The solution that we had implemented in the Kaggle Don’t overfit II competition was coded primarily using Python and utilizing the libraries of Pandas and scikit-learn. The overall steps we took was data ingestion, feature selection, and then applying various ML models and tuning them/comparing them to each other. The overall file structure of our solution was a scripts folder, a data folder, and a submission folder. It is important to note that none of the code that was used was copied from Github, thus it was much harder to score higher.

To go into more detail on our data ingestion, we first import the pandas library and ingest the training and test data as a pandas Dataframe. By ingesting the data as a dataframe, we are then able to more easily apply pre-processing techniques to it.This is due to the fact that a Dataframe acts similar to a CSV, where you have columns with labels taken from the original datafile along with rows determined by a numerical id.

For preprocessing, I used sklearn’s feature selection (RFECV) which find the optimal number of features for a more accurate prediction using recursive feature elimination and then applying a cross-validated selection. To do this I used both the Logistic Regression Model and Lasso Regression model as the estimator object. I then combined the results of both RFECV results to get the columns that should be taken from the models. I also implemented a StraifiedKFold function from sklearn in order to split up the data for RFECV.

For model tuning, we used various models but decided to stick with the Lasso and Logistic Regression models, along with MLP (Multi Layer Perceptron) Classification. These three algorithms all got identical scores in the competition. For Logistic Regression, the penalty was set to L1 and C was set to a smaller value, which makes data more sparse in it’s classification. The solver used for our model was the “liblinear” solver, which uses a coordinate descent algorithm. The liblinear solver essentially behaves as a multiclass classifier, so a separate binary classifier is trained for all classes present. Lasso is a variation of Linear Regression which also utilizes the L1 penalty along with selecting the best features as it creates its model to go through the data. Thus, because both the Logistic Regression and Lasso Regression run on a similar algorithm and penalty (L1), the results were identical for both. For utilizing Multi Layer Perceptron Classification, we set the hidden layer size to around 2000, this helps the model account for more complex datasets (the amount of features make this dataset fairly difficult to classify with less hidden layers). The activation function was set to “relu” (although logistic scored the same) and this was due to the fact that our dataset was extremely small and “relu” excels at smaller datasets through aggressive classification of data. “lbfgs” was used as the solver as it also is tuned for smaller datasets as it converges faster and performs better than the other options available.

Since our submissions were scored using an AUC (area under curve), submitting the probabilities of the data points yields a higher Leaderboard score. Thus, by having the models return the probability of each value being 1, we were able to get a higher leaderboard score. Our prediction jumped from .763 to .842 using this technique.