## Updated Methodology

This study will be performed in 2 stages. Stage 1 is replication of the study by Doopala et al., and stage 2 is development of improved classifiers using the study datasets. A workflow has been produced for both techniques to be completed at each stage. The stages will be completed in Python.

Each dataset under consideration in the study has been saved in a separate Python notebook as follows:

Initial Codes – Cleveland

* Documents code relevant to the Cleveland Clinic Dataset obtained from the UCI Machine Learning Repository

Initial Codes – IEEE

* Documents code relevant to the Comprehensive Heart Disease Dataset obtained from the IEEE Datamart

Initial Codes – Mendeley

* Documents relevant to the Cardiovascular Disease Dataset obtained from the Mendeley Data Center

### Stage 1: Study Replication

1. Remove blanks and duplicates for each dataset
2. Perform minmax scaling to numerical features in dataset
3. Perform one-hot encoding to categorical nominal features with more than 2 levels
4. Perform 60:40 test, training split
5. Use training set to develop the following algorithms:
   1. Decision Tree
   2. Random Forest
   3. Naive Bayes
   4. Logistic regression
   5. SVM
   6. Gradient boosting
   7. XGBoost
   8. Ensemble algorithm consisting of
6. Naive Bayes
7. Random Forest
8. SVM
9. Gradient Boosting
10. Apply algorithms to test set
11. Evaluate the following performance metrics:
    1. Specificity
    2. Precision
    3. Recall
    4. F1-score
    5. Matthew’s Correlation Coefficient (MCC)

Hyperparameters will be tuned if necessary to achieve an accuracy as close as possible to the study authors.

### Stage 1: Workflow Replicated from Study Paper

Start

Data Preprocessing: Eliminate blanks and duplicates

Perform min-max scaling to dataset

Perform 60:40 test, training split

Use training set to develop classification algorithms:

Decision Tree

Random Forest

Naive Bayes

Logistic regression

SVM

Gradient boosting

XGBoost

Ensemble algorithm

Apply developed classification algorithms to test set

Calculate performance metrics: Accuracy, Precision, Recall, Specificity, F1 Score, MCC

### Stage 2: Improved Classification

1. Remove duplicates and null values
2. Identify outliers in numerical features using z-scores
3. Leave outliers in dataset and perform RobustScaling to entire dataset
4. Perform one-hot encoding to categorical nominal features with more than 2 levels
5. Perform 60:40 test, training split
6. Perform feature selection to training set
   1. First Round: Simple Filter methods
      1. Apply chi-squared tests for categorical features with 3 or more levels
      2. Apply odds ratio tests for categorical features with 2 levels
      3. Apply basic t-tests and shrinkage t-test for integer features
   2. Second Round: Wrapper methods
      1. Apply Genetic Algorithm
      2. Apply backward elimination using sklearn.feature\_selection.SequentialFeatureSelector
   3. Third Round: Embedded methods
      1. Use impurity-based feature importance and permutation-based feature importance to perform feature selection for Random Forest, Gradient Boosting and XGBoost classifiers
7. Use training set subjected to feature selection to develop and tune hyperparameters the following algorithms:
   1. Decision Tree
   2. Random Forest
   3. Naive Bayes
   4. Logistic regression
   5. SVM
   6. Gradient boosting
   7. XGBoost
   8. Ensemble algorithm consisting of Random Forest and Gradient Boosting
      1. Determine if Random Forest and Gradient Boosting are behaving in the same way on the dataset. Possible evaluators include Mann-Whitney on ranked categorical data
   9. kNN
   10. ANN
8. Apply algorithms to test set
9. Perform k-folds cross-validation (where k=10)
10. Evaluate the following performance metrics:
    1. Specificity
    2. Precision
    3. Recall
    4. F1-score
    5. Matthew’s Correlation Coefficient
    6. Computational Speed
    7. Memory Consumption
11. Evaluate classifier stability by analyzing the variance between folds
12. Evaluate classifier scalability
    1. Randomly introduce noise to the dataset using np.random and a specified noise ratio for example 10% of the data and observe effects on performance metrics
    2. Manually introduce bias to a single record and test effects on performance metrics