**2.1 Logistic Regression**

At first, a logistical regression model was set up to perform up to 2000 iterations. It was provided with two sets of data for training and testing. Its effectiveness was evaluated using F1 score, accuracy, precision, and recall to address any class imbalances. Additionally, the influence of each feature on the model’s decisions was analyzed.

**2.2 Decision Tree Classifier**

Using a decision tree classifier optimized by a grid search, we determined the best set of hyperparameters. Specifically, we identified the maximum depth of the tree, the minimum number of samples required to split an internal node, and the minimum number of samples required to be at a leaf node. The optimal values we found were a maximum depth of 8, with a minimum of 1 leaf sample and 10 samples required to split a node. The model's performance was then evaluated on the same metrics as above, and feature influences were examined.

**2.3 Random Forest**

Third, we use a random forest model. In this random forest model, we conducted hyperparameter tuning sing grid search to optimize the model's performance. We set the maximum depth of each tree in the forest to be 12, the n-estimator of 200, random states to be 42. After the model was run, standard performance metrics were used to assess its robustness, and the most important features were studied.

**2.4 K-Nearest Neighbors (KNN)**

Next, we attempted to classify the data with a k-nearest neighbors classifier. The number of neighbors used was determined by taking the square root of the number of total samples. The KNN model utilized 265 neighbors for the classification. After being trained on the training data, this model underwent performance evaluation using F1 score, accuracy, precision, and recall.

**2.5 Support Vector Machine (SVM)**

The fifth method we used is the SVM. The SVM used a linear kernel with a regularization parameter of 1.0. The model's performance was also assessed using F1 score, accuracy, precision, and recall. An analysis was also performed to investigate the feature importance.

**2.6 XGBoost**

We did a lot of hyperparameter tuning for XGBoost, very carefully varying the learning rate, tree depth, number of trees in the ensemble, and regularization terms. We found that the optimal parameters were identified as a learning rate of 0.1, 200 estimators, and regularization terms reg\_alpha at 0.5 and reg\_lambda at 1. The model was then evaluated using the standard set of metrics, and feature importance were visually represented through a horizontal bar graph to highlight the most influential predictors.

**2.7 AdaBoost**

Finally, the AdaBoost classifier underwent hyperparameter tuning using grid search to optimize key parameters. After identifying the optimal settings, the model was trained and evaluated on the test set using the same performance metrics. We also investigated the resulting feature importance of the AdaBoost classification.