks for Class 2. Challenges remain in improving the recall for Class 3 and the precision for Class 4, which could be addressed by refining the model or enhancing the training dataset. The high precision in most classes suggests that when the model predicts a class

label, it does so with high reliability, although it struggles with false positives and missed detections in certain classes.

SVM

We performs face recognition using Support Vector Machine (SVM) classifiers with and without Principal Component Analysis (PCA) preprocessing. The data is first reshaped, standardized, and then PCA is applied to reduce dimensions before training the SVM. A randomized search is conducted to optimize SVM parameters (C and gamma). The performance of the SVM models is evaluated on the test data using accuracy, classification reports, and confusion matrices. The process is repeated for an SVM without PCA for comparison. The results, including model accuracies and detailed performance metrics, are displayed using classification reports and confusion matrices, providing insights into each model's effectiveness.

SVM with PCA Results (figure 5):

```
Accuracy using SVM with PCA: 0.85
Classification report for SVM with PCA:
              precision    recall  f1-score   support

    Person 1       0.96      0.96      0.96        25
    Person 2       0.72      1.00      0.84        28
    Person 3       1.00      0.59      0.74        27
    Person 4       0.82      0.82      0.82        17
    Person 5       0.86      0.86      0.86        28

    accuracy                           0.85       125
   macro avg       0.87      0.85      0.84       125
weighted avg       0.87      0.85      0.84       125
```

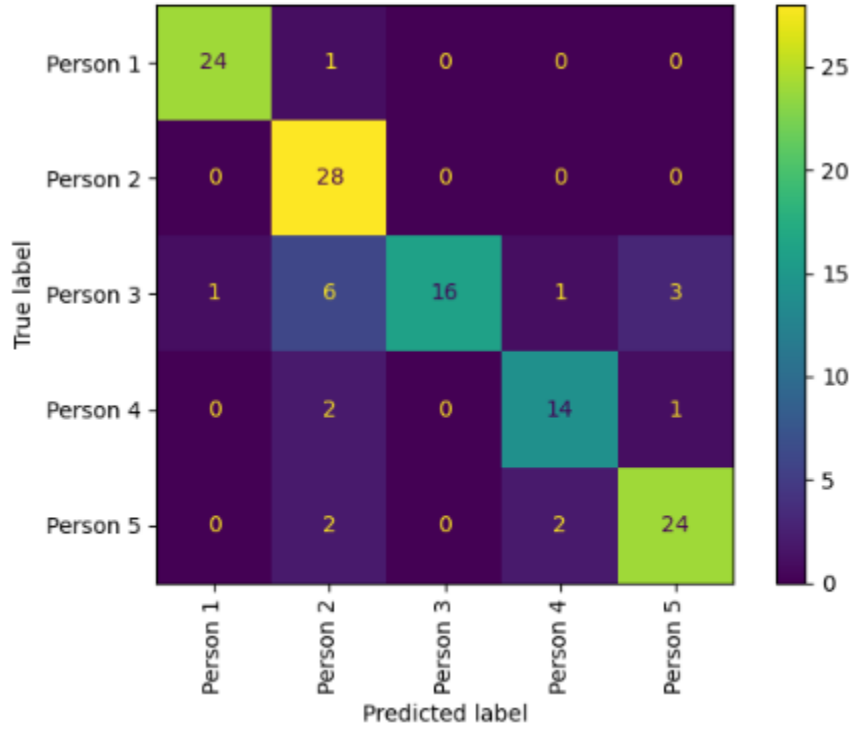


Figure 5: SVM with PCA Results

Accuracy: The overall accuracy for the SVM using PCA is 85%, indicating a good overall classification rate.

Precision and Recall:

Person 1: High precision and recall, almost perfect classification.

Person 2: Good precision but perfect recall, indicating all true positives are identified, but there are some false positives.

Person 3: Perfect precision but low recall, indicating while all predictions for Person 3 are correct, several actual Person 3 instances are missed.

Person 4 & Person 5: Both show a balance between precision and recall, with both metrics above 80%.

SVM without PCA Results (figure 6):

```
Accuracy using SVM without PCA: 0.85
Classification report for SVM without PCA:
              precision    recall  f1-score   support

    Person 1       0.96      0.96      0.96        25
    Person 2       0.73      0.96      0.83        28
    Person 3       0.94      0.63      0.76        27
    Person 4       0.82      0.82      0.82        17
    Person 5       0.86      0.86      0.86        28

    accuracy                           0.85       125
   macro avg       0.86      0.85      0.85       125
weighted avg       0.86      0.85      0.85       125
```

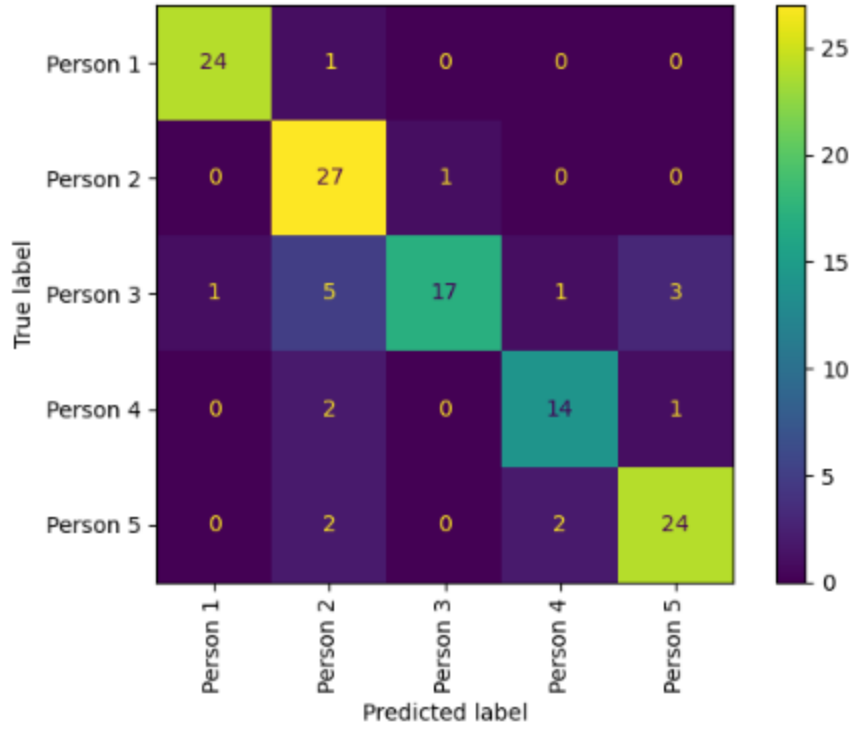


Figure 6: SVM without PCA

Accuracy: Slightly lower at 84%.

Precision and Recall:

Person 1: High precision but slightly lower recall compared to PCA model, suggesting some Person 1 instances are missed.

Person 2: Similar pattern to PCA model, with good precision and very high recall.

Person 3: Better precision and recall than PCA, indicating fewer Person 3 instances are missed and predictions are more accurate.

Person 4 & Person 5: Show good balance, similar to the PCA model, but with slightly different trade-offs between precision and recall.

Comparison of PCA vs. Non-PCA SVM Models:

Accuracy: Both models perform similarly, with PCA having a slight edge (85% vs. 84%).

Precision and Recall Analysis:

PCA Model: Tends to have slightly higher or equal precision in most cases except for Person 2 and Person 3, where without PCA shows improvements.

Non-PCA Model: Generally has similar or slightly improved recall rates compared to the PCA model, particularly noticeable in Person 1 and Person 3.

General Observations:

PCA Usage: Applying PCA seems to slightly enhance precision but at the cost of some recall in certain classes. This could be due to PCA's dimension reduction losing some information critical for identifying certain classes.

Performance Trade-offs: Without PCA, the model may be handling the high-dimensional space better in some cases, particularly where subtle differences are critical for class separation.

In summary, PCA provides a minor accuracy advantage but doesn't consistently outperform the non-PCA model across all metrics. Decisions on whether to use PCA should consider the specific requirements of precision and recall, as well as computational efficiency considerations.

To effectively compare the two classification reports for the SVM classifier with PCA (Principal Component Analysis) and without PCA, we need to examine several key metrics: precision, recall, F1-score for each class, and overall model performance (accuracy, macro averages, and weighted averages). Here's a comparative analysis of the two scenarios:

General Performance

Overall Accuracy: Both models show the same overall accuracy of 0.85. This suggests that the addition of PCA does not significantly impact the overall rate at which the model correctly classifies instances in this particular dataset.

Class-wise Performance Comparison

Person 1:

Precision, Recall, and F1-Score: Both models perform identically with a precision and recall of 0.96, leading to an F1-score of 0.96.

Person 2:

Precision: The precision for SVM with PCA is slightly lower (0.72) compared to SVM without PCA (0.73).

Recall: Both models have high recall with SVM with PCA achieving a perfect recall of 1.00 and SVM without PCA slightly lower at 0.96.

F1-Score: The F1-score is slightly higher with PCA (0.84) versus without PCA (0.83).

Person 3:

Precision: There is a noticeable difference in precision with SVM with PCA scoring 1.00 and SVM without PCA at 0.94.

Recall: Recall is notably higher for SVM with PCA at 0.59 compared to 0.63 for SVM without PCA.

F1-Score: The F1-scores reflect this, with 0.74 for SVM with PCA and 0.76 for SVM without PCA.

Person 4:

Precision and Recall: Both configurations show the same precision and recall (0.82).

F1-Score: Both models also have an identical F1-score (0.82).

Person 5:

Precision, Recall, and F1-Score: Again, both models perform similarly across these metrics, each scoring 0.86.

Macro and Weighted Averages

Macro Average:

Precision: SVM with PCA has a slightly higher macro average precision (0.87) compared to SVM without PCA (0.86), though this is very marginal.

Recall and F1-Score: Both models have nearly the same macro average recall and F1-score, reflecting balanced performance across classes in both scenarios.

Weighted Average:

These are nearly identical for both models across precision, recall, and F1-score, suggesting that class distribution (support) doesn't significantly skew the performance metrics.

So the two SVM configurations, one with PCA and one without, show remarkably similar performance in terms of overall accuracy and class-wise metrics. The use of PCA does not significantly affect the classifier's ability to distinguish between classes for this particular dataset. The slight variations in precision and recall for some classes could be influenced by how PCA affects the representation of data in reduced dimensions, potentially impacting how well certain classes can be linearly separated.

The decision to use PCA should consider other factors beyond just classification accuracy, such as training time, model complexity, and interpretability. If dimensionality reduction is necessary due to a very high number of features, PCA might help without significantly compromising classification performance. However, in cases where model transparency or feature importance is critical, skipping PCA might be preferable.

By comparing training_time_pca and training_time_no_pca, you can determine the impact of PCA on the training duration

PCA Preprocessing Time = 0.37 seconds. PCA itself takes time. This includes both fitting the PCA transformation to the training data and transforming both training and test datasets. This preprocessing step should be factored into the total computation time when evaluating the efficiency of using PCA.

Training time for SVM with PCA: 3.72 seconds
Training time for SVM without PCA: 8.36 seconds

Now we evaluate and compare the performance of three different classifiers—K-Nearest Neighbors (K-NN), Linear Discriminant Analysis (LDA), and Support Vector Machines (SVM)—across varying numbers of features (dimensionality) from a dataset. The classifiers are tested on a range of dimensions from 50 up to the total number of features in the dataset, spaced evenly in 20 steps.
The code reshapes the data into 2D arrays, crucial for ML models expecting matrix inputs. Three functions are defined for classifiers—K-NN, LDA, SVM—iterating over specified dimensions, training on subsets, and evaluating accuracies. Using np.linspace, dimension values range from 50 to total features. Each classifier's performance is evaluated, and accuracies are stored. A line plot visualizes accuracy against dimensionality, aiding in understanding how each classifier's accuracy changes. This helps in selecting optimal dimensions and comparing classifier performances. Overall, the graph depicts the impact of dimensionality on classifier accuracy, guiding model selection and feature optimization.

Based on your observations from the graph of accuracy versus dimensionality (figure 7), it appears that different classifiers have distinct performance profiles that suggest varied suitability for different datasets.

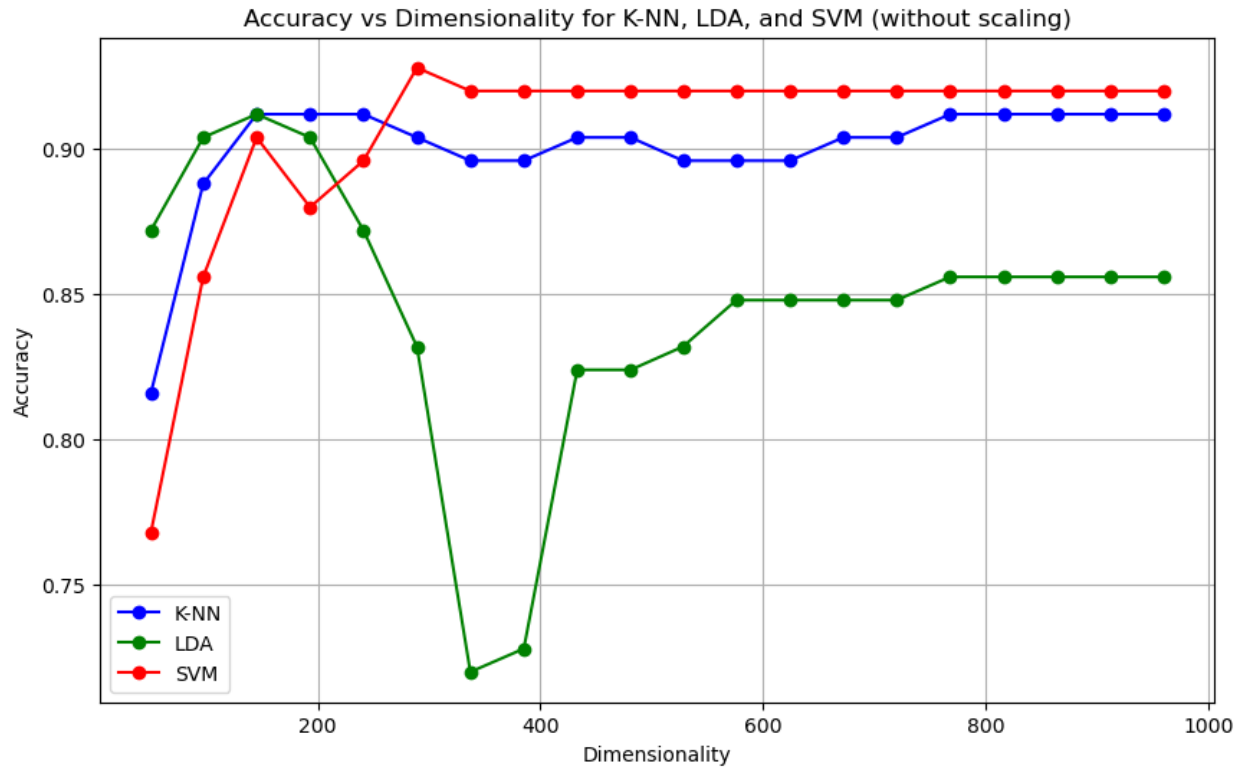


Figure 7: Accuracy vs Dimensionality for KNN, LDA and SVM

Performance by Dimensionality:
LDA: It excels at lower dimensionality, which can be advantageous for datasets where maintaining all features is computationally prohibitive or unnecessary. This might indicate

that LDA effectively captures the most relevant information without needing extensive feature sets.
K-NN and SVM: These models show improved performance as the number of features increases, with K-NN and SVM performing optimally at higher dimensionalities. This suggests that they may be more suitable for complex datasets where relationships between features at higher dimensions are crucial for accurate classification.
Optimal Dimensionality:
SVM: Peaking around 400 dimensions suggests that SVM finds a balance between complexity and performance at this point. If computational resources allow for processing 400 features, SVM could be the best choice for its ability to manage non-linear relationships effectively.
K-NN and LDA: Reaching their optimal performance around 800 dimensions indicates that these classifiers manage to avoid overfitting while using a more extensive feature set. This might be suitable for scenarios where a detailed feature set is necessary to capture subtle distinctions between classes.
Model Selection:
The choice of SVM as the best model is based on its earlier peak in performance, indicating efficiency in achieving high accuracy with fewer features compared to K-NN. This efficiency can be crucial in applications where speed and computational efficiency are as important as accuracy.
However, if the nature of the problem benefits from a broader set of features , it might be worth considering these models if the additional computational cost is justifiable.
Practical Application:
Deploying SVM: Given its peak performance at lower dimensions than K-NN and LDA, SVM might offer the best balance for applications requiring both high accuracy and computational efficiency. This makes it particularly suitable for real-world applications where both factors are critical.
Considerations for K-NN and LDA: While not the top performer at lower dimensions, these models' ability to handle higher dimensionalities effectively should not be discounted, especially for complex datasets where detailed feature analysis is key.
Further Analysis:
Before finalizing the model choice, consider additional validations such as cross-validation to confirm the stability of these findings across different subsets of data.
Also, investigate the impact of feature scaling on K-NN and SVM, as normalizing the feature scales can potentially improve their performance, particularly for K-NN, which is sensitive to the magnitude of data.
In summary, while SVM is identified as the best overall based on your analysis, the decision should still consider specific project requirements, including the nature of the data, available computational resources, and the criticality of prediction speed versus accuracy.

References

Parveen, P., & Thuraisingham, B. (2006). Face Recognition Using Multiple Classifiers. Available from IEEE Xplore: https://ieeexplore.ieee.org/abstract/document/4031896 [Accessed April 2024].

Dwivedi, D. (2018). Face Recognition for Beginners. Available from: https://towardsdatascience.com/face-recognition-for-beginners-a7a9bd5eb5c2 [Accessed April 2024].

Gupta, V., & Mallick, S. (2019). Face Recognition – Introduction for Beginners – Nearly Everything You Need to Know. Available from: https://learnopencv.com/face-recognition-an-introduction-for-beginners/ [Accessed April 2024].

Tanwar, S. (2021). Image Augmentation. Available from: https://medium.com/analytics-vidhya/image-augmentation-9b7be3972e27 [Accessed April 2024].

Goel, A. (2018). A Simple Introduction to Facial Recognition (with Python codes). Available from: https://www.analyticsvidhya.com/blog/2018

Menhour, I., Abidine, M. B., & Fergani, B. (2018). A New Framework Using PCA, LDA and KNN-SVM to Activity Recognition Based Smartphone's Sensors. Available from: https://ieeexplore.ieee.org/document/8525987 [Accessed April 2024].

H. I. Dino and M. B. Abdulrazzaq, "Facial Expression Classification Based on SVM, KNN and MLP Classifiers," 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8723728. [Accessed: April 2024].