Classifying Human-Written Digits with Machine Learning Methods

Authors: Noah Lee, Jeffrey Kong

Introduction:

With the rise in popularity in fields like computer vision, various stakeholders have been interested in using machines to read human made writings like digits. Additionally, companies are getting interested in digitizing data, so investments have been made to write algorithms that can transform human-written data into a digital form which can be more easily fed into a model like an LLM. We believed a good starting point to creating a model that classifies all digits would be to start with binary classification. For this, we chose digits 2 and 6. Using the widely used MNIST dataset, we were able to create a reliable Random Forest model that works to identify a 2 from a 6 and vice versa and provide insight into the most important factors in determining so.

Although this type of classification is typically covered through the use of a neural network, we believe that we were able to create an interpretable machine learning model that reliably identifies a human-written 2 against a 6. We aim to show the features we added that aided in our model and identify the key attributes that separated a 2 from a 6.

Data:

The MNIST dataset consists of images of handwritten digits from 0 to 9. We filtered the data to remove any digits not labeled as 2 or 6. Each image consists of a 28 by 28 matrix of 0-255 values where the higher values correspond to a darker pixel.Each record consisted of 785 columns, 1 for **Digit** and 784 pixel color values. While this created a high level of dimensionality for our dataset, we believed that due to the high amount of records in comparison, that we would be best off continuing leaving all the pixel values in our model.

In addition, we decided on creating some features to include as predictor variables in our model. We tested the amount of dark squares an image had, with the threshold being 50. After running initial analysis, through a function that calculates proportion of dark squares, histograms showed that there was a noticeable difference in proportions of “dark squares” between the images of the digit 2 versus images of the digit 6. (**Figure 7**).

We also outlined a special area to measure the count of dark squares with a threshold of 50 in all of the images. After manually looking at a few images, we settled on the region (Rows: 9:16, Columns: 6:13). This region was intended to catch the pixels in images of the digit 6, which may not be in that area in images of the digit 2. We also tested a couple of other regions, such as upper and lower images splits, but the difference in proportion of dark pixels from those splits were not statistically significant enough for us to justify including it in our model.

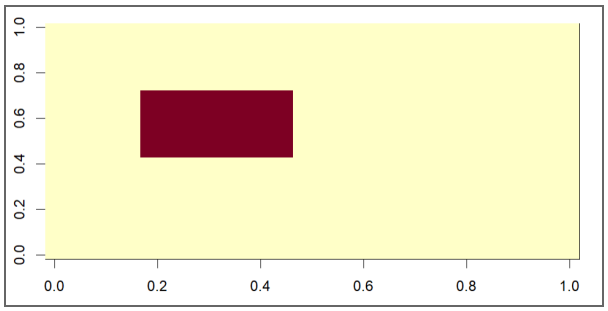

Figure 1 - Visualization of the Region on the Grid we used to calculate propArea

Our final dataset fed into the model consisted of the following 787 quantitative variables:

- **Digit:** Either a 2 or a 6 - response variable
- **V1-V784**: A number from 0-255 that represented the monotone color of a pixel
- **countBlack**: The number of dark pixels (>50) in the image
- **propArea:** The proportion of dark pixels (>50) in **Figure 1**

We split the data into training, validation, and testing sets, stratifying by **Digit**. We then proceeded with our model building process on the training data.

Model:

We started by plotting a KNN classification model due to its simplicity for a relatively large dataset. KNN also provided a straightforward baseline model to quickly assess separability between the two classes (digits 2 and 6). After tuning for the optimal value of k, the model performed well in terms of accuracy at over 99%. However, we ultimately found it limited in interpretability, as KNN does not provide information on variable importance.

We chose to move forward with a Random Forest Classifier because it combines high accuracy with the ability to evaluate feature importance, helping us understand which pixels or regions significantly contributed to distinguishing between the two digits. Random Forest's robustness to overfitting, especially with cross-validation, made it an ideal choice for capturing complex patterns in the high dimensional data while maintaining interpretability. This model allowed us to achieve a balance between performance and insights, making it a more suitable choice for our classification task.

As the data was all quantitative and there were no observable outliers or missing values in the dataset, we continued with our model building process without any further data preprocessing. We ran Grid Search Cross Validation on a Random Forest classification model in order to identify the optimal parameters in our model in terms of accuracy. We identified an optimal m = 21 and n = 125 where m is the number of predictors considered in each individual tree and n was the number of trees generated to create an averaged probability of 6 or 2.

We then fit the final model (m=21, n=125) on our training data using all the data discussed in the previous section and the parameters listed earlier. Using the vip package, we were able to identify the most important variables in classifying **Digit** in our Random Forest model. Using this, we identified V346, V298, V244, V348, V376, V270, V271, V375, V272, V374 as the most important variables in our model. Surprisingly, **propArea** and **countBlack** were not included in this list. From **Figure 2**, we can see that this creates two regions on the grid where if the left region is darker, it appears to indicate that **Digit** = 6 and the same is true for the right region for **Digit** = 2.

Figure 2 - Side-By-Side Diagram of the Most Important Pixels in determining Digit

An Out-of-Bag (OOB) error rate of 0.0116 means that the Random Forest model misclassifies approximately 1.16% of the observations on average across all the trees when each tree is tested on the data samples it did not train on. The low OOB error rate suggests that the model performs very well in distinguishing between the two classes in the training data and has a strong ability to generalize. Our ROC curve on the validation set also resembled the nearly perfect identifier (**Figure 9**) so we were rest assured that our model was accurate.

Results:

Our model performed well on the test set with minimal misclassifications. The AUC-ROC value of 0.9997 indicates that the Random Forest model effectively differentiates between the two classes (2 and 6), while the accuracy of 0.9933 shows that the model correctly predicts the digit about 99.33% of the time. Although these metrics suggest strong performance, they may also reflect potential overfitting, given the unusually high values. Future tests with a larger and more varied dataset could help validate the model's generalizability.

In terms of specificity and sensitivity, the test results were 0.9955 and 0.9912, respectively. This indicates that the model correctly classifies a "6" about 99.5% of the time and a "2" about 99.1% of the time, with similar performance across both classes. The confusion matrix (**Figure 10**) shows only 4 false negatives and 2 false positives out of 897 test records, indicating the model does not do particularly better with one of the digits over the other.

However, analyzing the misclassified images reveals a limitation in the model's reliance on specific pixel positions. **Figures 3** illustrate how variations in positioning and handwriting style

lead to errors. For example, an off-center "2" or a "6" with an extended loop can cause the model to misclassify due to its focus on exact pixel arrangements. This reliance on positional consistency limits the model's ability to handle common handwriting variations.

Looking at the misclassified images, we can see that the model does not do well against digits that are not drawn to be in the middle of the image. As we can see in the following example:

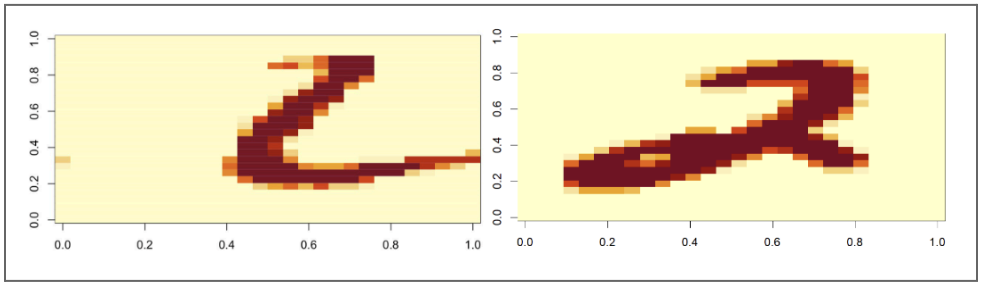

Figure 3 - Diagram of a Misclassified Image vs a Correctly Classified Image (Digit 2)

To address these challenges, future improvements could include incorporating features that capture shape rather than pixel positions alone. Additionally, using a Convolutional Neural Network (CNN) might enhance the model's robustness to variations in style and positioning by automatically learning hierarchical features that generalize better across diverse images (CARES, 2022). Expanding the dataset with augmented images, such as shifted or rotated digits, may also help reduce sensitivity to positional changes, improving overall performance and adaptability.

To conclude, although the model is heavily reliant on positional accuracy, we were still able to create a highly accurate model given certain constraints. If it is known beforehand that the numbers are expected to be around the same size and position - or if the figure is adjusted to fit this expectation - then we can expect our model to perform well. That being said, the model only currently works for digits of 2 or 6 so feeding a digit like 3 would have our model predict 2 or 6 seemingly arbitrarily. In future works, it is to be determined whether or not this same approach will have as good a performance with binary classification given the larger number of possible labels.

References:

MNIST Data for the Digits 2 and 6: https://aloy.github.io/stat270/data/digits26.csv, Adam Loy

CARES, L. (2022). *Handwritten Digit Recognition using Convolutional Neural Network (CNN) with Tensorflow*. [online] Medium. Available at:

https://learner-cares.medium.com/handwritten-digit-recognition-using-convolutional-neural-network-cnn-with-tensorflow-2f444e6c4c31.

Appendix:

Figure 4: Diagram of the Most Important Predictor Pixels

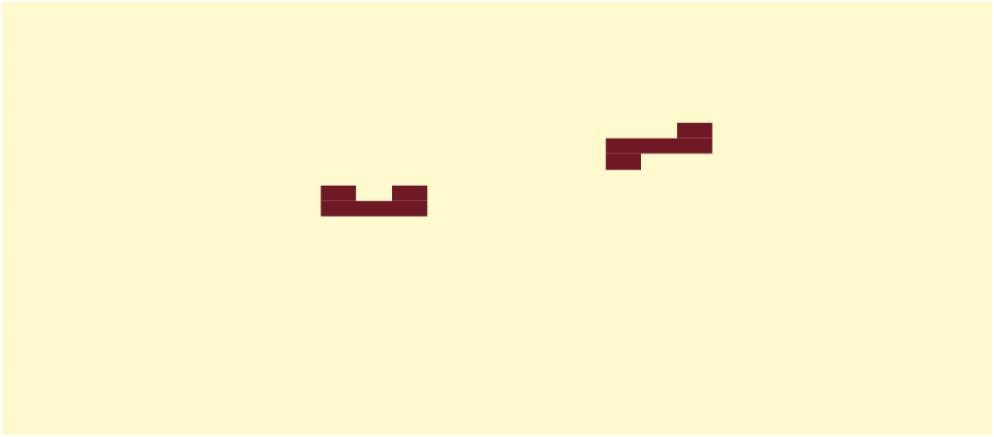

Figure 5: VIP plot of the Most Important Predictors

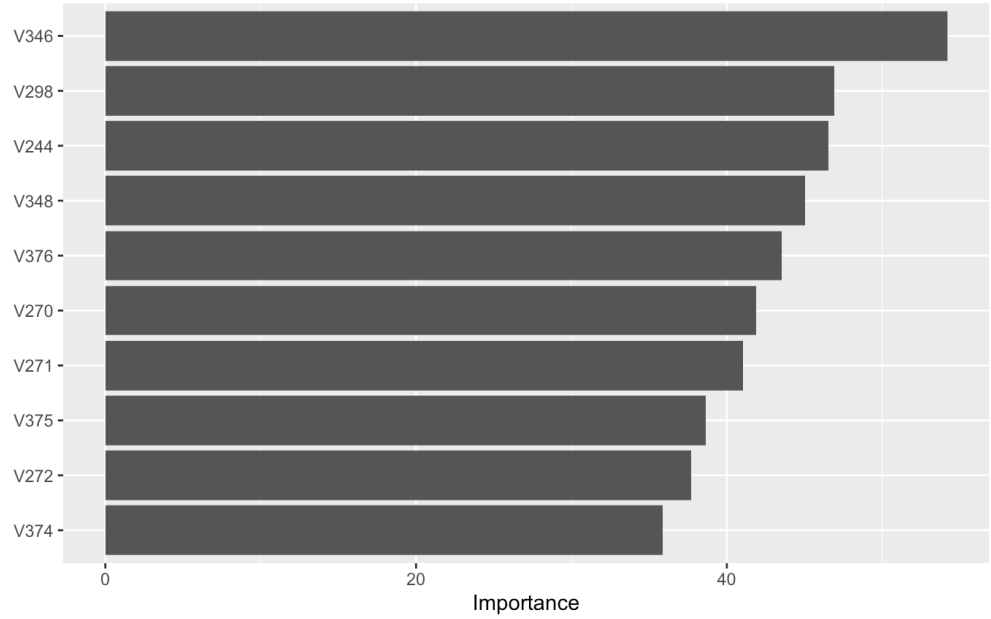

Figure 6 : Random Forest Model Summary

```
== Workflow [trained] ==========================================================
Preprocessor: Recipe
Model: rand_forest()

-- Preprocessor ----------------------------------------------------------------
0 Recipe Steps

-- Model -----------------------------------------------------------------------
Ranger result

Call:
 ranger::ranger(x = maybe_data_frame(x), y = y, mtry = min_cols(~4,      x), num.trees = ~200,
importance = ~"impurity", num.threads = 1,      verbose = FALSE, seed = sample.int(10^5, 1),
probability = TRUE)

Type:                             Probability estimation
Number of trees:                  200
Sample size:                      4188
Number of independent variables:  787
Mtry:                             4
Target node size:                 10
Variable importance mode:         impurity
Splitrule:                        gini
OOB prediction error (Brier s.):  0.02432444
```

Figure 7 : A Graph of the Distribution of Dark Pixel Counts for Digits 2 and 6

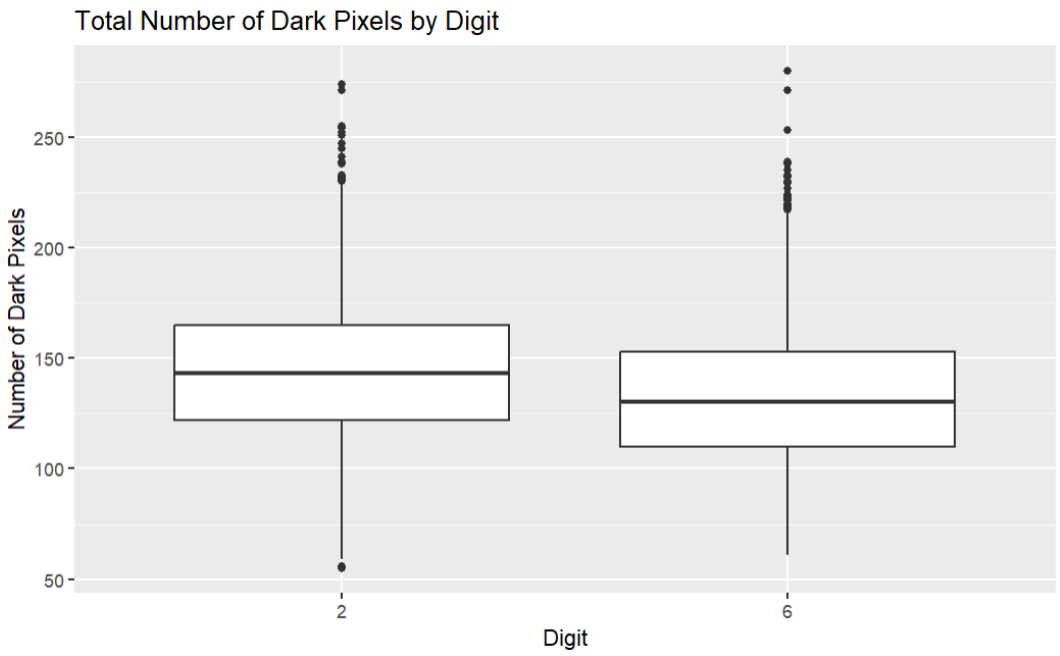

Figure 8: Diagram of a Misclassified Image vs a Correctly Classified Image (Digit 6)

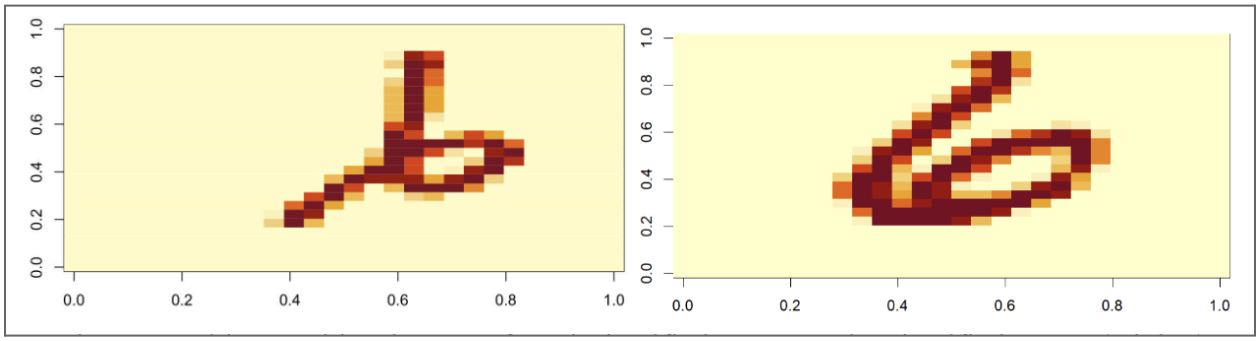

Figure 9: A plot of a ROC-AUC curve for the Random Forest Model

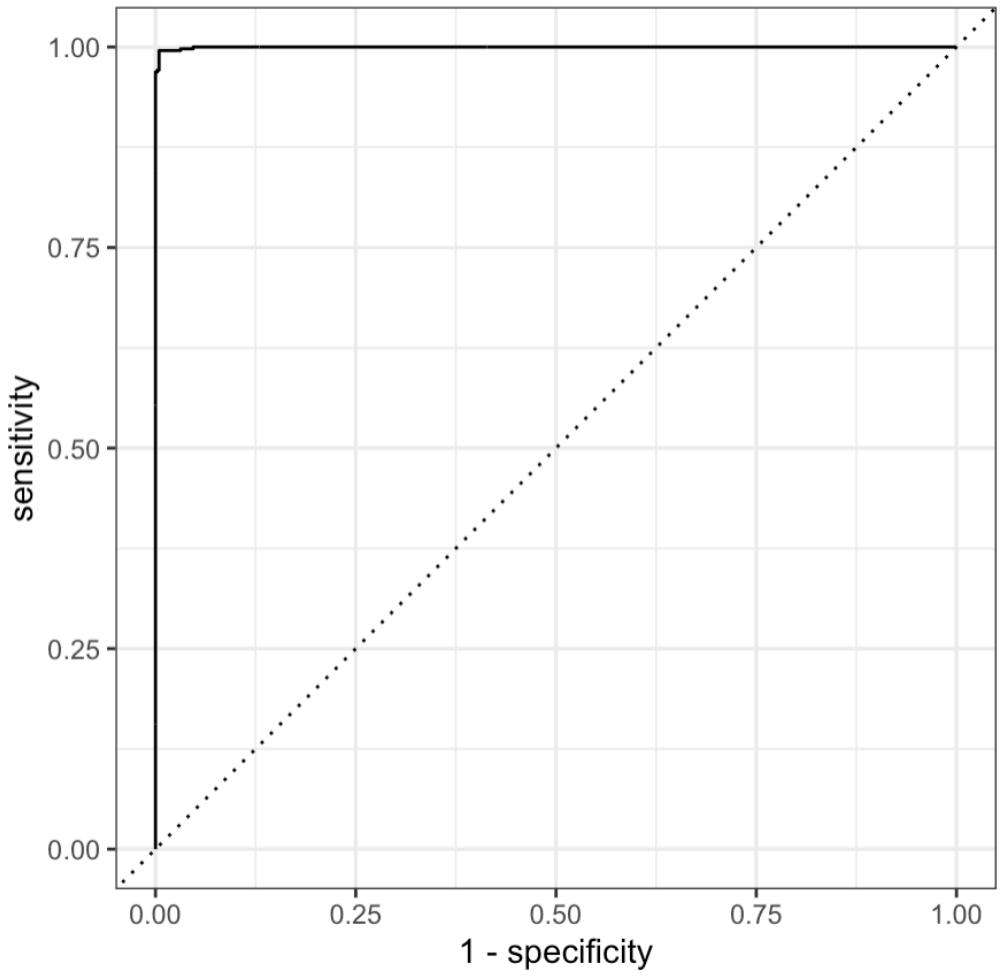

Figure 10: Confusion Matrix of the Random Forest Predictions

	True	
Predicted	2	6
2	448	4
6	2	443