Classifier Details

For this Project, 11 classifiers were chosen to evaluate the data. Brief explanations of each classifier are provided to explore their differences in operation as they provide different results during the model training.

Naive Bayes

The Naive Bayes Classifiers are based on the Bayes Theorem, a method that allows for the inversion of conditional probabilities. The Bayes Theorem describes the probability of an event based on prior knowledge of conditions potentially related to the event.

The Naive Bayes Classifiers are known as Naive because they assume certain aspects of the dataset to make the classification problems less computationally intensive. First, they assume all of the features equally contribute to the prediction. Second, they assume the features are all independent and unrelated.

The Gaussian Naive Bayes Classifier processes the data assuming that each data feature is distributed according to the gaussian distribution.

The Bernoulli Naive Bayes Classifier processes the data according to multivariate Bernoulli Distributions. Thus, each feature is processed as binary-valued. In case the variables are not binary, the instance of the classifier binarizes the input data before processing.

Decision Tree

The Decision Tree Classifier is a classification method that employs a greedy divide-and-conquer method to create the tree. The tree is split until the majority or all data records are classified under the class labels.

K Neighbors

The K Nearest Neighbor Classifier is a classification method that assigns labels to the data points based upon majority voting, where each data point is labeled according to the label of the majority of data points found to be closely related/represented to them.

Logistic Regression

The Logistic Regression Classifier is a classification method that estimates the probability of a classification outcome based on data with features assumed to be independent.

The Logistic Regression fits a logistic sigmoid function predicting binary values, assigning each data point a classification label based on a threshold value.

Random Forest

The Random Forest Classifier is a feature bagging ensemble method composed of decision trees where each tree in the ensemble is made of a data sample drawn from the training data set. The method then trains each tree independently. The predicted class is then outputted by the categorical variables with the majority of votes.

Gradient Boosting

The Gradient Boosting Classifier is a boosting ensemble method based on the gradient descent algorithm. The method iteratively adds predictors to the ensemble to correct the errors of the existing predictors by training on the residual errors of the existing predictor.

Histogram Gradient Boosting

The Histogram Gradient Boosting Classifier is a boosting ensemble method based upon the gradient boosting classifier. The Histogram Gradient Boosting Classifier significantly speeds up the gradient boosting method by binning the input data, speeding up the construction of the decision trees.

Light Gradient Boosting Machine

The Light Gradient Boosting Machine Classifier is a boosting ensemble method based on the histogram gradient boosting classifier. The implementation of this classifier focuses on two separate methods: gradient-based one-side sampling and exclusive feature bundling. Gradient-based one-sided sampling is a method that focuses on training examples that have a larger gradient. Exclusive feature bundling is an automatic feature selection method focused on bundling sparse mutually exclusive features.

Extreme Gradient Boosting

The Extreme Gradient Boosting Classifier is a boosting ensemble method based on the histogram gradient boosting classifier. Optimized with a focus on speed, the classifier has several features, some allowing missing data through the maximization of gain and using weighted quantiles to find the best node splits instead of analyzing all candidates.

Cat Boosting

The Cat Boosting Classifier is a boosting ensemble method based upon the histogram gradient boosting classifier. The implementation of this classifier focuses on forming symmetric decision trees. In addition, the classifier performs well with categorical datasets by focusing on encoding categorical features based on the label columns.

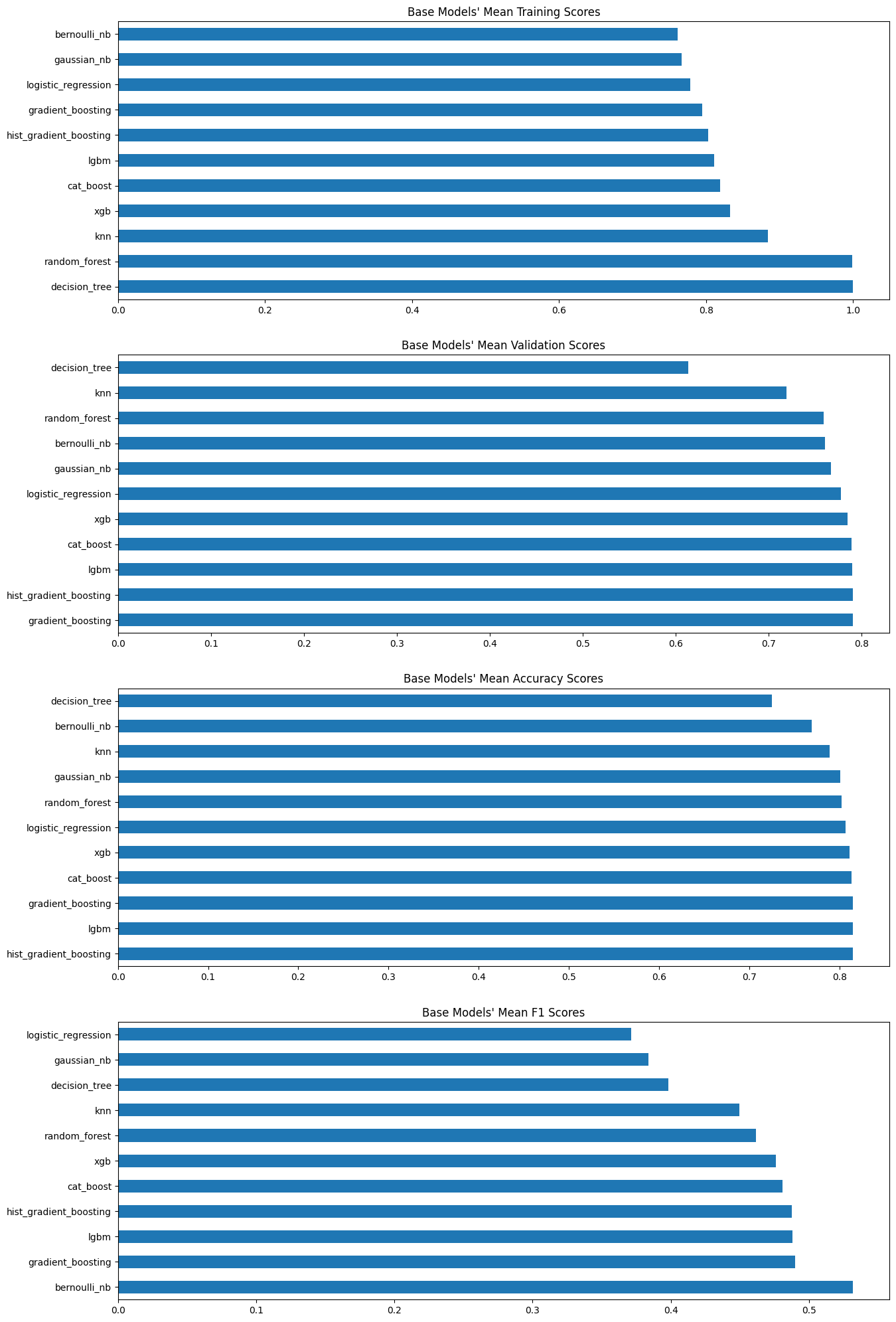
Cross Validation

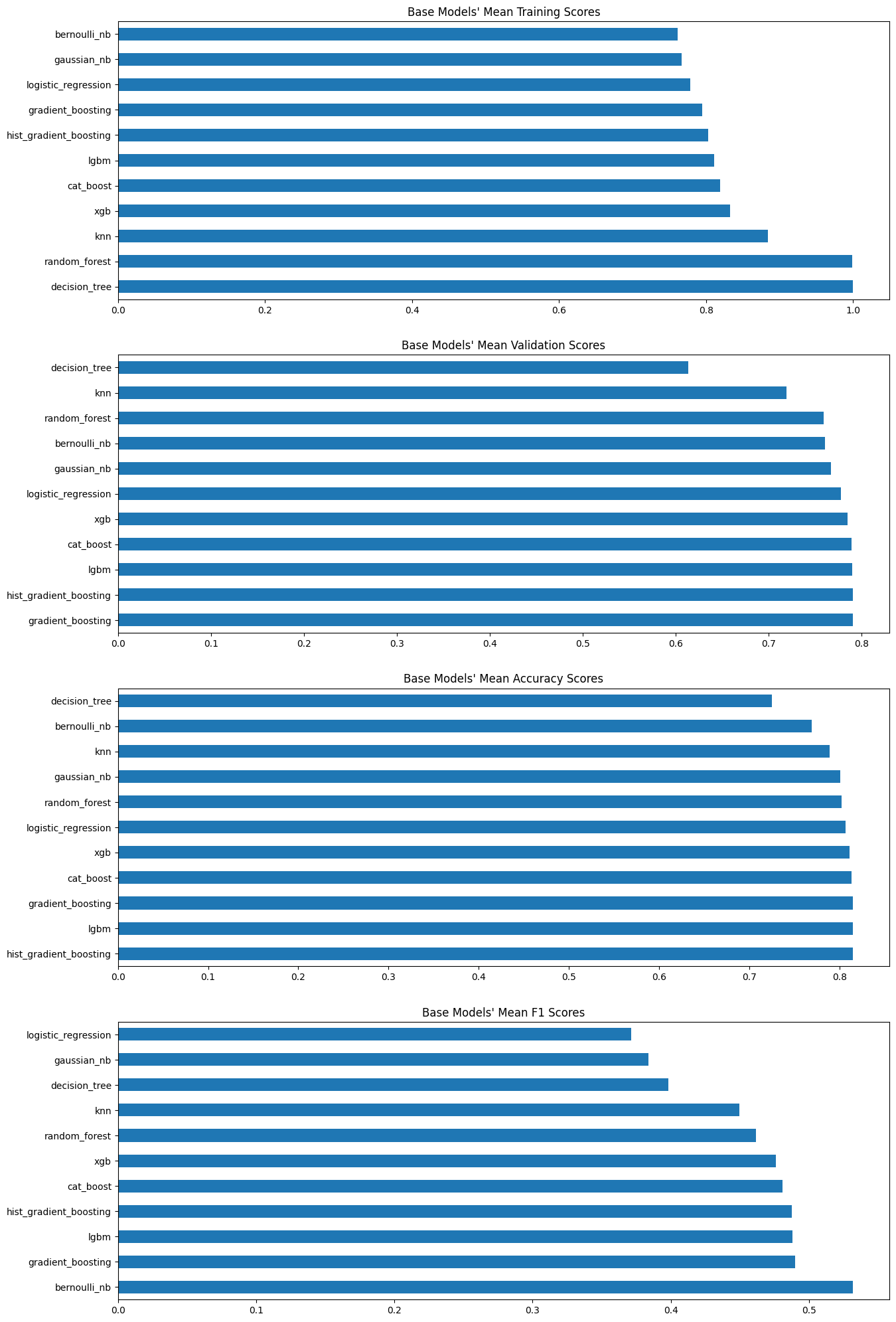
Once data feature selection and standardization were completed, a custom cross-validation process was created to evaluate each classifier and its performance. The repeated stratified k-fold cross-validation method was defined with five folds and ten repeats to ensure that during cross-validation, each fold contains the same proportion of input data with defect and non defect labels.

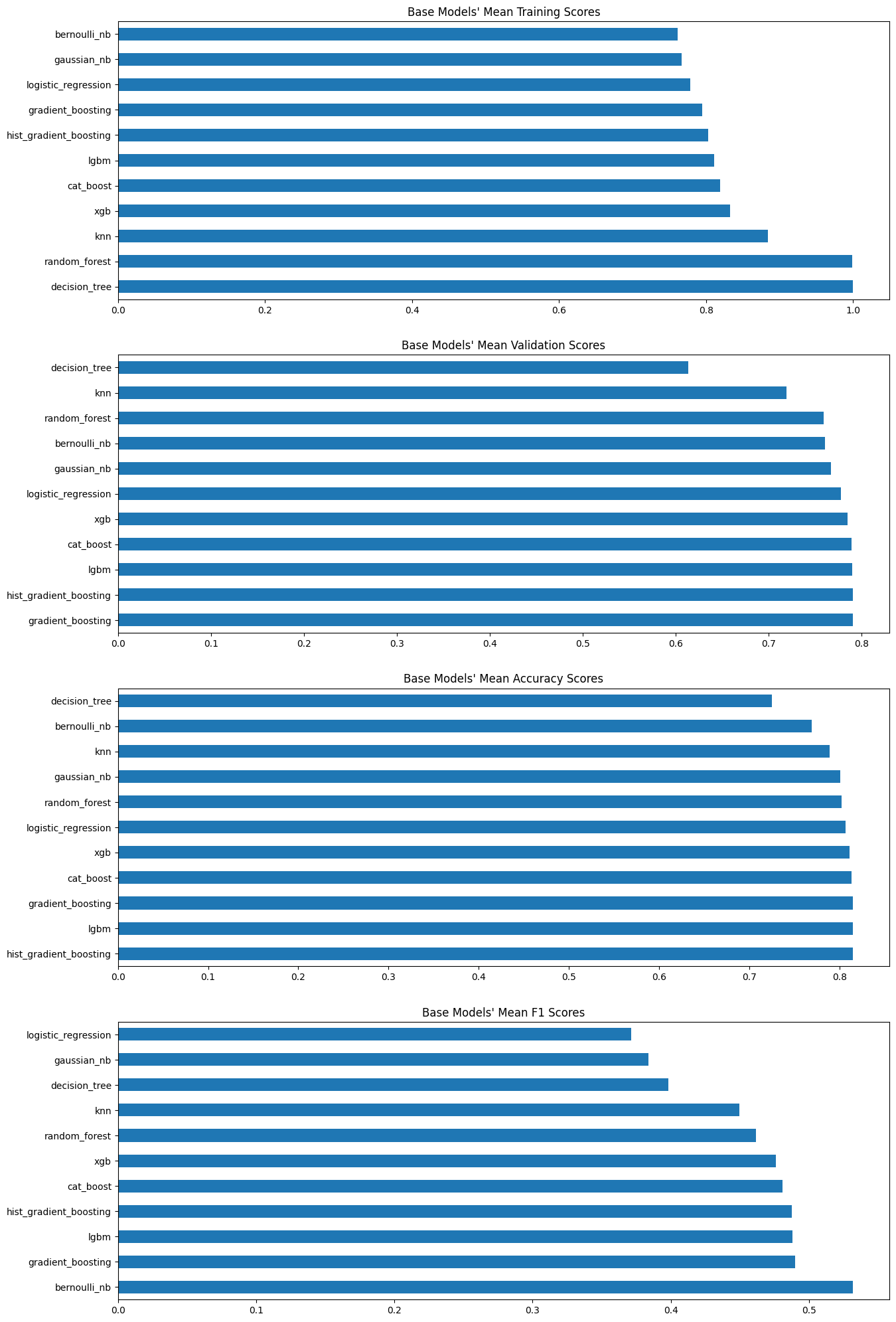
During evaluation, four metrics were analyzed, the average training ROC AUC score, average validation ROC AUC score, average accuracy, and average F1 score. The ROC AUC score is the area under the receiver operating characteristic curve, which plots the true positive rate against the false positive rate at varying classification thresholds. The ROC AUC score details how well a classifier can distinguish between the positive and negative classes. The accuracy describes the percentage of observations that are correctly classified. The F1 score is the harmonic mean of the precision and recall metrics to increase the F1 score is to maximize both metrics. Precision is a metric that measures the ratio of correct positive class predictions, and recall is a metric that measures the ratio of positive class samples correctly identified.

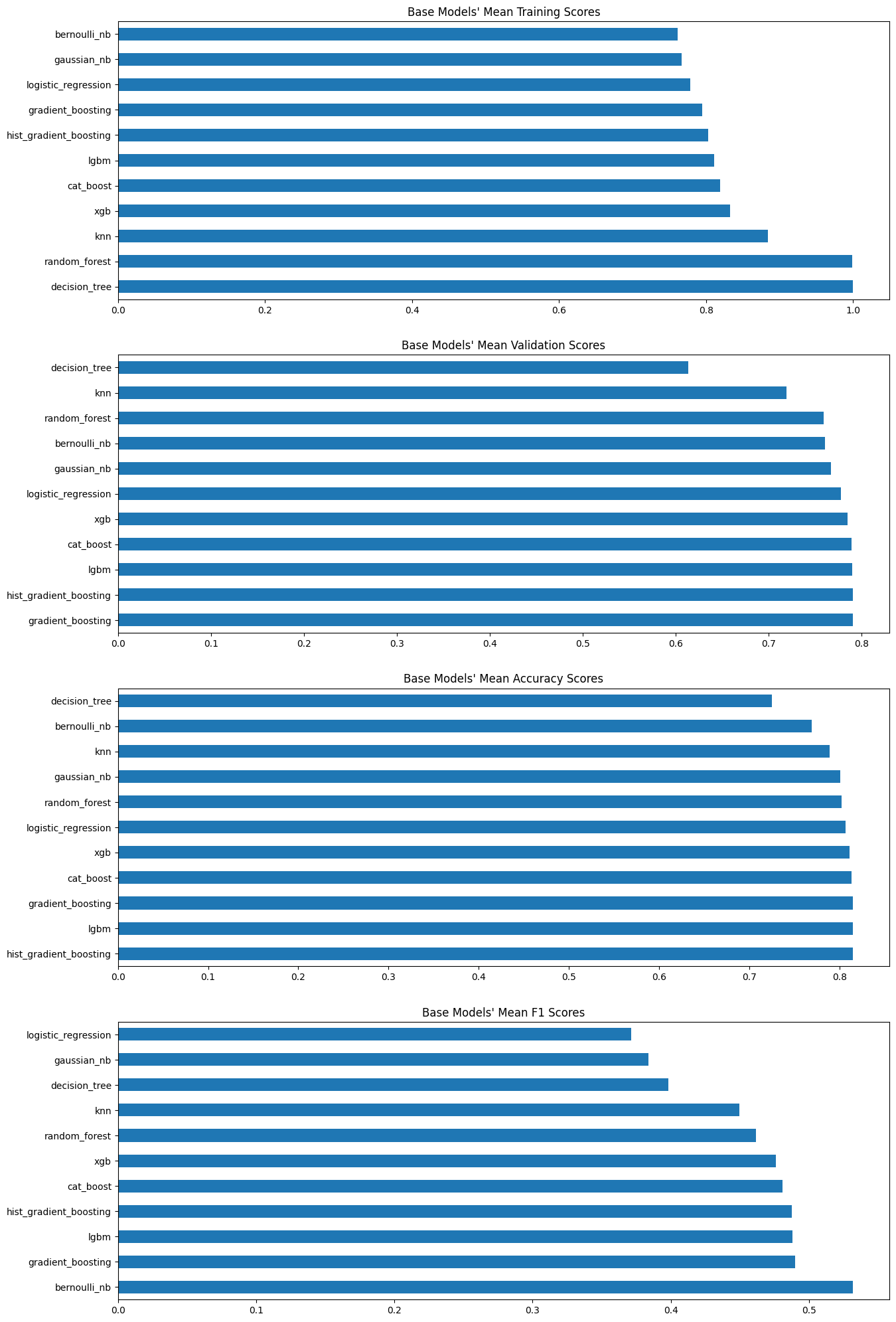
Base Models

Initially, cross-validation was performed on the 11 classifiers without modifying the model parameters beyond adding a seed to ensure consistent results after training. The results of cross-validation are displayed below. The average training ROC AUC scores range between .76 and .99. From these scores, there is an inclination to believe there is an aspect of overfitting in the training of the models. For the average validation ROC AUC, the scores range from .61 to .79. The average accuracy range for the models is .72 to .81. Although accuracy may not accurately represent each classifier’s performance since the distribution has a 77:23 skew. Average F1 scores were disappointing, with a range from .37 to .53.









Figures: Base Model Average Training/Validation ROC AUC, Accuracy, and F1 Score Bar Charts

| **Model Name** | **Ave. Training ROC AUC Score** | **Ave. Validation ROC AUC Score** | **Ave. Accuracy** | **Ave. F1 Score** |
| --- | --- | --- | --- | --- |
| Gaussian NB | 0.7671589 | 0.7671437 | 0.8006672 | 0.3837027 |
| Bernoulli NB | 0.760931 | 0.7608771 | 0.7692737 | 0.5317257 |
| Decision Tree | 0.9995711 | 0.6138554 | 0.7248548 | 0.3985115 |
| KNN | 0.884046 | 0.7194356 | 0.7890225 | 0.4495136 |
| Logistic Regression | 0.7781648 | 0.7780576 | 0.8067461 | 0.3711798 |
| Random Forest | 0.9985411 | 0.759114 | 0.8024754 | 0.461492 |
| Gradient Boosting | 0.7948394 | 0.7906932 | 0.8146203 | 0.4898802 |
| Hist Gradient Boosting | 0.8028327 | 0.7906572 | 0.8147686 | 0.4876493 |
| LGBM | 0.810897 | 0.7903612 | 0.8147549 | 0.4881741 |
| XGB | 0.8323212 | 0.7849803 | 0.8113244 | 0.4759125 |
| Cat Boost | 0.8193732 | 0.7892677 | 0.8130283 | 0.4809887 |

Figure: Average Training/Validation ROC AUC, Accuracy, and F1 Scores For All Base Model Classifiers

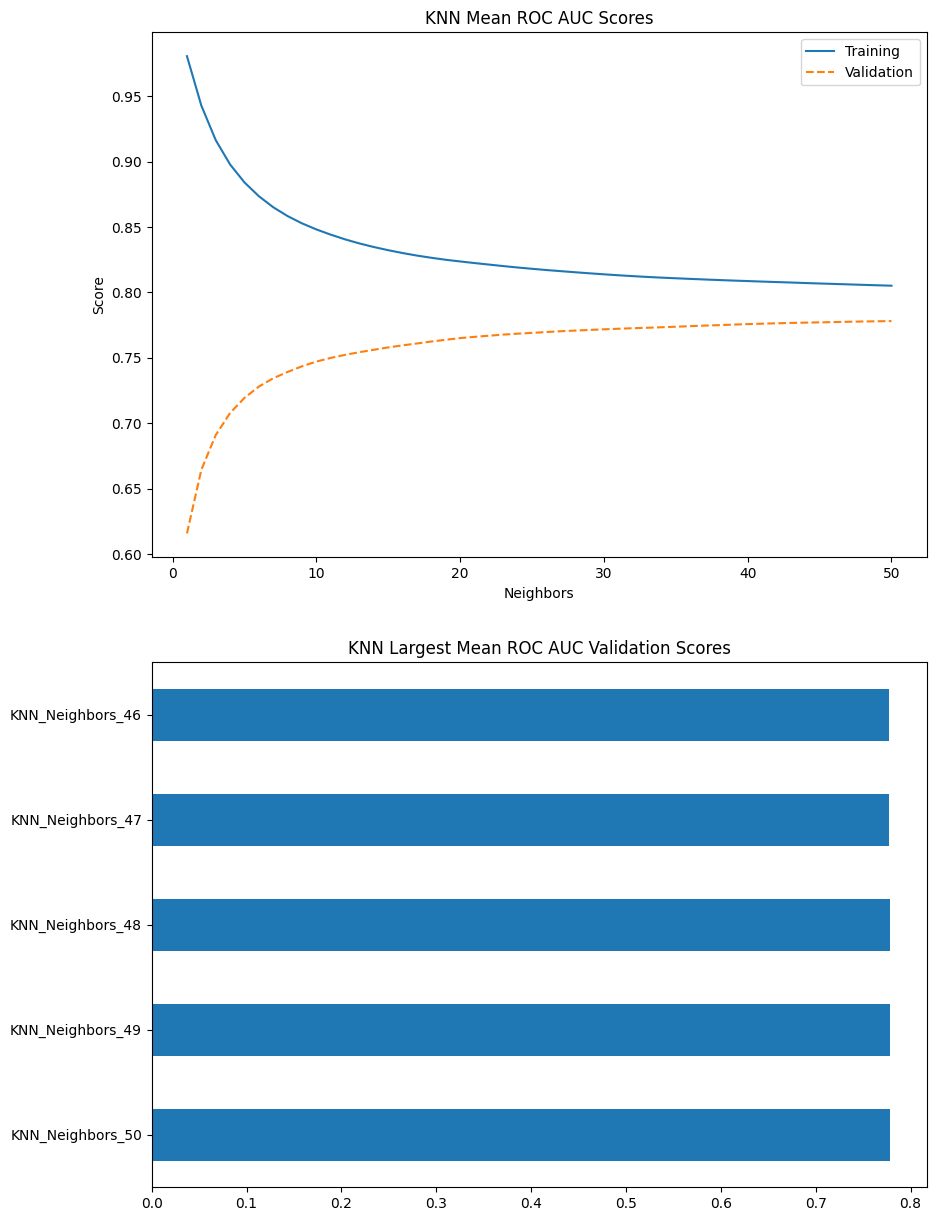
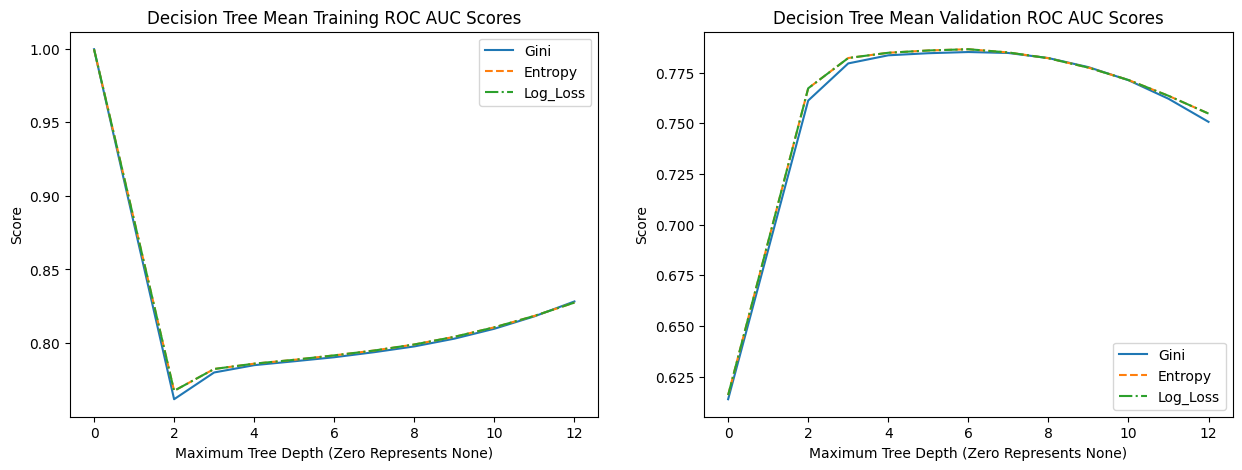
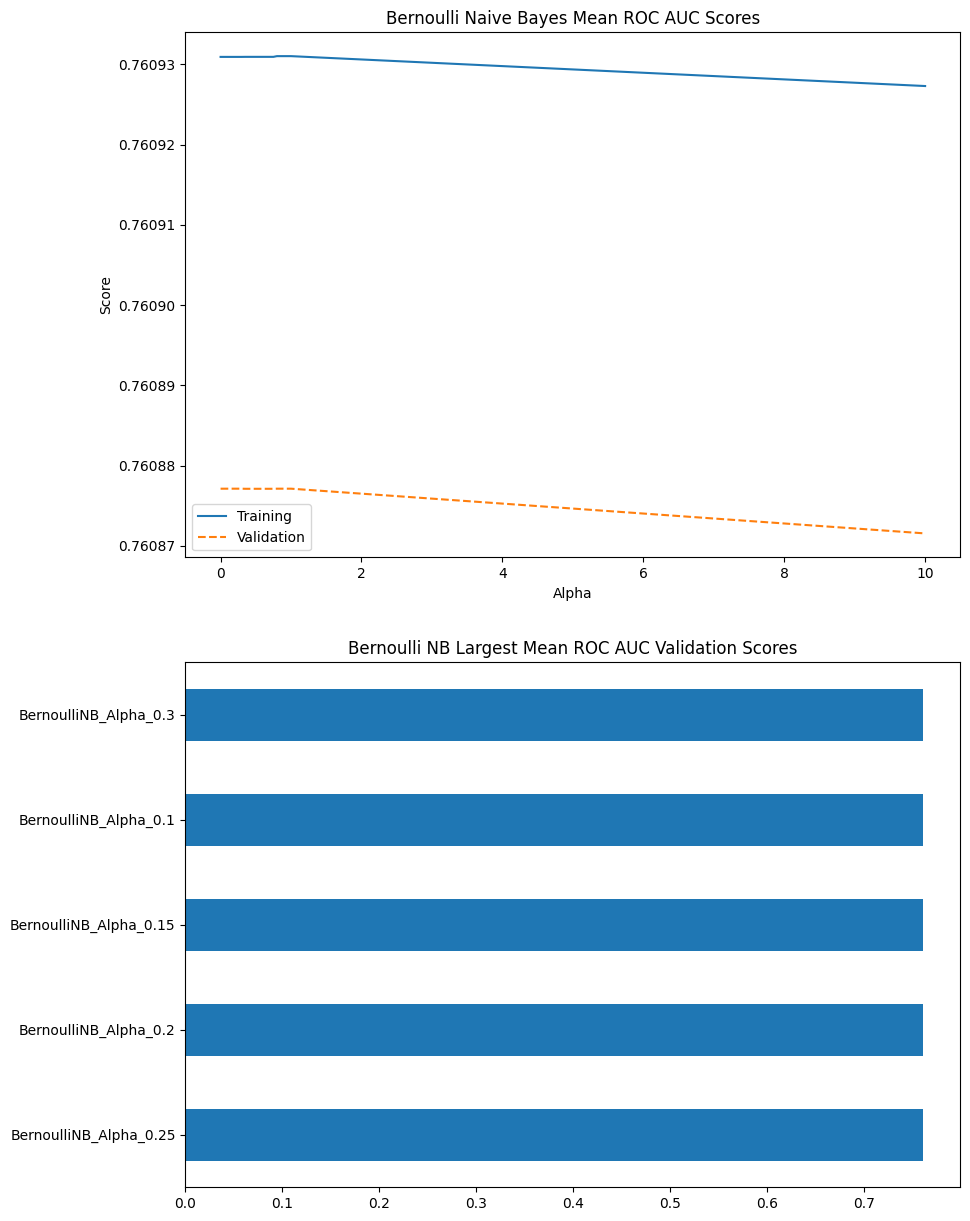
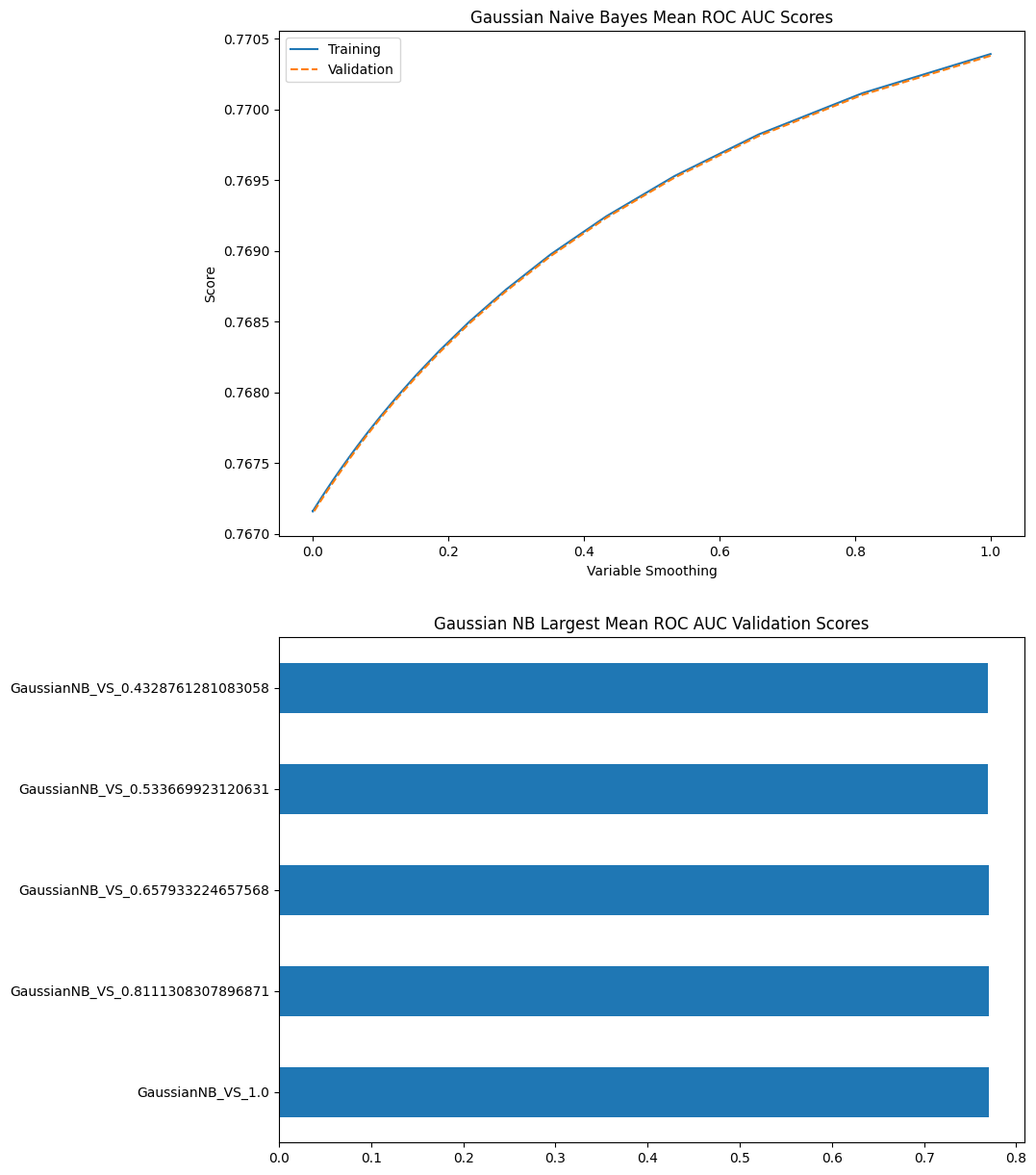
Classifier Optimization

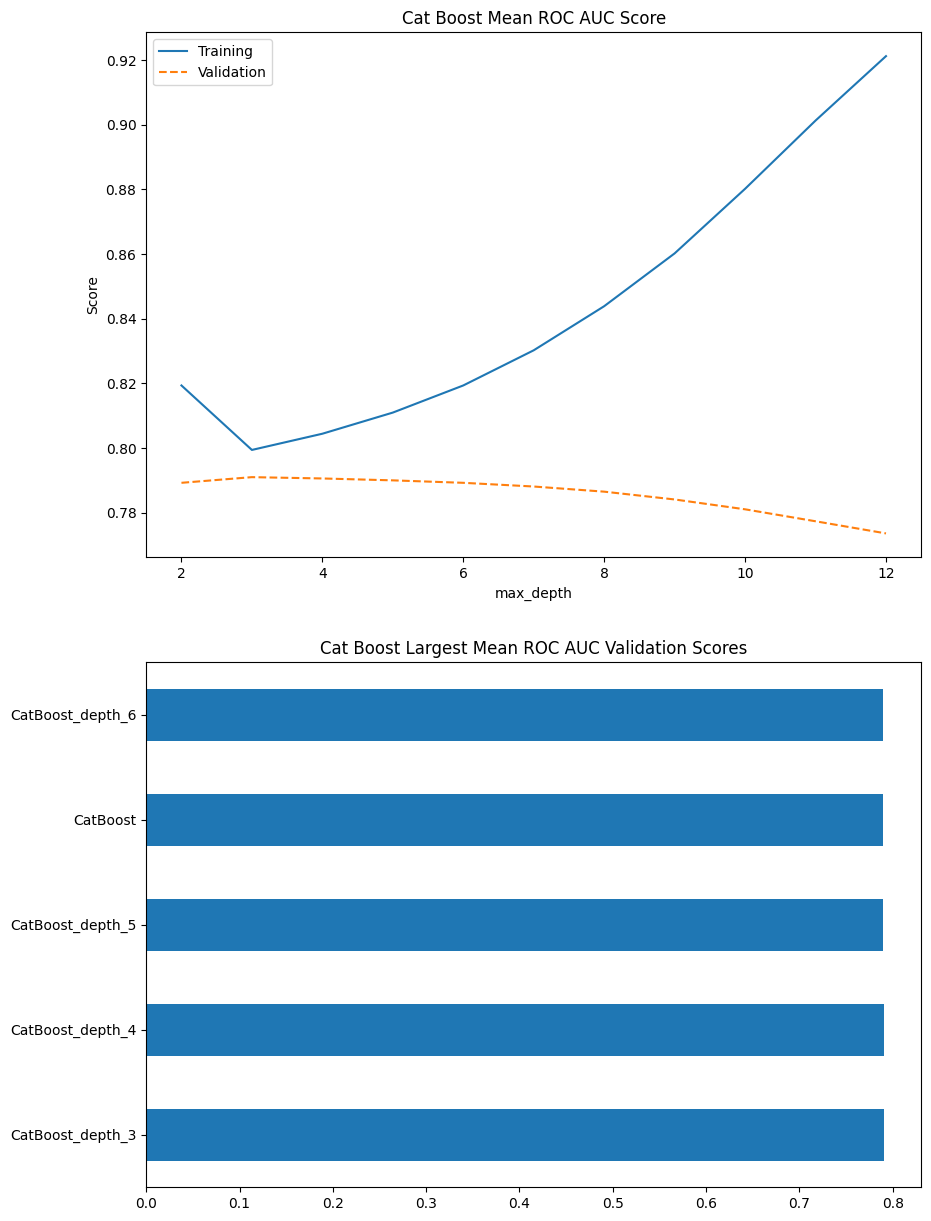
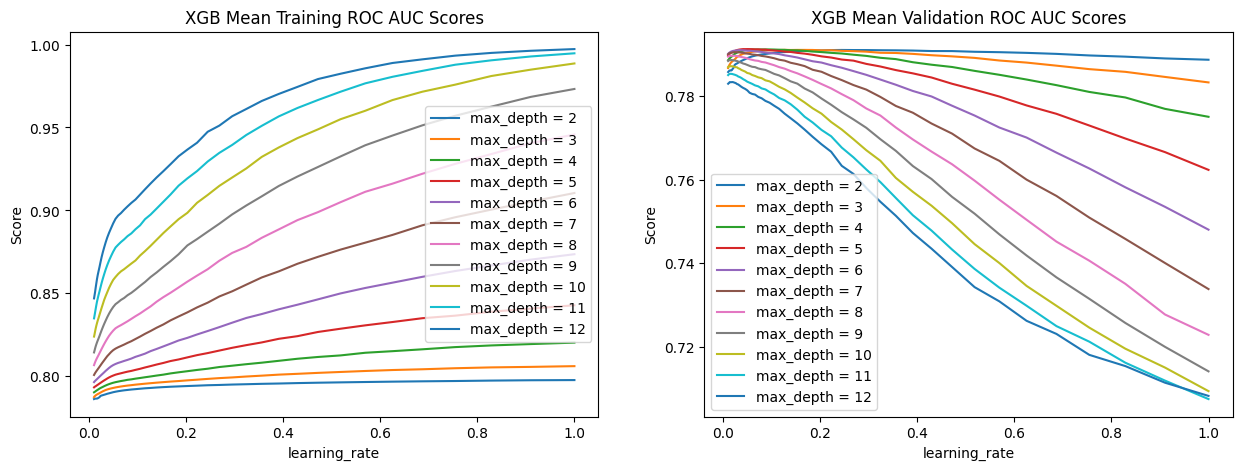
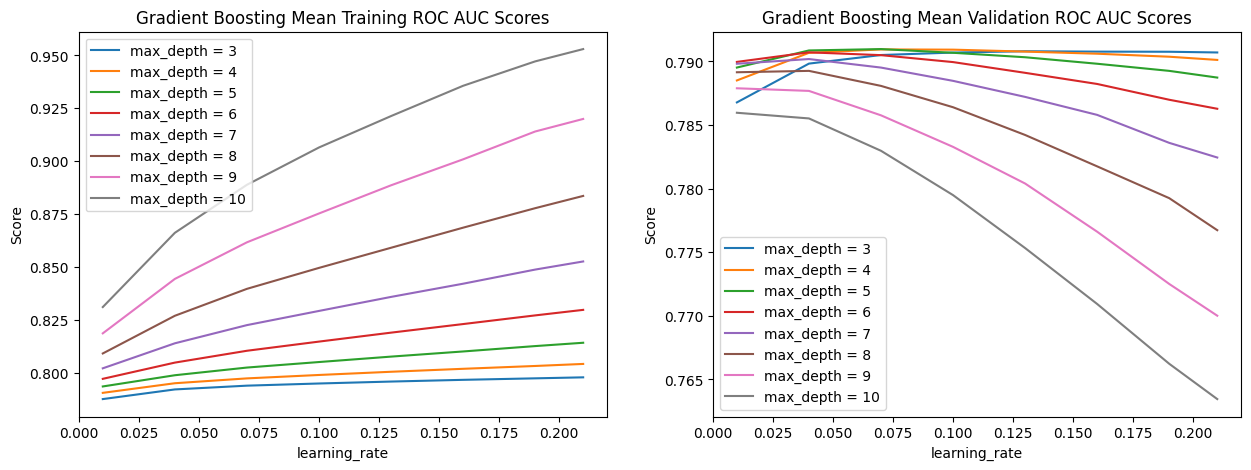
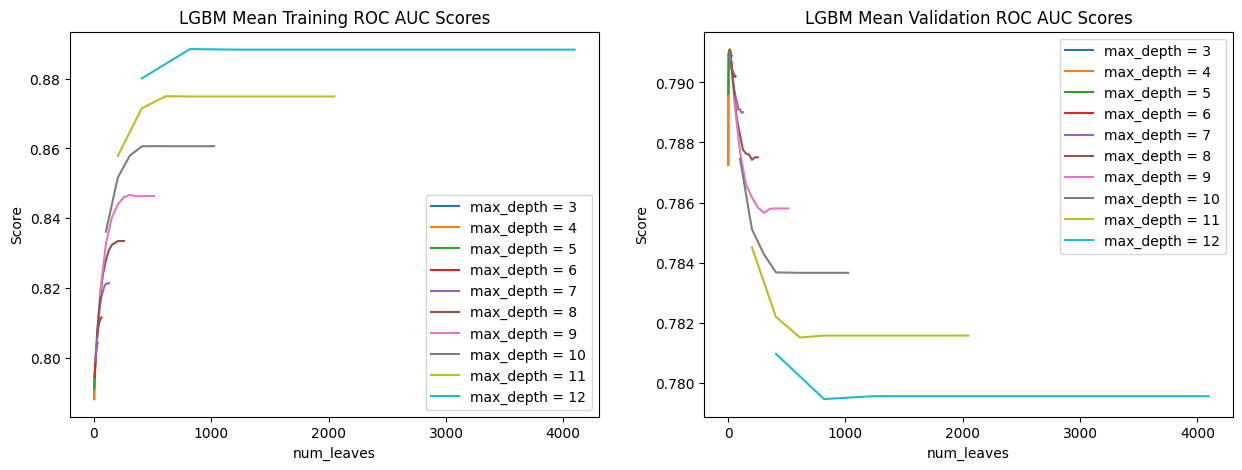
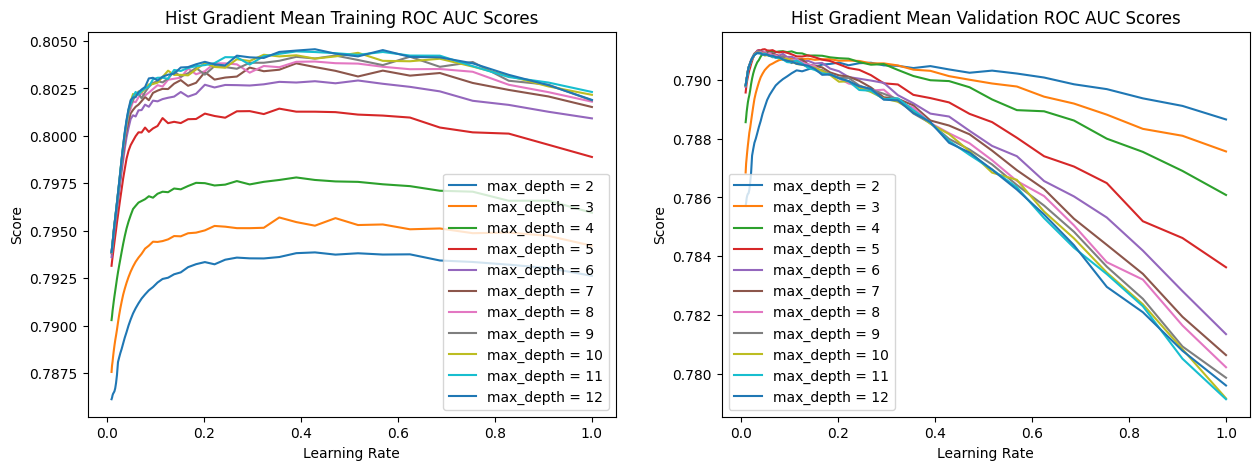
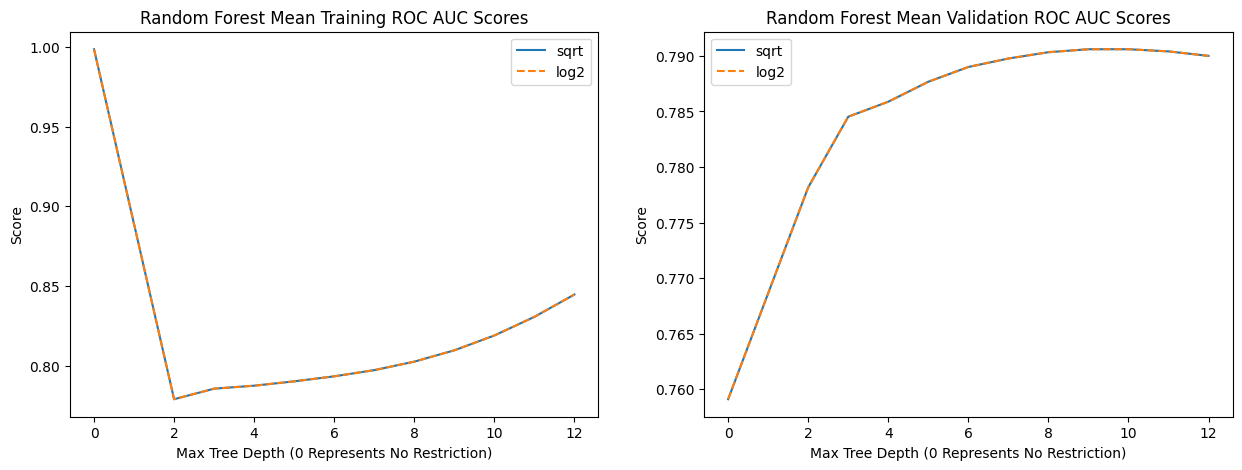
To improve each model and the results of their metrics, each classifier was independently cross-validated to tune their hyperparameters. They were each uniquely tuned based on the available hyperparameters, computational complexity, and the overall effects of the hyperparameter modification.

For each of the 11 classifiers, a model with specific hyperparameter values was selected based on the average validation ROC AUC score. The validation ROC AUC represents how well the model can distinguish between classes. Overall, it was believed that the validation ROC AUC would best represent each model’s performance with unknown testing data. The hyperparameters modified for each classifier are displayed in the table below. In addition, the mean validation ROC AUC score plots for each model are included below.

| **Model** | **Hyper Parameters Tuned** |
| --- | --- |
| Gaussian Naive Bayes | * Variance Smoothing |
| Bernoulli Naive Bayes | * Alpha |
| Decision Tree | * Criterion * Max Tree Depth |
| K Nearest Neighbor | * Neighbors |
| Logistic Regression | * Solvers * C Values |
| Random Forest | * Max Tree Depth * Max Features |
| Gradient Boosting | * Max Tree Depth * Learning Rate |
| Hist Gradient Boosting | * Max Tree Depth * Learning Rate |
| Light Gradient Boosting Machine | * Max Tree Depth * Max Number of Leaves |
| Extreme Gradient Boosting | * Max Tree Depth * Learning Rate |
| Cat Boosting | * Max Tree Depth |

Figure: Hyperparameters Tuned For Each Classifier



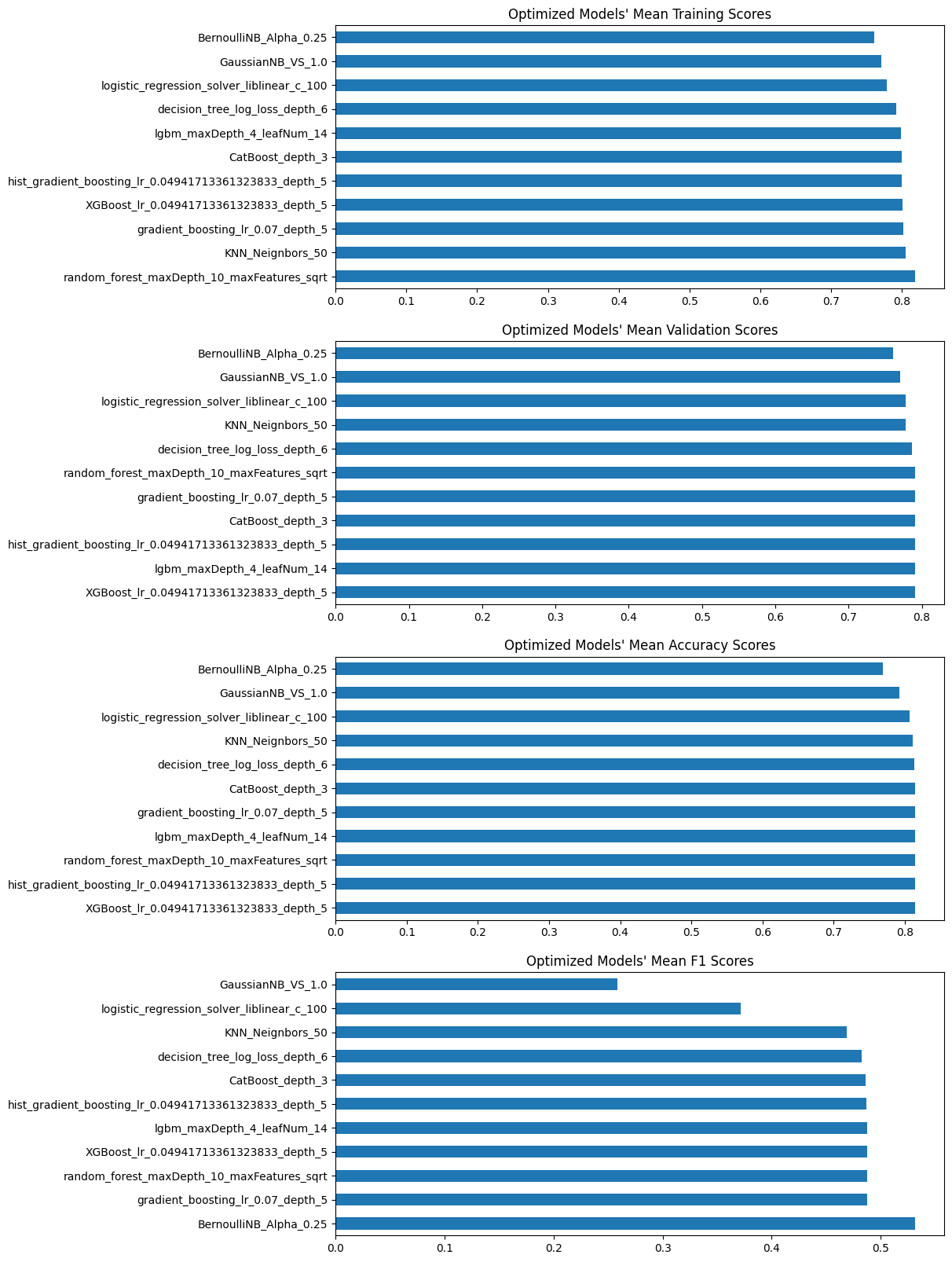


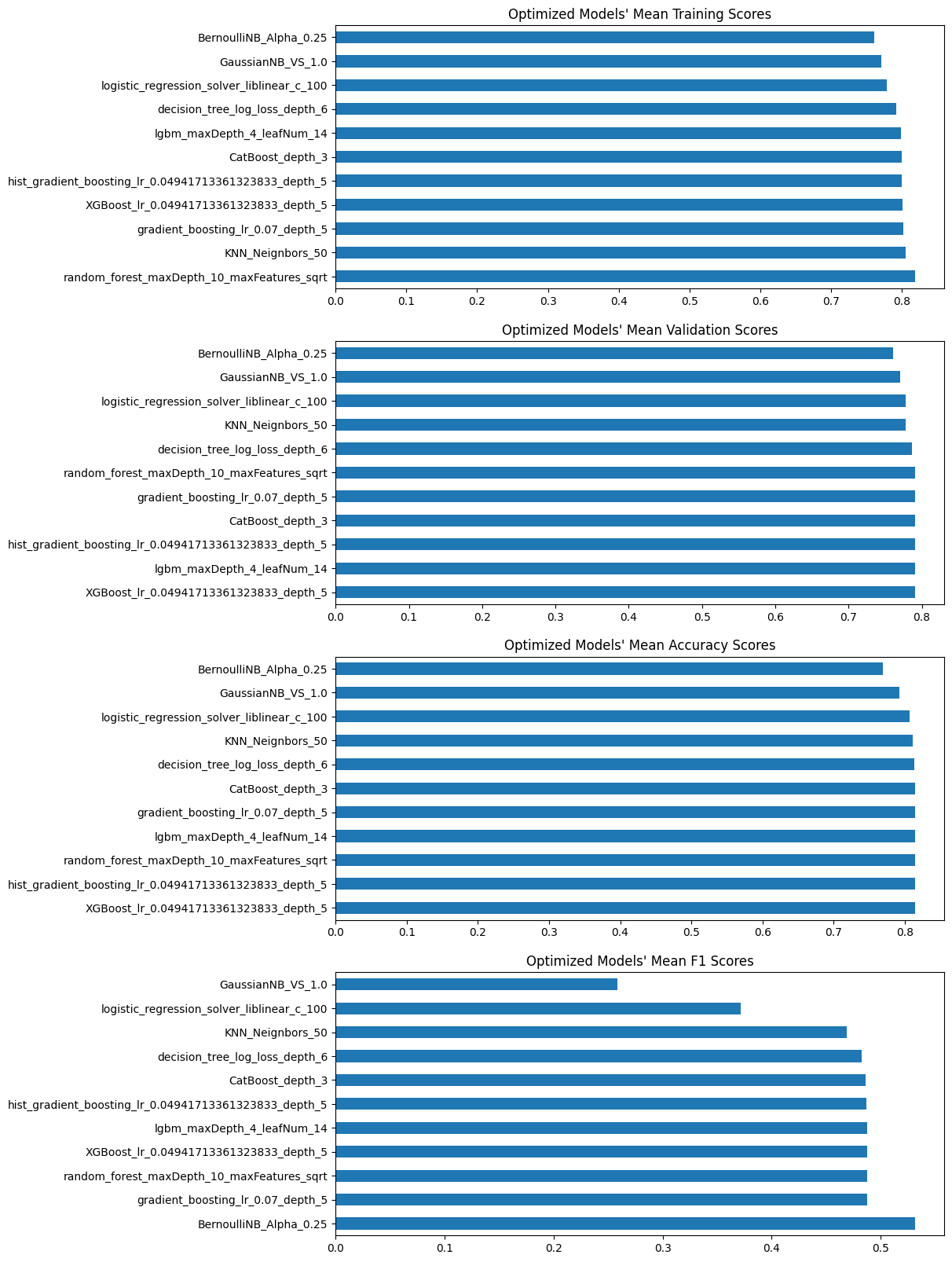
Figures: Hyperparameter Tuning Classifier Average Validation ROC AUC Scores

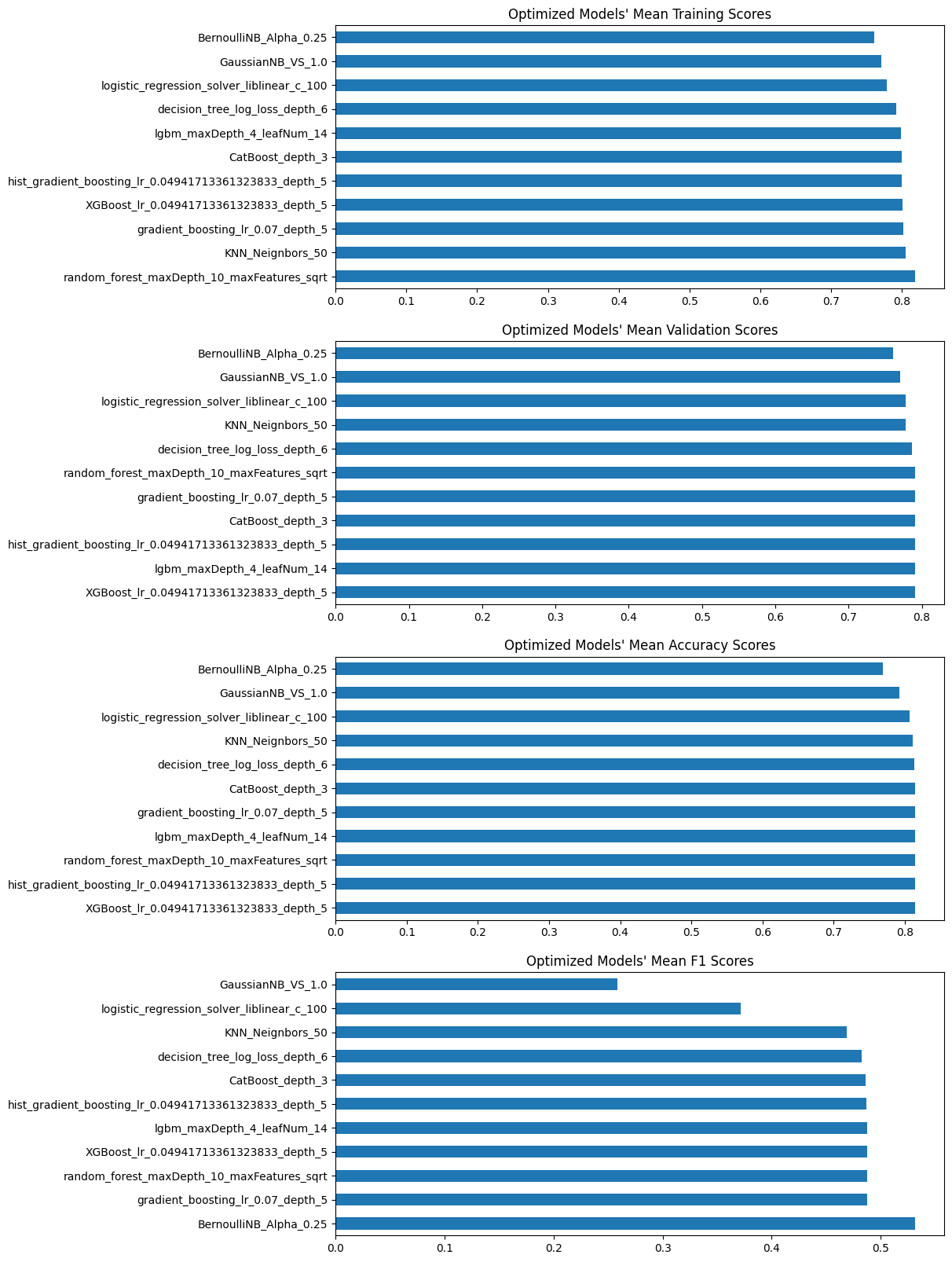
Evaluating Optimized Models

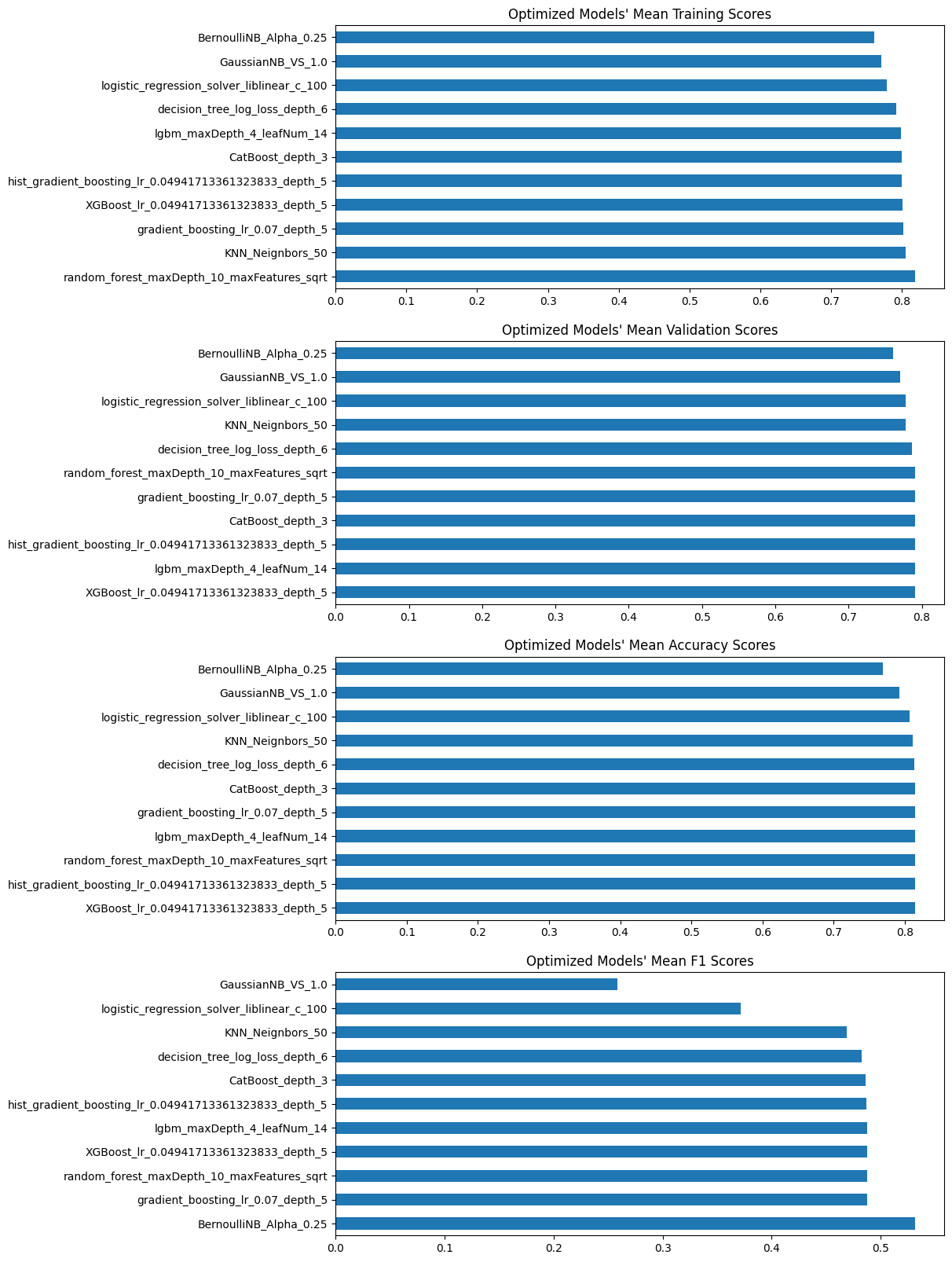
Upon selecting the models from each classifier with the largest mean validation ROC AUC scores, the models were cross-validated once more and evaluated. The average training ROC AUC scores for the optimized models range from .76 to .81. The top end of the range was decreased by .18, dramatically reducing the overfitting of the models. The average accuracy range is from .76 to .81, a slight improvement over the base model range of .72 to .81. The average F1 score range is from .25 to .53, which is a performance decrease from the base model average F1 range of .37 to .53. Now the average validation ROC AUC score range seemed to narrow up to between .76 and .79.

Overall, the optimization of the models seemed to have mainly narrowed the ranges of the metrics. The average F1 score range did see a .12 decrease from the lower end range, which does indicate a decline in either recall, precision, or both. Overall, the model optimized with preference to the average validation ROC AUC scores did not improve as much as hoped. Instead, there may have been an overall decrease in many of the models. Models like logistic regression did have slight improvement, but this appears to be the exception. The results of the optimized model cross-validation are below.









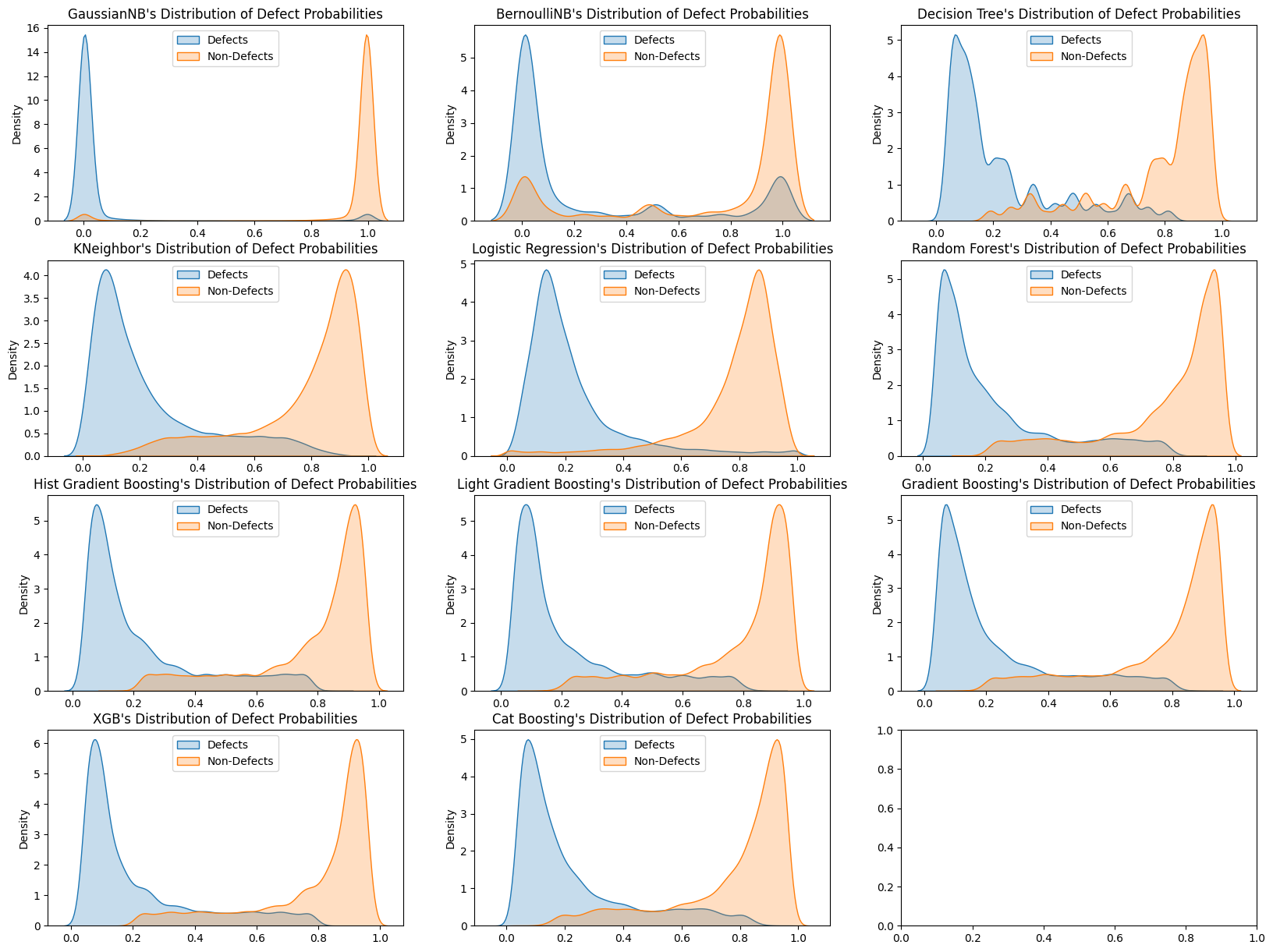
Figures: Optimized Model Average Training/Validation ROC AUC, Accuracy, and F1 Score Bar Charts

| **Model Name & Hyperparameter Values** | **Ave. Training ROC AUC Score** | **Ave. Validation ROC AUC Score** | **Ave. Accuracy** | **Ave. F1 Score** |
| --- | --- | --- | --- | --- |
| GaussianNB  Variable Smoothing: 1.0 | 0.7703925 | 0.7703802 | 0.7925759 | 0.2587808 |
| BernoulliNB  Alpha: 0.25 | 0.7609309 | 0.7608771 | 0.7692737 | 0.5317257 |
| Decision Tree  Criterion: log loss Max Depth: 6 | 0.7916948 | 0.7865117 | 0.8135423 | 0.4824395 |
| KNN  Neighbors: 50 | 0.8051304 | 0.7781011 | 0.8117106 | 0.4686465 |
| Logistic Regression Solver: liblinear  C Value: 100 | 0.7781662 | 0.7780591 | 0.80675 | 0.3712207 |
| Random Forest Max Depth: 10 Max Features: sqrt | 0.8188424 | 0.790596 | 0.8148463 | 0.4876912 |
| Hist Gradient Boosting  Learning Rate: 0.04941713361323833  Max Depth: 5 | 0.7995138 | 0.7910308 | 0.8148699 | 0.4870412 |
| LGBM  Max Depth: 4 Max Leaf Number: 14 | 0.7983258 | 0.7911049 | 0.8146743 | 0.4876798 |
| Gradient Boosting  Learning Rate: 0.07  Max Depth: 5 | 0.8023741 | 0.7909673 | 0.8146645 | 0.4877596 |
| XGBoost  Learning Rate: 0.04941713361323833  Max Depth: 5 | 0.800319 | 0.7911931 | 0.814921 | 0.4876838 |
| CatBoost  Max Depth: 3 | 0.7994325 | 0.791019 | 0.8144041 | 0.4859718 |

Figure: Average Training/Validation ROC AUC, Accuracy, and F1 Scores For All Optimized Model Classifiers

Testing Data

The final aspect of this project is analyzing the testing data with the validation ROC AUC optimized models. Since the testing data was provided for a Kaggle competition, the actual labels are unknown. Instead, the distribution of defect predictions was calculated for each model. The observation made on the distribution plots is that the ensemble models all appear to have similar distributions. The kernel density estimate plots are displayed below.



Figures: Distribution of Defect Predictions For Each Optimized Classifier

Conclusion

In conclusion, it was found that for the overall base model classifiers, the average F1 scores appear to be rather low, ranging between .37 and .53. For the base models, the ensemble classifiers and Bernoulli Naive Bayes performed better based upon the average F1 and average validation ROC AUC scores. The model optimization based upon validation ROC AUC did not appear to make much of an improvement, if any. The optimized classifiers that performed best appeared to be the ensembles, all with similar metric results. The lack of improvement through optimizing the average validation ROC AUC may be due to how imbalanced the training data was, skewing the results.

In the future, the group would like to further explore the data present in the data sets before continuing to cross-validate and test the classifier models. Exploring other pre-processing methods may be beneficial for each model to improve the results. The metrics used to evaluate the model would also need reevaluation since the optimization of the models did not yield the expected improvements to the results. In addition, based on the results of the cross-evaluated classifiers, the exploration of a custom classifier ensemble may create an improved classifier for the data.

References:

<https://www.ibm.com/topics/naive-bayes>

<https://www.ibm.com/topics/decision-trees>

<https://www.ibm.com/topics/knn>

<https://www.ibm.com/topics/logistic-regression>

<https://www.ibm.com/topics/random-forest>

<https://www.ibm.com/topics/boosting>

<https://machinelearningmastery.com/histogram-based-gradient-boosting-ensembles/>

<https://machinelearningmastery.com/light-gradient-boosted-machine-lightgbm-ensemble/>

<https://www.geeksforgeeks.org/ml-xgboost-extreme-gradient-boosting/>

<https://www.geeksforgeeks.org/gradientboosting-vs-adaboost-vs-xgboost-vs-catboost-vs-lightgbm/>