Northeastern University

Final Project

Heart Disease

Course: ALY 6040 – Data Mining

Professor: Shahram Sattar

Submitted by:

Bhanu Prathyusha Uppalapati

Lakshmi Sravya Vedantham

Pooja Savjibhai Viroja

Jay Haresh Vora

Jie Zhang

Date: May 20, 2022

Table of Contents

Objective

As per the 2020 annual CDC survey data, heart disease is one of the leading causes of death in Americans compared to the other races. So, from the dataset, our main objective will be to detect and predict the factors which have the highest impact to get heart disease. For example, will a person have heart disease, if he/she smoked at least 100 cigarettes in their lives? Or do they have any serious difficulty walking or climbing the stairs?

In this report, we will analyse dataset of heart disease and study whether there exist relationships among the dependent variable, HeartDisease, and other 17 independent variables of health conditions, and which variables have a significant effect on the likelihood of heart disease to predict if one person has heart disease or not with his or her health condition.

We will briefly focus on below areas to find out which variables have a significant effect on the likelihood of heart disease.

- Dataset introduction
- Exploratory data analysis
- Model selection and exploration
- Analysis and Interpretations
- Recommendations and Conclusions

Dataset Introduction

The data comes from the Centers for Disease Control and Prevention (CDC), which performs a phone survey on the health of US (United States) inhabitants. The dataset appeared to be the most current, having been conducted in 2022 and containing data from 2020. It has 319795 observations and 18 variables.

Variable Description

Field Name	Description	Type
HeartDisease	Respondents that have ever reported having coronary heart disease (CHD) or myocardial infarction (MI)	Categorical
BMI	Body Mass Index (BMI)	Decimal
Smoking	Have you smoked at least 100 cigarettes in your entire life? [Note: 5 packs = 100 cigarettes]	Categorical
AlcoholDrinking	Heavy drinkers (adult men having more than 14 drinks per week and adult women having more than 7 drinks per week	Categorical
Stroke	Do you ever have a stroke?	Categorical

PhysicalHealth	It includes physical illness and injury, for how many days during the past 30 days was your physical health not good?	Continuous
MentalHealth	How many days during the past 30 days was your mental health not good?	Continuous
DiffWalking	Do you have serious difficulty walking or climbing stairs?	Categorical
Sex	Are you male or female?	Categorical
AgeCategory	Fourteen-level age category	Character
Race	Imputed race/ethnicity value	Character
Diabetic	Do you ever have a diabetes?	Character
PhysicalActivity	Adults who reported doing physical activity or exercise during the past 30 days other than their regular job	Categorical
GenHealth	Describe your general health	Character
SleepTime	On average, how many hours of sleep do you get in a 24-hour period?	Continuous
Asthma	Do you ever have an asthma?	Categorical
KidneyDisease	Not including kidney stones, bladder infection or incontinence, were you ever told you had kidney disease?	Categorical
SkinCancer	Do you ever have a skin cancer?	Categorical

Table 1: Variable description details of dataset

In the given dataset, first we will look at the structure details and summary details and clean the data before moving forward. From the below figures, we can see that among 18 variables, 4 variables are characters, 1 is numerical and rest are integer.

Structural details

```
'data.frame':   319795 obs. of  18 variables:
 $ HeartDisease    : int  0 0 0 0 0 1 0 0 0 0 ...
 $ BMI             : num  16.6 20.3 26.6 24.2 23.7 ...
 $ Smoking         : int  1 0 1 0 0 1 0 1 0 0 ...
 $ AlcoholDrinking : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Stroke          : int  0 1 0 0 0 0 0 0 0 0 ...
 $ PhysicalHealth  : int  3 0 20 0 28 6 15 5 0 0 ...
 $ MentalHealth    : int  30 0 30 0 0 0 0 0 0 0 ...
 $ DiffWalking     : int  0 0 0 0 1 1 0 1 0 1 ...
 $ Sex             : chr  "Female" "Female" "Male" "Female" ...
 $ AgeCategory     : chr  "55-59" "80 or older" "65-69" "75-79" ...
 $ Race            : chr  "White" "White" "White" "White" ...
 $ Diabetic        : int  1 0 1 0 0 0 0 1 0 0 ...
 $ PhysicalActivity: int  1 1 1 0 1 0 1 0 0 1 ...
 $ GenHealth       : chr  "Very good" "Very good" "Fair" "Good" ...
 $ SleepTime       : int  5 7 8 6 8 12 4 9 5 10 ...
 $ Asthma          : int  1 0 1 0 0 0 1 1 0 0 ...
 $ KidneyDisease   : int  0 0 0 0 0 0 0 0 1 0 ...
 $ SkinCancer      : int  1 0 0 1 0 0 1 0 0 0 ...
```

Figure 1: Structural details of the datasets after cleansing

Summary details

```
 HeartDisease         BMI           Smoking       AlcoholDrinking      Stroke        PhysicalHealth
Min.   :0.0000   Min.   :12.02   Min.   :0.0000   Min.   :0.0000   Min.   :0.00000   Min.   : 0.000
1st Qu.:0.0000   1st Qu.:24.03   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.: 0.000
Median :0.0000   Median :27.34   Median :0.0000   Median :0.0000   Median :0.00000   Median : 0.000
Mean   :0.0856   Mean   :28.33   Mean   :0.4125   Mean   :0.0681   Mean   :0.03774   Mean   : 3.372
3rd Qu.:0.0000   3rd Qu.:31.42   3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu.:0.00000   3rd Qu.: 2.000
Max.   :1.0000   Max.   :94.85   Max.   :1.0000   Max.   :1.0000   Max.   :1.00000   Max.   :30.000
 MentalHealth     DiffWalking         Sex            AgeCategory            Race              Diabetic
Min.   : 0.000   Min.   :0.0000   Length:319795      Length:319795      Length:319795      Min.   :0.0000
1st Qu.: 0.000   1st Qu.:0.0000   Class :character   Class :character   Class :character   1st Qu.:0.0000
Median : 0.000   Median :0.0000   Mode  :character   Mode  :character   Mode  :character   Median :0.0000
Mean   : 3.898   Mean   :0.1389                                                           Mean   :0.1356
3rd Qu.: 3.000   3rd Qu.:0.0000                                                           3rd Qu.:0.0000
Max.   :30.000   Max.   :1.0000                                                           Max.   :1.0000
PhysicalActivity  GenHealth          SleepTime         Asthma       KidneyDisease       SkinCancer
Min.   :0.0000   Length:319795      Min.   : 1.000   Min.   :0.0000   Min.   :0.00000   Min.   :0.00000
1st Qu.:1.0000   Class :character   1st Qu.: 6.000   1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.00000
Median :1.0000   Mode  :character   Median : 7.000   Median :0.0000   Median :0.00000   Median :0.00000
Mean   :0.7754                      Mean   : 7.097   Mean   :0.1341   Mean   :0.03683   Mean   :0.09324
3rd Qu.:1.0000                      3rd Qu.: 8.000   3rd Qu.:0.0000   3rd Qu.:0.00000   3rd Qu.:0.00000
Max.   :1.0000                      Max.   :24.000   Max.   :1.0000   Max.   :1.00000   Max.   :1.00000
> |
```

Figure 2: Summary details of dataset after cleansing

Exploratory Data Analysis

Cleaning and processing

Many categorical variables appeared in the data, each with a value of "yes" or "no" for each medical condition. As a result, we opted to change the data by converting "yes" to 1 and "no" to 0. All categorical data are converted to factors, yielding numerical variables like sleep time, physical health, mental health, and BMI values. With only four missing numbers (NA), the data were clean. As a result, we attempted to eliminate all missing values and prepare the data for future analysis.

Outliers

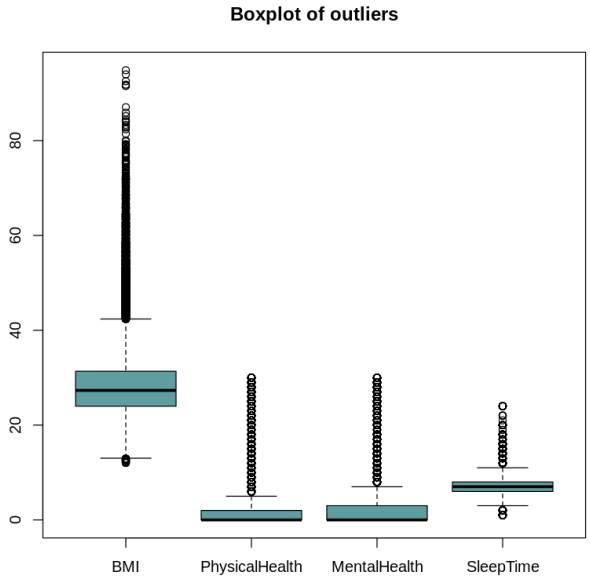


Figure 3: Outliers in BMI, Physical Health, Mental Health, and Sleep Time

As seen in Figure 3, most of the data is categorical, with only a few numerical values having outliers. The BMI ranges from 18 to 30 in most cases. All people over the age of 30 are considered obese. All the figures were exceeding 80. The maximum amount of sleep time for a normal human being is 16 hours. It would be hard to sleep for more than 16 hours every day. For the range of 0 to 10, physical and mental health parameters are tested and calculated. These outliers are those in the range above. They are dealt with by eliminating and altering the dataset.

Without Outliers

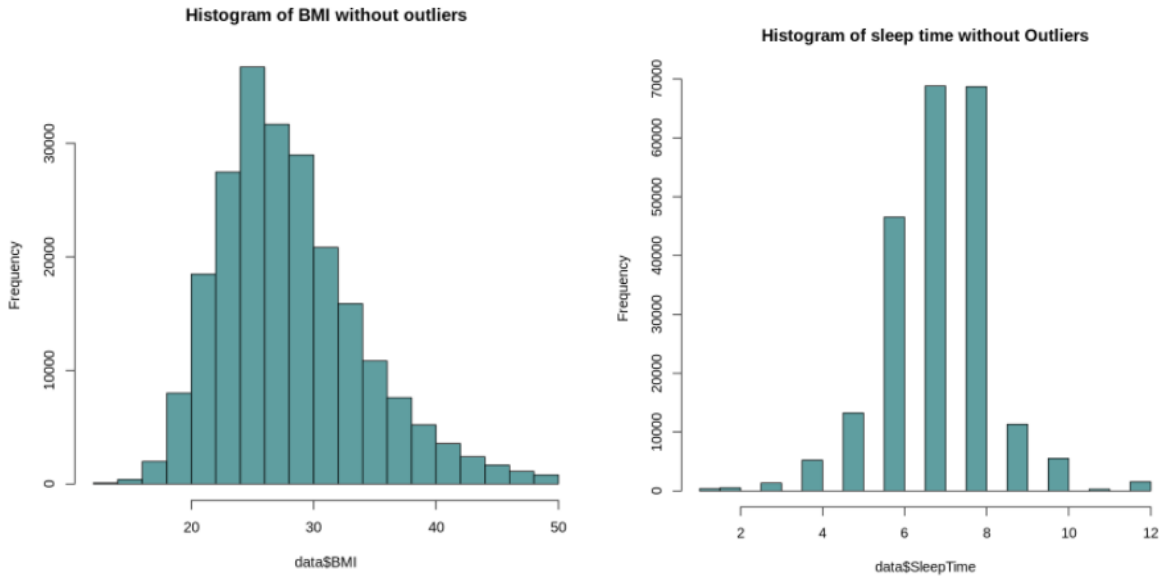


Figure 4: After removing the outliers in BMI and Sleep time

This left graph depicts the histogram of the BMI readings of US people who participated in the survey. Surprisingly, most US individuals have normal BMI readings between 18 and 30. However, many Americans have a BMI of greater than 30, and many are obese. And the right histogram depicts that the most people in the United States sleep for 6 to 8 hours every night. There are about 3000 people in the United States who would like to sleep more.

Summarizing the dataset

total	hasHeartDisease
<int>	<dbl>
319795	0.08559546

Figure 5: Applied summarize on full dataset

By summarizing the full dataset, the total number of test group and the percentage of people who have heart disease, as shown in the above figure, are 319,795 people and 8.56% out of the total population have heart disease.

Group by Gender

Sex	total	hasHeartDisease
<fct>	<int>	<dbl>
Female	167805	0.06694675
Male	151990	0.10618462

Figure 6: Gender wise summary details of dataset

Grouping the data by sex and summarizing it, the results in the Figure 6 above show that in the total population of the test group, 167,805 of them are female, where 6.69% of these women have heart disease, and 151,990 of them are males, where 10.62% of these men have heart disease. Thus, in general, male has a higher likelihood for having a heart disease than female.

Group by Race

Race	total	hasHeartDisease
<fct>	<int>	<dbl>
American Indian/Alaskan Native	5202	0.10419070
Asian	8068	0.03296976
Black	22939	0.07537382
Hispanic	27446	0.05257597
Other	10928	0.08107613
White	245212	0.09178588

Figure 7: Race wise summary details of dataset

Grouping the data by race and summarizing it, the results in the Figure 7 above show that in the total population of the test group, the White race has the largest population of 245,212, and the American Indian/Alaskan Native race has the smallest population of 5202. However, the American Indian/Alaskan Native race also has the highest proportion of people who have heart disease, while the Asian race has the lowest proportion. Thus, in general, the American Indian/Alaskan Native race has the highest likelihood of having heart disease than other races, while the Asian race has the lowest likelihood of having heart disease than other races

Group by Age

AgeCategory	total	hasHeartDisease
<fct>	<int>	<dbl>
18-24	21064	0.006171667
25-29	16955	0.007844294
30-34	18753	0.012051405
35-39	20550	0.014403893
40-44	21006	0.023136247
45-49	21791	0.034142536
50-54	25382	0.054487432
55-59	29757	0.073999395
60-64	33686	0.098765066
65-69	34151	0.120084331
70-74	31065	0.156027684
75-79	21482	0.188483381
80 or older	24153	0.225603445

Figure 8: Age wise summary details of dataset

Grouping the data by age categories and summarizing it, the results in above figure show that the oldest age group has the highest percentage of people who have heart

disease while the youngest age group has the lowest percentage. The likelihood of having heart disease increases with age

Graphical exploration

Correlation Plot

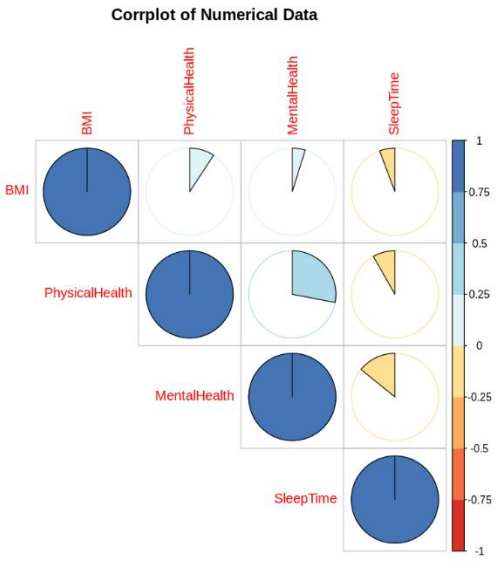


Figure 9: Correlation Plot among variables

We finally have the correlations for all numerical data, as shown in above figure, because the dataset contains several categorical factors. Even though all these variables are unrelated, physical, and mental health are intertwined, with a correlation value of 0.35.

Scatter Plots

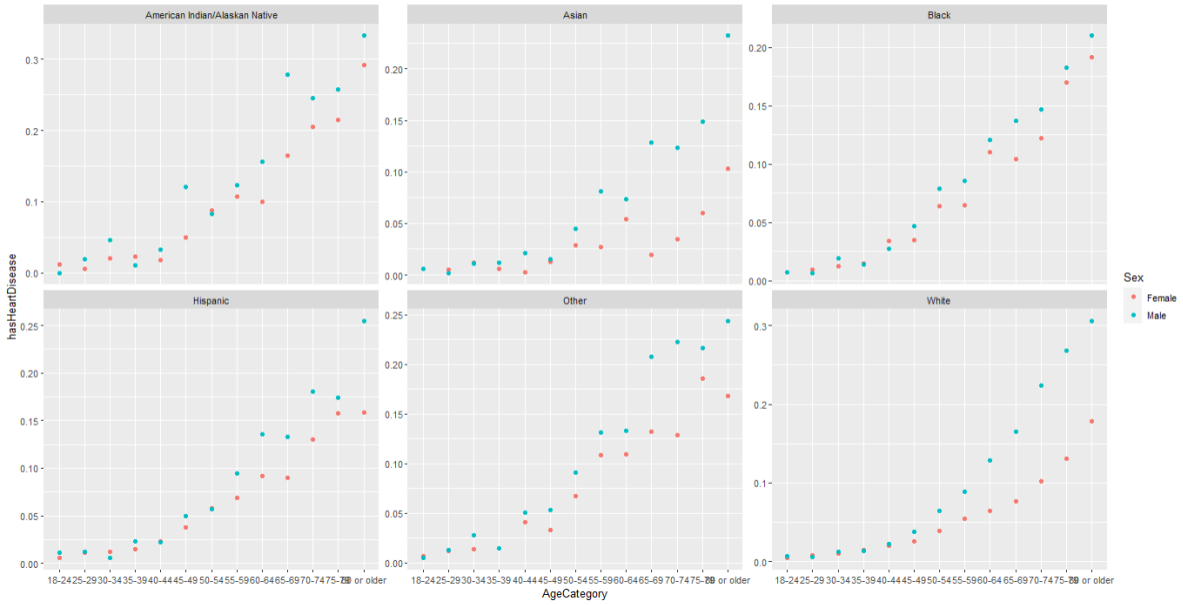


Figure 10: Scatter plot of % has heart disease in both genders by age for each race

The Figure 10 shows the percentage of having heart disease in both genders by age for each race. As we can see, the likelihood of having heart disease increases exponentially in both genders for each race. In addition, for the Asian, Hispanic, and White race, the gap for the percentage of having heart disease between men and women widens with age.

Bar plot

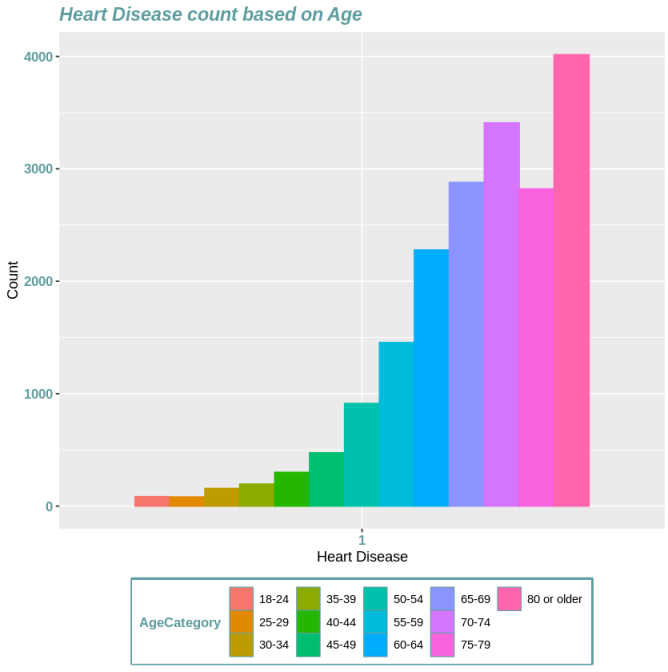


Figure 11: heart disease based on the Age

The details of the heart disease count dependent on age are shown in Figure 11. The graph shows that most US inhabitants suffer heart disease when they reach the age of 50. The older you get, the more likely you are to get heart disease. Around 4000 individuals over the age of 80 suffer from heart disease.

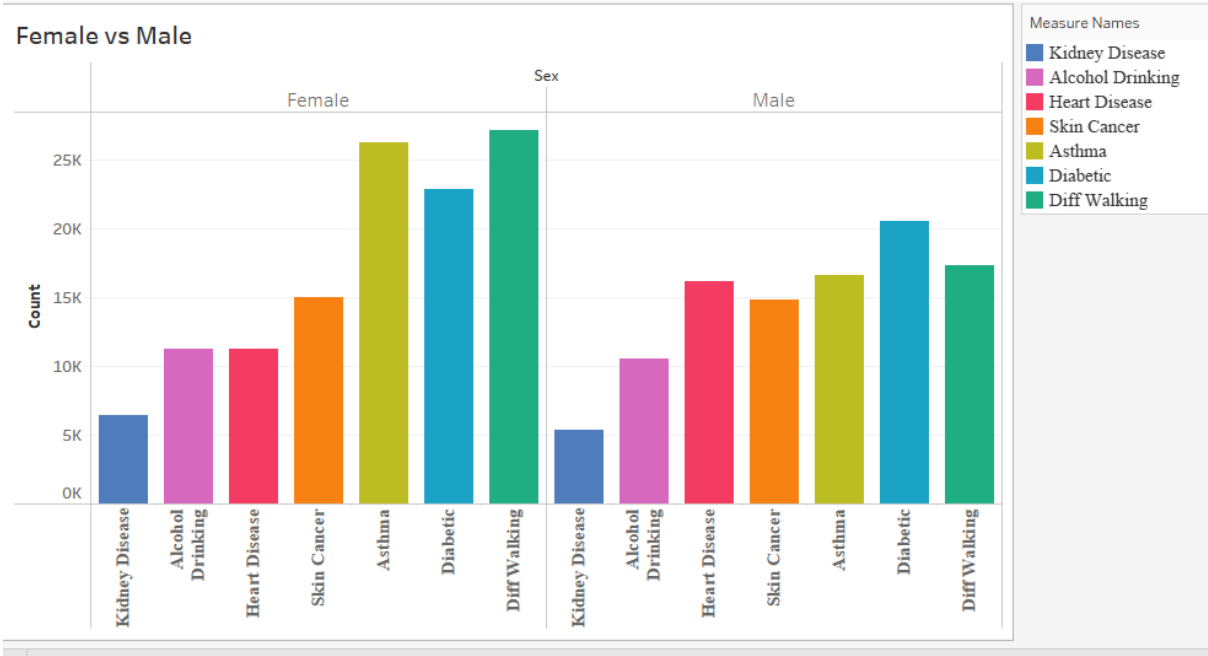


Figure 12: Comparison plot of Males and Females

The graph in above figure clearly depicts the comparison of all conditions among US inhabitants by gender. We can see that the rates of kidney illness, alcohol consumption, and skin cancer in both men and women are similar. We may also notice that males have a higher rate of heart disease than females. In comparison to males, a substantial number of females had asthma, diabetes, and trouble walking.

Model Selection and Exploration

We have looked at the plotting of different variables with mean of having a heart disease. We can see a relationship between dependent and independent variables. There are few methods which can be applied to get the answers of our predictive analysis.

As our dependent variable is categorical, binary in nature, we can use models such as logistic regression, random forest, SVM, K-means, XGBoost and many more. These models use the significant independent variables that we have studied above through plots to build a better model to predict the chances of having a heart disease in the test dataset when we fit the model there.

Logistic Regression

Logistic regression is considered amongst the simplest and best techniques for a binary classification dataset like the one we have one of heart disease. We have applied the model with different combinations of attributes with our target variable HeartDisease but we get better results when we applied it by considering all the variables.

Fit the model

```
> LR_model <- glm(HeartDisease ~. , data = trainset, family = "binomial")
> summary(LR_model)

Call:
glm(formula = HeartDisease ~ ., family = "binomial", data = trainset)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.1172  -0.4108  -0.2434  -0.1288   3.6200

Coefficients:
                           Estimate Std. Error z value Pr(>|z|)
(Intercept)              -6.2789632  0.1283963 -48.903  < 2e-16 ***
BMI                       0.0084312  0.0012815   6.579 4.73e-11 ***
Smoking                   0.3538421  0.0160826  22.002  < 2e-16 ***
AlcoholDrinking          -0.2463103  0.0375590  -6.558 5.46e-11 ***
Stroke                    1.0354929  0.0253506  40.847  < 2e-16 ***
PhysicalHealth            0.0035225  0.0009677   3.640 0.000272 ***
MentalHealth              0.0048896  0.0009875   4.951 7.37e-07 ***
DiffWalking               0.2019406  0.0203342   9.931  < 2e-16 ***
SexMale                   0.7282380  0.0162568  44.796  < 2e-16 ***
AgeCategory25-29          0.1008943  0.1402741   0.719 0.471978
AgeCategory30-34          0.4902865  0.1245546   3.936 8.27e-05 ***
AgeCategory35-39          0.5614323  0.1200049   4.678 2.89e-06 ***
AgeCategory40-44          0.9872137  0.1124752   8.777  < 2e-16 ***
AgeCategory45-49          1.3243415  0.1082713  12.232  < 2e-16 ***
AgeCategory50-54          1.7264785  0.1046654  16.495  < 2e-16 ***
AgeCategory55-59          1.9925071  0.1029126  19.361  < 2e-16 ***
AgeCategory60-64          2.2365744  0.1020253  21.922  < 2e-16 ***
AgeCategory65-69          2.4759123  0.1017285  24.338  < 2e-16 ***
AgeCategory70-74          2.7648249  0.1016417  27.202  < 2e-16 ***
AgeCategory75-79          2.9636193  0.1022600  28.981  < 2e-16 ***
AgeCategory80 or older    3.2203498  0.1019450  31.589  < 2e-16 ***
RaceAsian                -0.5082814  0.0927596  -5.480 4.26e-08 ***
RaceBlack                -0.3724812  0.0641507  -5.806 6.39e-09 ***
RaceHispanic             -0.2928093  0.0653753  -4.479 7.50e-06 ***
RaceOther                -0.1075444  0.0713912  -1.506 0.131962
RaceWhite                -0.1151863  0.0572136  -2.013 0.044087 *
Diabetic                  0.4713115  0.0183696  25.657  < 2e-16 ***

PhysicalActivity          0.0028235  0.0179286   0.157 0.874861
GenHealthFair             1.5196435  0.0369240  41.156  < 2e-16 ***
GenHealthGood             1.0579039  0.0332546  31.812  < 2e-16 ***
GenHealthPoor             1.9075841  0.0459553  41.510  < 2e-16 ***
GenHealthVery good        0.4925118  0.0340958  14.445  < 2e-16 ***
SleepTime                -0.0241090  0.0048466  -4.974 6.55e-07 ***
Asthma                    0.2757423  0.0214912  12.830  < 2e-16 ***
KidneyDisease             0.5659057  0.0272588  20.760  < 2e-16 ***
SkinCancer                0.1149675  0.0217866   5.277 1.31e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 149245  on 255770  degrees of freedom
Residual deviance: 115922  on 255735  degrees of freedom
AIC: 115994

Number of Fisher Scoring iterations: 7
```

Figure 13: Fit Logistics regression model by setting required parameters

From the above results, we can say that the model is fine as the difference is large between null and residual deviance. In addition to this, a lower AIC value indicates a better model so, moving forward we will apply this on our test dataset to compute the confusion matrix.

Create confusion matrix

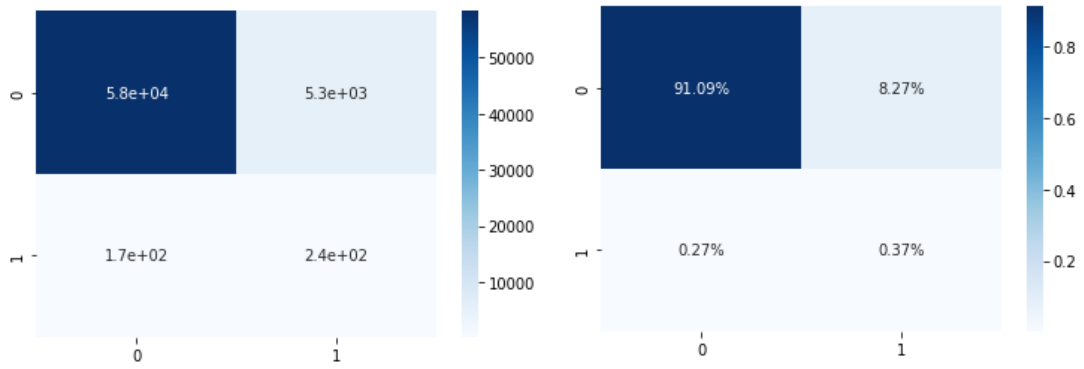


Figure 14: Confusion matrix – Accuracy, Precision and F1 score details

Here, with our dataset of binary classification, we applied the logistic regression model with all the attributes. Figure 14 shows the confusion matrix plot with true positive and true negative are 91.09% and 0.37%, respectively, whereas false positive and false negative are 8.27% and 0.27% With a precision score of 0.9167, recall value of 0.9970 and an F1 Score of 0.9552, the accuracy is 91.46%.

SVM - Support Vector Machine

Support Vector Machines (SVMs) is a supervised machine learning algorithm capable of performing classification and regression analysis. The first step we did is splitting the dataset into a training set, in which 80% of the data is randomly assigned, and a test set, in which contains the rest 20% of the data. Since our data set contains both categorical and numeric variables, we need to encode the categorical variables to the dummy variables and standardize the numeric variables before fitting SVM models to the training dataset.

Fit the model

```
> for (i in 1:100){
+   sample_train <- new_train[sample(nrow(new_train), 1000, replace = FALSE), ]
+   svm_model<- svm(as.factor(V1) ~ ., data = sample_train,
+                   type = "C-classification", kernel = "radial")
+   pred_test <- predict(svm_model, new_test)
+   accuracy[i] <- mean(pred_test == new_test$V1)
+ }
> #print average accuracy and standard deviation
> mean(accuracy)
[1] 0.9142237
> sd(accuracy)
[1] 0.0005524981
```

Figure 15: Fit RBF SVM models for 100 partitions

Since our dataset contains approximately 320,000 observations and even the training set itself contains more than 250,000 observations, it would take an exceptionally long execution time to fit a SVM model using the full training set. So, we used 100 different sample datasets, where each of them includes 1,000 observations randomly selected from the training set, for a Radial Basis Function kernel SVM model, and then calculated the average accuracy and standard deviation over all iterations. The average accuracy over 100 iterations is 91.42% with a standard deviation of 0.0005525.

Create confusion matrix

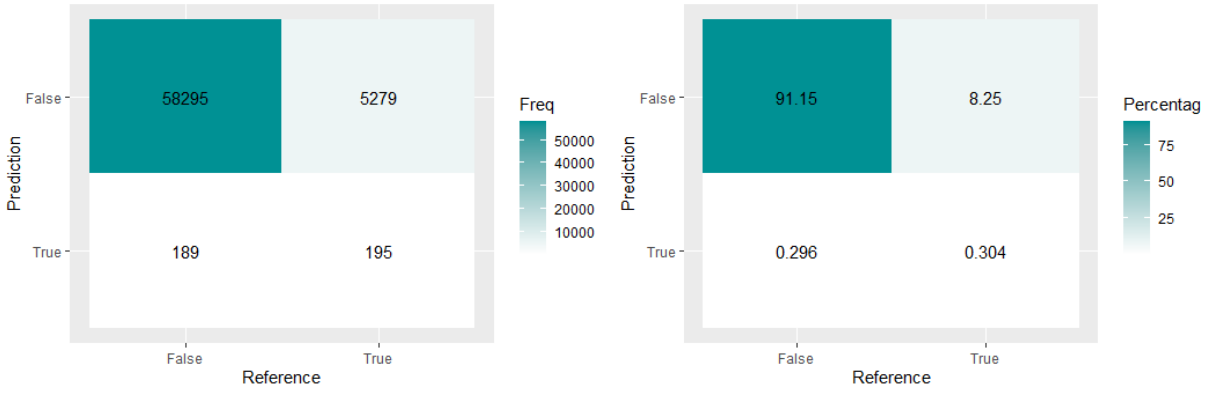


Figure 16: Confusion matrix – Accuracy, Precision and F1 score details

The confusion matrix plots in Figure 16 are results for one of the 100 iterations we generated previously. In this case, we can see that the model predicted true negative with 91.15% and true positive with 0.304%, which gives an accuracy of 91.453%. Besides, the precision score is 0.9170, the recall value is 0.99677, and the F1 score is 0.9552.

Random Forest

Fit the model

We have used the SVM model to obtain the average accuracy over 100 iterations is 91.42% with a standard deviation of 0.0005525. We wanted to see if there is any model that can give a higher accuracy, so we have taken the Random Forest model and wanted to find the accuracy of the dataset. Random Forest - builds and combines multiple decision trees to get more accurate predictions. Random Forest is typically the bagging method, so we wanted to find the Out of bag cross-validation error in which the higher the percentage to 10 the higher the chance of using the model in general. The out-of-bag cross-validation error was approximately 8.57 percent.

```
rf<-randomForest(HeartDisease~.,ntree=500,data=train)
rf
plot(rf$err.rate[,1],type="l",main="Random Forest Error Rate",xlab="Number of Trees"
varImpPlot(rf,main="Variable Importance Plot for Random Forest")
rfpred<-predict(rf,test,type="class")
rft<-table(test$HeartDisease,rfpred)
rft
sum(diag(rft))/nrow(test)
rft["No","Yes"]/sum(rft["No",])
rft["Yes","No"]/sum(rft["Yes",])
test$HeartDiseaseRF<-rfpred
```

Figure 17: Fit Random Forest model by setting required parameters

Create confusion matrix:

```
        OOB estimate of  error rate: 8.51%
Confusion matrix:
        No  Yes class.error
No  232228 1646 0.007037978
Yes  20131 1831 0.916628722
```

Figure 18: Confusion matrix – Accuracy, Precision and F1 score details

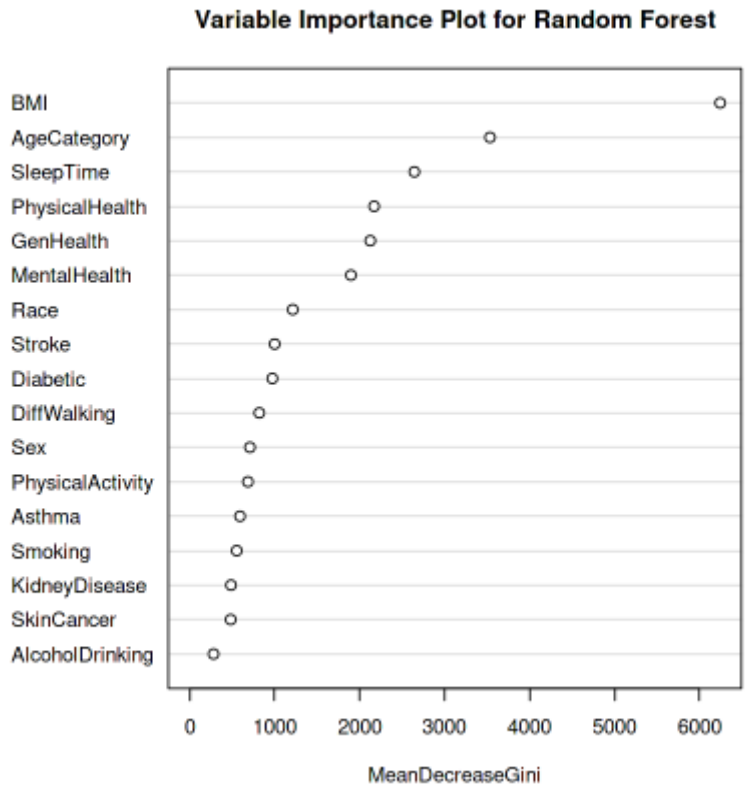
In the Random Forest section of the Appendix, there is a random forest with a default of 500 trees on the data. The false-positive rate was approximately 0.6 percent, while the false-negative rate was still high at approximately 91.4 percent. Overall, it looked to perform better than a random string of zeroes, with a slightly higher accuracy rate and a lower false-negative rate. I also constructed a variable relevance plot to visualize the influential variables in the random forest as the number of trees increased.

The accuracy rate is 91.7%.

The precision score is 0.992.

The F1 score is 0.9552.

After 100 trees, the inaccuracy no longer decreases.



From the above plot, In the random forest, BMI is the most relevant variable, followed by age and sleep time. The influential variables are general health and (6-7) days of poor physical and mental health.

XGBoost

Fit the model

XGBoost is a distributed gradient boosting library that has been developed to be extremely efficient, adaptable, and portable. It uses the Gradient Boosting framework to implement machine learning algorithms. XGBoost uses parallel tree boosting to address a variety of data science issues quickly and accurately. we used the XGboost tree to implement the categorization, which can be shown in Figure 19.

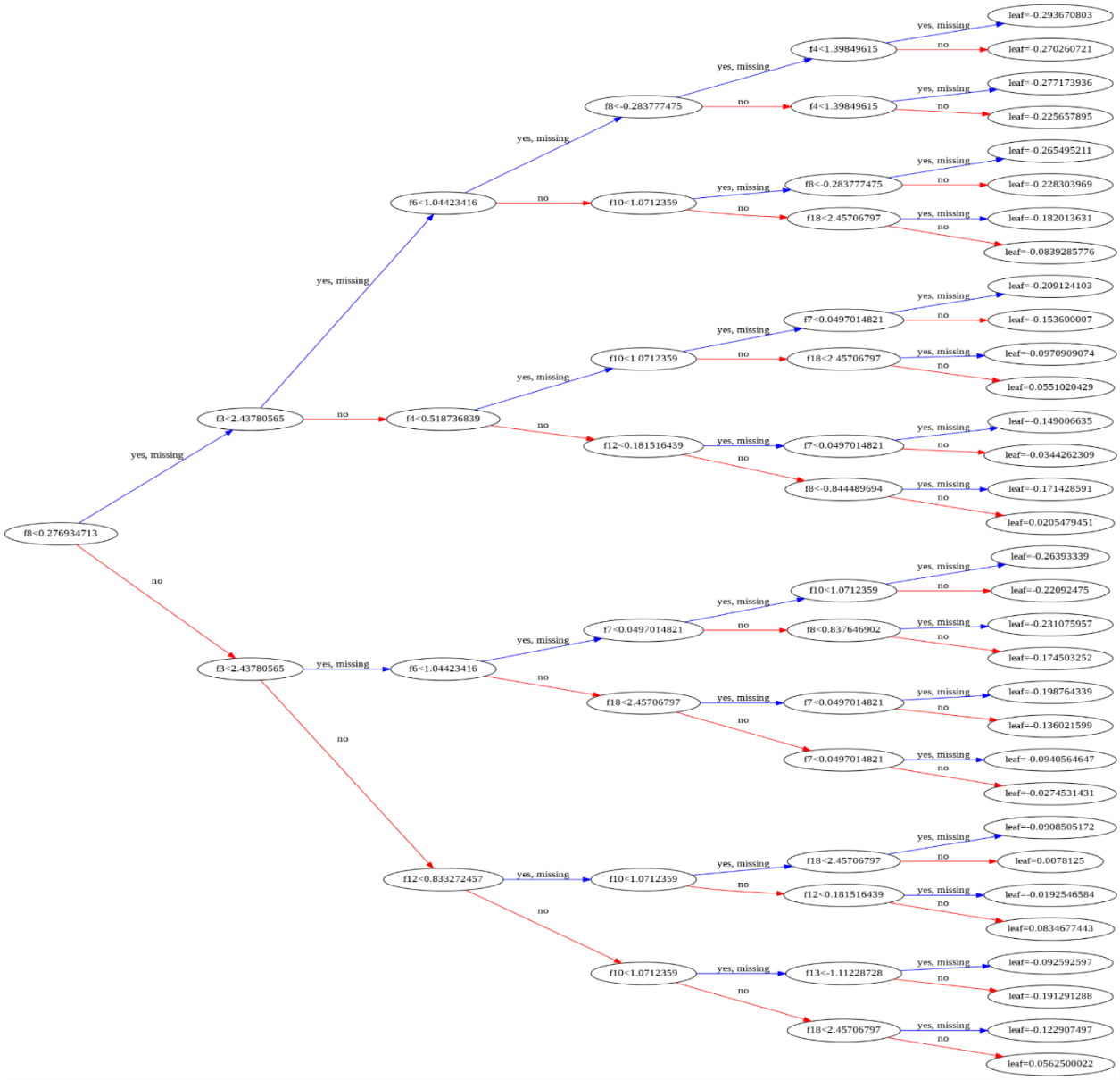


Figure 19: Fit XGBoost model by setting required parameters

Create confusion matrix

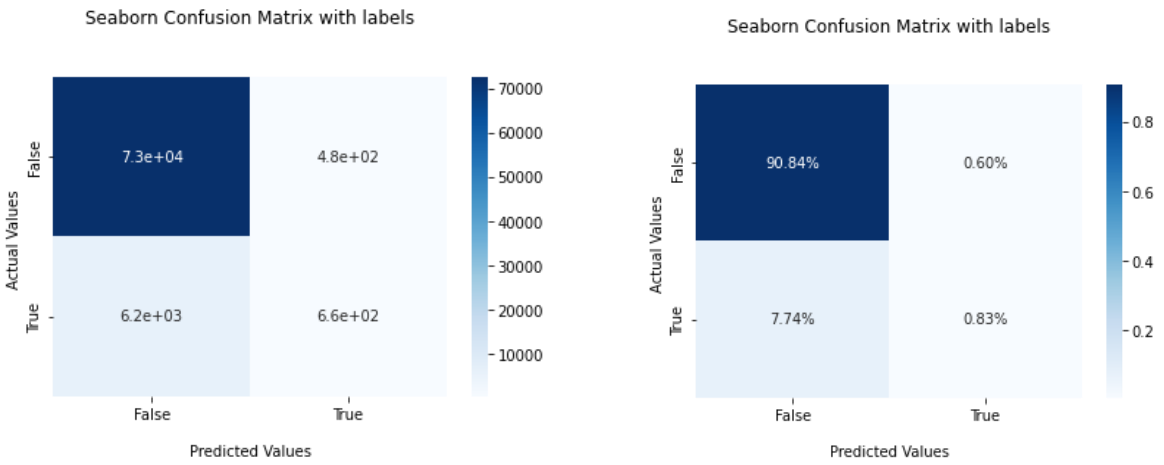


Figure 20: Confusion matrix – Accuracy, Precision and F1 score details

For the classification, we used the US XGboost classifier and fine-tuned it with a random search to find the optimal parameter. Figure 20 shows the confusion matrix plot with and without labels, showing that true positive and true negative are 91 percent and 0.83 percent, respectively, whereas false positive and false negative are 0.6 percent and 7.74 percent. With a precision score of 0.581 and an F1 Score of 0.166, the accuracy is 91.7 percent.

Comparison and Interpretation

Classifiers	Accuracy (%)	Precision Score	F1 Score
Logistic Regression	91.46	0.9167	0.9552
SVM	91.42	0.9170	0.9552
Random Forest	91.70	**0.9920**	0.9552
XGBoost	**91.71**	0.9900	**0.9561**

Table 2: Classifier comparison – Accuracy, Precision and F1 score

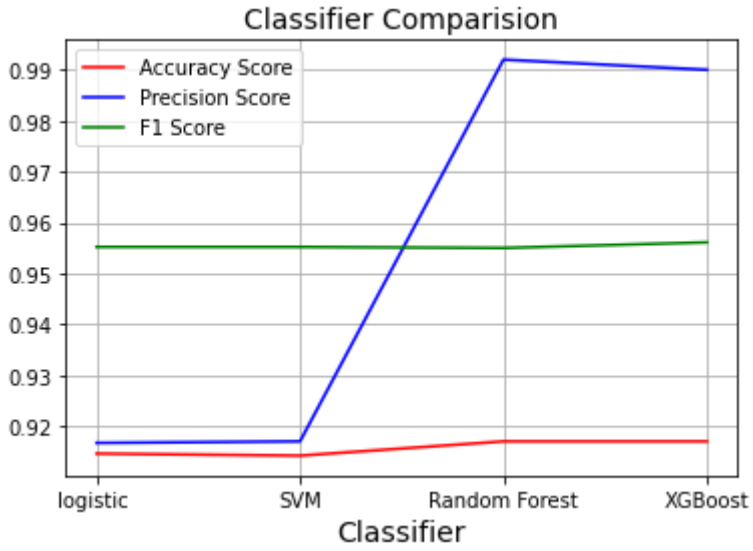


Figure 21: Classifier Comparison

As indicated in the preceding table and Figure 21, all the models have an Accuracy score of around 0.91, but since we are working with the medical business, predicting if a patient has heart disease is critical. If incorrectly forecast, the repercussions could be catastrophic. As a result, type 1 errors in the models should be minimized. As a result, precision is not a superior metric. As a result, we decide to consider the F1 and precision scores. The higher the F1 and precision score, the better the choice. All the F1 scores in Figure 21 are similar, however, the precision score is higher in the case of random forest.

Recommendations and Conclusion

The above analysis will help us understand that for the given dataset, machine learning models can accurately predict if a person has heart illness or not, as well as which parameters affect this heart disease. As a result, if BMI, sleeping time, age, physical health, and other factors are fed into the algorithm, it can predict whether a person has heart disease or not. Machine learning models such as random forest, SVM, XG Boost, and logistic regression were tried. When we examine the results in Figure 21, we can see that, in comparison to random forest, XG Boost looks to have superior accuracy, a higher F1-score, and a higher precision score. Moreover, we can also have fine-tuned it to be better than any other model that has been tested previously.

Though we attempted to provide a solution based on random forest and XGBoost models for predicting cardiac illness, there is always room for improvement. The better the results, especially when dealing with heart illness, the easier it is for doctors to examine the person's health status. As a result, we decided to work on alternative ML (Machine Learning) classification models in the future, such as naive Bayes, k means, and deep learning models, to find the optimal solution.

Appendix

R – Scripts

Logistic Regression

```
# Install required packages and libraries
        install.packages(c ("skimr", "countrycode","dplyr","ggplot2","broom","tidyr","purrr",
                "gmodels","rpart","rpart. plot","pROC","MASS"))
# Add double quotes for each package to install more than one packages
         library(dplyr)
        library(countrycode)
        library(ggplot2)
        library(broom)
        library(tidyr)
        library(purrr)
        library(gmodels)
        library(skimr)
        library(rpart)
        library(rpart.plot)
        library(pROC)
        library (MASS)
        library(corrplot)
# Import the cleaned dataset
         HeartData <- read.csv ("heart.csv", header=TRUE, sep=",", quote = "\"")
        head(HeartData)
        skim(HeartData)
        str(HeartData)
        summary(HeartData)
# Split the dataset into training and testing data with 80/20 proportion
        HeartData[, "train"] <- ifelse(runif(nrow(HeartData))<0.8, 1, 0)
 #Assign training rows to data frame trainset
        trainset <- HeartData[HeartData$train == 1,]
```

```
#Assign test rows to data frame testset
        testset <- HeartData[HeartData$train == 0,]
 #remove "train" column from train and test dataset
        trainset <- trainset [, -trainColNum]
        testset <- testset[, -trainColNum]
# Apply planning regression, setting required parameters
        model1 <- glm(HeartDisease ~ Diabetic + Asthma + KidneyDisease + SkinCancer , data =
        trainset, family = binomial (link = "logit"))
        summary(model1)
         model2 <- glm(HeartDisease ~ Smoking + AlcoholDrinking + Stroke, data = trainset, family
        = binomial (link = "logit"))
        summary(model2)
         LR_model <- glm(HeartDisease ~., data = trainset, family = "binomial")
        summary(LR_model)
 # Analysis:
# The model is fine when the difference is large between null and residual deviance.
# A lower AIC value indicates a better model.
# The AIC for LR_model has the lowest among them
# The difference between null and residual deviance has also higher compared to other two models
# So, moving forward we will proceed with "LR_model" as it is better than others


# Compute test accuracy
        pred_test <- predict(LR_model, testset)
        pred_test
        pred = as.factor(ifelse(pred_test >= 0.5, 1,0))
        pred
        mean (pred == testset$HeartDisease)
        testset$HeartDisease = as.factor(testset$HeartDisease)
        class(testset$HeartDisease)
# Confusion matrix
        confusionMatrix(pred, testset$HeartDisease)
 # Analysis
# After taking all the variables or attributes, the model developed an accuracy
# Of prediction as 91.46%.
```

A dispersion is being observed between Sensitivity & Specificity which indicates

There may be chances of wrong classification.

SVM

```
        library(caret)
        library(lattice)
        library(e1071)
        set.seed(123)
#Split the data into an 80/20 training and test set
        index <- createDataPartition(data$HeartDisease, p = 0.8, list = FALSE)
        train <- data[index,]
        test <- data[-index,]
#Set dummy variables to represent the categorical variables
        dummies1 <- dummyVars(~., data = train [, -c (1, 2, 6, 7, 15)])
        c1 <- predict (dummies1, train [, -c (1, 2, 6, 7, 15)])
        new_train <- as.data.frame(cbind(train$HeartDisease, c1))
        dummies2 <- dummyVars(~., data = test [, -c (1, 2, 6, 7, 15)])
        c2 <- predict (dummies2, test [, -c (1, 2, 6, 7, 15)])
        new_test <- as.data.frame(cbind(test$HeartDisease, c2))
#Feature scaled training and test dataset
        new_train[, 46:49] = scale (train [c (2, 6, 7, 15)])
        new_test[, 46:49] = scale (test [c (2, 6, 7, 15)])
#Create vector to store accuracies and set random number seed
        accuracy <- rep (NA, 100)
#Calculate accuracies for 100 training/test partitions
        for (i in 1:100) {
          sample_train <- new_train[sample(nrow(new_train), 1000, replace = FALSE),]
          svm_model<- svm(as.factor(V1) ~., data = sample_train,
                          type = "C-classification", kernel = "radial")
          pred_test <- predict(svm_model, new_test)
          accuracy[i] <- mean(pred_test == new_test$V1)
        }
```

```
#Print average accuracy and standard deviation
        mean(accuracy)
        sd(accuracy)
#Random select 1000 samples from the new training set
        sample_train <- new_train[sample(nrow(new_train), 1000, replace = FALSE),]
#Fit SVM to the training set
        svm_model <- svm(as.factor(V1) ~., data = sample_train,
                type = "C-classification", kernel = "radial")
        svm_model
        pred_test <- predict(svm_model, new_test)
        mean(pred_test == new_test$V1)
        cm <- confusionMatrix(pred_test, as.factor(new_test$V1), mode = "everything")
        cm
        plt <- as.data.frame(cm$table)
        plt$Prediction <- factor(plt$Prediction, levels = rev(levels(plt$Prediction)))
        ggplot(plt, aes(Reference, Prediction, fill = Freq)) +
          geom_tile() + geom_text(aes(label = Freq)) +
          scale_fill_gradient(low = "white", high = "#009194") +
          labs (x = "Reference", y = "Prediction") +
          scale_x_discrete(labels=c("False","True")) +
          scale_y_discrete(labels=c("True","False"))
        value <- as.data.frame(plt$Freq / nrow(test) * 100)
        value <- as.data.frame(c (91.15, 0.296, 8.25, 0.304))
        sta <- cbind(plt$Prediction, plt$Reference, value)
        names(sta)[1] <- "Prediction"
        names(sta)[2] <- "Reference"
        names(sta)[3] <- "Percentage"
        ggplot(sta, aes(Reference, Prediction, fill = Percentage)) +
          geom_tile() + geom_text(aes(label = Percentage)) +
          scale_fill_gradient(low = "white", high = "#009194") +
          labs (x = "Reference", y = "Prediction") +
          scale_x_discrete(labels=c("False","True")) +
          scale_y_discrete(labels=c("True","False"))
```

Random Forest

```
#Loading the required Libraries.
        install.packages("randomForest")
        library(randomForest)
        install.packages("caret")
        library(caret)
        set.seed(2904)
#Split the data into an 80/20 training and test set
        index <- createDataPartition(data$HeartDisease, p = 0.8, list = FALSE)
        train <- data[index,]
        test <- data[-index,]
#Assigning the training dataset into the Random Forest model.
        rf<-randomForest(HeartDisease~.,ntree=500, data=train)
        Rf
        plot(rf$err.rate[,1], type="l",main="Random Forest Error Rate",xlab="Number of
        Trees")
#Plotting the Important Variable in the dataset.
        varImpPlot(rf,main="Variable Importance Plot for Random Forest")
        rfpred<-predict(rf,test,type="class")
#Creating the Confusion Matrix.
        rft<-table(test$HeartDisease,rfpred)
        rft
        sum(diag(rft))/nrow(test)
        rft["No","Yes"]/sum(rft["No",])
        rft["Yes","No"]/sum(rft["Yes",])
        test$HeartDiseaseRF<-rfpred
```

Python-Scripts

XG Boost

```
        from sklearn import preprocessing
        le = preprocessing.LabelEncoder()
# Label encoding all the categorical columns that have more than 2 unique values
        df['Sex']=le.fit_transform(df['Sex'])
        df['AgeCategory']=le.fit_transform(df['AgeCategory'])
        df['Race']=le.fit_transform(df['Race'])
        df['Diabetic']=le.fit_transform(df['Diabetic'])
        df['GenHealth']=le.fit_transform(df['GenHealth'])
        cols=
["BMI","Smoking","AlcoholDrinking","Stroke","PhysicalHealth","MentalHealth","DiffWalki
ng","Sex","AgeCategory","Race","Diabetic","PhysicalActivity","GenHealth",""
        "SleepTime","Asthma","Asthma","KidneyDisease","Asthma","KidneyDisease","SkinCancer"
]
        X=df[cols]
        y=df['HeartDisease']
        X=np.array(X)
# Splitting
        from sklearn import neighbors, datasets, preprocessing
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import accuracy_score
        X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=33)
        scaler = preprocessing.StandardScaler(). fit(X_train)
        X_train = scaler.transform(X_train)
        X_test = scaler.transform(X_test)
# Making an object of the XGBoost class
        from sklearn.metrics import confusion_matrix
        from xgboost import XGBClassifier
        clf = XGBClassifier()
        clf.fit(X_train, y_train)
        y_pred = clf.predict(X_test)
        cm = confusion_matrix(y_test,y_pred)
        print(cm)
```

```
        print(accuracy_score(y_test,y_pred))
#Random Search hyper parameter tuning
        from sklearn.model_selection import RandomizedSearchCV, GridSearchCV
        from sklearn.metrics import roc_auc_score
        import random
        from sklearn.model_selection import StratifiedKFold
        xgb = XGBClassifier()
        params = {
         "learning_rate" : [0.05,0.10,0.15,0.20,0.25,0.30],
         "max_depth": [ 3, 4, 5, 6, 8, 10, 12, 15],
         "min_child_weight" : [ 1, 3, 5, 7],
         "gamma": [ 0.0, 0.1, 0.2, 0.3, 0.4],
         "colsample_bytree": [ 0.3, 0.4, 0.5, 0.7]
        }
        folds = 3
        param_comb = 5
        skf = StratifiedKFold(n_splits=folds, shuffle = True, random_state = 1001)
        random_search = RandomizedSearchCV(xgb, param_distributions=params,
        n_iter=param_comb, scoring='roc_auc', n_jobs=4, cv=skf.split(X,y), verbose=3,
        random_state=1001)
        random_search.fit(X, y)
# Best params
        random_search.best_estimator_
#Classification after tuning
        clf = XGBClassifier(colsample_bytree=0.7, gamma=0.0, learning_rate=0.15, max_depth=5,
                min_child_weight=3)
        clf.fit(X_train, y_train)
        y_pred = clf.predict(X_test)
        cm = confusion_matrix(y_test,y_pred)
        print(cm)
        print(accuracy_score(y_test,y_pred))
##plotting tree
```

```
        from sklearn import tree
        from xgboost import plot_tree
        from matplotlib.pylab import rcParams
##set up the parameters
        rcParams['figure.figsize'] = 40,40
        plot_tree(clf, num_trees=0, rankdir='LR')
        from sklearn.metrics import precision_score, recall_score, f1_score
        print ('Precision: %.3f' % precision_score(y_test, y_pred))
        print ('Recall: %.3f' % recall_score(y_test, y_pred))
        print ('Accuracy: %.3f' % accuracy_score(y_test, y_pred))
        print ('F1 Score: %.3f' % f1_score(y_test, y_pred))
##Confusion Matrix
        import seaborn as sns
        import matplotlib.pyplot as plt
        ax = sns.heatmap(cm, annot=True, cmap='Blues')
        ax.set_title('Seaborn Confusion Matrix with labels\n\n').
        ax.set_xlabel('\nPredicted Values')
        ax.set_ylabel('Actual Values ').
## Ticket labels - List must be in alphabetical order
        ax.xaxis.set_ticklabels(['False','True'])
        ax.yaxis.set_ticklabels(['False','True'])
## Display the visualization of the Confusion Matrix.
        plt.show()
        ax = sns.heatmap(cm/np.sum(cm), annot=True,
                fmt='.2%', cmap='Blues')
        ax.set_title('Seaborn Confusion Matrix with labels\n\n')
        ax.set_xlabel('\nPredicted Values')
        ax.set_ylabel('Actual Values ').
## Ticket labels - List must be in alphabetical order
        ax.xaxis.set_ticklabels(['False','True'])
        ax.yaxis.set_ticklabels(['False','True'])
## Display the visualization of the Confusion Matrix.
```

```
            plt.show()
# Classifier Comparison
            Data = {'Classifier': ["logistic ","SVM","Random Forest","XGBoost"],
               'Accuracy Score': [0.9146,0.9142,0.917,0.917],
               'Precision Score': [0.9167,0.9170,0.992,0.990],
               'F1 Score': [0.9552,0.9552,0.955,0.9561]
                 }
            df = pd.DataFrame(Data,columns=['Classifier','Accuracy Score','Precision Score','F1 Score'])
            plt.plot(df['Classifier'], df['Accuracy Score'], color='red', label="Accuracy Score")
            plt.plot(df['Classifier'], df['Precision Score'], color='blue',label="Precision Score")
            plt.plot(df['Classifier'], df['F1 Score'], color='green',label="F1 Score")
            plt.title('Classifier Comparision', fontsize=14)
            plt.xlabel('Classifier', fontsize=14)
            plt.grid(True)
            plt.legend()
```

References

- Allan G. Bluman, 2018. Elementary Statistics: A step by step Approach. Mc Graw Hill Education. Edition 10th
- Robert I. Kabacoff, 2011. R in Action: Data Analysis and Graphics with R. Manning Publications Co.
- Personal Key Indicators of Heart Disease. (2022). Retrieved 27 April 2022, from https://www.kaggle.com/datasets/kamilpytlak/personal-key-indicators-of-heart-disease?select=heart_2020_cleaned.csv
- Aruchamy, V. (2021, December 15). *How To Plot Confusion Matrix in Python and Why You Need To?* Stack Vidhya. https://www.stackvidhya.com/plot-confusion-matrix-in-python-and-why/#:%7E:text=Plot%20Confusion%20Matrix%20for%20Binary%20Classes%20With%20Labels&text=You%20need%20to%20create%20a,matrix%20with%20the%20labels%20annotation.
- Brownlee, J. (2020, August 27). *How to Visualize Gradient Boosting Decision Trees with XGBoost in Python*. Machine Learning Mastery. https://machinelearningmastery.com/visualize-gradient-boosting-decision-trees-xgboost-python/
- *Hyper parameters tuning XGBClassifier*. (2019, April 23). Data Science Stack Exchange. https://datascience.stackexchange.com/questions/49746/hyper-parameters-tuning-xgbclassifier

- Kumar, A. (2022, April 8). *Accuracy, Precision, Recall & F1-Score - Python Examples*. Data Analytics. https://vitalflux.com/accuracy-precision-recall-f1-score-python-example/
- Pal, J. K. (2022, January 6). *Hyperparameter Tuning in XGBoost using RandomizedSearchCV*. Medium. https://jayant017.medium.com/hyperparameter-tuning-in-xgboost-using-randomizedsearchcv-88fcb5b58a73
- Cheng, X. (2020, August 17). *Preprocessing of categorical predictors in SVM, KNN and KDC*. Statistics LibreTexts. Retrieved May 14, 2022, from https://stats.libretexts.org/Bookshelves/Computing_and_Modeling/RTG%3A_Classification_Methods/4%3A_Numerical_Experiments_and_Real_Data_Analysis/Preprocessing_of_categorical_predictors_in_SVM%2C_KNN_and_KDC
- Rashkiany, H. (2020, May 15). *Support Vector Machine in R: Using SVM to predict heart diseases*. Edureka. Retrieved May 14, 2022, from https://www.edureka.co/blog/support-vector-machine-in-r/

Table of Figures