Blake McFarlane
Homework 2

Implementing Machine Learning for Face Recognition

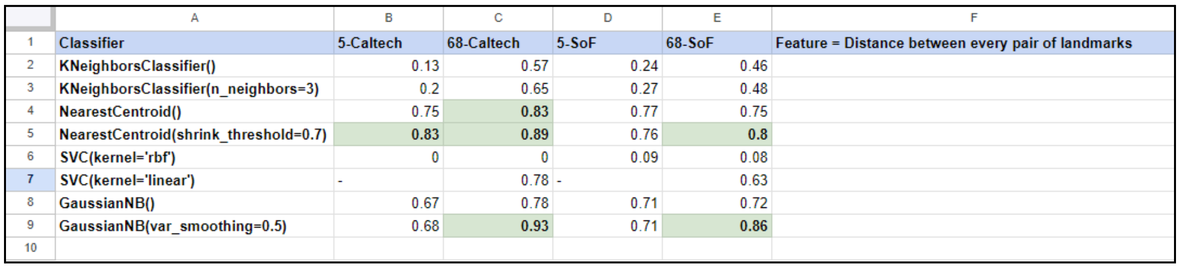
I. Introduction

The goal of this assignment was to implement and improve a simple face recognition system using the Caltech Faces Dataset and Specs on Faces (SoF) Dataset. The objective was to explore different machine learning classifiers and feature extraction techniques to achieve an authentication accuracy of at least 80% on each dataset. This report summarizes the process of classifier selection, parameter tuning, and feature representation modification, focusing on the experimental results and insights gained.

II. Classifier Exploration

As a beginner with Scikit-learn, I first familiarized myself with its documentation and explored various classifiers, focusing on adjusting their parameters to improve accuracy. I created a Google spreadsheet to track the performance of each classifier across the four datasets provided, allowing for easy comparison of results.

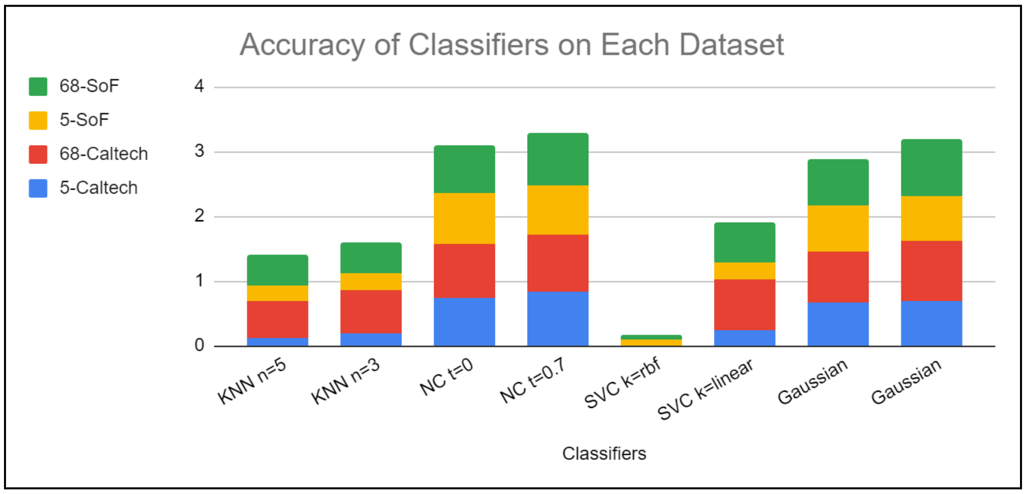
Figure 1. A screenshot of the spreadsheet after experimentation



	A	B	C	D	E	F
1	Classifier	5-Caltech	68-Caltech	5-SoF	68-SoF	Feature = Distance between every pair of landmarks
2	KNeighborsClassifier()	0.13	0.57	0.24	0.46	
3	KNeighborsClassifier(n_neighbors=3)	0.2	0.65	0.27	0.48	
4	NearestCentroid()	0.75	0.83	0.77	0.75	
5	NearestCentroid(shrink_threshold=0.7)	0.83	0.89	0.76	0.8	
6	SVC(kernel='rbf')	0	0	0.09	0.08	
7	SVC(kernel='linear')	-	0.78	-	0.63	
8	GaussianNB()	0.67	0.78	0.71	0.72	
9	GaussianNB(var_smoothing=0.5)	0.68	0.93	0.71	0.86	
10						

As shown in Figure 1, aside from KNN, I primarily focused on NearestCentroid(), SVC(), and GaussianNB() classifiers. The Nearest Centroid classifier achieved the highest accuracy across all datasets by assigning data points to the nearest centroid, which is the mean vector of each class. By setting a shrink threshold of 0.7, I reduced the influence of irrelevant features, contributing to its success even in datasets with fewer features.

Figure 2 shows how each classifier ranked up.



The GaussianNB performed well, achieving 93% accuracy on the 68-Caltech dataset and 86% on the 68-SoF dataset with its var_smoothing parameter set to 0.5. I also experimented with SVC; the 'linear' kernel yielded 78% accuracy on the 68-Caltech dataset, while the 'rbf' kernel struggled, often resulting in lower accuracy. I suspect the 'rbf' kernel was not well-suited for the smaller datasets, leading to potential overfitting or inadequate feature separation.

III. Classifier Parameters Tuning

Below is a list of the classifiers I chose to work with along with a brief summary of what went into tuning their paramaters. Keep in mind these results were without making changes to the feature extractor which created a feature space using the Euclidean distance.

- NearestCentroid: Adjusting the shrink_threshold significantly impacted the model's accuracy. A threshold of 0.7 yielded the best results, with the classifier performing consistently well across multiple datasets.
- GaussianNB: The var_smoothing parameter had a noticeable impact, with a value of 0.5 resulting in higher accuracy across the larger datasets (68-Caltech and 68-SoF).
- SVC: By reducing the n_neighbors parameter from its default value of 5 to 3, I observed a moderate improvement in accuracy.

IV. Feature Representation Modification

Initially, the provided code used Euclidean distances between every landmark pair. To explore alternative features, I focused on the 5-landmark datasets to keep things simple. Below is a list of a few of the feature extraction methods I took along with my findings..

- Areas: I used the landmarks to calculate several facial areas including the face itself. I calculated this by measuring the distance between the outer points of each eye and the mouth. Figure 3 shows how I was able to calculate 4 features unique to each user.
- Facial Ratios: I experimented with several ratios such as taking the eye widths and comparing them to the gap in between the eyes. I also used ratios to Areas.
- Landmark Displacement: After plotting several landmarks I noticed they were not all centered. This observation led me to use the origin of the graph as a reference point when calculating Euclidean distance.

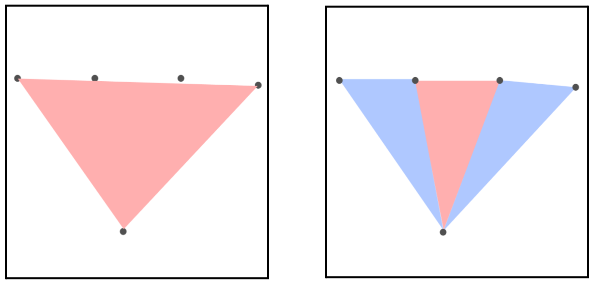


Figure 3. A visual representation of some area features

V. Summary and Conclusion

Overall, this project was a great learning experience, especially since it was my first time working with machine learning and face recognition systems. I experimented with different classifiers and feature extraction techniques, like calculating facial areas and ratios, which gave me a much deeper understanding of how models operate. Through trial and error, I found that classifiers like Nearest Centroid and GaussianNB, when properly tuned, could produce impressive results, consistently surpassing the 80% accuracy target. While SVC didn't perform as well with its default settings, I gained valuable insights into how parameter adjustments can make a big difference. This project not only helped me build my technical skills, but it also gave me confidence in tackling more complex machine learning problems in the future.