Digital Paleography of the Ávila Bible:
Classification of handwriting in a medieval manuscript

Alexander Berry
Brown University
Tuesday, December 3, 2019
Data set, code, figures, and reports are available at:
https://github.com/ABerry057/avila/

Introduction

The goal of this project was to investigate the Ávila Bible handwriting data set while practising the skills needed to implement a data science project from start to finish. The Ávila Bible is a medieval manuscript that was created in the 12th-century in both Italy and Spain by 12 copyists and is therefore a mix of different styles of writing and decoration. Researchers at the University of Cassino and Southern Lazio in Italy proposed using machine learning to better identify and understand portions of the manuscript. This is a classification problem with the goal of assigning a sample of text to one of the copyists according to the writing style. To that end, the researchers created a data set of over 20,000 data points by deriving features from 800 high-definition images of the pages of the manuscript (Fig. 1). The choice of features generated from the images was informed by the expertise of paleographers who work with similar manuscripts. The data set is published on the UCI ML Repository: https://archive.ics.uci.edu/ml/datasets/Avila.

Fig.1. Example of a feature derived from digital image of the manuscript, in this case the number of peaks in the pixel projection histogram on the horizontal axis for a row.

Exploratory Data Analysis

The first step in this project's data science pipeline was to perform exploratory data analysis (EDA) to get a better understanding of the data set. As this was a classification problem with 12 classes, one of the most important pieces of information to consider is the balance of the data set. I found the data set to be very imbalanced, with over 40% of the data points belonging to class 0 (Fig. 2). I imagine the reason for the imbalance of the classes is that there have been

one or maybe two copyists who wrote the majority of the text, with the other 10 copyists working on smaller sections or rewriting some of the original text.

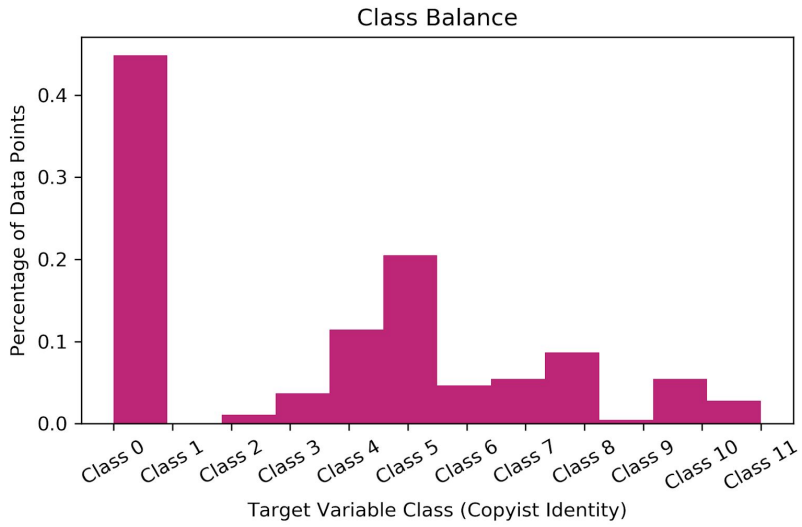

Fig. 2. Histogram visualizing the imbalanced distribution of copyist classes

In addition to visualizing the class balance as part of EDA, I wanted to investigate the distribution of some of the features that the researchers suggested would be informative in classifying scripts, based on the advice of domain experts in paleography. These features are exploitation, weight, and the number of peaks. I used box plots to visualize the distributions of each feature according to class. The resulting figures show distinct means and variances, which suggest these features will be useful for classification (Fig. 3, Fig. 4, Fig. 5). While I made visualizations for other features, these were the most informative.

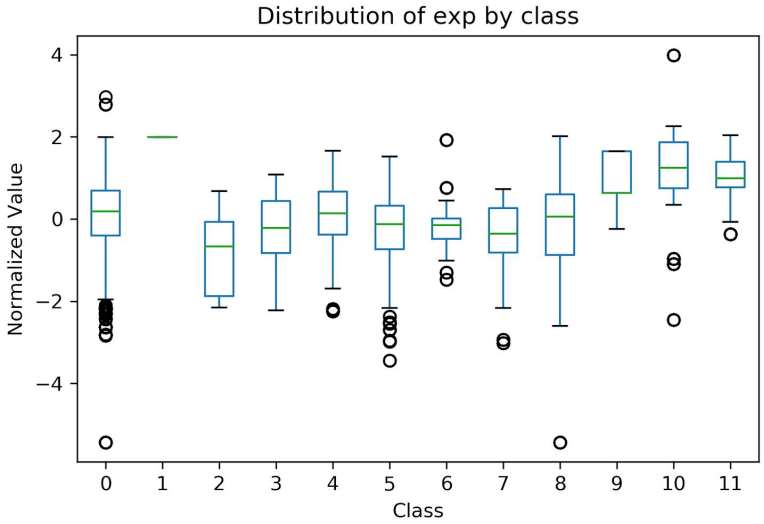

Fig. 3. Boxplots showing distribution of exploitation by class

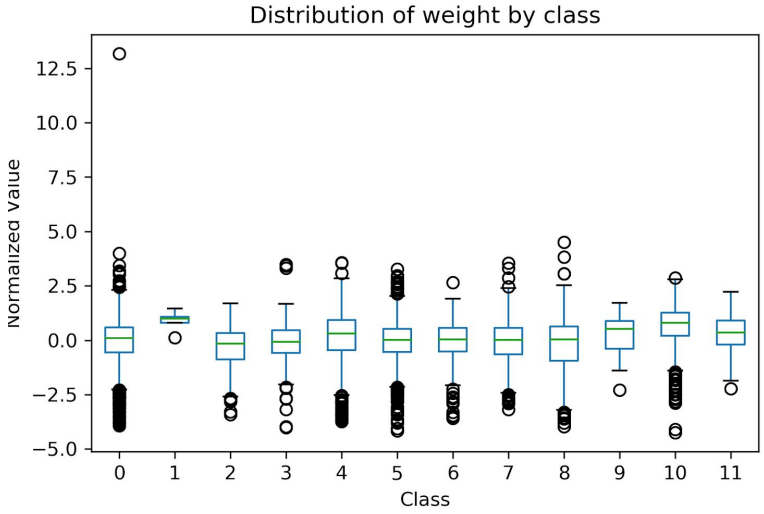

Fig. 4. Boxplots showing distribution of weight by class

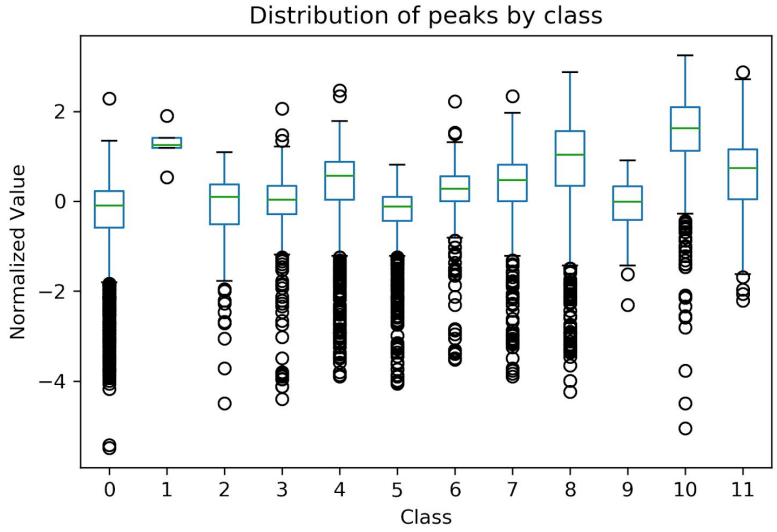

Fig. 5. Boxplots showing distribution of peaks by class

One final visualization I used to better understand the data set was t-distributed stochastic neighbor embedding which uses dimensionality reduction to exhibit pairwise relationship between data points. The resulting figure (Fig. 6) shows that there are some distinct clusters (classes 8 and 10, for example) but overall, the classes are intertwined and this led me to believe that the classification will be difficult.

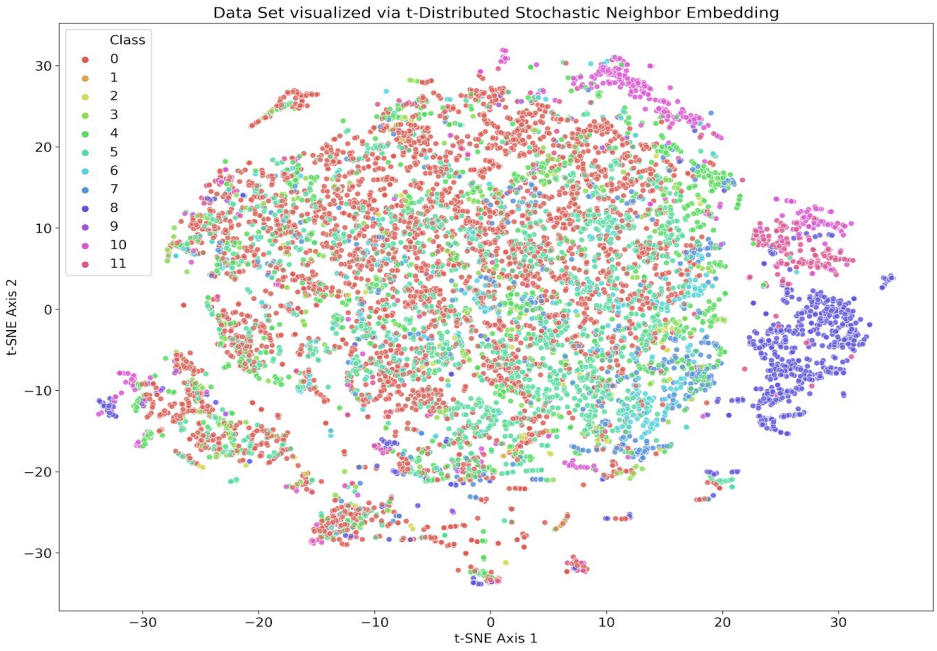

Fig.6. t-SNE visualization of data set

Methods

The next step in the pipeline was to preprocess the training data and then perform cross validation to select the best machine learning (ML) algorithm and corresponding hyperparameters. Because the data set is published with its features already having been normalized by Z-score, the only preprocessing I performed outside of the cross-validation pipeline was label encoding the class names from letters to integers 0-11. Inside the cross-validation pipeline, I first used standard scaling on the training set (in the event that the previous scaling was performed across the whole data set at once) and I then used the k-folds method to tune hyperparameters with a k value of 5. To account for uncertainty in the random splits for k-folds and for non-deterministic ML algorithms, I computed accuracy scores for 6 iterations (6 was chosen to account for uncertainty but minimize training time) of the cross-validation process and obtained the mean and standard deviation for the 6 scores (Fig. 8).

Because this is a multi-class classification problem, I used ML algorithms that are inherently multi-class (k-neighbors, random forest) or whose implementation in Scikit-learn support multi-class classification (logistic regression). The associated hyperparameters for these models are k, max depth and minimum samples split, and C, respectively. (Fig. 7) As an evaluation metric, I used the accuracy score because this problem is interested in the number of

correctly-classified points for all classes; there is no reason to value correctly-classified points in class 0 over points in class 8, for example.

Algorithm	Hyperparameter ranges	Hyperparameter descriptions
k-neighbors	k ∈ 2-15, linear	k is the number of neighbors to consider when classifying
logistic regression	C∈10e-4 - 10e3, logarithmic	C is the penalty for lasso regression
random forest	max_depth ϵ 18-22, linear min_sample_split ∈2-10, linear	Max depth is the max number of splits the model can make, min samples split is min samples required to split an internal node

Fig. 7. Summary of the hyperparameters of different ML algorithms

Results

Given the mean scores and standard deviations of the accuracy scores, I calculated the number of standard deviations the mean scores are away from the baseline score of about 40%. The number of standard deviations a mean accuracy is from the baseline describes the level of certainty that the accuracy is consistent. The resulting table (Fig. 8) shows the model with the highest accuracy to be random forest.

Algorithm	Standard deviations from baseline	Mean score on test set	Standard deviation	Best hyperparameters
k-neighbors	181.9797	0.7415	0.0018	k = 4
logistic regression	582.3914	0.5620	0.0003	C = 10
random forest	287.4444	0.9841	0.0020	max_depth = 21, min_samples_split = 4

Fig. 8. Summary of ML algorithms' performances

Given more time, I would average the results of the confusion matrices and feature importance tests for all of the 6 random forest models, but in this case I randomly chose one of the six for the analysis. To better understand the performance of this model, I considered the confusion matrices for its classification. Scikit-learn provides a multi-class confusion matrix method for models and using this resulted in the following matrices (for brevity, I only list the matrices for the classes with the most data points and the fewest data points, 0 and 1, respectively):

	True Positive	**True Negative**
Predicted Positive	4241	45
Predicted Negative	73	6078

Fig. 9. One-vs-all confusion matrix for class 0

	True Positive	**True Negative**
Predicted Positive	5	0
Predicted Negative	0	10432

Fig. 10. One-vs-all confusion matrix for class 1

These matrices exhibit generally favorable classification from the chosen random forest model. For class 0, the most populous class, the model avoids making too many false positives with a precision of 0.9898. For class 1, the model has a recall score of 1.0, meaning it does a good job of detecting all the instances of the scarce and difficult to classify class 1 (again, I am uncertain about the possibility of data leakage but I have done all that I can to control for it given the format of my data).

Given this model, I sought to calculate the global feature importance to assess if the features that appeared to be the most distinct during EDA were indeed the most useful in the classification performance of the model. Using the random permutation feature importance test, I generated boxplots (Fig. 11) that visualize impact the random permutation of each feature's values has on the model's performance.

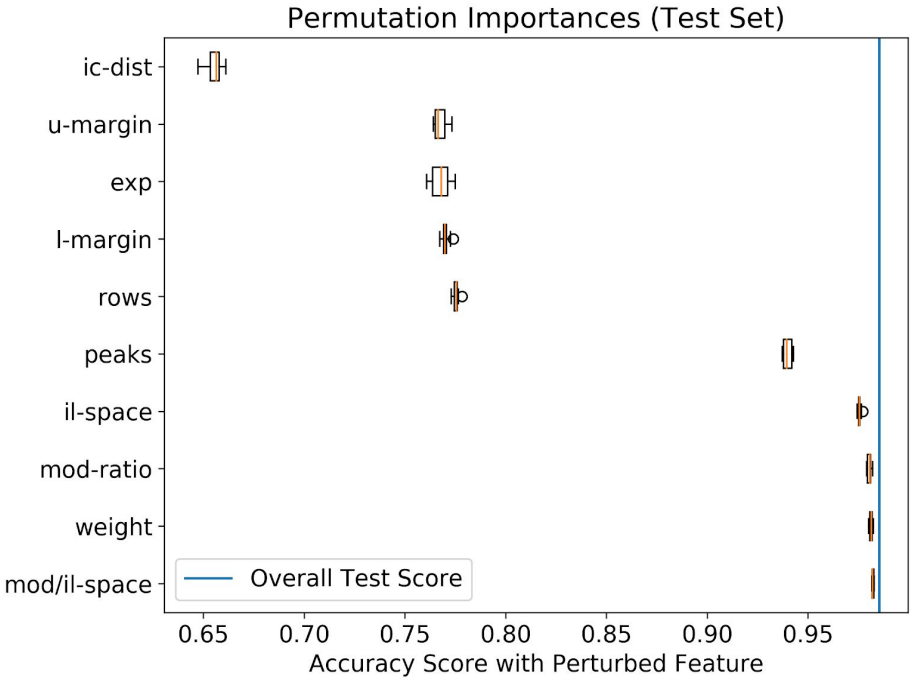

Fig. 11. Global feature importance via permutation test

Peaks and weight are not very important at all, which is not what I expected based on the EDA. This visualization suggests that intercolumnar distance is the most important feature, followed by upper margin, and exploitation. In real world terms, this suggests that the most distinctive features of a copyist's handwriting are how much distance the copyist leaves between columns of text on a page, how much distance the copyist leaves between the top edge of page and text, and how densely a copyist fills the vertical space of a page. While the other features do carry some importance, it is interesting to note that the most informative aspects of handwriting with regards to classification in this manuscript are features that are very easily measured and interpreted. Ideally, this model will allow for fast and reliable classification of copyist handwriting and could generalize to other manuscripts.

Outlook

If I were to take another look at this data set, there are a few different approaches I could take to address weaknesses in my analysis. One thing I would attempt would be to obtain the un-preprocessed data from the researchers. When I originally performed cross validation, I discovered I was dealing with data leakage because the way I was splitting my data into train, CV, and test sets meant that there was information from the training set included in the test set. The authors published their data preprocessed and already split into train and test, but I was originally shuffling these two sets and re-splitting. Respecting the original train and test split

provided by the authors seemed to correct for this data leakage, but I remain suspicious of the accuracy of the random forest, as the best test accuracy the researchers obtained in their work was 92.46%, and that was using a neural network with 100 hidden neurons and 1000 learning cycles. Again, having the un-preprocessed, unsplit data would allow me to better control for data leakage.

Another approach I would take would be to use a sequential "one-vs-all" plan to handle multi-class classifications in order of class balance. This would mean training 10 classifiers, one first to classify data points as either part of the most common class (0) or not, then another model to classify a data point as a part of either class 5 (and not 0), and so on. Each model in this sequence could be a different classifying algorithm with appropriately-tuned hyperparameters. This would require much more cross-validation and potentially more data for the last few models (the current data set only contains 10 points for class 1). However, this sequential model has the potential to have better performance because it breaks down the multiclass classification task into smaller binary classification tasks.

A final approach I would try would be to create new features from the existing features. The researchers who first created the data set did this when they generated a new feature from the modular ratio and the interlinear spacing, the ratio of the modular ratio to interlinear spacing. A possible new feature to generate would be the ratio of the weight to the exploitation, because both weight and exploitation measure the density of script along an axis. A feature combining both of these densities could describe the average density of a copyist's handwriting and might be predictive of copyist identities.

References

De Stefano, Claudio, Francesco Fontanella, Marilena Maniaci, and Alessandra Scotto di Freca. "A Method for Scribe Distinction in Medieval Manuscripts Using Page Layout Features." *ICIAP*, 2011, 393–402.

De Stefano, Marilena Maniaci, Francesco Fontanella, Alessandra Scotto di Freca. "Reliable writer identification in medieval manuscripts through page layout features: The 'Avila' Bible case." *Engineering Applications of Artificial Intelligence*, Volume 72, 2018, 99-110,