Homework 5 Report
Calvin Schaul
IDSN542 Machine Intelligence

Dataset & Background

For this assignment I returned to the UC Irvine Machine Learning Repository and selected a Letter Recognition dataset [1]. This dataset felt like a natural continuation of the introduction to classification we covered in class with the MNIST dataset for number recognition. Specifically I will be doing multiclass classification on this dataset.

The 20,000 entries of this dataset include 16 numerical attributes describing the positioning and number of edges identified from an image. Unlike the MNIST dataset, the data here did not include actual images, just numeric information describing images of letters. The letters in this dataset were capital english letters from 20 different fonts with many different transformations applied on them to distort the letters, creating a wide range of forms for a given letter.

The target attribute to classify was simply the “lettr” of a given row of data - a single character. Based on Geron’s Chapter 3 information on Multiclass Classification, a RandomForestClassifier was chosen to do this classification due to its native support of multiple classes [2].

Data Preparation

The data from UCI was provided as two files. First, “letter-recognition.data” included the 16 numeric attributes for all 20,000 rows of data. Second, “letter-recognition.names” included the target letter from each row of data.

This differed from the MNIST dataset, which used a built-in method to import the dataset. Pandas read_csv function was able to handle these files. Further data preparation was done to check if there was any missing data in the dataset.

After no missing values were found, the next step of data preparation was to add labels to the data based on the information provided from UCI, as they were not included in the original .data file. This included the “lettr” target attribute as well as the numerical attributes such as “width” and “height” of the letter images.

With our data properly formatted, it was split into training and test sets. UCI identified that the first 16,000 entries were to be used for training and the last 4,000 entries were to be used for testing. Once the data was split, a final preparation check was done to ensure that all 26 letters existed within the target column. [1]

Precision vs. Recall

A random forest classifier was fit to the X and y training data and tested on a single row of data. Right away, the model correctly identified an “N” from the dataset.

Random Forest Classifier - First entry test prediction: ['N']
Expected letter for first test case: N

Further training was conducted using Stratified K Fold cross-validation was done with 3 folds and a clone of the random forest classifier. A base accuracy calculation was performed on each of the three folds based on the number of correct predictions divided by the total number of predictions.

Random Forest Classifier accuracy for fold: 0.9495688038995126
Random Forest Classifier accuracy for fold: 0.9579973748359273
Random Forest Classifier accuracy for fold: 0.9555597224826552

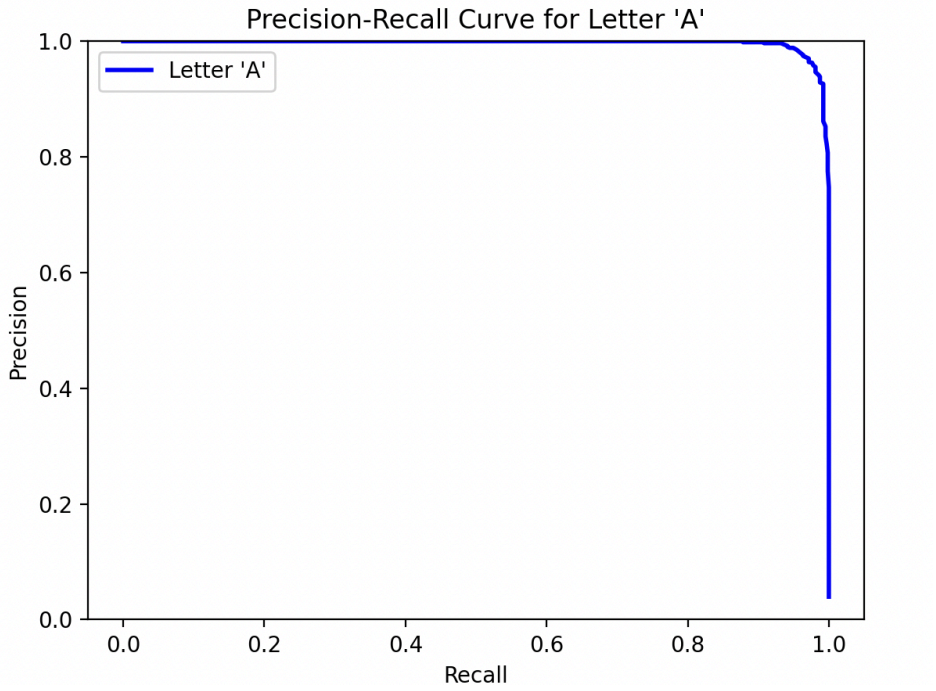
With these base accuracies in mind, cross-validated predictions were made on the training data using SciKitLearn’s cross_val_predict function. These predicted values were fed into more SciKitLearn functions to calculate the following precision, recall, and f1 scores:

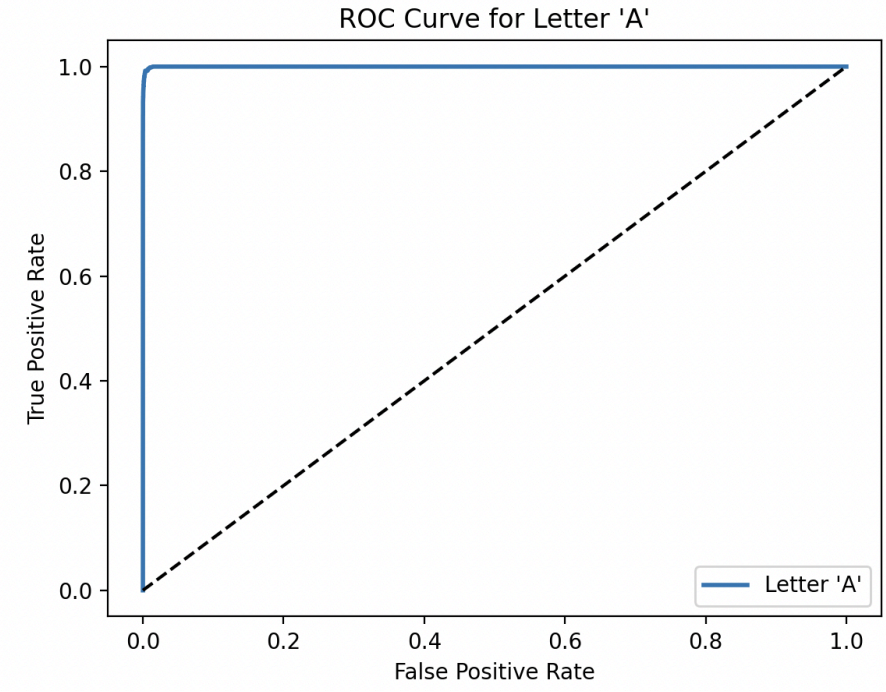
Random Forest Evaluation

Precision Score	0.9534247058240044
Recall Score	0.9524529254705468
F1 Score	0.9527199128850885

In the case of letter identification, a balance of both precision and recall would be important to ensure that our predictions actually align with their corresponding letter and that we actually can identify the letter in the first place. However, recall should be considered slightly more important. Recall measures how many letters are correctly identified out of the entire dataset (whereas precision measures the true positives out of selected prediction set), so a high recall would be helpful if this model was used to decipher messages or read text from an image.

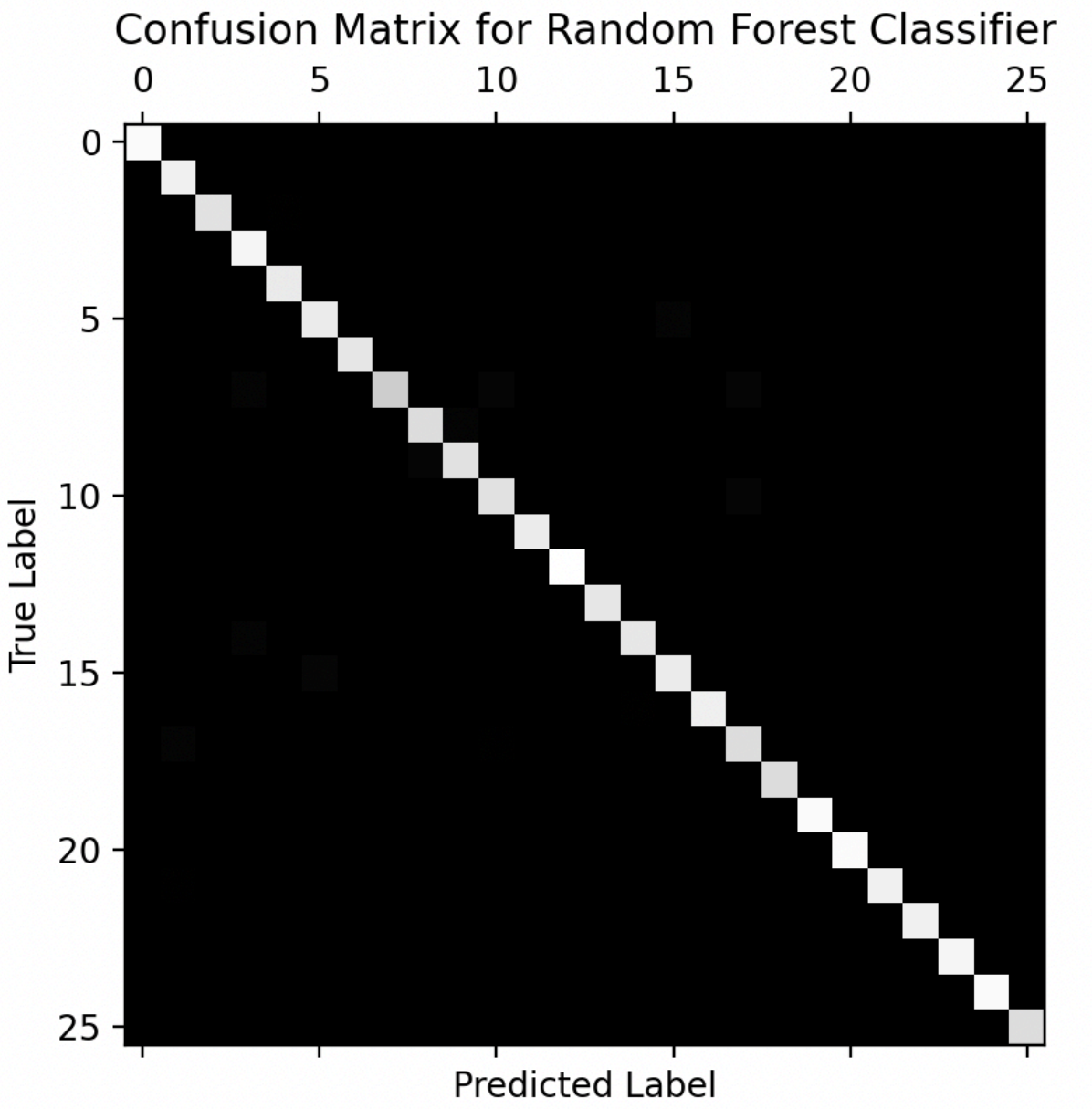
To show an example of these performance metrics, a single class was chosen to compare to the binary classifier examples we completed in class. Below are teh Precision-Recall and ROC graphs for the classifier's performance on the letter "A"



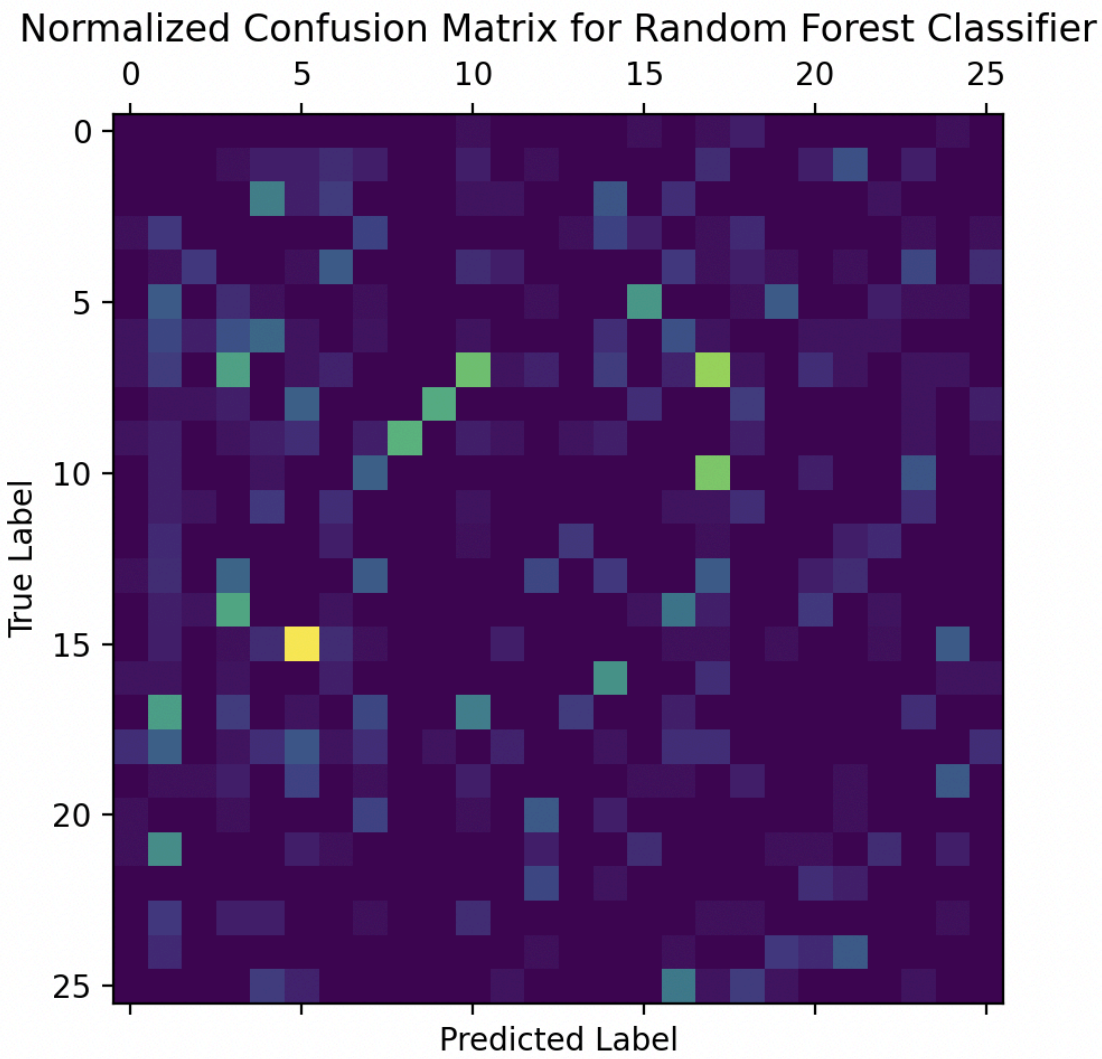


Error Analysis with Normalized Confusion Matrix

More error analysis was conducted using confusion matrices. First, the training data was scaled using a StandardScaler. The base confusion matrix below shows that nearly all letters were correctly identified, as they sit on the main diagonal of the plot. Slightly darker entries were “C”, “H”, and “Z”, meaning those were more likely to be missed by the model.



After normalizing the confusion matrix by dividing each prediction by the number of total elements of that class, it becomes easier to see specific errors from the model. Brighter colors indicate common errors. Based on the plot below, “F” and “P” were one of the most confused pairs of letters. Other common errors were with “R” and “H” as well as “R” and “K”.



With these specific errors happening often, more numeric or categorical data could be helpful to include with this dataset to identify when the tops or sides of letters are closed. Further, around 600 examples of each letter were used in the training dataset, so more data, perhaps 1000 of each letter, could help give the model more context to decipher between these problematic letters.

References

[1] UCI Machine Learning Repository Letter Recognition Dataset
https://archive.ics.uci.edu/dataset/59/letter+recognition

[2] Hands-On Machine Learning with Scikit-Learn, A. Geron.
Chapter 3 Classification, Section 4 Multiclass Classification