University of Houston
EDS 6342 – Introduction to Data Science

Classification Semester Project on the Avila Bible Dataset

Mansib Mursalin – 2254670
Jahnavi Chintala – 2149068
Mani Sabarish Akula –2148717
Asha Collie - 2163111

I. Introduction

Our group was tasked with choosing a substantial dataset and develop predictive models using various machine learning techniques. Our chosen dataset contains features extracted from 800 images of the *Avila Bible*. The Avila bible is a Latin copy of the Bible produced during the twelfth century by both Italy and Spain. Each sample in the dataset contains 11 features corresponding to a group of 4 consecutive rows averaged together. Each group of rows will henceforth be called a *pattern.* The features of each pattern represent characteristics that were inferred from a digital scan of each page. Rubricated letters and miniatures were excluded from the scans (*De Stefano, 2018*).

The paleographic analysis by De Stefano, et. al revealed that the Avila Bible was written by 12 different scribe monks. Our goal is to predict which scribe wrote a given pattern using the available features of the pattern. These features include intercolumnar distance, upper margin, lower margin, exploitation, row number, modular ratio, interlinear spacing, weight, peak number, modular ratio/interlinear spacing, and monk. The full dataset contains ~20,000 records and a description of each feature is found in Figure 1.

Figure 1. Description of Feature Variables

Variable	Description
Intercolumnar Distance	Distance between two columns of a page
Upper Margin	Distance between upper margin of the page and first line of text
Lower Margin	Distance between lower margin of the page and last line of text
Exploitation	Fraction of the column filled with ink. Formally computed as
Modular Ratio	Estimate of the dimension of the handwritten character
Interlinear Spacing	Distance between two rows, in pixels
Weight	Fraction of row filled with ink. Analogous to exploitation, but for a single row
Peak Number	Estimate of the number of characters in a row
Modular Ratio / Interlinear Spacing	Ratio of the two preceding attributes

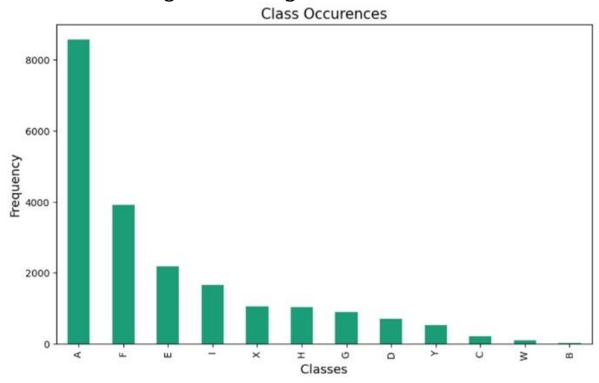
The breakdown of our analysis is as follows; Preprocessing of the data to prepare for modeling, the creation of several base models for preliminary analysis, performing variable selection on the data using several techniques, performing cluster analysis, and developing an ensemble of models for final predictions. Following our analysis is a brief discussion of our findings and thoughts after completion.

II. Pre-Processing

The dataset was downloaded directly from the University of California-Irvine's Machine Learning Repository. The data source's original format was 'txt' and was converted to csv for easier readability to a python data frame. Column names were not included in the file however, additional dataset documentation provided in the repository detailed this information and it was added during pre-processing.

From observation it was discovered that the original data source was that the set labeled for testing contained more samples than the training set. As a result, the two sets were combined to be re-split later, 70/30, by the team. We also noted that the distribution of the *monk* variable, containing 12 classes, was imbalanced. Some classes had thousands more instances than others and likewise one particular class accounted for less than 0.05% of the samples, as can be seen in Figure 2.

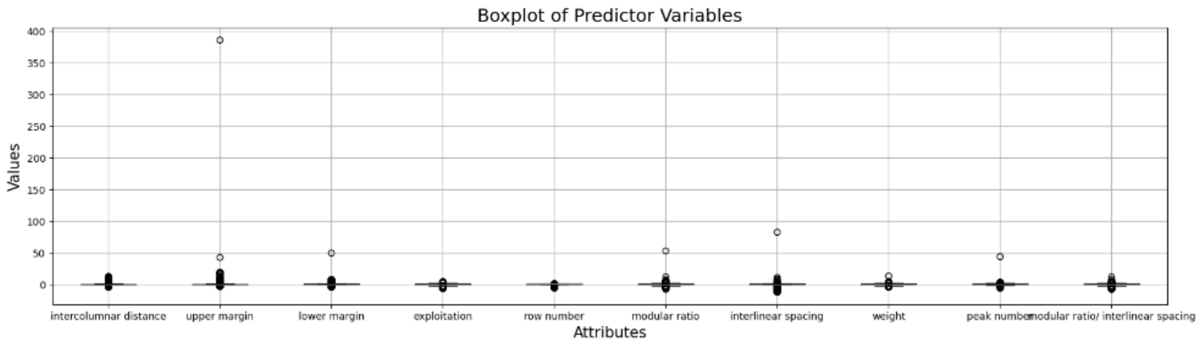
Figure 2. Target Class Distribution



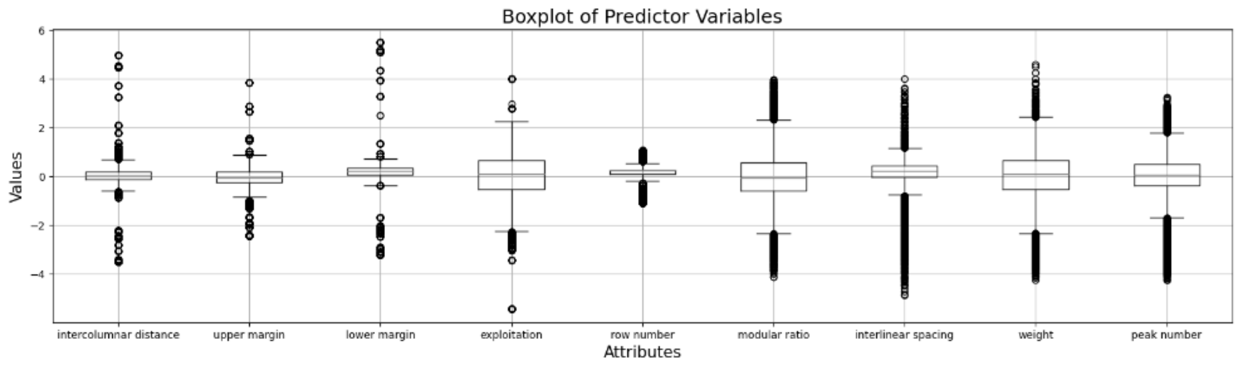
At this time, the combined set was then checked for missing values and contained none. The next step in pre-processing was to check for highly correlated variables, as these can lead to redundancy in modeling and the unnecessary inclusion of features. The variable *modular ratio/interlinear spacing* expectedly had a high correlation with both modular ratio and interlinear spacing. As a result, this variable was dropped from the dataset and not included in any analysis.

The documentation provided with the dataset claimed that the variables were all scaled and normalized. However, using a series of plots and graphs, many outliers were detected. The solution proposed was to replace those values that fall outside of a given quantile interval. Figure 3a and 3b displays the uncleaned dataset with noise and the final cleaned dataset, respectively.

Figure 3a & 3b. Uncleaned & Cleaned Boxplots.



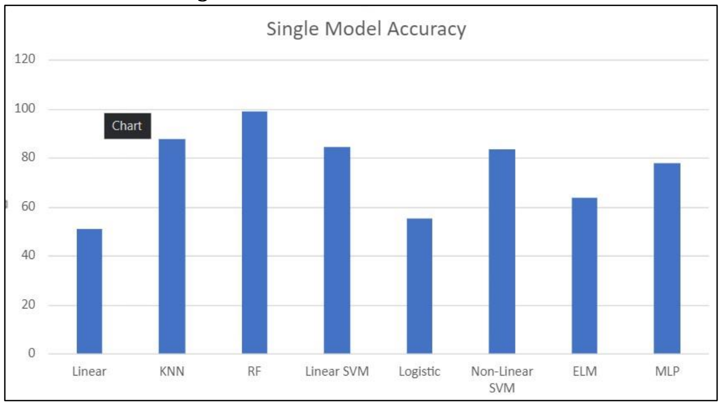
3a)



3b)

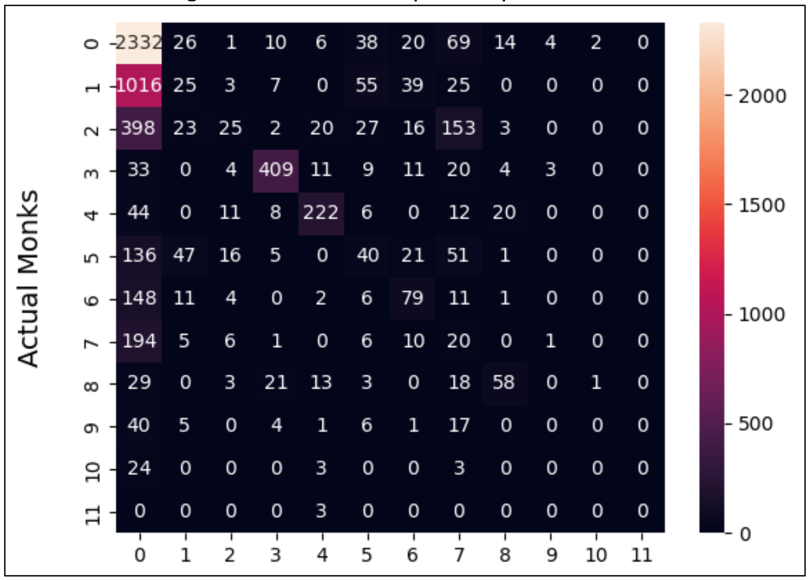
III. All Models

Figure 4. Individual Model Results



The data was modeled by 7 different algorithms individually, the first of which was our linear regression classifier. The team used *sklearn*'s SGDRegressor model as our linear classifier using stochastic gradient descent to update coefficients. The structure of this model was selected using grid search and 5-fold cross validation. The optimal parameters were found to use a learning rate of 0.1, hinge loss for error calculations and no shrinkage methods. This model performed with 52.05% accuracy and was henceforth the baseline model.

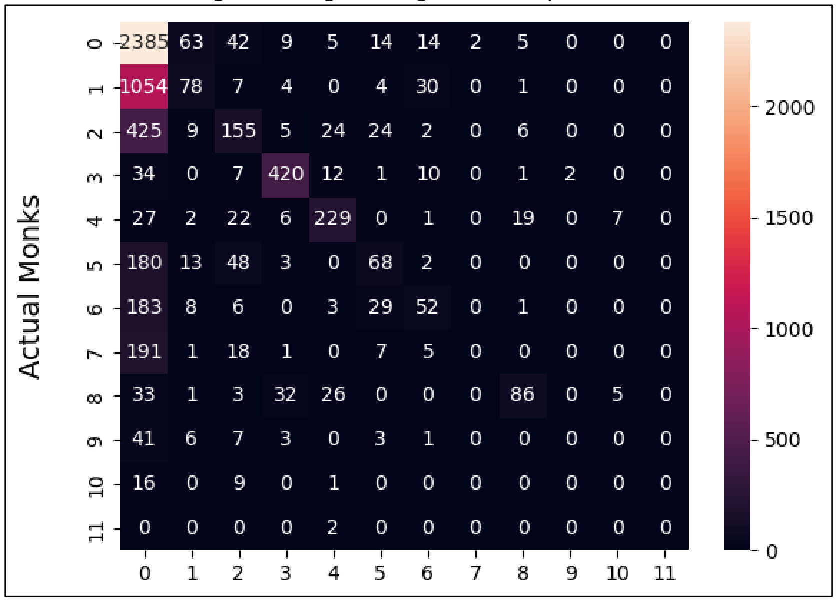
Figure 5. Linear Classifier Confusion Matrix



	0	1	2	3	4	5	6	7	8	9	10	11
0	2332	26	1	10	6	38	20	69	14	4	2	0
1	1016	25	3	7	0	55	39	25	0	0	0	0
2	398	23	25	2	20	27	16	153	3	0	0	0
3	33	0	4	409	11	9	11	20	4	3	0	0
4	44	0	11	8	222	6	0	12	20	0	0	0
5	136	47	16	5	0	40	21	51	1	0	0	0
6	148	11	4	0	2	6	79	11	1	0	0	0
7	194	5	6	1	0	6	10	20	0	1	0	0
8	29	0	3	21	13	3	0	18	58	0	1	0
9	40	5	0	4	1	6	1	17	0	0	0	0
10	24	0	0	0	3	0	0	3	0	0	0	0
11	0	0	0	0	3	0	0	0	0	0	0	0

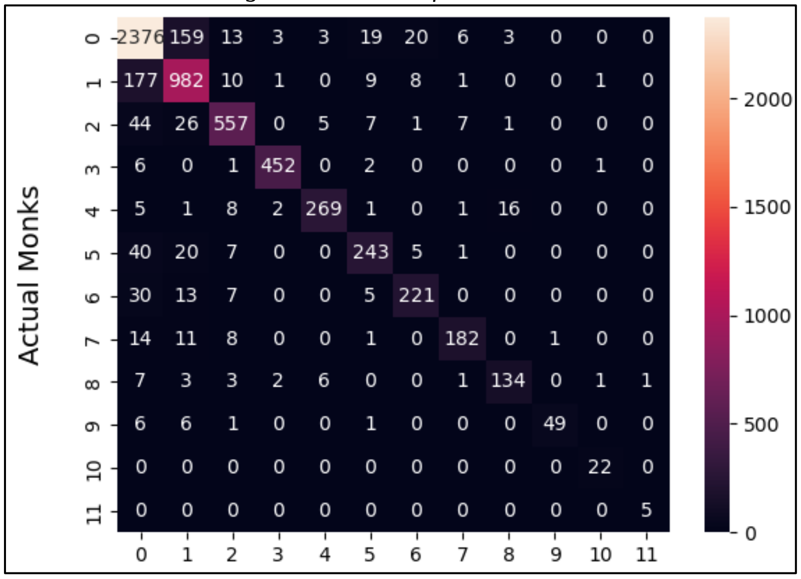
A logistic regression model was also designed and optimized using grid search and repeated cross validation. The structure of this model includes the use of a shrinkage penalty to shrink the coefficients of the model, inverse regularization parameter, and the addition of an optimization solver. The final hyperparameters were ridge regression (l2 penalty), a small regularization parameter (C) of 0.001, and the newton-cg method as optimization solver. The test accuracy of this model was 55.33%.

Figure 6. Logistic Regressor Confusion Matrix



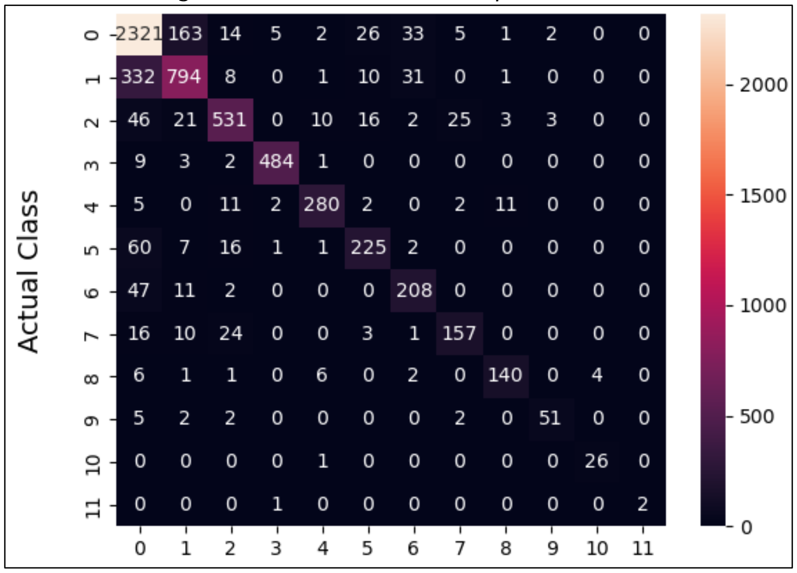
K-Nearest Neighbors was the third model trained and optimized using grid search. The team considered which distance function to use, the number of neighbors to consider for new query points, and how weight should be distributed between neighbor values in the model structure. The optimal hyperparameters use the Manhattan distance function, 3 neighbors in query point decisions, and weights distributed based on distance, making them inversely correlated. KNN performed fairly well on the test set with an accuracy of 88.27%.

Figure 7. KNN Confusion Matrix



The next model trained was a support vector machine using *sklearn*'s Support Vector Classifier algorithm. Randomized grid search was used to tune hyperparameters and define the model's structure because of SVM's large time complexity for training. The structure uses a radial basis function kernel, with a kernel coefficient of 0.1 and a regularization parameter of 1000. This model also performed fairly on the test set with an accuracy of 84.4%.

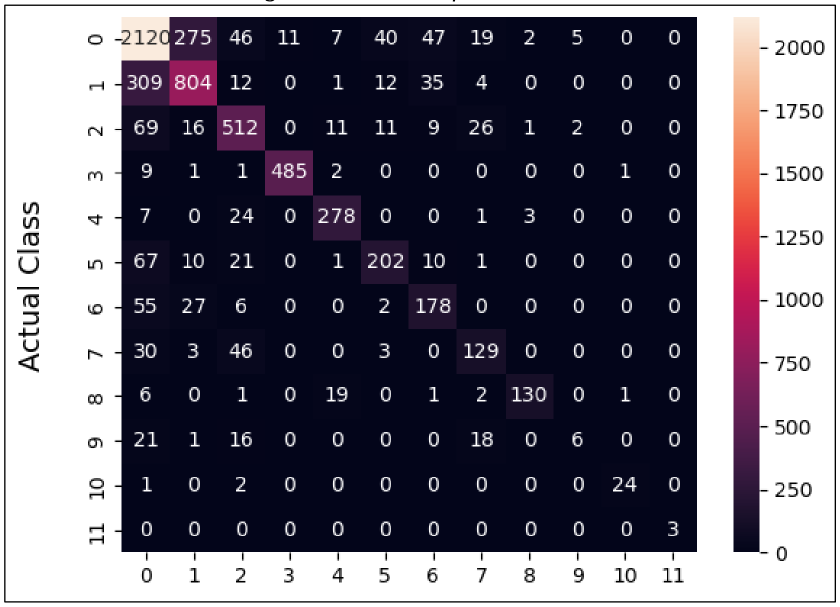
Figure 8. Non-Linear SVM Confusion Matrix



	0	1	2	3	4	5	6	7	8	9	10	11
0	2321	163	14	5	2	26	33	5	1	2	0	0
1	332	794	8	0	1	10	31	0	1	0	0	0
2	46	21	531	0	10	16	2	25	3	3	0	0
3	9	3	2	484	1	0	0	0	0	0	0	0
4	5	0	11	2	280	2	0	2	11	0	0	0
5	60	7	16	1	1	225	2	0	0	0	0	0
6	47	11	2	0	0	0	208	0	0	0	0	0
7	16	10	24	0	0	3	1	157	0	0	0	0
8	6	1	1	0	6	0	2	0	140	0	4	0
9	5	2	2	0	0	0	0	2	0	51	0	0
10	0	0	0	0	1	0	0	0	0	0	26	0
11	0	0	0	1	0	0	0	0	0	0	0	2

A deep neural network was also trained using *sklearn's* Multi-Layer Perceptron algorithm. The hyperparameters considered for this network were the number of layers, number of neurons in each layer, and the type of learning rate. The optimal hyperparameters were found by grid search to be 2 hidden layers with 20 neurons in each layer and a constant learning rate instead of an adaptive one. The test accuracy for this model was 77.8%.

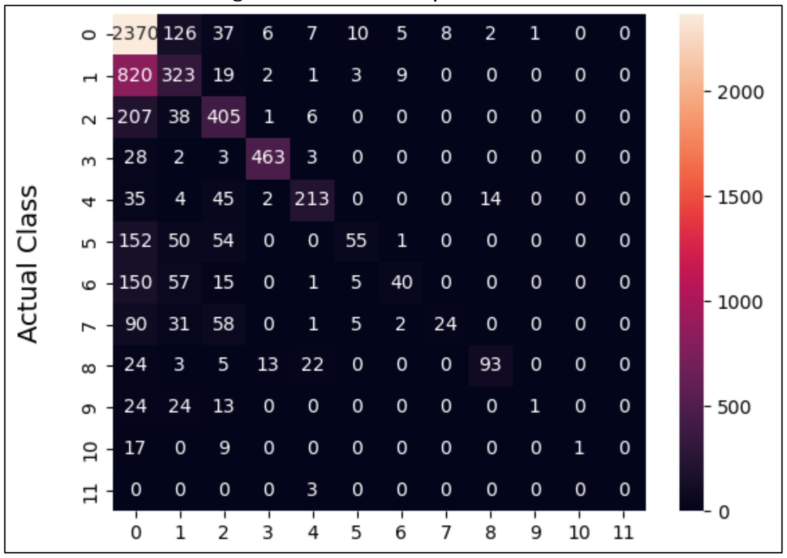
Figure 9. MLP Confusion Matrix



	0	1	2	3	4	5	6	7	8	9	10	11
0	2120	275	46	11	7	40	47	19	2	5	0	0
1	309	804	12	0	1	12	35	4	0	0	0	0
2	69	16	512	0	11	11	9	26	1	2	0	0
3	9	1	1	485	2	0	0	0	0	0	1	0
4	7	0	24	0	278	0	0	1	3	0	0	0
5	67	10	21	0	1	202	10	1	0	0	0	0
6	55	27	6	0	0	2	178	0	0	0	0	0
7	30	3	46	0	0	3	0	129	0	0	0	0
8	6	0	1	0	19	0	1	2	130	0	1	0
9	21	1	16	0	0	0	0	18	0	6	0	0
10	1	0	2	0	0	0	0	0	0	0	24	0
11	0	0	0	0	0	0	0	0	0	0	0	3

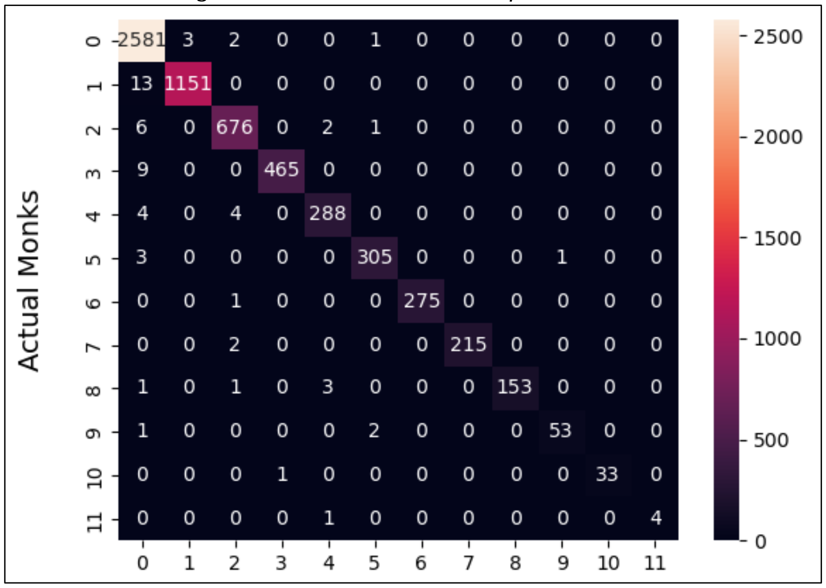
The sixth model trained was an extreme learning machine which uses 200 neurons. While research suggests that there are better ways to tune ELM's than using grid search, there still aren't many published methods to use with python. As such, the number of neurons used was decided by grid search.

Figure 10. ELM Confusion Matrix



The final and best model of all single models was the Random Forest. Because this algorithm is just an ensemble of overfit decision trees, and has capabilities of performing variable selection in training, it was not surprising that it performed the best on the test set. Model structure was defined by the number of trees in the forest and the maximum number of features to be considered in fitting. Grid search also found these optimal hyperparameters to be 1000 trees and the maximum number of features to be log(n), the log of the number of samples. The model tested extremely well with 98.8% accuracy.

Figure 11. Random Forest Confusion Matrix



IV. KNN Variable Selection

After running all models and obtaining results, the next step in optimizing predictions was performing feature selection. The first selection method was to be chosen from either correlation techniques, lasso regression, or k-nearest neighbor's algorithm. Correlation observations made in the pre-processing stage—including the use of scatter plots and heatmaps—proved that the data is not linear and would not be linearly separable. Additionally, our linear models, namely logistic regression and SGD Regressor, performed at ~50% accuracy, which is almost synonymous to random guessing. Therefore, using correlation for variable selection would be a poor choice. Our best model, random forest, is incompatible with using lasso regression in its algorithm. Since the task after preliminary modeling would be to improve upon the best model, using lasso regression would also not be a fitting selection technique. For these reasons, the first feature selection method chosen was KNN.

The proposed method was a modified best subset selection, only considering combinations of 5 variables instead of all possible combinations. Optimal hyperparameters were found during grid search and using stratified k-fold cross-validation. Grid search determined the optimal hyperparameters to be "Manhattan" for the distance metric, 1 neighbor as the nearest neighbor's parameter, and uniform weights as the weight parameter. The feature-selection KNN model iterates through each combination of 5 variables and was evaluated based on balanced accuracy. The best subset of variables is displayed in figure 13.

Random Forest and SVM were both trained with the subset of variables determined by KNN. Random Forest improved accuracy on the test set by 0.86% to 99.66% but SVM performed worse on the test set with the selected variables than without at ~75%. From deduction, it seems that the selected variables oversimplified the Support Vector Machine, an algorithm praised for its performance on high-dimensional data. Thus, this may have contributed to a high bias and reduced accuracy.

V. Bi-Directional Elimination

Bi-Directional Elimination also known as Stepwise Selection is a wrapper method of feature selection. It is a combination of forward selection and backward elimination and follows a greedy search approach by evaluating all possible combinations of features with the given machine learning algorithm. It is similar to forward selection but while adding a new feature, it checks the significance of already existing features. If it finds any feature insignificant, it removes it.

The random forest classifier with optimal hyperparameters found previously was used to evaluate the effect of bi-directional elimination based on accuracy. The five features seen in Figure 12 obtained from this method happened to be the same obtained from the KNN selection technique. It was not surprising, then, that it performed with a similar accuracy of ~99.6% on the test set.

Figure 12. Final Variable List

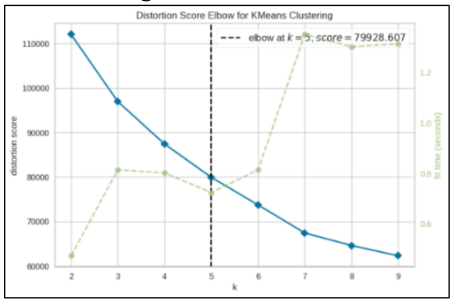
Final Variable List
Intercolumnar Distance
Upper Margin
Lower Margin
Exploitation
Weight

VI. Clustering

Clustering is a machine learning technique that involves grouping together similar data points based on a similarity metric or distance measure. The purpose of performing clustering on this data is to help identify patterns and relationships. Two clustering techniques were used on this dataset to find similarities.

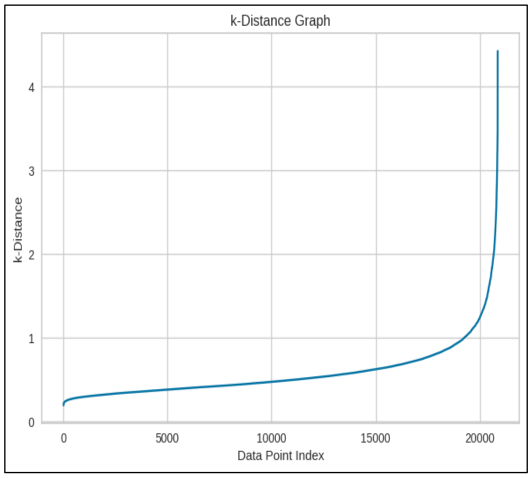
K-means is a popular clustering algorithm that aims to partition a dataset into a fixed number (k) of clusters, where each data point belongs to the cluster whose center is nearest to the point. Because it is necessary to explicitly state the number of clusters (k), the elbow method was used to determine the optimal number of clusters. The elbow method involves plotting the graph of sum of squared errors vs. number of clusters and selecting k such that the increasing number of clusters will not significantly reduce the sum of squared errors. The graph plots values of k between 2 and 9 and the optimal k select was k=5.

Figure 13. SSE vs K



Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is another popular clustering algorithm that groups together data points based on their spatial density. The algorithm defines a radius around each data point and searches for other points within that radius. DBSCAN can be more useful for highly dimensional data because it can identify clusters of arbitrary shape and can also detect noise points that do not belong to any cluster. To determine the best value for epsilon, the radius of each data point, the k-distance graph was plot and observed for where the function's gradient shifts toward infinity. The optimal epsilon was found to be 3 and the minimum number of points to consider a cluster was set at 10.

Figure 14. K-Distance Graph



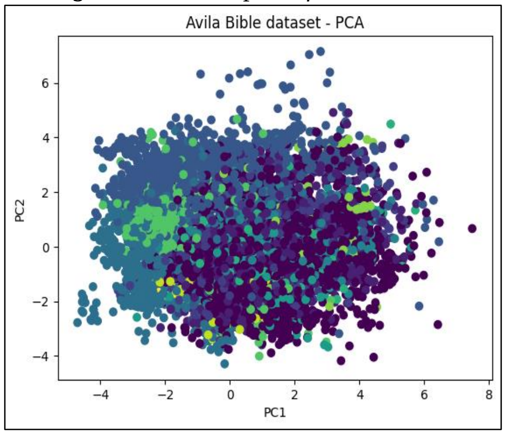
K-Means and DBSCAN were evaluated based on their silhouette score. The silhouette coefficient is a measure of how similar a data observation is to its own cluster compared to other clusters, and the silhouette score is the average coefficient over all data instances. The silhouette score ranges between -1 and +1 and is positively correlated with confidence in the data points' cluster assignments. K-Means clustering had an average silhouette score of 0.215 and DBSCAN had a score of 0.401. This means that the DBSCAN has more confidence that the data points were assigned to their correct clusters. This might have been useful to build a predictive model had we returned to variable selection and considered variable combinations of 3, however the team did not have enough time to perform this.

VII. Visualization using Dimensionality Reduction

Dimensionality of a dataset is defined by the number of features it contains. In machine learning, high dimensional data introduces the problem of interpretability, and is known as the curse of dimensionality. Principal component analysis (PCA) is a popular technique for reducing the dimensions of a large dataset, increasing the interpretability of data while preserving the maximum amount of information necessary to build a general model, without discarding any of the existing features. To our benefit, it also enables the visualization of multidimensional data.

Unfortunately, the visual representation of the first two principal components did not reveal any patterns or other trends found by observation. PCA might have been effective in building a predictive model if the first two components were used in our best two models and produced better results. However, this was not something that could be accomplished in our amount of time.

Figure 15. Scatterplot of PC1 vs. PC2

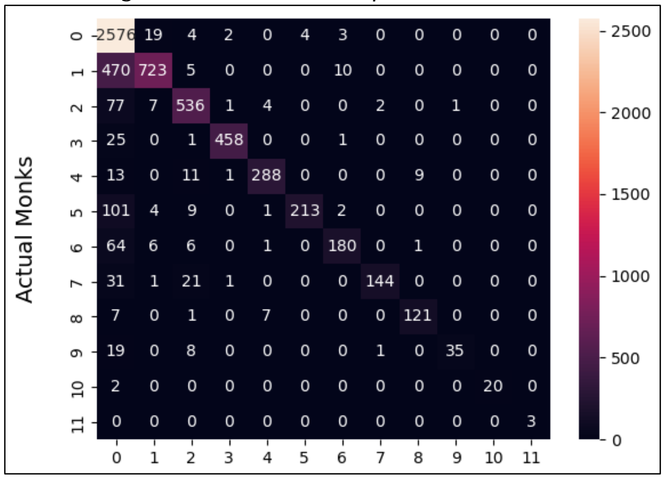


VIII. Ensemble

Ensemble modeling is the combining of multiple machine learning algorithms to better the performance of any individual model. They typically have better accuracy, and reduce both bias and variance. The basic idea of ensemble learning comes from the principles of 'the wisdom of the crowd', in that having a group of models evaluate predictions increases the likelihood that the prediction is correct.

The ensemble model created with this dataset uses all individual models built thus far with their respective optimal hyperparameter and combines them using a majority voting classifier. This means that an instance is assigned to the class that most of the models assigned it to. However, our ensemble model performed worse than the best individual model, only achieving an accuracy of ~85%. This was presumed to be because of the poor performance of the individual linear classifiers used in the ensemble, which both had accuracies of ~50%.

Figure 16. Ensemble Confusion Matrix



IX. Discussion

There were several tasks our team believes may have helped in determining the best predictive model if given more time. One of these tasks includes oversampling the class labeled ‘B’ or under sampling the class labeled ‘A’ to obtain a more representative population, and modeling to measure the difference in accuracy. It is believed that the Extreme Machine Learning Model might have performed closer to that of SVM if there was more time to dedicate towards developing an algorithm in python to tune hyperparameters using theory described in the Optimally Pruned ELM *(Miche et.al, 2010)* study. Another task to perform that might have contributed to better results was using the clustering method with the highest silhouette score—DBSCAN in our case—to determine the number of variables to consider in the best subset KNN feature selection. It is presumed that including these tasks or combination of them might improve the results of our best models.

X. Conclusion

In conclusion, our Random Forest Classifier with feature selection performed the best with an accuracy of 99.66%. Realistically, the model does an excellent job at predicting which scribe monk wrote which pattern on new data the model has not seen. Although it is not highly likely that more time would have resulted in better results, we were eager to build the experiment and believe that we might have obtained more robust results.

REFERENCES

De Stefano, C., Maniaci, M., Fontanella, F., & Scotto di Freca, A. (2018). Reliable writer identification in medieval manuscripts through page layout features: The “Avila” bible case. *Engineering Applications of Artificial Intelligence*, *72*, 99–110. https://doi.org/10.1016/j.engappai.2018.03.023

M-Pana. (n.d.). *Avila/a digital approach to palaeography.ipynb at master · m-PANA/avila*. GitHub. Retrieved May 5, 2023, from https://github.com/m-pana/avila/blob/master/A%20digital%20approach%20to%20palaeography.ipynb

Verma, V. (2022, June 22). *A comprehensive guide to feature selection using Wrapper methods in python*. Analytics Vidhya. Retrieved from https://www.analyticsvidhya.com/blog/2020/10/a-comprehensive-guide-to-feature-selection-using-wrapper-methods-in-python/

Yoan Miche, Sorjamaa, A., Bas, P., Simula, O., Jutten, C., & Lendasse, A. (2010). Op-elm: Optimally pruned extreme learning machine. *IEEE Transactions on Neural Networks*, *21*(1), 158–162. https://doi.org/10.1109/tnn.2009.2036259