

Stone Barrett

CSE 590: Introduction to Machine Learning

Section 53

Final Project (First Report)

## **Introduction**

For this project, I have been asked to choose the optimal model and parameters for classifying samples in a provided dataset. The provided dataset contains feature extraction data from over 2000 artworks from 8 different artists. The artists serve as the classes, and the model I select and train should most accurately classify the artworks compared to the other options. I have been asked to choose from many different tools this course has covered in previous assignments. I am asked to test the following:

- 2 options for data normalization (preprocessing)
- Principle Component Analysis (PCA) with 2 options for number of components
- 2 options for clustering methods
- 4 options for classifier methods
- 2 ensemble methods
- K-fold cross validation
- Pipelines and grid search

The ensemble methods and k value for cross validation have been predetermined, but the remaining options are to be decided by me. As such, I have made a list of the methods and values I would like to test as a sort of map to plan out this project. I will be using the following:

- Preprocessing: Standard Scaling and Min Max Scaling
- PCA:  $n = 2$  and  $n = 8$
- Clustering: k-means and agglomerative
- Classification: K-Nearest Neighbors (KNN), Logistic Regression, Linear Support Vector Machines (SVM), and Random Forest
- Ensemble: Bagging and Adaptive Boost
- K-fold cross validation with  $k = 4$
- Pipelines and grid search

By multiplying the count of options at each step ( $2 \times 2 \times 2 \times 4 \times 2$ ) we get a product representative of the total number of unique combinations possible with these options (which does not even consider combination options in ensemble, pipelines, and grid search). To run at

least 64 iterations of these tests blindly would essentially be brute forcing to find the highest accuracy and would not be an efficient use of time or computational resources. Therefore, I have planned out how I will proceed step-by-step in the following flowchart:

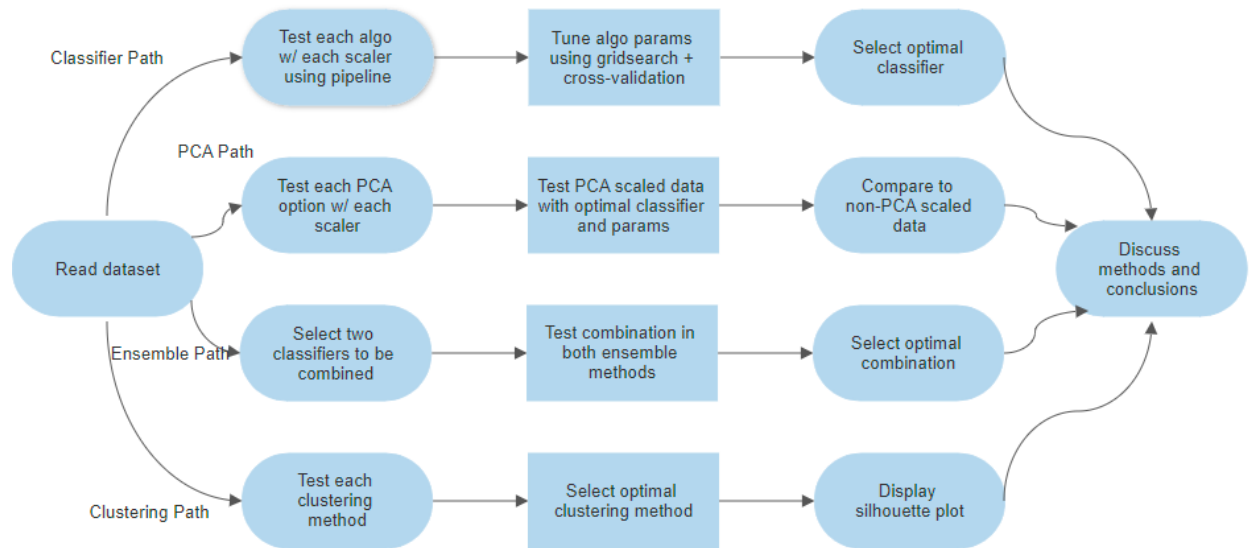


Figure F.0

With this roadmap, I have planned for there to be 6 distinct sections of this project:

- 1) Introduction
- 2) Testing each classification algorithm of my choosing with each data normalization method of my choosing utilizing grid search, pipelines, and cross-validation
- 3) Testing each PCA option of my choosing with each data normalization method of my choosing
- 4) Testing each required ensemble method with classification algorithms of my choosing
- 5) Testing each clustering method of my choosing
- 6) Discussion and wrap-up

Each of these sections will focus on one particular set of requirements for this project but will reference each other frequently, as comparison is key when attempting to locate an optimal strategy for classification. Also with this plan, I estimate there will be roughly 8 tests ran which is a huge improvement over the aforementioned 64. Of course, however, some of these tests will encapsulate multiple algorithms and/or multiple parameters, while still avoiding the idea of brute forcing or throwing everything into one grid search or pipeline. This plan should work through

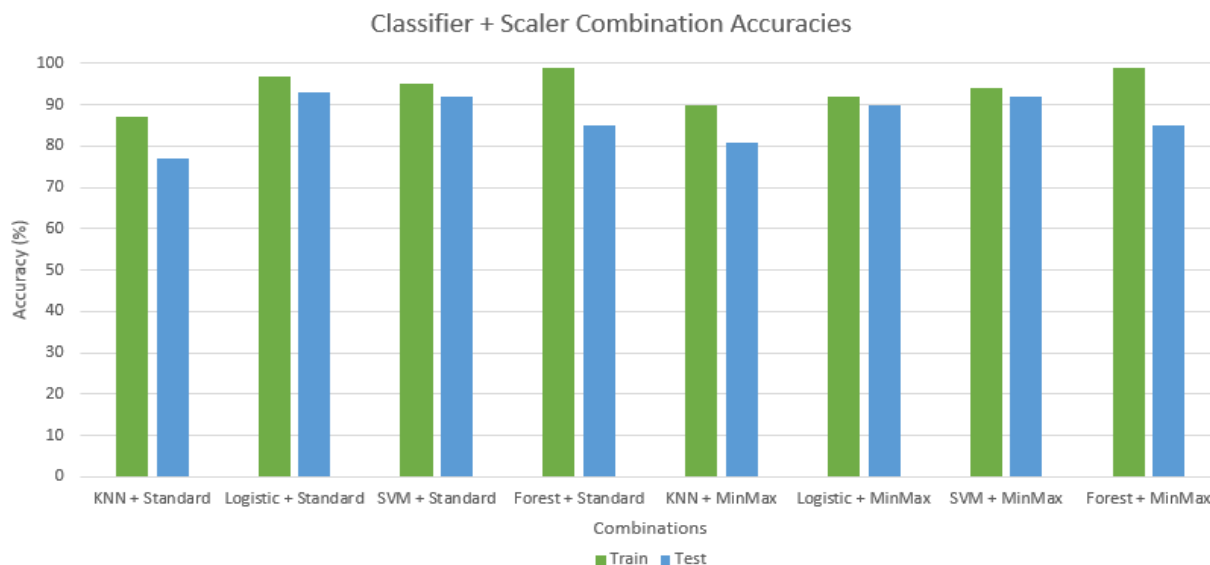
each set of options intelligently and weed out suboptimal approaches, with each pathway containing a sort of multidimensional tournament bracket. Let us begin.

## **Part 1: Classification Algorithms**

As stated in the introduction, I have chosen 4 classification methods and 2 preprocessing methods. The classification methods are KNN, Logistic Regression, SVM, and Random Forest. The preprocessing methods are Standard and MinMax. This section will focus on running these options through a pipeline with default parameters, then using grid search and cross-validation to find optimal parameters on whichever method scored highest in default. Once there is a clear highest accuracy, those methods and parameters will be selected to move forward with.

Train score for Standard + KNN:	0.87
Test score for Standard + KNN:	0.77
Train score for Standard + Logistic:	0.97
Test score for Standard + Logistic:	0.93
Train score for Standard + SVM:	0.95
Test score for Standard + SVM:	0.92
Train score for Standard + Forest:	0.99
Test score for Standard + Forest:	0.85
Train score for MinMax + KNN:	0.90
Test score for MinMax + KNN:	0.81
Train score for MinMax + Logistic:	0.92
Test score for MinMax + Logistic:	0.90
Train score for MinMax + SVM:	0.94
Test score for MinMax + SVM:	0.92
Train score for MinMax + Forest:	0.99
Test score for MinMax + Forest:	0.85

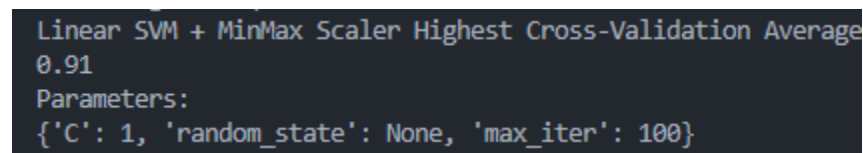
*Figure F.1*



*Figure F.2*

Figures F.1 and F.2 show the pipelines and their accuracy scores on the training and test datasets. These are all relatively high scores, though some stand above the rest as noteworthy for one reason or another. The first thing of note is Random Forest's massive 99% accuracy on the training set with each scaler. This would be exciting news; however, the associated testing accuracy is 14 points lower. This is an instance of overfitting – seemingly the most glaring example of the bunch. On the other hand, there do not seem to be any examples of underfitting, as the lowest training score is 87%. More importantly, there are pipelines that returned very promising results. We want to look for the smallest difference between training and testing score. There are a some 4% and even 3% differences, but there are also two 2% differences. MinMax + SVM and MinMax + Logistic both returned scores with a 2% difference, which is the smallest difference in scores. Both of these pipelines' scores are also above 90%, which means these models not only are not overfitting, but are also high in generalization and learning. We want to take the higher scores of the two, which happens to be MinMax + SVM at 94% and 92%. Now that the optimal combination of classification algorithm and scaler method have been chosen, the next step is to use cross-validation and grid search to find the optimal parameters.

Thankfully, Linear SVM only has a couple parameters to be tuned, so we will not have many nested for-loops in the grid search. If we were testing something like Random Forest, that would not be the case. The parameters to be considered in this scenario are C (regularization), `random_state`, and `max_iter`. I will test C at .001, .01, .1, 1, 10, and 100. I will test `random_state` at None and 42. I will test `max_iter` at 100, 1000, 10000, 100000, and 1000000.



```
Linear SVM + MinMax Scaler Highest Cross-Validation Average
0.91
Parameters:
{'C': 1, 'random_state': None, 'max_iter': 100}
```

*Figure F.3*

After 60 different versions were tested (6 C options x 2 `random_state` options x 5 `max_iter` options) and 240 models trained (60 versions x 4 cross fold sections), one combination had the highest average cross-validation score at 91%. Based on these findings, we can say with some confidence that the optimal classification algorithm to be used on this type of data is the Linear Support Vector Machine Classifier with parameters C = 1, `random_state` = None, and `max_iter` = 100 and that the optimal preprocessing method for the data is the MinMax Scaler.

## **Part 2: Principal Component Analysis**

Now we will test modifying the data with PCA. As stated previously, I have decided to test PCA with `n_components` = 2 and `n_components` = 8. It is common practice to scale the data using Standard Scaler before performing dimensionality reduction with PCA, so we will have to abandon the MinMax Scaler for the time being.

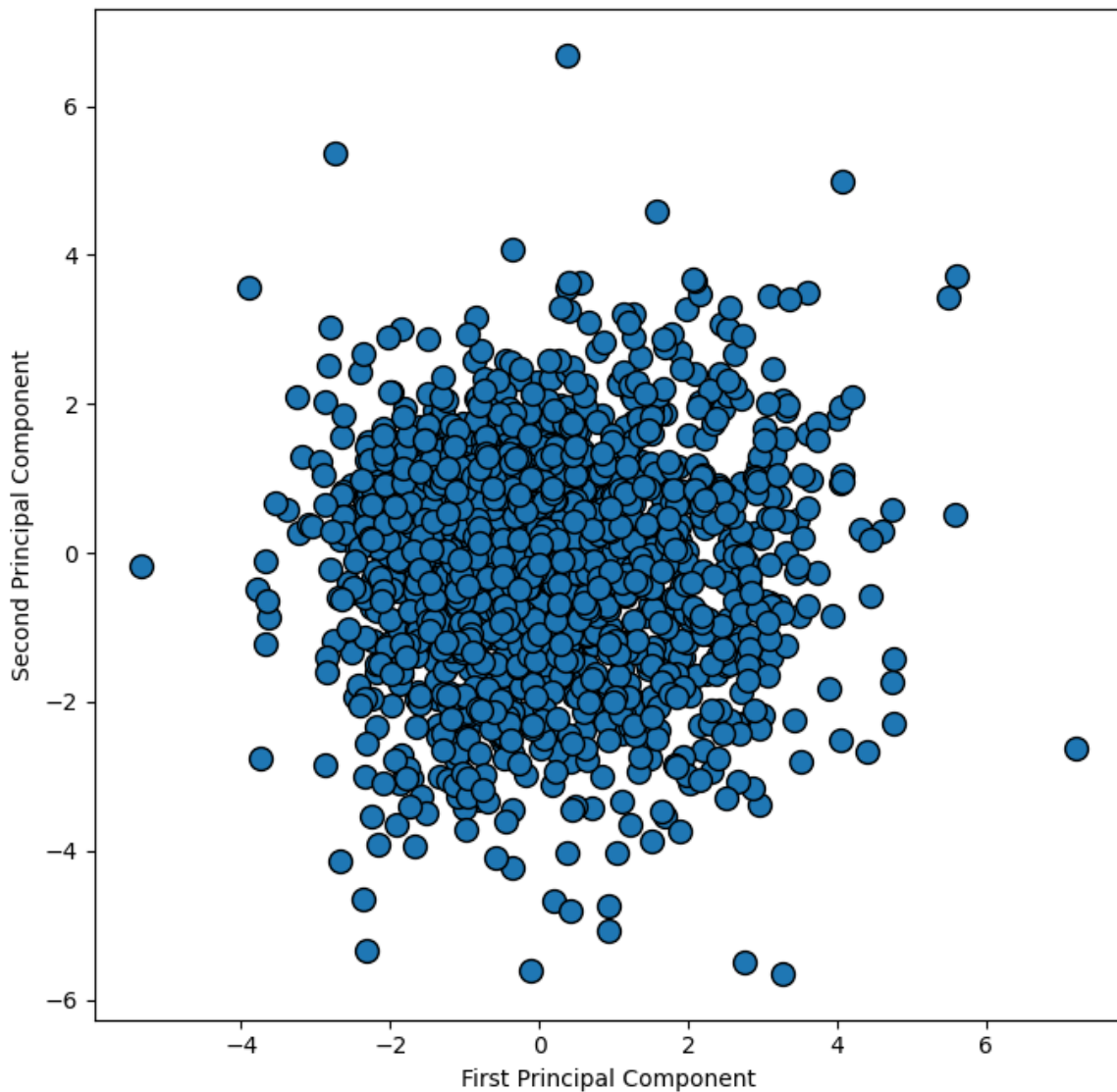
```
Original Shape: (1664, 64)
Reduced Shape: (1664, 2)
```

Figure F.4

As we can see in Figure F.4, PCA modified the data to focus on two principal components. We can actually see the principal components themselves in Figure F.5.

```
PCA Component Shape: (2, 64)
PCA Components:
[[-1.36928791e-17 -6.03821595e-02 -6.00704457e-02  2.70617787e-01
  1.27964724e-01 -2.03566851e-01  8.78179809e-02 -9.58590523e-02
 -8.54713165e-03 -2.55940992e-02  2.06827698e-02 -2.80777354e-02
  6.72846573e-02 -2.29292301e-02  0.00000000e+00  6.89869481e-02
 -1.92731234e-01 -2.04452385e-02 -1.68522890e-02 -1.03568372e-01
 -2.44737432e-01  1.04865898e-01  8.40201658e-02 -3.81548508e-02
 -5.43388694e-02  0.00000000e+00  0.00000000e+00 -4.97964248e-02
 -1.19117084e-01 -1.36284030e-01  3.57686772e-02 -4.93009514e-02
  8.88282703e-03 -4.69906072e-02  3.49154818e-01  0.00000000e+00
  3.60062470e-01  1.03393063e-01 -1.40297891e-01  0.00000000e+00
 -1.75747625e-01 -9.04271204e-02 -3.27085406e-02 -1.01589152e-01
  1.12944744e-01  2.51029449e-01  9.95474148e-02  3.23757502e-02
  1.19566931e-02 -2.72104279e-02  2.76748702e-01  1.43259889e-01
  1.85563997e-02 -6.36783803e-02 -6.61050248e-02 -3.26054314e-02
  2.13715564e-01  1.13245956e-01  1.42880788e-01  0.00000000e+00
 -6.63031279e-02  1.76430225e-01  0.00000000e+00  1.39596035e-01]
[[-1.58449914e-17  2.87963015e-01  5.31740237e-02 -1.36669265e-01
 -1.32612889e-01 -9.65447971e-03 -1.09562179e-01  2.50019286e-01
  3.89426123e-02  8.54546855e-02 -2.17977248e-03 -2.97730008e-02
  4.43624546e-02 -1.81961934e-01 -0.00000000e+00  2.72858117e-02
  5.89523515e-02 -1.96531561e-02 -2.25370290e-02 -7.93054401e-03
  6.98234231e-02 -1.24677897e-01 -1.12265552e-01 -7.28280364e-02
 -7.86485766e-02 -0.00000000e+00 -0.00000000e+00 -2.47654103e-02
 -6.43588637e-02 -4.45294985e-02  2.88975502e-01 -1.01841156e-01
 -5.10454257e-02 -2.12018689e-02 -5.69984688e-02 -0.00000000e+00
  5.22927678e-02  1.59915798e-01 -5.14182108e-02 -0.00000000e+00
  1.88963747e-02  2.87898935e-02 -5.27277341e-02 -7.15333804e-02
 -2.21598207e-01  5.08090062e-02  6.33874786e-02  8.97998518e-02
  4.48964195e-01 -2.24575782e-02  1.41995324e-01  3.18547635e-01
 -6.92419973e-02  1.21118641e-01 -1.04697823e-02  2.46368199e-01
  2.21532282e-01 -8.46491679e-04  9.71768919e-03 -0.00000000e+00
  6.97244780e-02 -1.77072691e-02 -0.00000000e+00 -1.99622706e-01]]
```

Figure F.5



*Figure F.6*

Since we are testing `n_components = 2` right now, we can actually visualize them on a two-dimensional scatterplot. Now, let us test the accuracy scores on the PCA-modified data using the optimal selection from Part 1.

```
Score on PCA Data: 0.36036036036036034
```

*Figure F.7*

This new score is alarmingly lower than the score from the non-PCA data. I imagine this has to do with removing many dimensions from a large dataset. Given that the data originally had 64 attributes to learn from, a lot of information has been removed to reduce the set to 2 attributes. Next, I will test `n_components = 8` to see if the higher count reserves some accuracy.

Score on PCA Data: 0.6864864864864865

Figure F.7

Given only 6 more dimensions, the score has rocketed back up to 69%. This makes me curious what a graph of score vs. number of dimensions would look like.

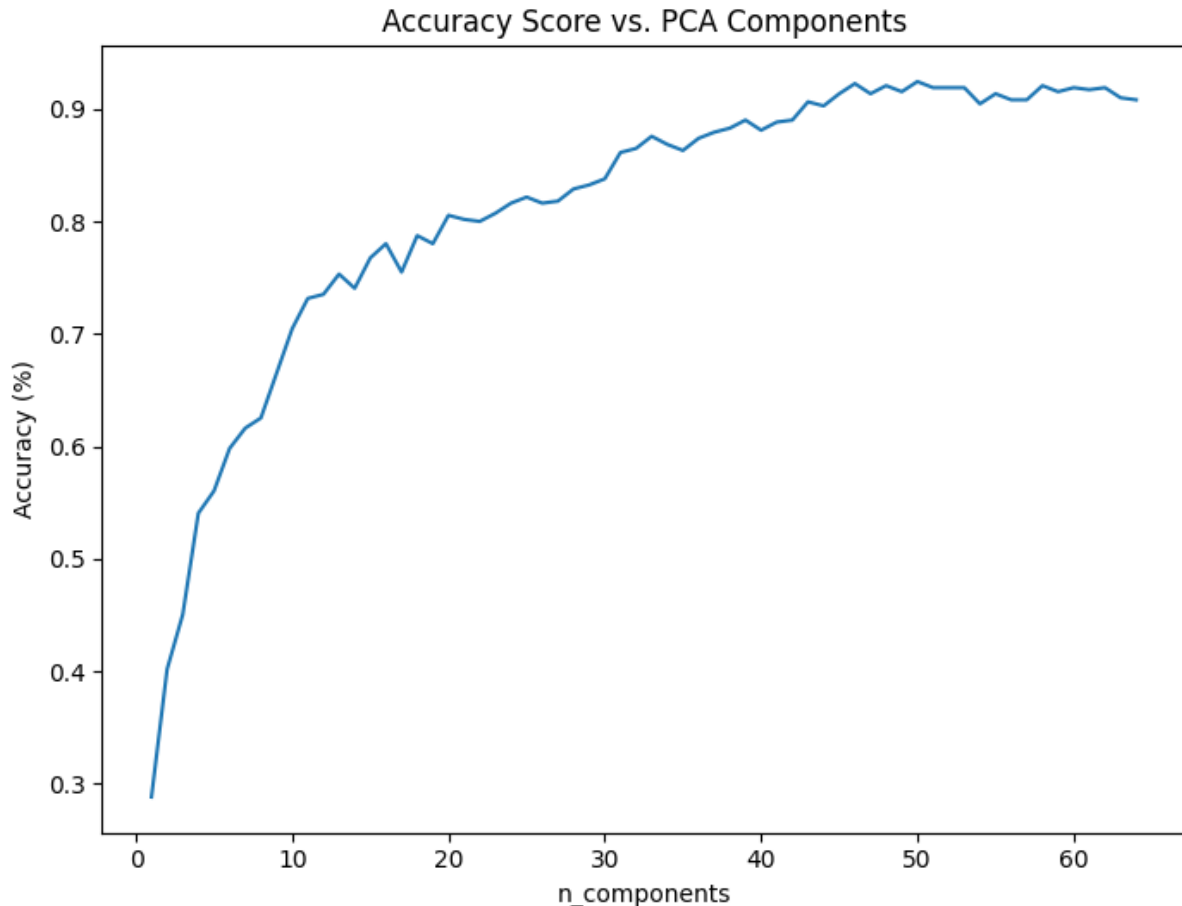


Figure F.8

Interestingly, Figure F.8 shows that the scores start to plateau around  $n\_components = 20$ . Perhaps the most important features for prediction can be located in only 20 of the 64 dimensions. If I understand correctly, the point of using PCA in a broad sense is to show that some of the least-important features in a set can actually throw off the predictions; it can show that homing in on a specific few attributes can actually improve accuracy. In this case, however, it seems that maximum accuracy is reached around the  $n\_components = 40$  mark. The accuracy does slightly dip after this, but not by more than a couple percentage points. I imagine that some features (or combinations of features) in this range are detrimental to prediction and that others are significantly beneficial.

## **Temporary Conclusion**

As this report is meant to be the first part of a complete project, I am asked to submit my findings at the halfway point. This is the halfway point for my plan of experiments. There are 2 Parts of my plan left to be explored: Ensemble Methods and Clustering. Hopefully, the forthcoming experiments continue to add to the discussion and findings generated thus far.

Regardless, at this moment, it seems that the Linear Support Vector Machine classifier with a specific set of parameters is optimal for classifying the samples in the given dataset. PCA could not improve the score higher than the non-PCA dataset, BUT the same high score can be reached with fewer attributes considered using PCA.

In the final version of this report (due one week from now) we will test ensemble methods (adaptive boosting and bagging), clustering methods (k-means and agglomerative), and (hopefully) viewing some misclassified samples to discuss why they may have been misclassified.