Objective: Linear Regression and Random Forest Classification on MNISTmini {4, 8}

This project explores the training and tuning of hyperparameters in logistic regression and random forest classifiers for two digits (4 and 8) from the MNISTmini dataset. Furthermore, we will compare the models' results.

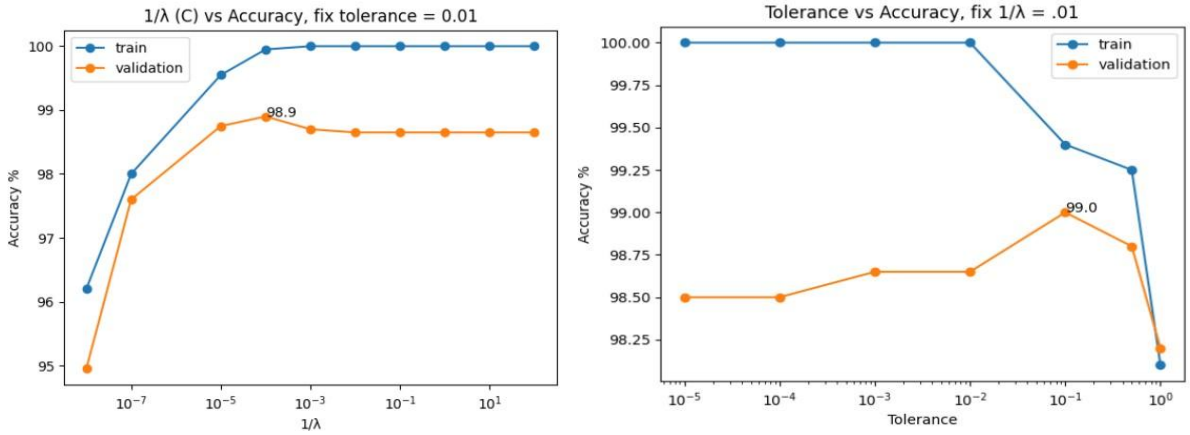
Preprocessing Steps:

1. Load MNISmini.mat file into a python dictionary using *scipy.io.loadmat*.
2. Create NumPy arrays for training and validation sets.
3. Reduce the data to the target digits (4s and 8s) by iterating over the ground truth array and append only the values equal to the target digits (ref. *utils.reduce_data*).
4. Split the reduced data into train and validation Numpy arrays (ref. *util.get_data*).
    a. Train set = first 1000 rows of 4 concatenated with first 1000 rows of 8.
    b. Static validation set = second 1000 rows of 4 concatenated with second 1000 rows of 8.

    Train slices: [0:999]+[5842:6841] Validation slices: [1000:1999]+[6842:7841]
5. We also tried scaling the data from [0, 255] to [0, 1], but this had no noticeable effect on the outcome of the models. This is left commented in the code. We chose not to use StandardScalar because the data does not seem well-represented by a Gaussian distribution.

Logistic Regression:

Logistic regression is a binary classification model of the form $\theta = \sigma(\mathbf{w}^T\mathbf{x} + w_0)$ with a cross-entropy loss function. Using scikit-learn's LogisticRegression(), we graphed various C (1/λ) values vs accuracy using an L2 penalty, solver = 'liblinear' and tolerance of 0.01, as well as tolerance vs accuracy fixing C = 0.01 also with L2 penalty and liblinear.



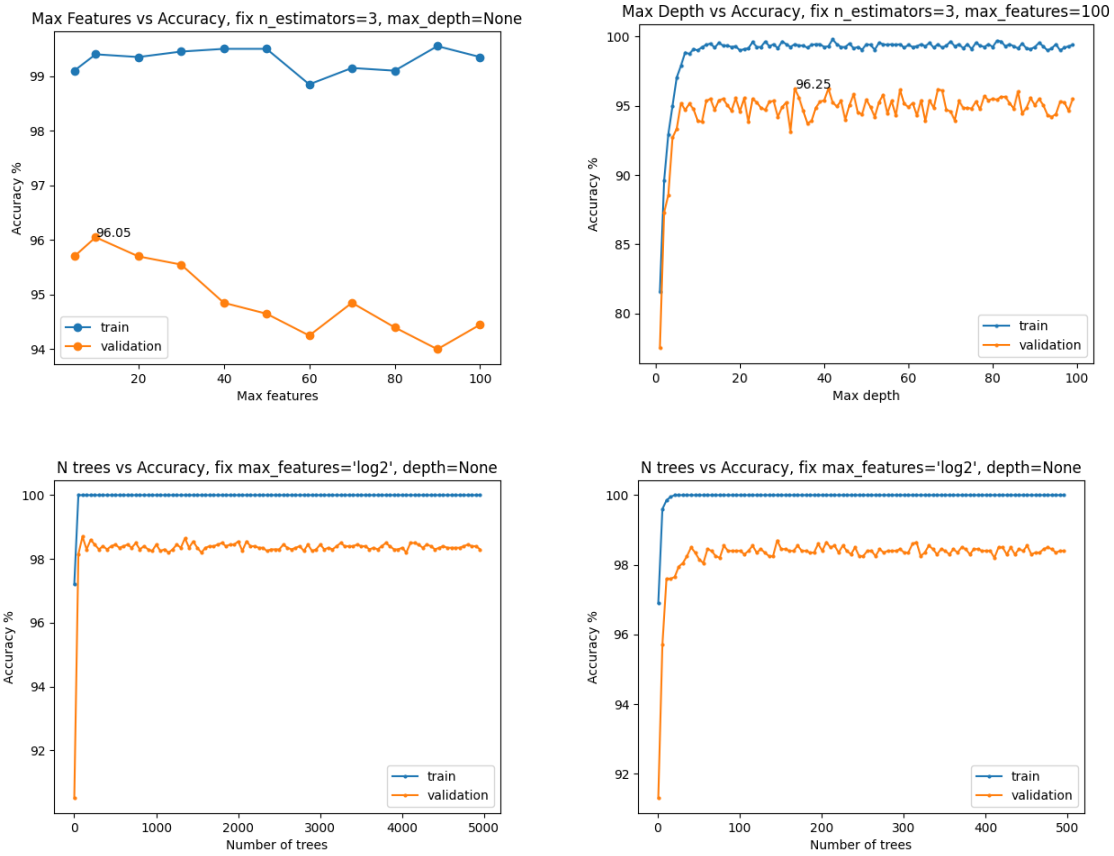
From the first graph, it is apparent that not penalizing **w** with the L2 penalty in the loss function adequately will overfit on the training data and perform worse in validation. The same is true if we allow the gradient descent to converge with a low tolerance; we must early-stop this process with a tolerance of about .01 to avoid overfitting our model.

We used these observations to create a grid search over C ∈ $[10^{-5}, 10^{-3}]$ by 0.00001 and tol ∈ $[10^{-2}, .5]$ by 0.01.

Our resulting ideal model was {C = .0003, tol = .09} with a validation accuracy of 0.99.

Random Forest:

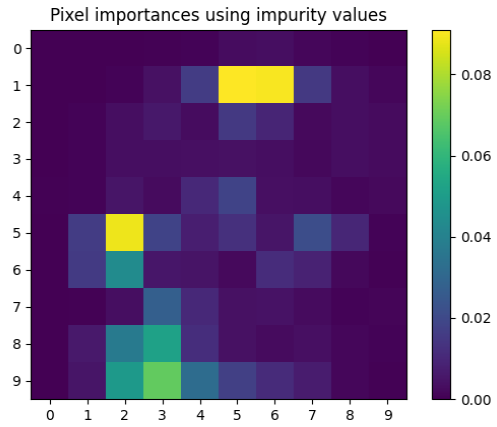
Random forest algorithm is a regression and classification method and consists of an ensemble of *n* of decision trees, each with a different random subset of the dataset and features (bootstrap). In order to come up with the best model using scikit-learn's RandomForestClassifier, we first tuned max_depth and max_features using 3 trees.



Using only 3 trees proved somewhat random if the graphs are reproduced, especially for the max_features plot, but on average we saw trends like those above. As advised in the course pdf and other sources online, the sqrt(#features) and log2(#features) for max_features seemed to perform well, and more than this would lower validation performance.

No long-term trend was seen as the number of trees was increased to 5000 by 50. Similarly for a window of [1, 500] by 5, no noticeable trend of an increase in accuracy was observed. We used insights from the graphs to then create a grid search over n_estimators $\in$ [3, 200] by 1, max_features $\in$ {'log2', 'sqrt'}, and a fixed max_depth = None (because of the erratic nature of the depth vs accuracy graph).

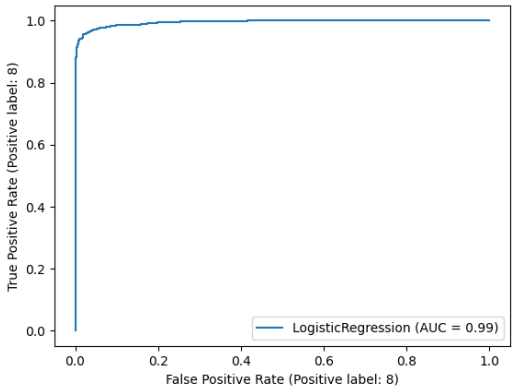
Our resulting ideal parameters were {n_estimators = 160, max_features='log2'} with a validation accuracy of 0.987.

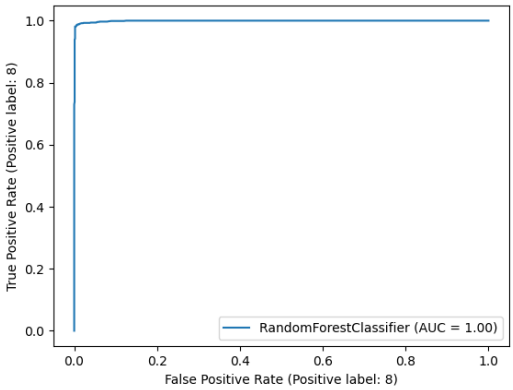


Next, to get some idea of how the decision trees are built we can use feature_importances (features closer to the root are more important globally), which measures how much each feature reduces the impurity across all the trees in the forest.
In this case we can see how the most important pixels are in the middle left and top which are the pixels where a split produces children containing instances of the same class (less overlap between 4 and 8 class).

Summary and comparison of the models:

Logistic Regression: The logistic regression model with C = .0003 and tolerance of .09 resulted in a test accuracy of 0.9893.
Random Forest: The random forest model of 160 trees with $\log_2(100)$ max features resulted in a test accuracy of 0.9897.
These accuracies were very close in value to each other and to the validation accuracies, and seem relatively high; we believe this is due to the fact that {4, 8} are relatively easy to distinguish from each other when compared to {4, 9} or {3, 8}, for example.
The ROC curves for both can be seen below, both are close to theoretically ideal ROC curves and do not aid in differentiating the two models on this data. (AUC=.99 and =1)





The logistic regression had the advantage of providing more consistent results in training compared to the forests' randomness. If we retrained the forests with the same hyperparameters and same data, the results change slightly each time. Still, due to the fact that we are able to produce the heatmap of feature importance, we slightly favor the random forest algorithm for our data; even though there are 160 trees in our optimal forest voting, it is theoretically possible to look into the trees and see exactly the reasons for their decisions.