AI HW4

Joshua Wiseman

October 2024

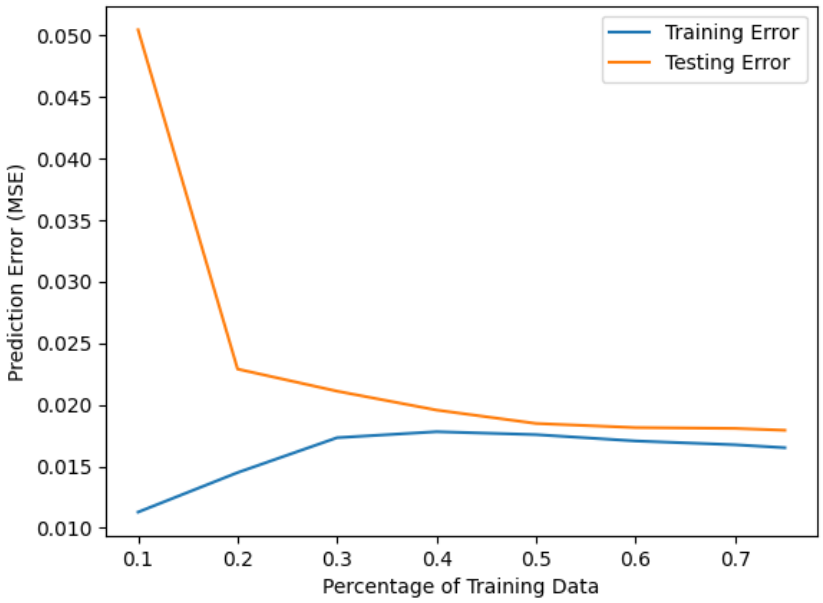

Figure 1: Plot of Prediction Error (MSE) vs Percentage of Training Data

The less data you are training on, the more prone to overfitting you are. We can see this by the Lower MSE on the training data resulting in a High MSE on the test data. As more data is used, the training MSE goes up, but the MSE of the test data goes down. Leading the the model being more useful in terms of generalizing various data inputs.

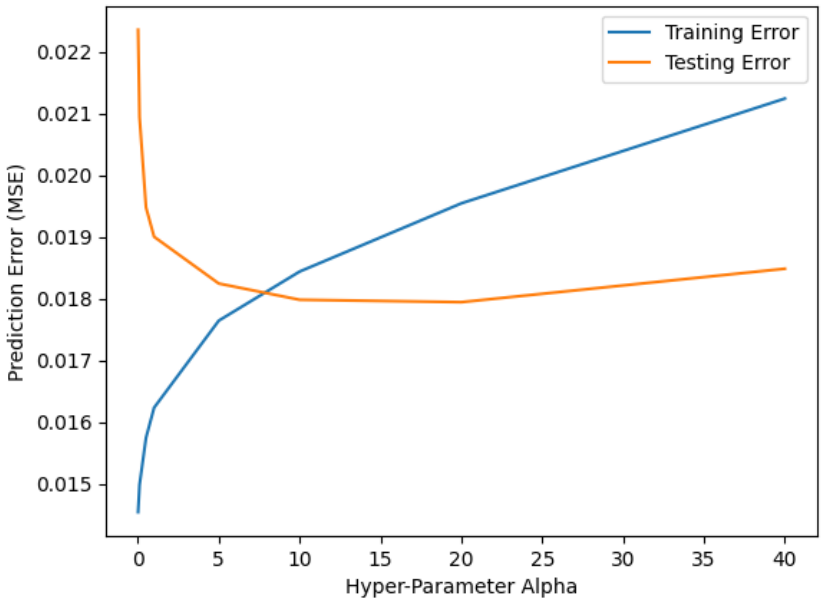

Figure 2: Plot of Prediction Error (MSE) vs Hyper-Parameter α

At the beginning we have some overfitting going on with the Training Error being substantially Lower than the Test Error (model too complex). However the higher the alpha value, up to a certain point, the model becomes better at predicting the Test Error (sweet spot). If you further increase alpha though, the model becomes to simple (underfitting), leading to an increase in Test MSE.

Hyper-Parameter (α)	0.1	0.5	1	5	10
Validation Error (R^2)	0.627	0.632	0.633	0.631	0.628

Table 1: Hyper-Parameter (α) value vs Validation Error (R^2)

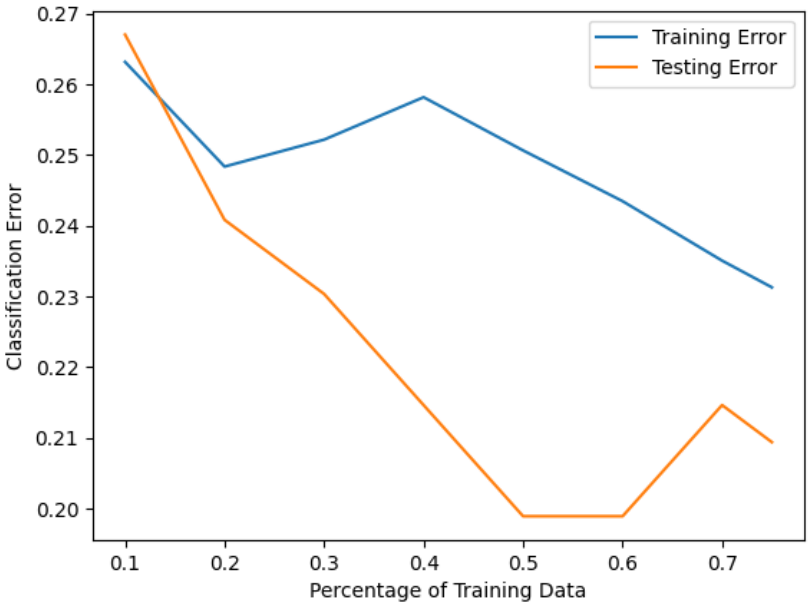

Figure 3: Classification Error vs Percentage of Training Data

The lower the training error (more data points used), the lower the test error to a certain point. At this point, overfitting comes into play where a lower training error doesn't mean a lower test error and actually leads to a higher test error.

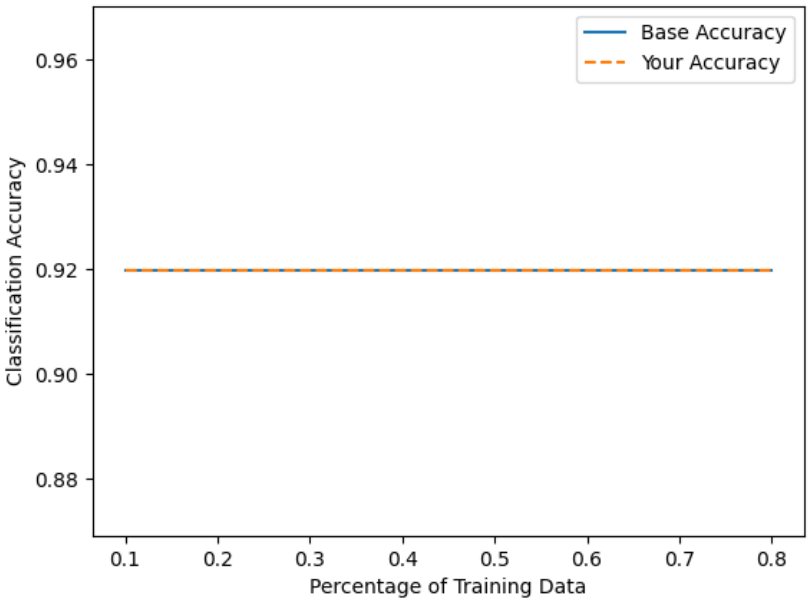

Figure 4: Classification Accuracy vs Percentage of Training Data

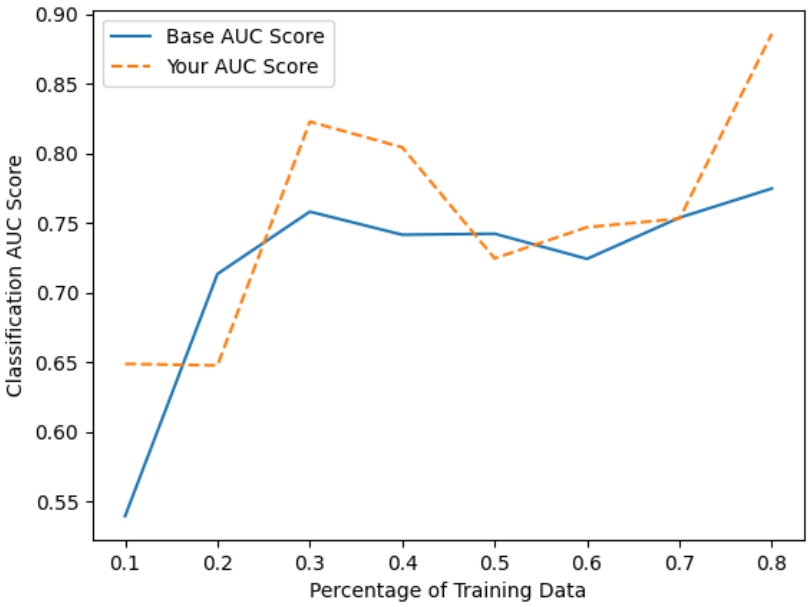

Figure 5: Classification AUC Score vs Percentage of Training Data

Multiple hyper-parameters were tested on the Logistic Regression method, but most changes were just bad to negligible. So instead I tried using XGBClassifiers (which are boosted decision trees) to make the AUC score higher. At depth 5 it matches the accuracy of the Logistic Regression Model, but performs substantially better at AUC score the more data you train on. (This was trained on the unbalanced data set, `diabetes_new.csv`)

```
default_model = LogisticRegression()
my_model = XGBClassifier(n_estimators=100, learning_rate=0.01, max_depth=5) #
```

Figure 6: Differences in My Model vs Default Model