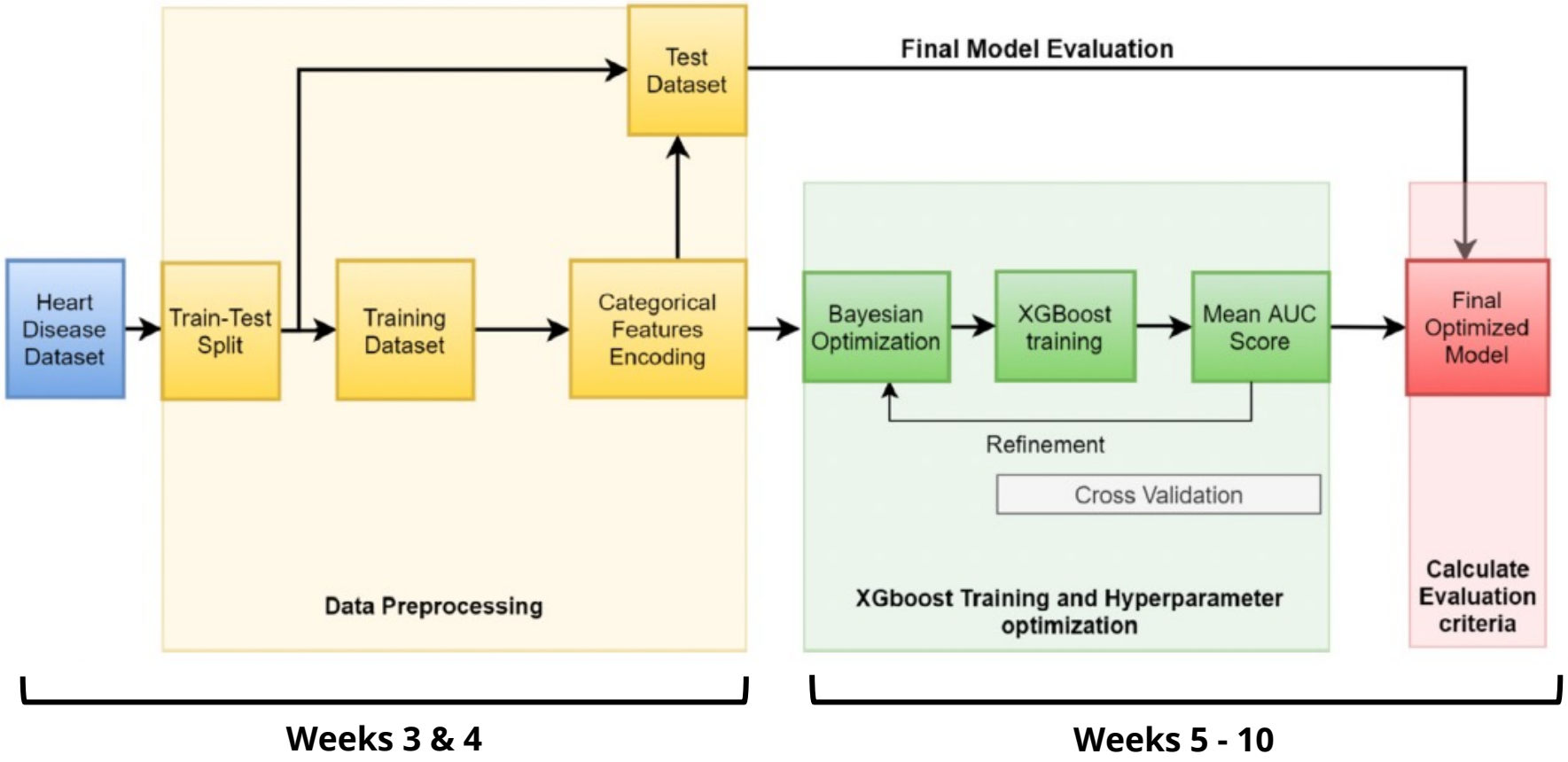
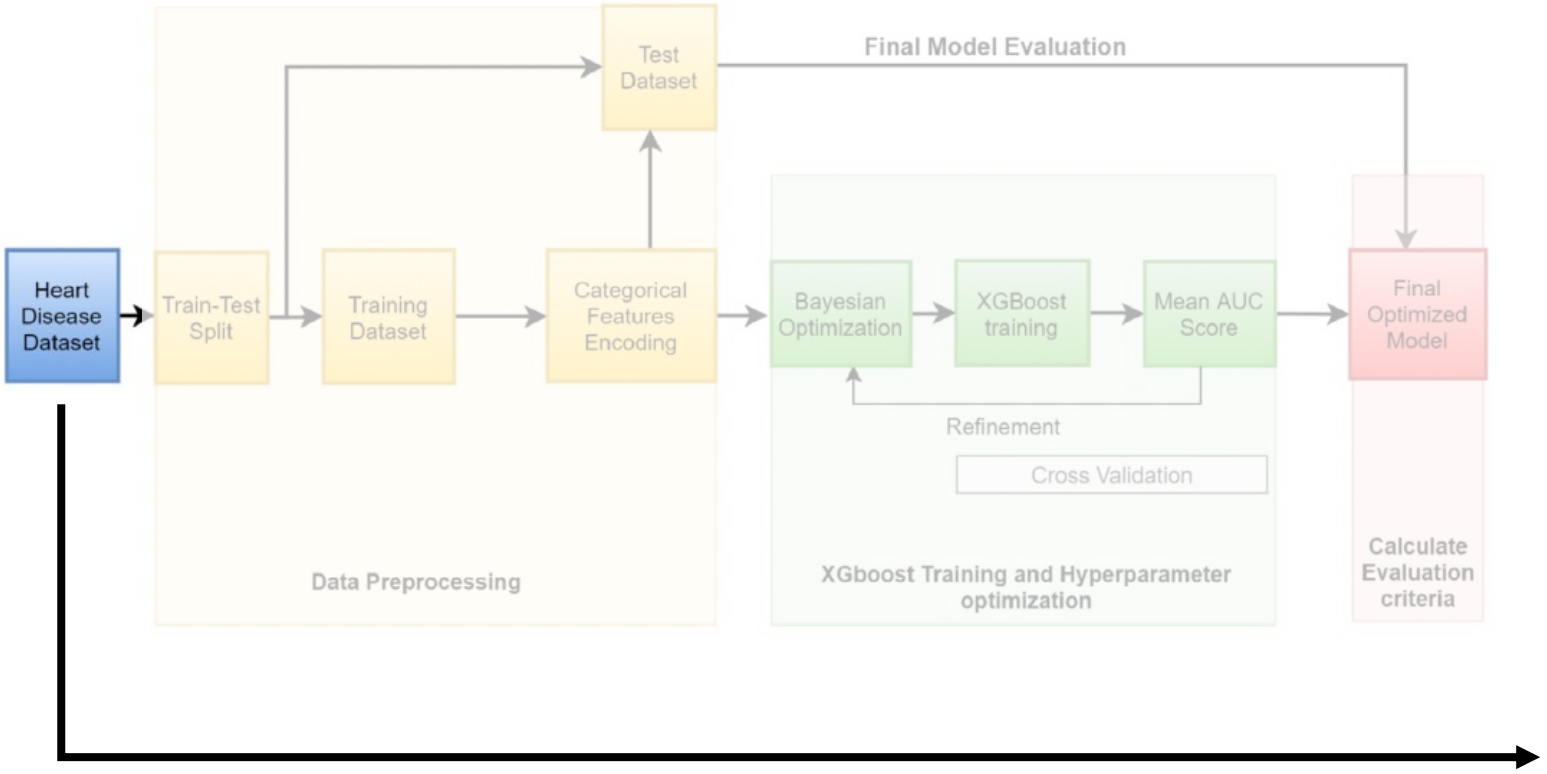
The Author's Framework



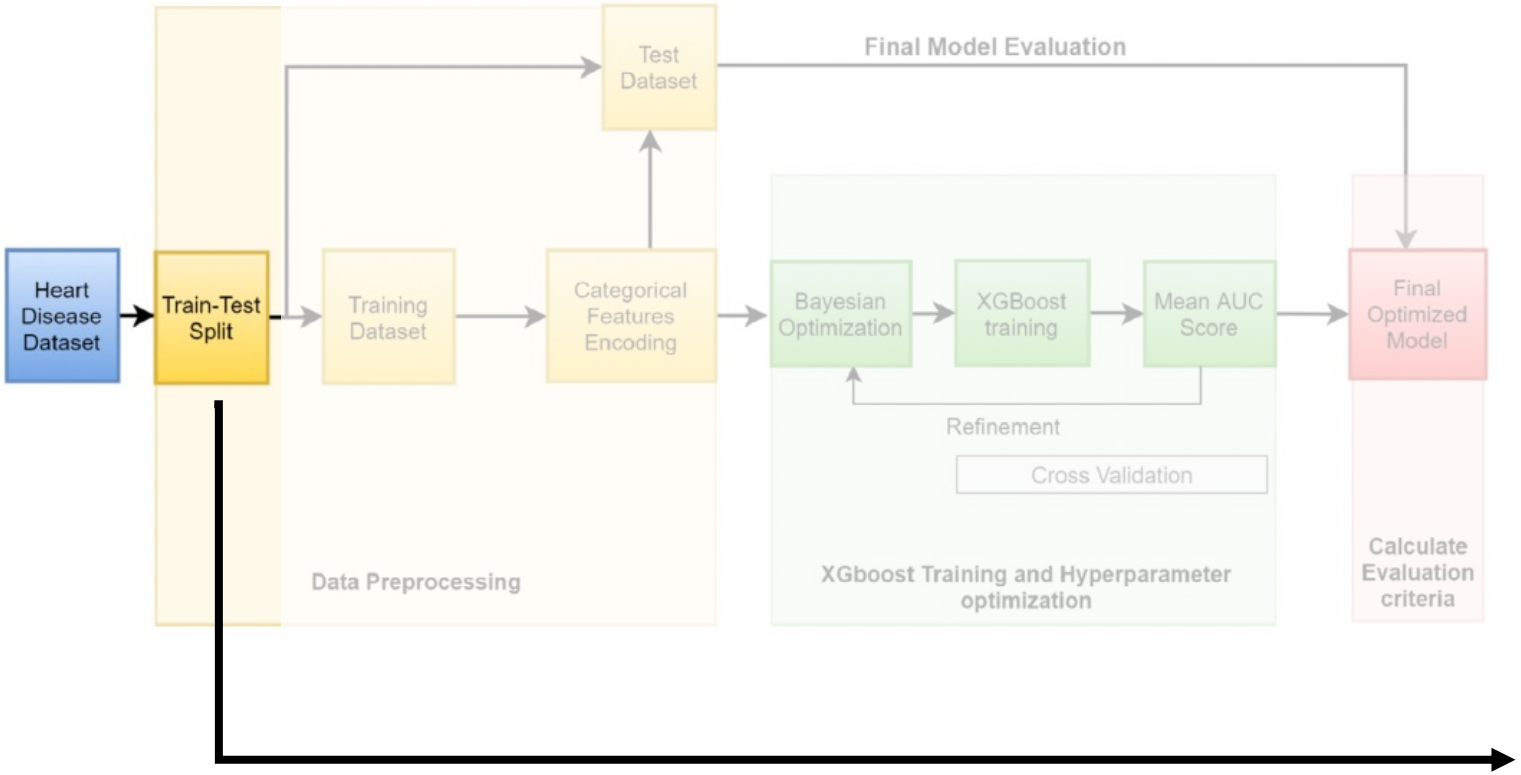
Step 1: Explore the Data



Before doing any modeling or model prep, it's ~~a good idea~~ essential to explore your data. You should:

- Take a look at each variable in the raw data to understand its composition and distribution.
- Explore the response variable to see if you need to transform it in any way + get a sense of any class imbalance
- Create some charts to visualize aspects of the data that you think might be interesting
- Review the data dictionary!

Step 2: Create a Train and Test Set

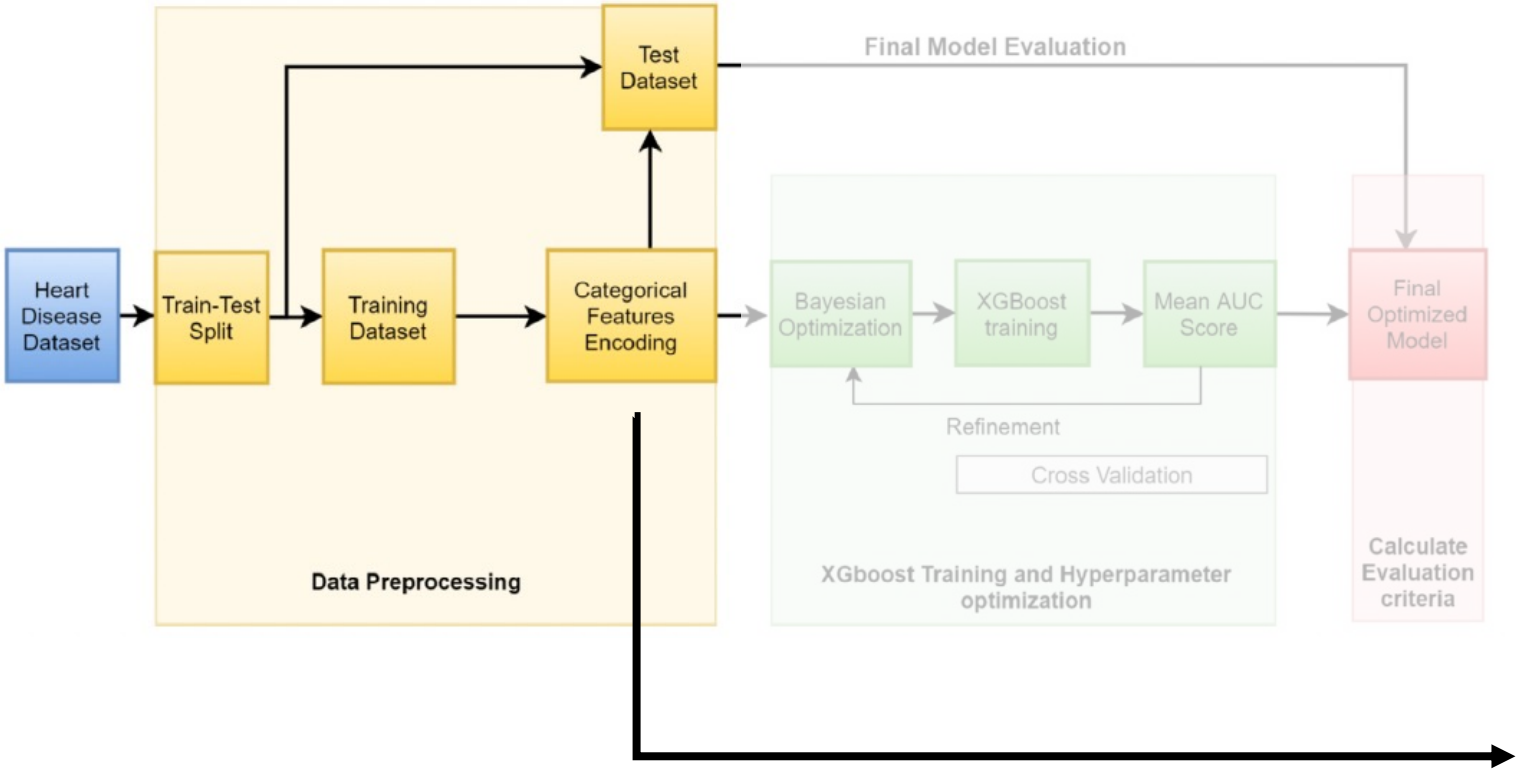


We need to split the data into a training set and a testing set so that we can build and evaluate our model. The authors use an 80/20 split.

One way to do this is to use the `test_train_split` function from `sklearn.model_selection`

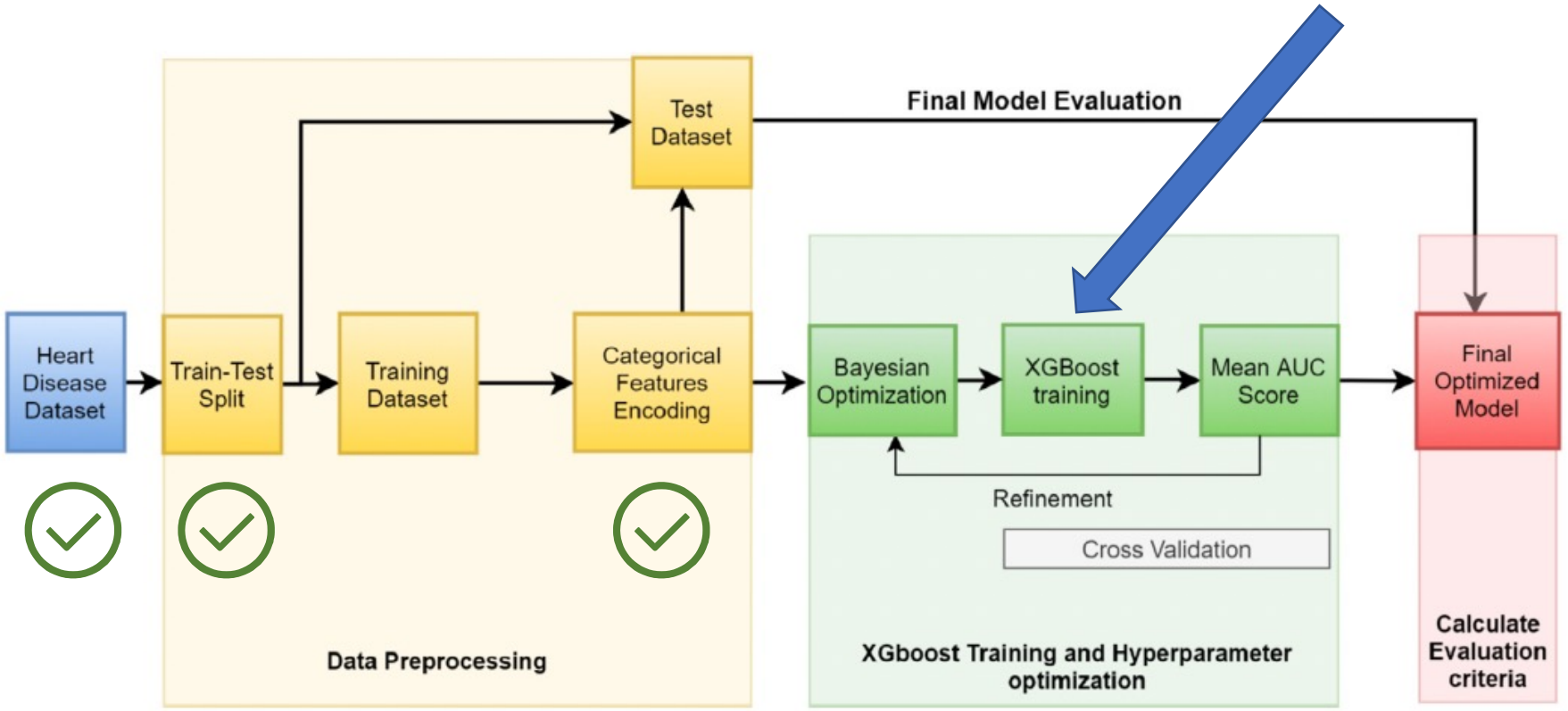
NOTE: The authors split their data before the encoding step, but that's not necessary. You could transform your features using the full dataset and then execute the test/train split.

Step 3: Encode the Categorical Features



We need to transform the categorical features in the dataset into numerical features, and one hot encoding is a standard approach for doing this (assuming that your categorical features don't have inherent order and don't have too many unique values).

What's Next?

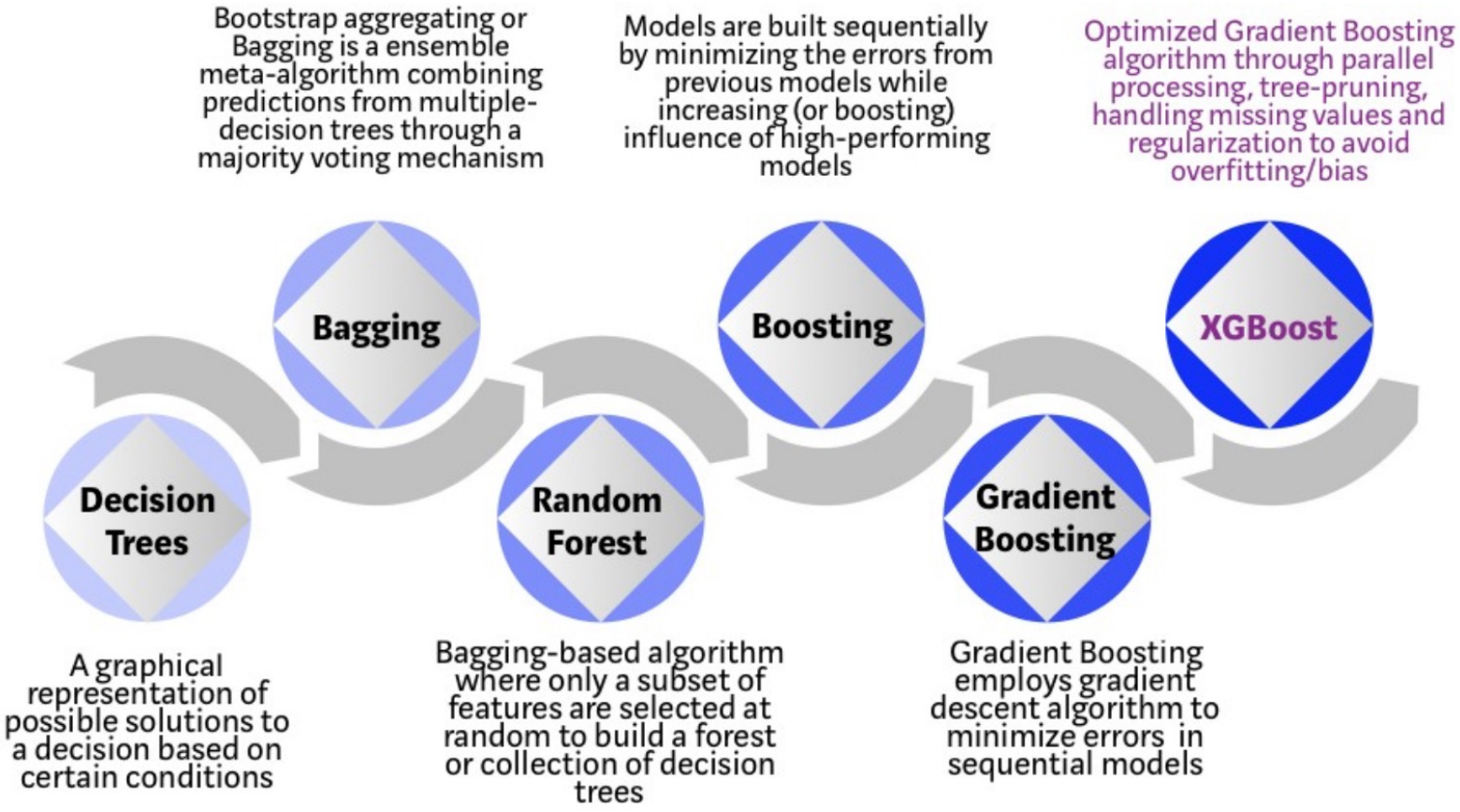

Heart Disease Dataset
Train-Test Split
Training Dataset
Categorical Features Encoding
Test Dataset
Final Model Evaluation
Data Preprocessing
Bayesian Optimization
XGBoost training
Mean AUC Score
Refinement
Cross Validation
XGboost Training and Hyperparameter optimization
Final Optimized Model
Calculate Evaluation criteria

Supervised Learning

Bias-Variance Tradeoff

Ensemble Methods

From Trees to XGBoost

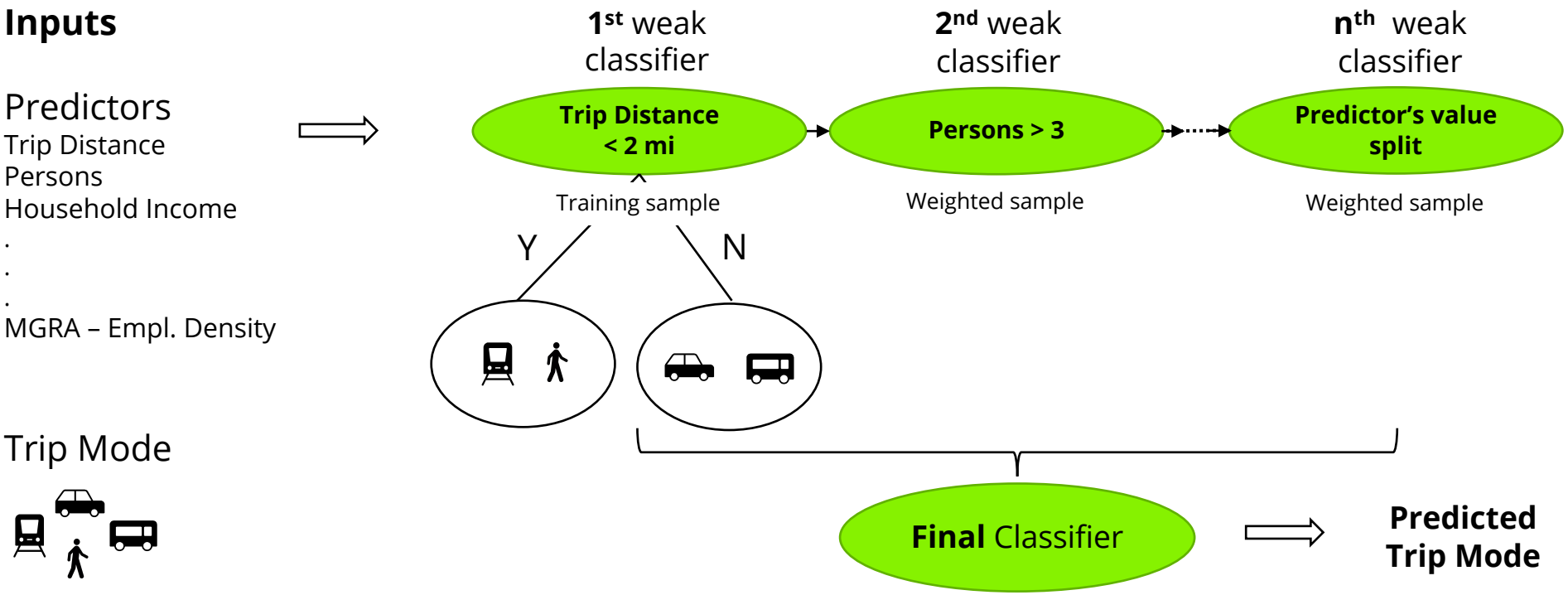


Evolution of XGBoost Algorithm from Decision Trees

Extreme Gradient Boosting Classification

Boosting algorithms fit many weak classifiers to reweighted versions of the training data. When using the boosting technique, all instances in the dataset are assigned a score based on classification difficulty. At each iteration, the algorithm pays more attention (e.g., assigns bigger weights) to instances that were wrongly classified previously. Boosting can reduce the variance by averaging many different trees. XGBoost is a more regularized version of Gradient Boosted Trees.

Boosting Overview



Model Advantages

A major benefit of using gradient boosting algorithms like XGBoost for mode choice modeling is that they readily provide estimates of feature importance from the trained model.

In addition, XGBoost offers the following advantages over similar algorithms:

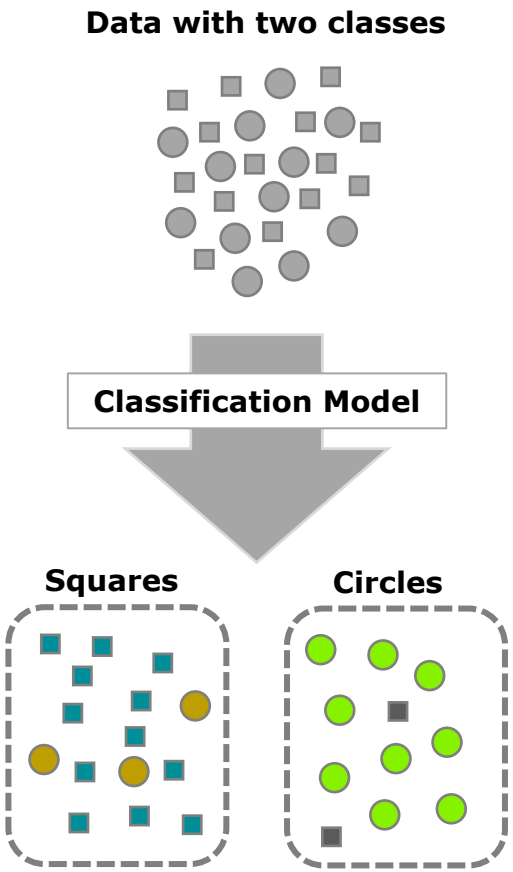
Fast: Originally written in C++, it is comparatively faster than other ensemble classifiers.

Parallelizable: The XGBoost algorithm is parallelizable and can harness the power of multi-core computers, GPUs, and networks of computers making it feasible to train on very large datasets.

Performant: XGBoost consistently outperforms other algorithm methods and has shown better performance on a variety of machine learning benchmark datasets.

Highly Tunable: XGBoost has internal parameters for cross-validation, regularization, user-defined objective functions, missing values, tree parameters, etc.

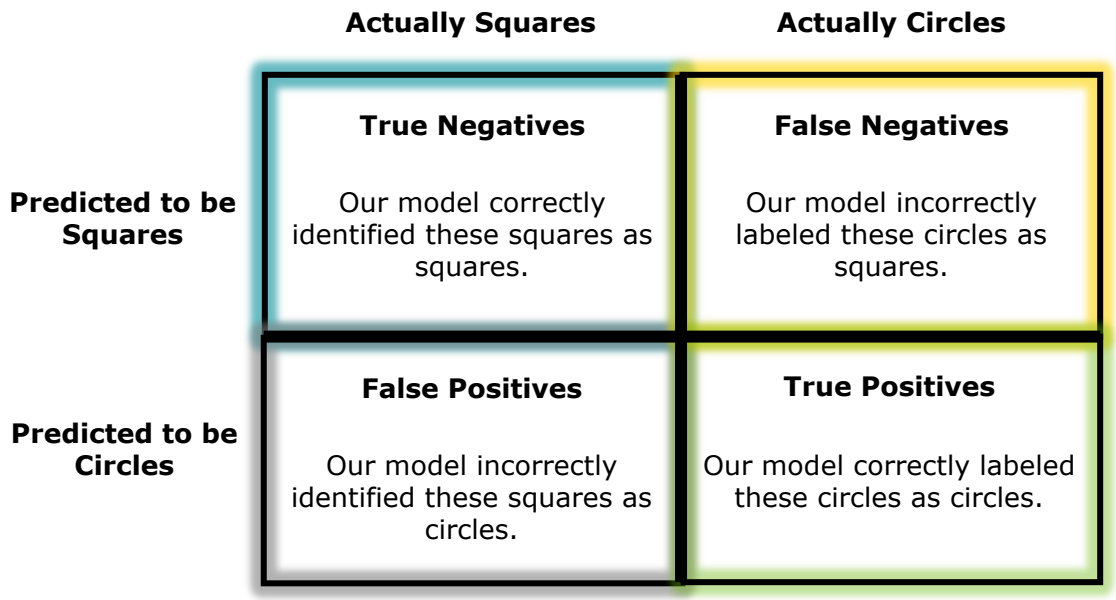
The Confusion Matrix



	Actually Squares	Actually Circles
Predicted to be Squares	**True Negatives** Our model correctly identified these squares as squares.	**False Negatives** Our model incorrectly labeled these circles as squares.
Predicted to be Circles	**False Positives** Our model incorrectly identified these squares as circles.	**True Positives** Our model correctly labeled these circles as circles.

Evaluation Metrics

Accuracy:	What percentage of all circles and squares were labeled correctly? Higher accuracy indicates that the model can more readily distinguish circles from squares.
Sensitivity (Recall):	What percentage of all circles were labeled as circles? Higher recall indicates that the model can more readily identify circles in the data.
Specificity:	What percentage of all squares were labeled as squares? Higher specificity indicates that the model can readily identify squares in the data.
Precision:	What percentage of all shapes predicted as circles were actually circles? Higher precision indicates that the model can more readily distinguish circles.



Evaluation Metrics

Accuracy: $\frac{\sum TP + \sum TN}{\sum TP + \sum TN + \sum FP + \sum FN}$ **0.68**

Sensitivity (Recall): $\frac{\sum TP}{\sum TP + \sum FN}$ **0.75**

Specificity: $\frac{\sum TN}{\sum TN + \sum FP}$ **0.60**

Precision: $\frac{\sum TP}{\sum TP + \sum FP}$ **0.65**

Consider a case where we had 20 squares and 20 circles in our dataset. We trained a classifier and here's the result:

	Actually Squares	Actually Circles
Predicted to be Squares	True Negatives: 12	False Negatives: 5
Predicted to be Circles	False Positives: 8	True Positives: 15

ROC Curve and AUC