Data Modeling Overview

Our goal is to determine the best classification model that will assist medical professionals in accurately predicting a heart failure patient's survival based on an unseen input vector of medical record features. Preprocessing the original raw dataset resulted in a smaller subset of data containing 224 rows, 7 feature columns and 1 target feature as displayed in table 1.

Table 1 Summary of Each Feature of the Processed Dataset

#	Feature	Description	Data Type	Domain
1	Age	Patient's age	Integer	[40, 95]
2	Anemia	Presence of anemia	Boolean	0, 1
3	Creatine Phosphokinase	Level of enzyme in blood	Integer	[23, 7861]
4	Ejection Fraction	Percentage of blood ejected per heartbeat	Integer	[14, 80]
5	Platelets	Platelet count in blood	Float	[25100, 850000]
6	Serum Creatinine	Level creatinine in blood	Float	[0.5. 9.4]
7	Serum Sodium	Level of sodium in blood	Integer	[113, 148]
8	Patient Deceased **(Targert Feature)**	Patient died	Boolean	0, 1

The target feature column, 'patient_deceased', is a binary classification task where class 0 indicates that the patient survived and class 1 signifies the patient's death. We have observed that there is an unequal distribution of the target feature's class with 162 patients that survived and 61 patients that are deceased. The preprocessed dataset was further partitioned into training and test subsets, with the test subset comprising 30% of the total data. Partitioning the data is crucial as it allows us to train our models, tune each model's respective hyperparameters and evaluate the effectiveness of our models on unseen test data.

Four supervised classification machine learning models were chosen for this task which consists of Logistic Regression, K-Nearest Neighbors, Random Forests, and Gradient Boosted Trees. Each model was evaluated by first tuning hyperparameters using cross validation with training and validation folds using the training subset. Then with the optimal hyperparameters the classification models are evaluated on the test subset and test metrics obtained were compared across all models. Performance of each classification model chosen depends significantly on the hyperparameters used. Hyperparameter tuning was performed using Grid Search Cross Validation or Randomized Search Cross Validation in tandem with Stratified K-Fold Cross Validation using 10 folds. Cross validation is performed using 10 folds meaning that the training set

comprising 70% of the total data is divided into 10 groups called folds. The model will iterate through the K folds, use the i'th fold as the validation set and remaining K - 1 folds as the training set. This technique provides a broad evaluation of the model's performance with 1 fold of data set aside for validation resulting in medium bias and medium variance as the training sets are correlated.

Stratified K-Fold Cross Validation is a variation of K-Fold Cross Validation that utilizes stratified sampling. Stratified sampling is employed to address significant imbalances in the target feature's distribution of classes. It ensures that each training and validation fold maintains a similar proportion of class frequencies as the complete dataset. Grid Search Cross Validation is an exhaustive search which generates models with all possible combinations of hyperparameters given a parameter grid input. The tradeoff of ensuring that we determine the best combination of hyperparameters is that this search is computationally expensive and results in a long training time. In contrast, Randomized Search Cross Validation randomly selects combinations of hyperparameters to evaluate model performance. Randomized Search is computationally less expensive and is useful when a model requires a large number of hyperparameters to tune. We determined that Logistic Regression and K Nearest Neighbors will use Grid Search Cross Validation for hyperparameter tuning as both only require 1 hyperparameter to tune. Random Forests and Gradient Boosted Trees will utilize Randomized Search Cross Validation for hyperparameter tuning. Although this does not guarantee that the best hyperparameters will be found for these models, a tradeoff has to be made as the number of hyperparameters that require tuning make the training time unfeasible to be completed in a reasonable amount of time.

Performance of our classification models on the training, validation folds and test subsets were evaluated using a confusion matrix. The confusion matrix provides metrics of accuracy, precision, recall and F1 score and receiver operating characteristic curve (ROC) area under the curve (AUC). Accuracy is the measure of the proportion of correct classifications calculated by adding true positive and negative predictions

		Predicted condition	
	Total population = P + N	Positive (PP)	Negative (PN)
Actual condition	Positive (P)	True positive (TP)	False negative (FN)
	Negative (N)	False positive (FP)	True negative (TN)

Figure 1 Confusion Matrix

divided by the total number of predictions. Precision measures how many of the positive predictions are actually correct by dividing the number of true positives by the sum of true and false positives. Recall measures how many positive predictions are identified correctly, calculated using the number of true positives divided by the sum of true positives and false negatives. The F1 score is the harmonic mean of precision and recall metrics, representing both precision and recall equally in a single metric. The ROC AUC is an aggregate measure of performance across all classification thresholds which compares the true positive rates as the y axis vs the false positive rate as the x axis. All evaluation metrics range from a lower bound of 0 and upper bound of 1, where 1 represents a perfect prediction. The best model with respect to classifying a heart failure patient's survival given their medical records will maximize both accuracy and ROC AUC.

Model Evaluations

Before training, hyperparameter tuning and performing predictions on the test subset, a pipeline is made which first standardizes the features then performs model training. Standardization is a preprocessing technique that transforms each continuous numerical feature individually to have a mean of 0 and a standard deviation of 1. This is important as some machine learning algorithms perform worse when features are not normally distributed.

Logistic Regression is a linear model for classification that models the probability $p(X) = p(Y \mid X)$ that the predicted class Y belongs to a specific category given the independent variables or features X. Classification of an unknown set of features is determined by the largest probability output. Logistic Regression uses the logistic function that transforms the log-odds to a probability value that will always fall between 0 and 1 giving it a S shape curve. This model is a parametric model that makes assumptions about the data's distribution with fixed coefficient parameters determined during training resulting in low variance and high bias. Optimal regression coefficients are chosen to maximize the likelihood function. The Logistic Regression model in Scikit-Learn uses L2 regularization. Regularization is utilized to penalize the complexity of a model in order to reduce overfitting. With L2 regularization the penalty term is the sum

of the squared feature weights and the impact of the penalty term is determined by the hyperparameter value lambda or C the inverse of regularization strength in Scikit Learn. Smaller values of C will result in stronger regularization reducing the weights of less important features to approximately zero but does not eliminate them.

Hyperparameter tuning of experimental C values of 0.001, 0.01, 0.1, 0.5, 1, 10 and 25 revealed that the most optimal hyperparameter value of C is 1. The average of the metrics across 10 folds with cross validation are depicted in figure 2. The optimal hyperparameters are reconfigured within the same model intrinsically with Scikit-Learn after hyperparameter tuning is completed. Model performance with C = 1 was then evaluated on the unseen test subset as shown in figure 3. With this data we can observe that there is no presence of overfitting on the training subset with this model. However, there is a trade-off involved in maximizing accuracy and ROC AUC comes at the expense of lower recall and F1 scores.

Accuracy	Precision	Recall	F1	ROC AUC
0.8	0.555	0.763	0.589	0.799

Figure 2 Logistic Regression Mean Training and Validation Metrics

Accuracy	Precision	Recall	F1	ROC AUC
0.761	0.6	0.176	0.273	0.568

Figure 3 Logistic Regression Test Metrics

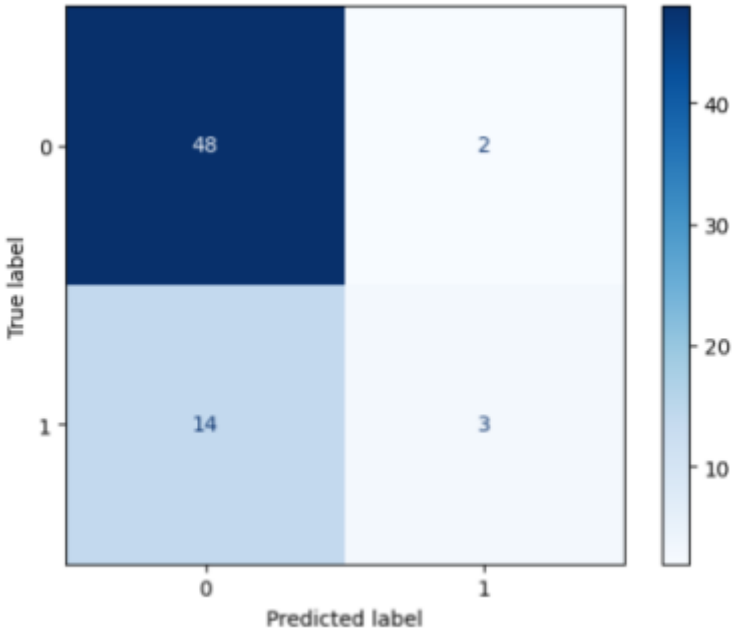

Figure 4 Logistic Regression Confusion Matrix on Test Subset

K Nearest Neighbors (KNN) is an non parametric model and is a type of instance based learning. The only training step in KNN is to identify the nearest neighbors. Classification using KNN of an unknown data point is performed by finding the K closest neighbors based on the Euclidean distance from the unknown point. Then the model assigns the class to the point based on the majority class among these K neighbors. The hyperparameter K influences the corresponding bias and variance of the

model. Small values of K results in low bias and high variance. In contrast, high values of K results in high bias and low variance.

Hyperparameter tuning with experimental K values of 1, 3, 9, 17, 25 and 50 revealed that the optimal value of K is 3. Comparing the mean training metrics across 10 folds given and the model performance on the unseen test subset with K = 3 reveal that there is not a strong presence of overfitting on the training subset of data. From the test metrics, we can observe that the most optimal KNN model sacrifices precision, recall, and F1 scores in favor of higher accuracy and ROC AUC metrics.

Accuracy	Precision	Recall	F1	ROC AUC
0.774	0.525	0.662	0.529	0.703

Figure 5 KNN Mean Training and Validation Metrics

Accuracy	Precision	Recall	F1	ROC AUC
0.672	0.273	0.176	0.214	0.508

Figure 6 KNN Test Metrics

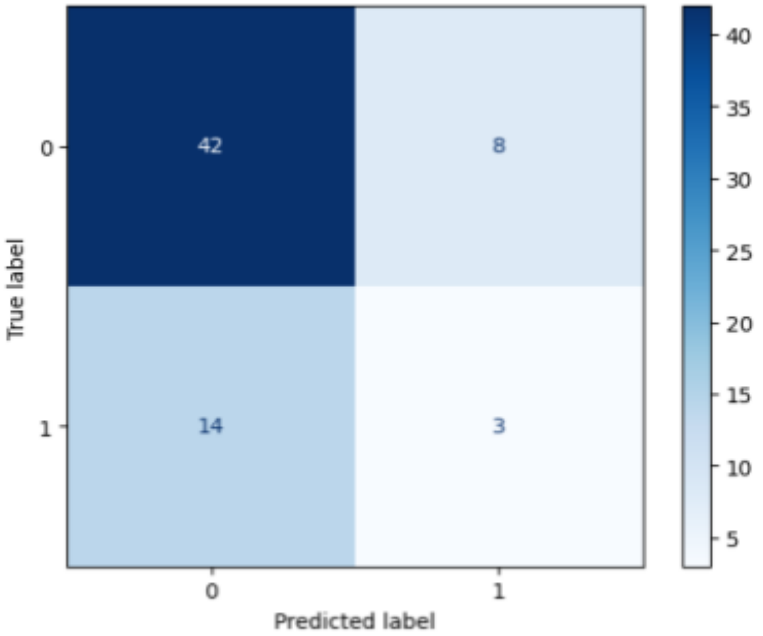

Figure 7 KNN Confusion Matrix on Test Subset

Random Forests Classifier is an ensemble tree based classification model that creates multiple trees through generation of bootstrap samples with replacement from the training subset. Ensemble methods are more effective than normal decision trees because individual decision trees tend to overfit the training data and exhibit high variance. Random Forests introduces a source of randomness to decrease correlation of bootstrap sampled trees by randomly selecting a smaller subset of features from the original set at each split. Decorrelating the trees in return decreases the variance of the forest estimator, improving overall model performance on unseen data. Classification

using Random Forests is performed through taking the majority class vote prediction among all the generated trees.

Random Forests model hyperparameters consist of the number of bootstrapped trees created in the forest (n_estimators), the max depth of all trees (max_depth), the minimum number of samples needed to split an internal node (min_samples_split), the minimum number of samples required at a leaf node (min_samples_leaf) and number of features selected at an internal node, for example max_features = sqrt(n_features), when looking for the best split known as the source of randomness (max_features). To reduce computation time of performing hyperparameter tuning, Randomized Search is favored instead of Grid Search. After hyperparameter tuning we can observe that the optimal parameters are n_estimators = 125, min_samples_split = 5, min_samples_leaf = 1, max_features = sqrt and max_depth = 5. We can observe by comparing the mean metrics over all training and validation folds against the test metrics there is no presence of overfitting on the training set. From the test metrics in figure 10, we can observe that the most

Parameter	Values
n_estimators	[25, 50, 100, 125, 150, 200]
max_depth	[None, 5, 10, 25]
min_samples_split	[2, 5, 10]
min_samples_leaf	[1, 2, 5]
max_features	['sqrt', 'log2']

Figure 8 Random Forests Hyperparameter Tuning

Accuracy	Precision	Recall	F1	ROC AUC
0.776	0.445	0.685	0.477	0.809

Figure 9 Random Forests Mean Training and Validation Metrics

Accuracy	Precision	Recall	F1	ROC AUC
0.746	0.5	0.176	0.261	0.558

Figure 10 Random Forests Test Metrics

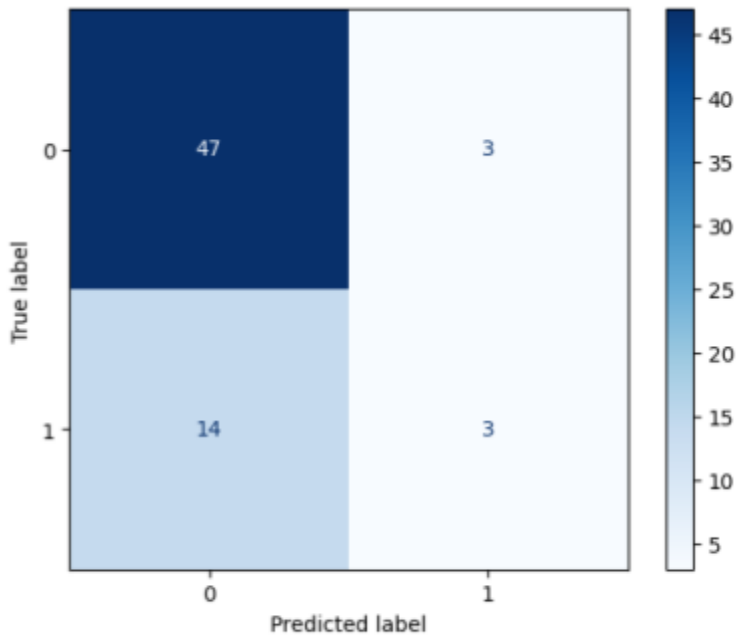

Figure 11 Random Forests Confusion Matrix on Test Subset

optimal Random Forests model sacrifices recall, and F1 scores in favor of higher accuracy and ROC AUC metrics.

Gradient Boosted Trees is another ensemble method that is an iterative process starting with an initial weak decision tree model and fits a new model to reduce the errors or misclassifications of the prior model. This iterative process that occurs N times produces an overall strong model composed of multiple weak models with poor accuracies. In contrast to Random Forests which creates bootstrap samples from the training data, in Gradient Boosting new samples are created sequentially based on where the error occurred in the previous sample. Classification using Gradient Boosted Trees is performed by taking all the N models generated iteratively and taking a weighted class prediction vote. Models generated that have more error have smaller weights and less error have larger weights.

Gradient Boosted Trees model hyperparameters consist of the number of boosting stages or iterations to perform (n_estimators), shrinkage of contribution of each tree (learning_rate), the max depth of all trees (max_depth), the minimum number of samples needed to split an internal node (min_samples_split), the minimum number of samples required at a leaf node (min_samples_leaf), the fraction of samples to used for fitting (subsample) and number of features selected, for example max_features = sqrt(n_features), to determine the best split (max_features). To reduce computation time of performing hyperparameter tuning, Randomized Search is favored instead of Grid Search. After hyperparameter tuning we can observe that the optimal parameters are n_estimators = 200, learning_rate = 0.01, max_depth = 3, min_samples_split = 2, min_samples_leaf = 2, subsample = 0.75 and max_features = log2.

Parameter	Values
n_estimators	[100, 200, 300]
learning_rate	[0.01, 0.1, 1.0]
max_depth	[3, 4, 5]
min_samples_split	[2, 3, 4]
min_samples_leaf	[1, 2, 3]
subsample	[0.5, 0.75, 1.0]
max_features	['sqrt', 'log2', None]

Figure 11 Gradient Boosted Trees Hyperparameters

Accuracy	Precision	Recall	F1	ROC AUC
0.795	0.435	0.75	0.499	0.813

Figure 12 Gradient Boosted Trees Mean Training and Validation Metrics

We can observe by comparing the mean metrics over all training and validation folds against the test metrics there is no presence of overfitting on the training set. From the test metrics in figure 10, we can observe that the most optimal Gradient Boosted Tree model has a significant drop in recall score in favor of higher accuracy and ROC AUC metrics.

Accuracy	Precision	Recall	F1	ROC AUC
0.776	0.667	0.235	0.348	0.598

Figure 13 Gradient Boosted Trees Test Metrics

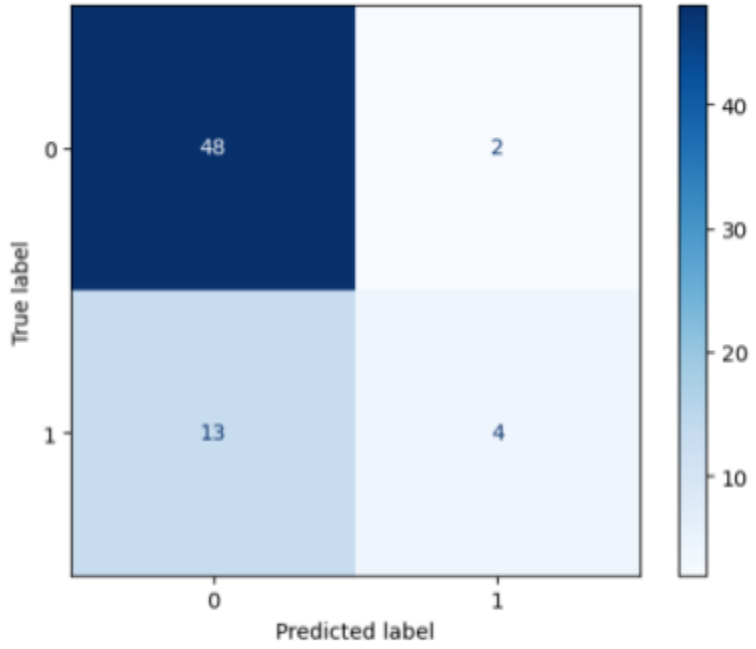

Figure 14 Gradient Boosted Trees Confusion Matrix on Test Subset

Model Comparisons

We can observe that after hyperparameter tuning and model evaluation on the unseen test subset the best performing classification model is Gradient Boosted Trees based on the given metrics in figure 15. Before determining the best model overall

	Accuracy	Precision	Recall	F1	ROC AUC	Model
0	0.761	0.600	0.176	0.273	0.568	Logistic Regression
1	0.672	0.273	0.176	0.214	0.508	KNN
2	0.746	0.500	0.176	0.261	0.558	Random Forest Classifier
3	0.776	0.667	0.235	0.348	0.598	Gradient Boosted Tree

Figure 15 Model Test Set Metrics

the bias and variance tradeoff of each classification model must be taken into account. Bias represents how far off the best model is off from the true target model and variance is the variability of a model's prediction on different datasets or subsets. The ideal model aims to have low bias and low variance. Logistic Regression the simplest classification model used has high bias and low variance because it is a parametric model that makes strong assumptions about the data. If this assumption is not true, the model can underfit the data, leading to high bias. K Nearest Neighbors bias and variance significantly depend on the value of K chosen. Small values of K results in low bias and high variance. While, high values of K results in high bias and low variance. The ensemble machine learning of methods are able to effectively balance the bias vs variance tradeoff overcoming a singular decision tree's weakness of exhibiting high variance. Both Random Forests and Gradient Boosted Trees have a hyperparameter max_features that reduces the feature space at every split at an internal node, reducing variance for an increase in bias. Given this we have decided that the best model for our task balances the bias vs variance tradeoff, has optimal accuracy and optimal ROC AUC is Gradient Boosted Trees.

Our decision to choose Gradient Boosted Trees to effectively classify the survival of heart failure patients is not perfect. We must take into account that the training and testing subsets are quite small. In addition, the metrics of the Gradient Boosted Tree model on the test subset are not ideal. To support our findings, we will validate our results using a larger dataset.