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, we first used sklean’s RobustScaler which standardizes the data in order to cope against outliers. We combined both the train and test data together for this in order to have a more accurate representation of the data when standardizing. Next, we 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 we used the Lasso Regression model as the estimator object. I set a larger step in order to avoid overfitting and used a StratifiedKFold approach in order to take smaller samples of the data to also avoid overfitting.

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. Lasso and Logistic regression got similar score for this competition with Lasso being slightly higher while MLP scored 4% lower. Thus, we ruled out MLP in our solution. 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. We set the “alpha” parameter to .031 as it is the multiplier for the L1 penalty (Similar to C). The tolerance is used as a stopping criteria for Lasso Regression, thus when the specified “Tol” of .01 is reached, the regression stops. Thus, because both the Logistic Regression and Lasso Regression run on a similar algorithm and penalty (L1), the results were nearly identical with Lasso coming out on top.

Lasso scored considerably higher than MLP and Logistic Regression so we went with the Lasso Regression for our submission. Even with MLP and Logistic Regression returning probabilities, we were unable to tune them to beat out our Lasso Regression.