**MD2201 Data Science**

**Course Project**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **S.No** | **Div** | **Batch No** | **Group No** | **Roll No** | **Gr.No** | **Name of Student** |
| 1 | C | 3 | 6 | 56 | 12211422 | Abheerav Patankar |
| 2 | 58 | 12210336 | Aaditya Patil |
| 3 | 72 | 12211491 | Srujan Patwardhan |
| 4 | 77 | 12211501 | Swaraj Phand |

**Date of performance:**

# Project Title: Cardiovascular Disease Prediction using Machine Learning

# Data Set Name: NHANES (National Health and Nutrition Examination Survey.)

# Data Set Source: Kaggle

# Data set Link: [CardiacPrediction](https://github.com/Aaditatgithub/The-Framingham-Heart-Study/blob/master/CardiacPrediction.csv) (download the raw file)

1. **Data Set Description:**
2. **Description**

No. of rows = 37,079

No. of columns = 51

* Gender, Vigorous work, Moderate work, Health Insurance, Diabetes, CoronaryHeartDisease, etc. are categorical while others are numerical variables.
* There are **no missing values** in any of the columns.
* A **major class imbalance exists**, since the y-variable (CoronaryHeartDisease) has only 1508 1-values

and the remaining 35,571 are 0-values.

1. **Explanatary Variables in the Dataset**

$ SEQN *<int>* 2, 5, 12, 13, 14, 15, 16, 20, 24, 25, 29, …

$ Gender *<int>* 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 1, 1, 2, 2, …

$ Age *<int>* 77, 49, 37, 70, 81, 38, 85, 23, 53, 42, 62…

$ Annual.Family.Income *<int>* 8, 11, 11, 3, 5, 8, 1, 6, 6, 5, 3, 5, 5, 4…

$ Ratio.Family.Income.Poverty *<dbl>* 5.00, 5.00, 4.93, 1.07, 2.67, 4.52, 0.38, …

$ X60.sec.pulse *<int>* 68, 66, 64, 102, 72, 68, 66, 72, 82, 74, 1…

$ Systolic *<int>* 98, 122, 174, 130, 136, 109, 139, 103, 114…

$ Diastolic *<int>* 56, 83, 99, 66, 61, 69, 60, 60, 71, 85, 71…

$ Weight *<dbl>* 75.4, 92.5, 99.2, 63.6, 75.5, 81.6, 41.5, …

$ Height *<dbl>* 174.0, 178.3, 180.0, 157.7, 166.2, 174.9, …

$ Body.Mass.Index *<dbl>* 24.90, 29.10, 30.62, 25.57, 27.33, 26.68, …

$ White.Blood.Cells *<dbl>* 7.6, 5.9, 10.2, 11.6, 9.1, 7.6, 7.4, 5.6, …

$ Lymphocyte *<dbl>* 21.1, 37.8, 23.7, 13.1, 29.8, 20.8, 28.0, …

$ Monocyte *<dbl>* 7.1, 6.2, 9.0, 3.8, 5.6, 5.6, 6.3, 9.1, 7.…

$ Eosinophils *<dbl>* 4.4, 3.4, 3.2, 0.4, 1.7, 1.0, 0.6, 1.0, 1.…

$ Basophils *<dbl>* 0.5, 0.4, 0.6, 0.4, 0.4, 0.6, 0.5, 0.2, 0.…

$ Red.Blood.Cells *<dbl>* 4.73, 5.13, 5.76, 5.53, 5.32, 4.14, 3.57, …

$ Hemoglobin *<dbl>* 14.1, 14.5, 16.0, 16.8, 16.6, 13.3, 10.9, …

$ Mean.Cell.Vol *<dbl>* 88.5, 84.9, 83.5, 91.1, 90.4, 97.1, 93.0, …

$ Mean.Cell.Hgb.Conc. *<dbl>* 29.7, 28.3, 27.8, 30.3, 31.3, 32.1, 30.4, …

$ Mean.cell.Hemoglobin *<dbl>* 33.6, 33.3, 33.3, 33.3, 34.5, 33.1, 32.7, …

$ Platelet.count *<dbl>* 214, 209, 357, 228, 160, 255, 219, 220, 33…

$ Mean.Platelet.Vol *<dbl>* 7.7, 10.4, 7.9, 8.8, 9.0, 7.7, 8.0, 8.0, 8…

$ Segmented.Neutrophils *<dbl>* 66.8, 52.2, 63.7, 82.4, 62.5, 72.0, 64.6, …

$ Hematocrit *<dbl>* 41.8, 43.6, 48.1, 50.4, 48.1, 40.2, 33.3, …

$ Red.Cell.Distribution.Width *<dbl>* 13.7, 13.1, 13.6, 14.4, 12.4, 11.9, 14.1, …

$ Albumin *<int>* 45, 45, 47, 40, 45, 44, 41, 44, 47, 43, 46…

$ ALP *<int>* 62, 63, 63, 103, 110, 31, 42, 50, 114, 53,…

$ AST *<int>* 19, 22, 17, 24, 23, 28, 24, 19, 18, 22, 17…

$ ALT *<int>* 16, 28, 35, 35, 18, 30, 16, 18, 20, 24, 26…

$ Cholesterol *<dbl>* 5.25, 7.16, 3.90, 7.94, 4.42, 4.91, 4.22, …

$ Creatinine *<dbl>* 61.9, 70.7, 88.4, 61.9, 88.4, 53.0, 79.6, …

$ Glucose *<dbl>* 4.330, 5.273, 4.163, 7.882, 6.384, 5.162, …

$ GGT *<int>* 20, 34, 32, 24, 24, 32, 17, 10, 57, 25, 18…

$ Iron *<dbl>* 11.28, 24.54, 11.28, 12.18, 11.82, 16.66, …

$ LDH *<int>* 140, 133, 131, 181, 150, 121, 193, 119, 13…

$ Phosphorus *<dbl>* 1.066, 1.033, 1.130, 0.904, 1.033, 1.195, …

$ Bilirubin *<dbl>* 12.0, 8.6, 6.8, 8.6, 10.3, 8.6, 6.8, 8.6, …

$ Protein *<dbl>* 72, 73, 72, 66, 79, 72, 72, 75, 76, 74, 78…

$ Uric.Acid *<dbl>* 362.8, 404.5, 339.0, 410.4, 368.8, 226.0, …

$ Triglycerides *<dbl>* 1.298, 3.850, 1.581, 3.635, 0.756, 0.756, …

$ Total.Cholesterol *<dbl>* 5.56, 7.21, 4.03, 8.12, 4.50, 5.15, 4.24, …

$ HDL *<dbl>* 1.39, 1.08, 0.98, 1.28, 1.04, 1.49, 1.41, …

$ Glycohemoglobin *<dbl>* 4.7, 5.5, 5.2, 7.6, 5.8, 4.6, 4.6, 4.7, 5.…

$ Vigorous.work *<int>* 3, 1, 2, 3, 1, 1, 1, 2, 2, 2, 2, 2, 2, 1, …

$ Moderate.work *<int>* 3, 1, 1, 3, 1, 1, 2, 1, 2, 1, 2, 2, 1, 2, …

$ Health.Insurance *<int>* 1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 1, 1, 2, 1, …

$ Diabetes *<int>* 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, …

$ Blood.Rel.Diabetes *<int>* 2, 2, 1, 1, 2, 2, 2, 1, 1, 2, 1, 2, 2, 1, …

$ Blood.Rel.Stroke *<int>* 2, 2, 1, 2, 2, 2, 2, 2, 2, 1, 1, 2, 2, 2, …

$ CoronaryHeartDisease *<int>* 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

1. **Correlation Between Variables:**

We identified the highly corelated features using the correlation matrix for performing feature selection.

A diagram of red text

Description automatically generated with medium confidence

# Description of Work Done:

* **Classification Algorithms Used:**
  + XG-Boost (Extreme Gradient Boosting)
  + SVM (Support Vector Machine)
  + LR (Logistic Regression)
  + Random Forest
* **Handling Class Imbalance:**
  + Utilized oversampling using the RACOG algorithm initially with a total of 16K rows to address the imbalance in the dataset.
  + Experimented with under-sampling by including 1.5K rows of each class, but this resulted in very low accuracy, prompting a switch back to the RACOG algorithm.
* **Feature Selection Methods:**
  + Employed the Mean Decrease Gini Impurity method to select relevant features and mitigate the risk of overfitting.
  + Also utilized LASSO regression for feature selection to further address potential overfitting issues in the models.

# Literature Survey:

# Data Preprocessing:

* 1. **Data Splitting**: The dataset was split into training and testing sets with a 75:25 ratio*.*
  2. **10-fold Cross-Validation:** Employed to ensure robust model evaluation, the dataset was systematically divided into ten subsets. The model was trained on nine subsets and validated on the remaining one iteratively, reducing overfitting risks and improving performance estimation.

# Handling Imbalanced Data:

# Initial Separation: The dataset was segregated into two dataframes based on the target variable ('CoronaryHeartDisease').

# Oversampling Process: Random sampling of 8000 rows from the majority class dataframe ('B') created a new dataframe ('C'). Merging 'A' and 'C' balanced class distribution.

# Imbalance Ratio Calculation: The ratio of the target variable in the final dataset was computed.

# RACOG Oversampling Algorithm:

# Minority Distribution Approximation: Employing the Gibbs Sampler, the algorithm processes a discretized and numeric dataset.

# Iterative Sampling: Through a Markov chain, new samples are generated iteratively, discarding initial burn-in iterations.

# Minority Example Generation: Validation of examples as new minority examples occurs at regular intervals.

* 1. **Result Storage**: The resulting oversampled dataframe was stored in a CSV file for further analysis or utilization.

# Feature Selection:

# We employed two feature selection methods to address the suspicion of potential overfitting and enhance model accuracy:

# Mean Decrease Gini Impurity

# A graph of a number of people Description automatically generatedThis algorithm assesses feature importance in decision tree models by measuring how much each feature reduces the Gini impurity. It helps prioritize features based on their contribution to classification accuracy.

# LASSO Regression

# By penalizing the absolute size of regression coefficients, LASSO regression shrinks less important features towards zero, identifying and prioritizing the most relevant predictors in the model.

# Using feature selection methods, we observed a slight increase in the accuracy of various algorithms.

1. **Algorithms Implemented:**
2. **Logistic Regression**: Logistic regression is a binary classification algorithm that estimates probabilities using a logistic function. It's widely used for binary classification problems. In our case, we can extend it to handle multiclass classification by using techniques like onevsrest.
3. **Support Vector Machine (SVM):** SVM is a powerful supervised machine learning algorithm used for classification tasks. It finds the hyperplane that best separates different classes in the feature space. SVMs are effective in high dimensional spaces and can handle nonlinear decision boundaries using kernel functions.
4. **Random Forest**: Random forest is an ensemble learning method that uses multiple decision trees to improve classification accuracy. It builds each tree independently and combines their predictions through averaging or voting. Random forests are robust against overfitting and perform well in various applications.
5. **XGBoost (Extreme Gradient Boosting)** is a powerful ensemble learning method renowned for its speed, scalability, and superior performance in various machine learning tasks. It operates by sequentially constructing a series of decision trees, each aiming to rectify the errors of its predecessors. Unlike traditional gradient boosting methods, XGBoost incorporates regularization techniques such as shrinkage and pruning to mitigate overfitting, ensuring robustness and generalization capability
6. **Bagged-CART (Bootstrap Aggregating - Classification and Regression Trees):** Bagged-CART is an ensemble learning method that builds multiple decision trees on bootstrapped samples of the dataset and combines their predictions through averaging or voting to improve classification or regression accuracy.
7. **GBM (Gradient Boosting Machine)**: GBM is a boosting algorithm that sequentially builds a series of decision trees, with each tree focusing on correcting the errors made by the previous ones, aiming to minimize the overall prediction error.
8. **C5.0:** C5.0 is a decision tree algorithm that builds a classification tree by recursively partitioning the feature space based on the most significant attribute at each node, using entropy or Gini impurity as the splitting criterion, to maximize information gain and minimize misclassification.

# Code*:*

**# Install and load necessary packages**

install.packages(c('caret', 'xgboost', 'C50', 'gbm'))

library(caret)

library(xgboost)

library(C50)

**# Read the dataset**

df <- read.csv('Final\_Dataset.csv')

**# Split the data into training and testing sets**

set.seed(123)

train\_index <- createDataPartition(df$CoronaryHeartDisease, times = 1, p = 0.75, list = FALSE)

train\_data <- df[train\_index, ]

test\_data <- df[-train\_index, ]

**# Preprocess the data**

train\_data$CoronaryHeartDisease <- as.factor(train\_data$CoronaryHeartDisease)

test\_data$CoronaryHeartDisease <- as.factor(test\_data$CoronaryHeartDisease)

**# Set up the train control**

trainControl <- trainControl(method = "repeatedcv", number = 10, repeats = 3, verboseIter = TRUE)

**# Train the model using the random forest algorithm**

set.seed(123)

model <- train(CoronaryHeartDisease ~ ., data = train\_data, method = "rf", trControl =

trainControl, metric = "Accuracy")

**# Extract variable importance using Mean Decrease in Gini (MDG)**

importance <- varImp(model, scale = FALSE)

**# Select top 20 features based on importance scores**

top\_features <- rownames(importance$importance[1:20, , drop = FALSE])

**# Correlations for multicollinearity concerns ----**

# # Subset the training data to include only the top features

# train\_data\_subset <- train\_data[, c("CoronaryHeartDisease", top\_features)]

#

# # Calculate the correlation matrix for the selected features

# correlation\_matrix <- cor(train\_data\_subset[, -1]) # Exclude the target variable

#

# # Melt the correlation matrix for plotting

# melted\_correlation <- melt(correlation\_matrix)

#

# # Plot the heatmap

# ggplot(data = melted\_correlation, aes(Var2, Var1, fill = value)) +

# geom\_tile(color = "white") +

# scale\_fill\_gradient2(low = "blue", high = "red", mid = "white",

# midpoint = 0, limit = c(-1,1), space = "Lab",

# name="Correlation") +

# theme\_minimal() +

# theme(axis.text.x = element\_text(angle = 45, vjust = 1, size = 10, hjust = 1)) +

# coord\_fixed()

**# Features to remove**

features\_to\_remove <- c("Segmented.Neutrophils", "Total.Cholesterol", "Glucose")

# Remove specified features from top\_features vector

top\_features <- top\_features[!top\_features %in% features\_to\_remove]

**# Select top 10 features**

top\_10\_features <- top\_features[1:10]

**# Prepare the data using the selected features**

train\_data\_subset <- train\_data[, c("CoronaryHeartDisease", top\_10\_features)]

**# Train models**

**# XGBoost**

set.seed(123)

xgb\_model <- train(

CoronaryHeartDisease ~ .,

data = train\_data\_subset,

method = "xgbTree",

trControl = trainControl,

metric = "Accuracy"

)

**# Bagged CART**

set.seed(123)

bagged\_cart\_model <- train(

CoronaryHeartDisease ~ .,

data = train\_data\_subset,

method = "treebag",

trControl = trainControl,

metric = "Accuracy"

)

**# C5**

set.seed(123)

c5\_model <- train(

CoronaryHeartDisease ~ .,

data = train\_data\_subset,

method = "C5.0",

trControl = trainControl,

metric = "Accuracy"

)

**# GBM**

set.seed(123)

gbm\_model <- train(

CoronaryHeartDisease ~ .,

data = train\_data\_subset,

method = "gbm",

trControl = trainControl,

metric = "Accuracy"

)

**# Train the model using the Random Forest algorithm**

set.seed(123)

rf\_model <- train(

CoronaryHeartDisease ~ .,

data = train\_data\_subset,

method = "rf",

trControl = trainControl,

metric = "Accuracy"

)

**# Train the model using Support Vector Machines (SVM**)

set.seed(123)

svm\_model <- train(

CoronaryHeartDisease ~ .,

data = train\_data\_subset,

method = "svmRadial",

trControl = trainControl,

metric = "Accuracy"

)

**# Train the model using Generalized Linear Model (GLM)**

set.seed(123)

glm\_model <- train(

CoronaryHeartDisease ~ .,

data = train\_data\_subset,

method = "glm",

trControl = trainControl,

metric = "Accuracy"

)

**# Predictions and evaluation**

models <- list(xgb\_model, bagged\_cart\_model, c5\_model, gbm\_model, rf\_model, svm\_model, glm\_model)

model\_names <- c("XGBoost", "Bagged CART", "C5", "GBM", "Random Forest", "SVM", "GLM")

for (i in seq\_along(models)) {

predictions <- predict(models[[i]], newdata = test\_data)

conf\_matrix <- confusionMatrix(predictions, test\_data$CoronaryHeartDisease)

cat(model\_names[i], "Model Performance on Test Data:\n")

cat("Accuracy:", conf\_matrix$overall["Accuracy"], "\n")

cat("Precision:", conf\_matrix$byClass["Precision"], "\n")

cat("Recall:", conf\_matrix$byClass["Recall"], "\n")

cat("F1 Score:", conf\_matrix$byClass["F1"], "\n\n")

}

**# Gather accuracy values**

accuracies <- sapply(models, function(model) {

conf\_matrix <- confusionMatrix(predict(model, newdata = test\_data), test\_data$CoronaryHeartDisease)

conf\_matrix$overall["Accuracy"]

})

**# Plot bar graph with better color theme and scale**

bar\_plot <- ggplot(data.frame(Model = model\_names, Accuracy = accuracies), aes(x = Model, y = Accuracy, fill = Model)) +

geom\_bar(stat = "identity") +

coord\_cartesian(ylim = c(0.7, 0.95)) + # Adjust y-axis limits

labs(title = "Model Accuracies", y = "Accuracy", x = "Model") +

theme\_minimal() +

theme(axis.text.x = element\_text(angle = 45, hjust = 1)) +

scale\_fill\_viridis\_d() + # Use viridis color palette

scale\_y\_continuous(expand = c(0, 0)) # Remove padding

**# Show the plot**

print(bar\_plot)

**# Line graph for Comparative accuracies of models**

# Vectors with your data

algorithms <- c("Treebag", "GBM", "C5.0", "XGBoost", "RF", "SVM", "GLM")

accuracy\_all\_features <- c(0.8484, 0.8481, 0.8582, 0.8739, 0.8449, 0.8297, 0.82)

accuracy\_10\_features <- c(0.8211, 0.8235, 0.8220, 0.8371, 0.8318, 0.8270, 0.7950)

# Create a data frame for plotting

data <- data.frame(

Algorithm = rep(algorithms, 2),

Accuracy = c(accuracy\_all\_features, accuracy\_10\_features),

Features = factor(rep(c("All Features", "10 Features"), each = length(algorithms)))

)

**# Plot**

ggplot(data, aes(x = Algorithm, y = Accuracy, color = Features, group = Features)) +

geom\_line() +

geom\_point() +

scale\_color\_manual(values = c("All Features" = "blue", "10 Features" = "red")) +

labs(title = "Algorithm Accuracy Comparison",

subtitle = "Comparison of Accuracy with All vs. 10 Selected Features",

x = "Algorithm",

y = "Accuracy (%)",

color = "Feature Set") +

theme\_minimal() +

theme(axis.text.x = element\_text(angle = 45, hjust = 1))

# Shiny app:

# 1. UI Definition:

# The UI is defined using the `fluidPage` function from the Shiny package.

# It includes a title panel, sidebar layout with input parameters, and a main panel for future enhancements.

# 2. Server Logic:

# The server logic loads an XGBoost model from an RDS file.

# A screenshot of a computer Description automatically generatedIt defines an observer that triggers prediction when the "Predict" button is clicked, using input data provided by the user. The prediction result is then displayed below the button.

# Evaluation Parameters:

# We have utilized several evaluation parameters to assess the performance and effectiveness of our model. These evaluation parameters help us gauge how well our model is performing and provide insights into its strengths and weaknesses. Here's an explanation of the evaluation parameters used:

# Accuracy: Accuracy measures the proportion of correctly classified instances out of the total instances. It provides an overall assessment of how well the model performs across all classes

# Precision: Precision measures the proportion of true positive predictions out of all positive predictions. It indicates the model's ability to correctly identify relevant instances from all instances predicted as positive.

# Recall (Sensitivity): Recall measures the proportion of true positive predictions out of all actual positive instances. It assesses the model's ability to identify all relevant instances.

# Specificity: Specificity measures the proportion of true negative predictions out of all actual negative instances. It indicates the model's ability to correctly identify negative instances.

# F1 Score: The F1 score is the harmonic mean of precision and recall. It provides a balance between precision and recall, especially in scenarios where class imbalance exists.

# Experimentation Done:

# We started by application of certain classification algorithms namely BaggedCART, GBM, C5.0, SVM, Linear Regression on the dataset with ~37K rows divided into 75:25 ratio of training and testing.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Bagged-CART | GBM | C5.0 | SVM | GLM |
| Accuracy | 0.9599 | 0.9600 | 0.9606 | 0.9606 | 0.9606 |
| Precision | 0.9613 | 0.9608 | 0.9606 | 0.9606 | 0.9628 |
| Recall | 0.9984 | 0.9991 | 1 | 1 | 0.9975 |
| F1 Score | 0.9795 | 0.9796 | 0.9799 | 0.9799 | 0.9799 |

# Due to the similarity in the accuracy that we got during the testing of all 5 algorithms, we presumed that the model was overfitting. We needed to reduce the number of variables first. Thus results obtained after Mean Decrease Gini Impurity are as follows:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Bagged-CART | GBM | C5.0 | SVM | GLM |
| Accuracy | 0.9581 | 0.9606 | 0.9606 | 0.9606 | 0.9603 |
| Precision | 0.9617 | 0.9606 | 0.9606 | 0.9606 | 0.9610 |
| Recall | 0.9960 | 1 | 1 | 1 | 0.9992 |

# After encountering the fact that the dataset was imbalanced. We trained the models with only ~1500 rows of patients prone to cardiovascular disease and the remaining out of ~37K not prone to the same. We performed undersampling of class 0 (Not prone to CVD) equal to that of class 1(prone to CVD) and used the following algorithms.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Logistic Regression | XG-Boost | SVM | RF | KNN |
| Accuracy | 0.7719 | 0.7838 | 0.7626 | 0.7732 | 0.5464 |
| Precision | 0.7971 | 0.8110 | 0.8018 | 0.8239 | 0.5452 |
| Recall | 0.7294 | 0.7401 | 0.6976 | 0.6950 | 0.5597 |

# After application of Mean Decrease Gini Impurity over the 3K Rows with 13 explanatory variables. (After balancing the dataset we chose the algorithms with better accuracy which are as follows):

|  |  |  |  |
| --- | --- | --- | --- |
|  | Logistic Regression | SVM | RF |
| Accuracy | 0.7719 | 0.7626 | 0.7759 |
| Precision | 0.8006 | 0.8133 | 0.8023 |
| Recall | 0.7241 | 0.6817 | 0.7321 |

# Code for experimentation: [*Link to the Github Repository of the code.*](https://github.com/Aaditatgithub/The-Framingham-Heart-Study)

# Results and Discussion:

# Since there was a sudden decrease in the performance of the models due to a reduced dataset. We opted to oversample the data using the RACOG algorithm with a total of 13.5K rows and a balanced dataset.

# By considering all the explanatory features, the following results were obtained:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Accuracy | Precision | Recall | F1 Score |
| Bagged-Cart | 0.8484 | 0.8812 | 0.8050 | 0. 8414 |
| GBM | 0.8481 | 0.8715 | 0.8162 | 0.8430 |
| C5 | 0.8582 | 0.9037 | 0.8014 | 0. 8495 |
| XG-Boost | 0.8739 | 0.8890 | 0.8542 | 0.8712 |
| RF | 0.8449 | 0.8926 | 0.7867 | 0. 8363 |
| SVM | 0.8297 | 0.8610 | 0.7860 | 0.8156 |
| GLM | 0.82 | 0.8349 | 0.7973 | 0. 8156 |

# A relative comparison of their accuracies is given as follows:

# A graph of a graph showing a bar chart Description automatically generated with medium confidence

# Upon usage of mean decrease gini impurity, we found out the top 20 features that contributed to predicting the target variable in the dataset. For ease of usage with the shiny app, we trained the models on the top 10 features.

# A diagram of blood cells Description automatically generatedTo further improve the model performance, we analysed the correlations between the top 20 variables:

# Upon analysis, Glucose-GlycoHaemoglobin, Lymphocyte-Segmented Neutrophils, and Cholesterol-Total Cholesterol had high correlation coefficients. Thus we removed the predictor variables with lower Gini-importance scores. This resulted in increased model accuracy since multicollinearity concerns were eliminated.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Accuracy | Precision | Recall | F1 Score |
| Bagged-Cart | 0.8211 | 0.8460 | 0.7848 | 0.8142 |
| GBM | 0.8235 | 0.8503 | 0.7848 | 0.8162 |
| C5 | 0.8220 | 0.8563 | 0.7735 | 0.8128 |
| XG-Boost | 0.8371 | 0.8679 | 0.7949 | 0.8298 |
| RF | 0.8318 | 0.8727 | 0.7765 | 0.8218 |
| SVM | 0.8270 | 0. 8738 | 0. 7640 | 0. 8153 |
| GLM | 0. 7950 | 0. 8072 | 0. 7747 | 0. 7906 |

# A graph of different colored squares Description automatically generatedAccuracies of the models for the above conditions have been plotted below:

# Comparing the model performance trained on all features and the model trained on 10 selected features (removing multicollinearity concerns) is as follows:

# Comparison of model performances on all explanatory variables and 10 feature selected variables.

# Conclusions:

# In conclusion, the project "Cardiovascular Disease Prediction using Machine Learning" utilized various classification algorithms and techniques to predict the risk of cardiovascular disease (CVD) based on the NHANES dataset obtained from Kaggle. Initially, classification algorithms including Bagged-CART, GBM, C5.0, SVM, and Logistic Regression were applied, revealing high accuracy but potential overfitting issues. To address this, feature selection methods such as Mean Decrease Gini Impurity and LASSO regression were employed, followed by handling class imbalance through oversampling using the RACOG algorithm. The evaluation of different models indicated that oversampling significantly improved performance, particularly demonstrated by Random Forest. However, further experiments were conducted to assess the impact of reduced datasets and feature selection. Despite reduced performance with fewer features and data points, algorithms like Logistic Regression, SVM, and Random Forest still showed promising accuracy levels.

# Overall, the project showcased the importance of rigorous evaluation, feature selection, and handling class imbalance in building effective predictive models for cardiovascular disease. Future work may involve refining feature selection techniques, exploring other oversampling methods, and possibly incorporating additional data sources to further enhance the predictive performance and generalizability of the models.

# References: