Capstone Project : Disease Diagnostics

Emmanuel Antiri

2022-07-20

1. Introduction

Disease diagnosis using machine learning has gained prominence in recent years. Though the golden standard in health is to be certain through actual tests of the presence or absence of disease, performance of machine learning in diagnosis particularly with image diagnosis have proven useful. In this study, however, I examine the performance of different machine learning algorithms applied on preliminary medical examinations of potential diabetic patients in a Sub-African country. The Random Forest algorithm achieved the highest accuracy on the test set, *66.88%*, better than all other algorithms studied including the the baseline guessing accuracy of *52.76%*.

1.1 Description of Dataset

The data contains the results of preliminary medical examination of potential diabetic patients. The data is made up of 60000 observations. There are 9 variables; an outcome variable and 8 feature variables. Disease status is the outcome variable; a binary variable which indicates whether the individual was diagnosed with diabetes or not. The feature variables include gender, region, glucose level, creatinine level, blood urea nitrogen level, platelets count, bilirubin and age. *Note that the code for importing libraries,importing data and partitioning are concealed using chunk options for brevity.* A snapshot of the top six(6) observations of the data is shown below:

```
## # A tibble: 6 x 9
##   glucose_level bilirubin creatinine_level blood_urea_nitrogen   age
##           <dbl>     <dbl>            <dbl>                <dbl> <dbl>
## 1            33     0.405              0.3                    5    35
## 2           185     0.487              0.3                    9    65
## 3           124     0.442              0.3                   10    70
## 4            57     0.301              0.3                   12    40
## 5            96     0.432              0.3                   28    63
## 6            94     0.301              0.3                   10    33
## # ... with 4 more variables: platelets_count <dbl>, gender <chr>, region <chr>,
## #   disease_status <dbl>
```

The description of the various variables is shown below:

variable	description
glucose_level	Level of glucose (sugar) in blood measured in mmol/L
bilirubin	Average bilirubin concentration measured in micromol/L
creatinine_level	Creatinine Concentration in blood measured in micromol/L
blood_urea_nitrogen	Average Blood Urea Nitrogen Concentration in serum measured in millimol/L

variable	description
age	Age of Person measured in years
platelets_count	Average platelets count in blood measured in microliter
gender	Gender of the Person (Male/Female)
region	Region of Person (North/South)
disease_status	Disease Status (1 meaning Diagnosed/ 0 meaning Not Diagnosed)

1.2 Goal

The main goal of the project is to propose a supervised learning algorithm that achieves the highest accuracy in classifying diabetes status among a list of selected algorithms. The paper would also consider imbalance metrics such as F_1-score and Balanced Accuracy to account for the about 60%/40% imbalance in the target variable. Consequently, the chosen model should perform better than guessing. Also, the paper would examine the most important feature variables. Additionally, the paper would try as much as possible to demonstrate the practicability and interpretability of the methods on real world data. It should be noted that the paper would mainly use methods studied within the HarvardX Data Science Professional Certificate.

1.3 Key Steps

The key steps used in the paper are outlined below: 1. Data is split into train and test set. 2. Descriptive and Exploratory Analysis are examined to ascertain characteristics of variables. 3. Pre-processing of feature variables (Min-Max scaling of continuous features and one-hot binary encoding of string features). 4. Fit and ascertain performance of baseline (guessing) model. 5. Fit and ascertain performance of the following machine learning models: Logistics Regression, KNN, QDA, Decision Tree and Random Forest. 6. The algorithm with the highest accuracy is chosen as the final model. This algorithm should also have better Balanced Accuracy and F_1-score.

It should be noted at each step of the process, the performance of the model in terms of accuracy was evaluated on the test set. The R software was used in performing the analysis. The caret package is used for training and fitting the models.

2. Methods/Analysis

2.1 Data Cleaning/Data Splitting

The data was randomly split into two: the train set (80%) and the test set (20%). The algorithm would be trained exclusively on the training set, and the test set would be used for evaluation. Thus, we need as many observations as possible to train the algorithms, 80% in this case. On the other hand, we need a good portion of the data to evaluate our models to obtain stable estimates, the reason for 20% portion for the test set. Numerically, the train set contained 48000 observations whiles the test set contained 12000 observations.

2.2 Summary Statistics

The summary statistics of the continuous features of train set is shown below. The statistics are represented in the rows as the following: minimum (0%), first quartile (25%), median (50%), third quartile (75%), maximum (100%) and mean.

	glucose_level	bilirubin	creatinine_level	blood_urea_nitrogen	age	platelets_count
0%	33.0000	0.2000000	0.300000	3.5000	16.0000	22.7750
25%	91.0000	0.4081159	0.760000	12.5000	53.0000	147.5000
50%	109.0000	0.4823737	1.000000	19.0000	65.0000	191.0000
75%	134.0000	0.5759169	1.500000	31.5000	75.0000	241.0000
100%	288.0000	1.3556851	11.180000	119.5450	89.0000	571.2250
mean	116.1837	0.5058003	1.505818	25.5147	62.6866	200.9316

The summary statistics of the variables shown that there are numerical differences between the mean and median values of the variables, an indication of possible skewness. There are other interesting observations that can be made from the summary statistics. First, we note that individuals are aged from 16 years to 89 years old, where majority of the individuals are 53 years and above. Also, majority of individuals fall within the normal creatinine level of 0.74 to 1.35 mg/dL for healthy men and 0.59 to 1.04 mg/dL for healthy women (Mayo Clinic, 2022). A more comprehensive analysis of the distribution of the variables would be examined during the exploratory analysis. The summary statistics based on disease status are examined in the tables below.

Summary Statistics of Patients Diagnosed

	glucose_level	bilirubin	creatinine_level	blood_urea_nitrogen	age	platelets_count
0%	33.000	0.2053191	0.300000	3.50000	16.00000	22.775
25%	90.000	0.4331562	0.820000	14.50000	56.00000	150.000
50%	115.000	0.5140237	1.100000	23.00000	66.00000	191.000
75%	148.000	0.6161717	1.890000	38.00000	75.00000	244.000
100%	288.000	1.3556851	11.180000	119.54500	89.00000	571.225
mean	123.482	0.5384149	1.767105	29.67211	64.32023	203.421

Summary Statistics of Patients Not Diagnosed

	glucose_level	bilirubin	creatinine_level	blood_urea_nitrogen	age	platelets_count
0%	33.0000	0.2000000	0.300000	3.50000	16.00000	22.7750
25%	91.0000	0.3943914	0.710000	11.00000	51.00000	145.5000
50%	106.0000	0.4631811	0.990000	17.00000	64.00000	191.0000
75%	126.0000	0.5472936	1.300000	27.00000	75.00000	240.0000
100%	288.0000	1.3556851	11.180000	119.54500	89.00000	571.2250
mean	111.2843	0.4839057	1.330414	22.72379	61.58994	199.2604

From the tables, we note that the median and mean values of glucose level, bilirubin level, creatinine level, blood urea nitrogen and age of individuals diagnosed as diabetic are higher than those not diagnosed with diabetes.

2.3. Exploratory Data Analysis

In this section, we examine the distribution of the data using graphical representation. The paper explored bar plots, histogram, density plots and box plots to visually inspect how the data vary based on the outcome variable. First, we examine the outcome variable and its relation to the categorical variables of gender and region. *The code for the barplot of disease status is shown below. For brevity, the codes for similar barplots are concealed using chunk options*

```r
##Disease Status
##Plot bar plot on Outcome variable using ggplot
#the code allows for plotting in percentages.
##aesthetic set to disease_status(for plotting),
## geom_bar for barplot, y = (..count..)/sum(..count..)
## to calculate percentage (count/sum of count)
#geom_text to add percentage in plot; scale_y_continuous to set y to percentage,
## theme_light() to generate light/white background
#labs to set title, x and y labels
#theme(plot.title = element_text(face = "bold")) to bolden title text.
train %>% ggplot(aes(x = disease_status)) +
  geom_bar(aes(y = (..count..)/sum(..count..))) +
  geom_text(aes(y=((..count..)/sum(..count..)), label = scales::percent((..count..)/sum(..count..))),
            stat = "count", vjust = -0.25) +
  scale_y_continuous(labels = percent) +
  theme_light() +
  labs(title = "Figure 1. Disease Status in Percentage", y = "Percent", x = "Disease Status") +
  theme(plot.title = element_text(face = "bold"))
```

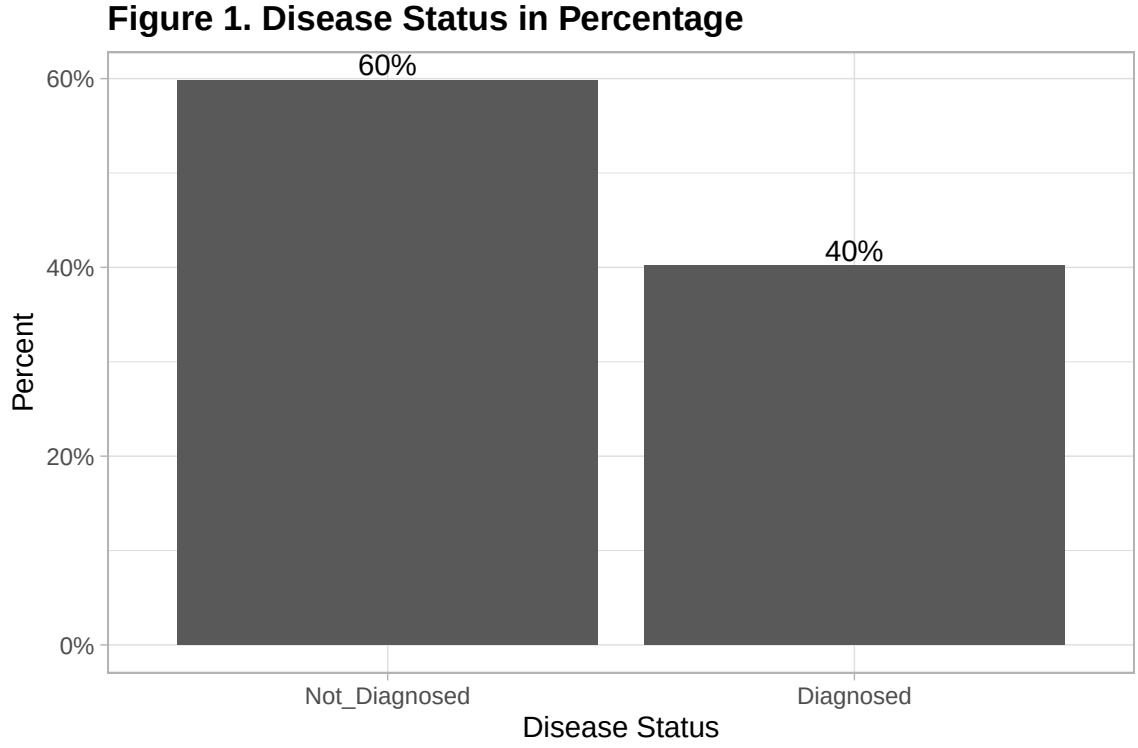


Figure 2. Gender in Percentage

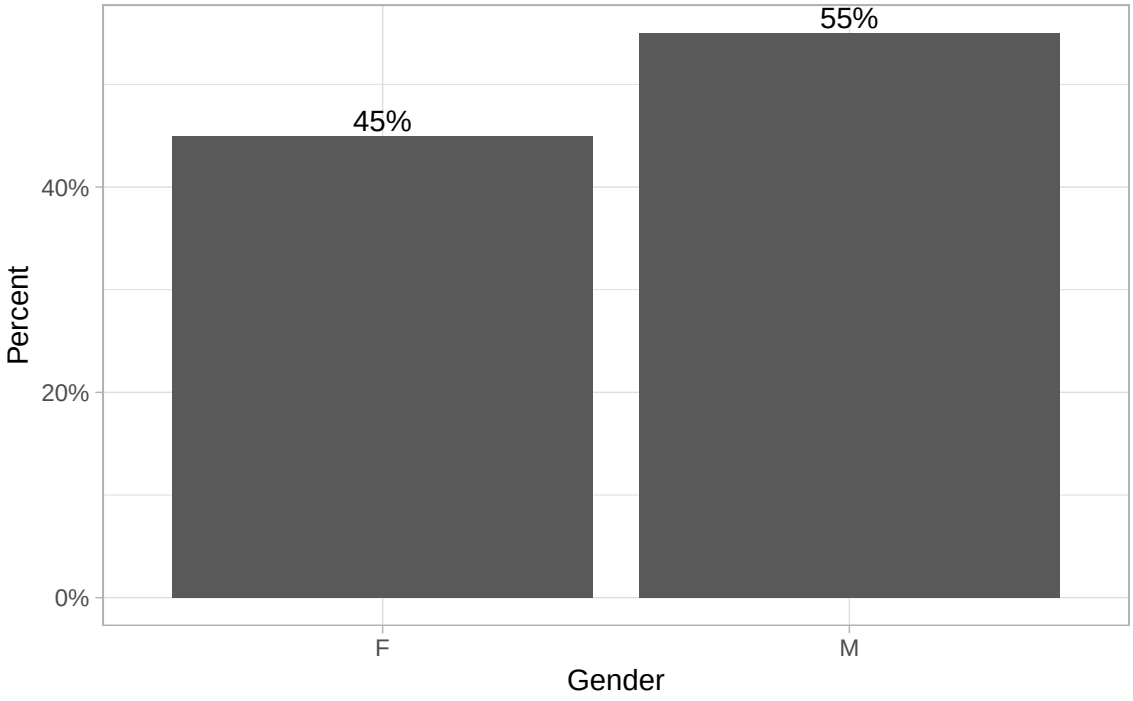


Figure 3. Region in Percentage

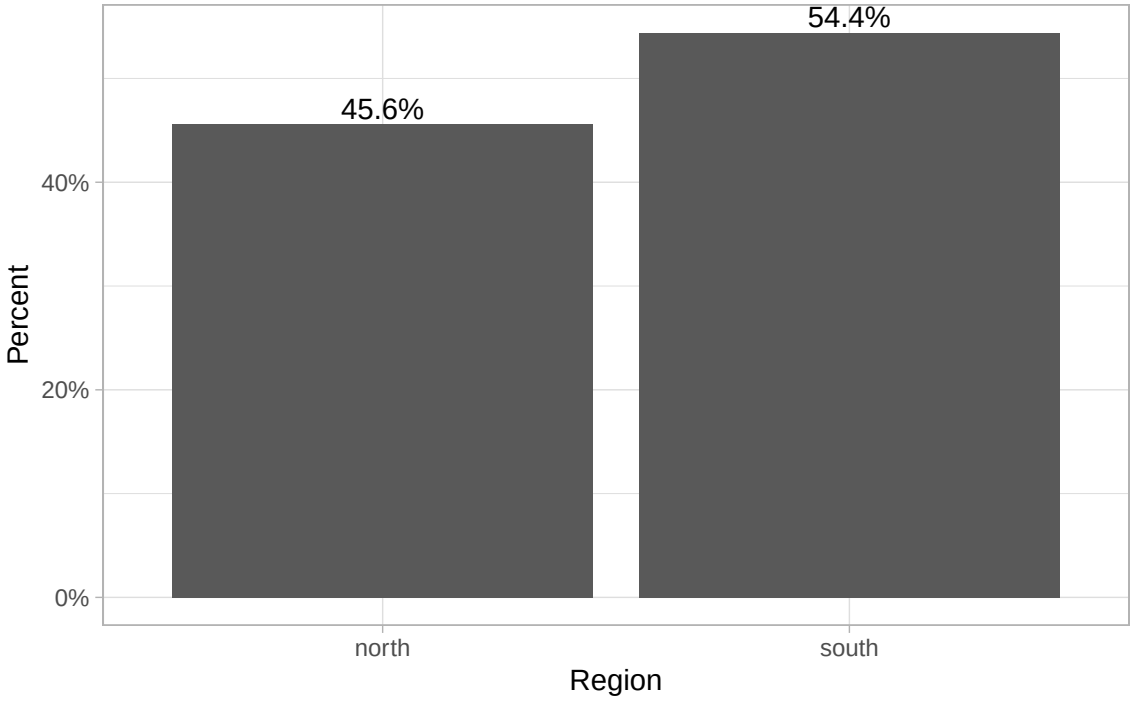


Figure 1 shows the percentage of disease status in the train set. We note that 60% of the patients were not diagnosed with the disease whereas the minority 40% were diagnosed. Figure 2 shows that majority of the individuals are males (55%) and Figure 3 also shows that majority of the individuals are from the Southern part of the country (54.4%).

The code for the barplot of disease status using facet wrap on gender is shown below. For brevity, the codes

for similar barplots are concealed using chunk options

```
##Using Facet Wrap Function (Gender on Disease Status)
#facet_wrap to generate plots by group (gender)
train %>% ggplot(aes(x = disease_status, fill = disease_status, y = gender)) +
  geom_bar(aes(y = (..count..)/sum(..count..)), position = "dodge") +
  geom_text(aes(y=((..count..)/sum(..count..)), label = scales::percent((..count..)/sum(..count..))),
            stat = "count", vjust = -0.25) +
  scale_y_continuous(labels = percent) +
  theme_light() +
  labs(title = "Figure 4. Percentage of Disease Status by Gender", y = "Percent", x = "Gender") +
  facet_wrap(~gender) +
  theme(plot.title = element_text(face = "bold"))
```

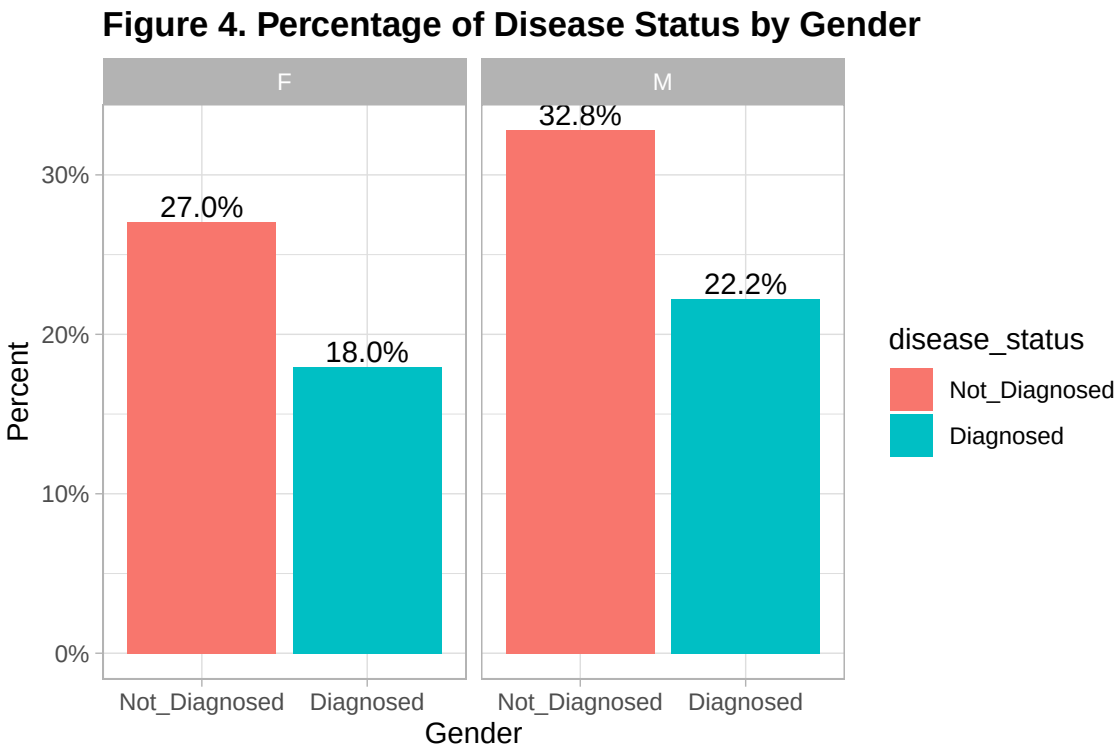


Figure 5. Percentage of Disease Status by Region

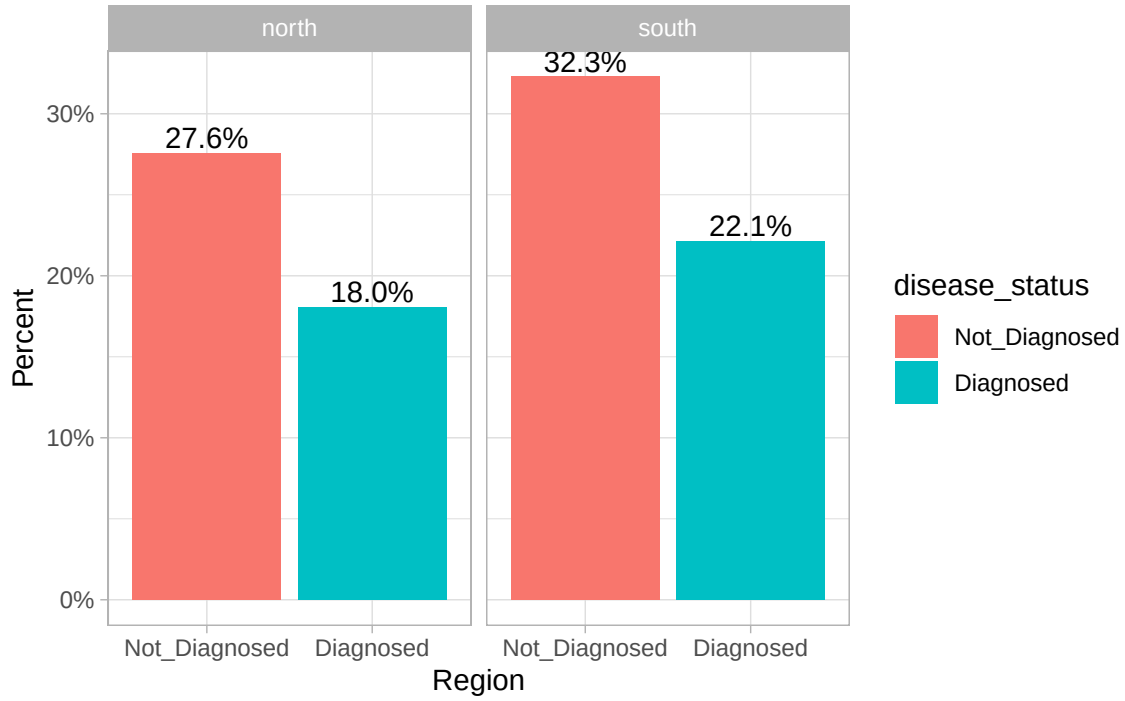
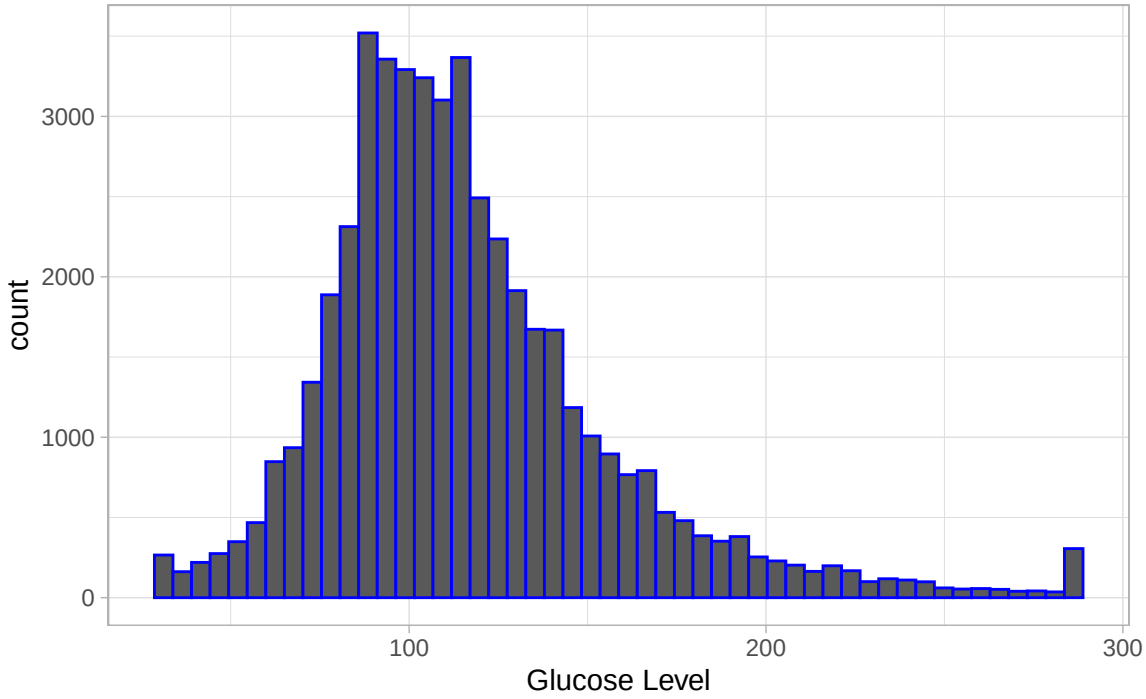


Figure 4 shows that in either gender, the percentage of individuals who are not diagnosed with the condition is more than those who are diagnosed with it. However, we note that for males, the difference in percentage between those who are not diagnosed and those who are diagnosed is about 10.6% whiles for females, the difference is about 9.0%. Figure 5 shows percentage of disease status by region; north and south. In both cases, individuals who have not been diagnosed with the disease are more than those who have been diagnosed with the disease. The difference for the Southern part of the country is about 10.2% whiles that for the Northern part of the country is about 9.6%.

The code for plotting histogram, density plot and box plot of glucose on disease status is shown below. For brevity, the codes for similar plots are concealed using chunk options

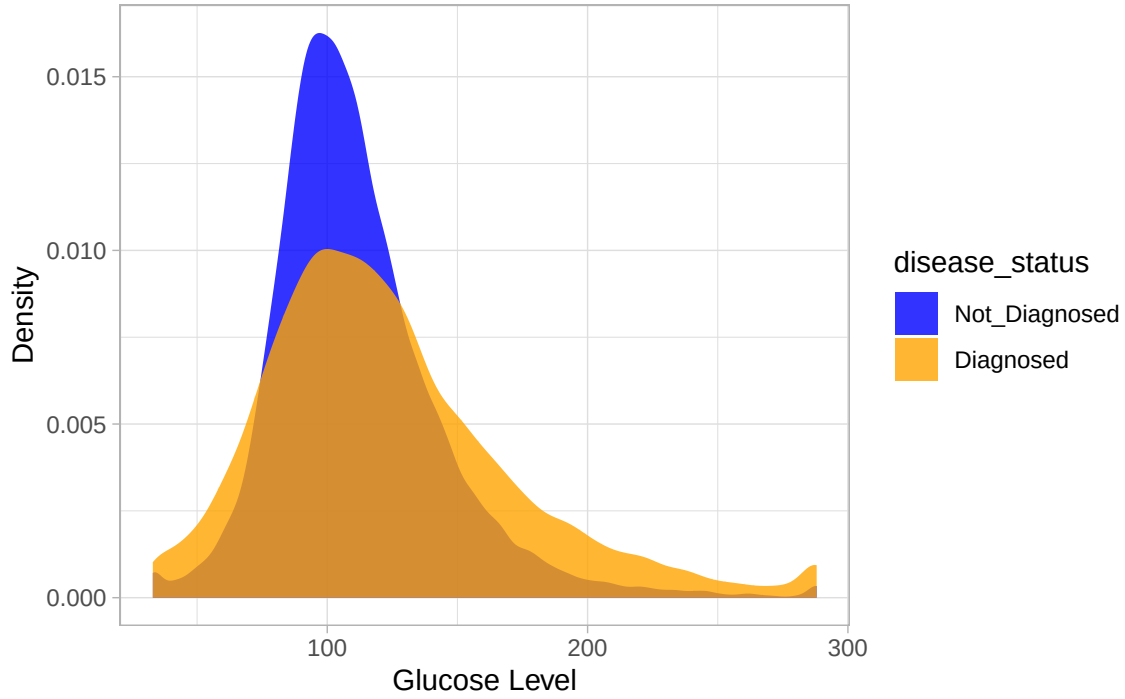
```
##Distribution of Glucose Level
##Histogram
# Plotting the Distribution of Glucose; using ggplot
##geom_histogram to generate histogram
train %>%
  ggplot(aes(x = glucose_level)) +
  geom_histogram(bins = 50, color = "blue") +
  theme_light() +
  ggtitle("Figure 6. Distribution of Glucose Level") +
  xlab("Glucose Level") +
  theme(plot.title = element_text(face = "bold"))
```

Figure 6. Distribution of Glucose Level



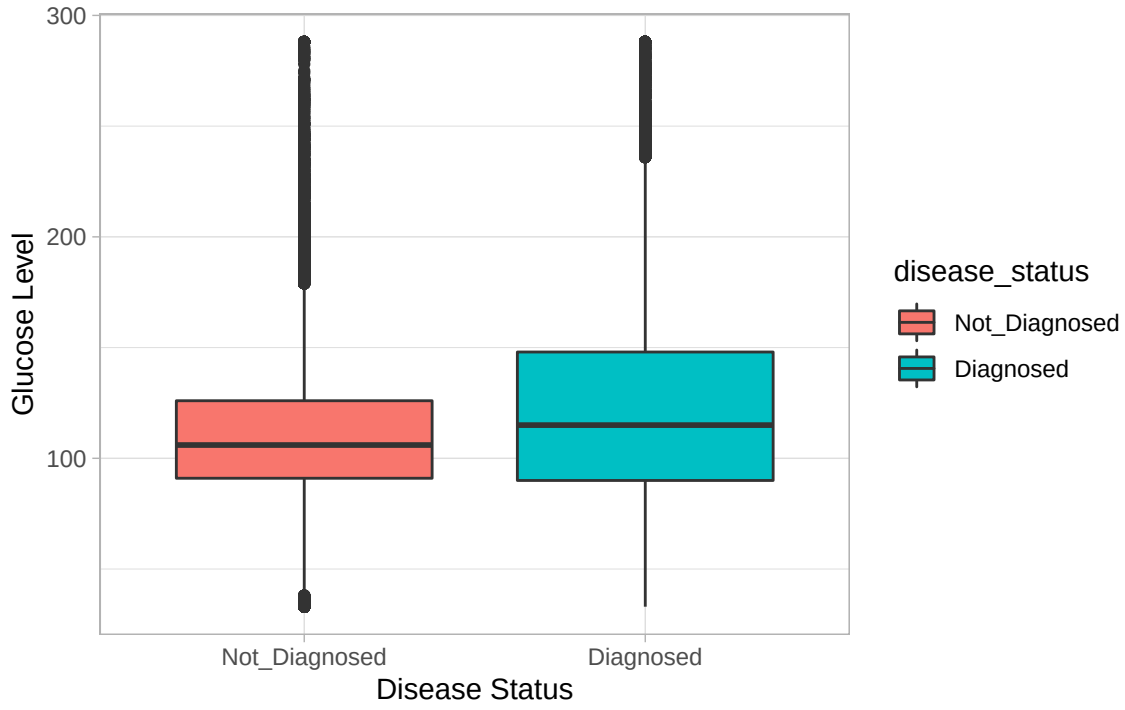

```
##Distribution of Glucose Level by Disease Status
# Using Density Plot
#geom_density to generate density plot
train %>%
  ggplot(aes(x = glucose_level, fill = disease_status)) +
  geom_density(alpha = 0.8, color = NA) +
  scale_fill_manual(values = c("blue", "orange")) +
  theme_light() +
  labs(title = "Figure 7. Density plot of Glucose Level on Disease Status",
       y = "Density", x = "Glucose Level") +
  theme(plot.title = element_text(face = "bold"))
```

Figure 7. Density plot of Glucose Level on Disease Status




```
##Distribution of Glucose Level by Disease Status
# Using Box plot
# geom_boxplot to generate box plot
train %>%
  ggplot(aes(x = disease_status, y = glucose_level, fill = disease_status)) +
  geom_boxplot() +
  theme_light() +
  labs(title = "Figure 8. Box plot of Glucose Level on Disease Status",
       y = "Glucose Level", x = "Disease Status") +
  theme(plot.title = element_text(face = "bold"))
```

Figure 8. Box plot of Glucose Level on Disease Status



The level of glucose is positively skewed (Figure 6). We note that the range of glucose level for the middle 50% of individual diagnosed with disease is wider than those not diagnosed with the disease (Figure 7 and Figure 8). The narrow inter-quartile range for individuals who are not diagnosed with the disease reveals outliers at both tails of the boxplot.

Figure 9. Distribution of Bilirubin

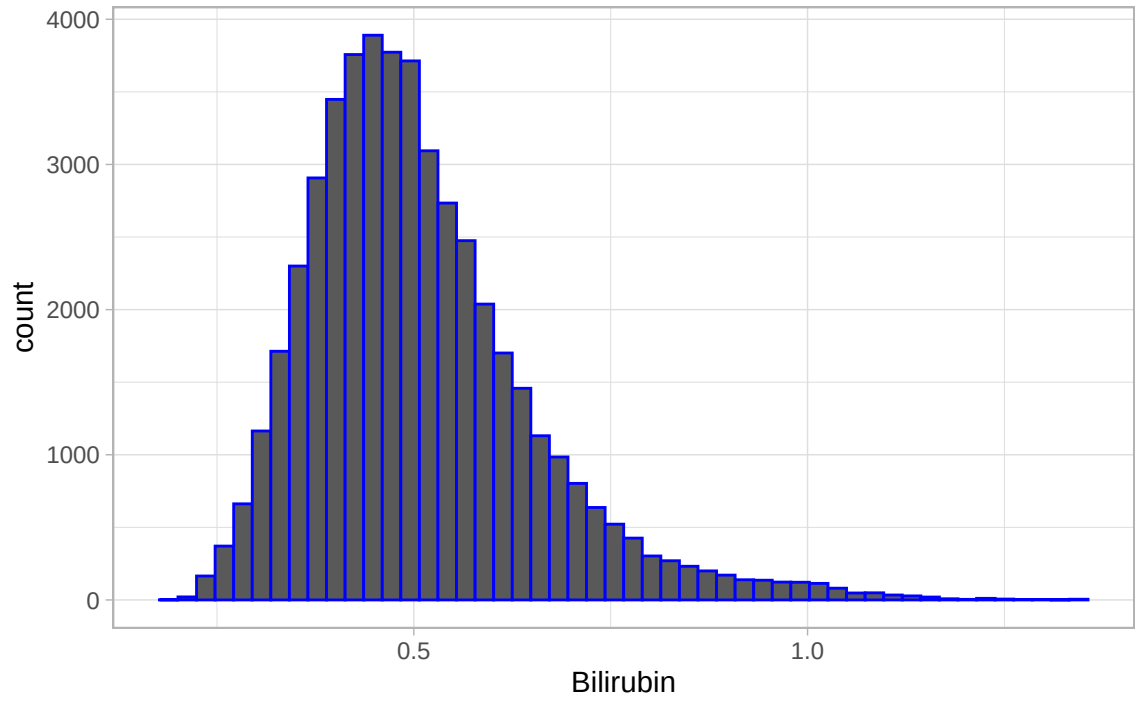


Figure 10. Density plot of Bilirubin on Disease Status

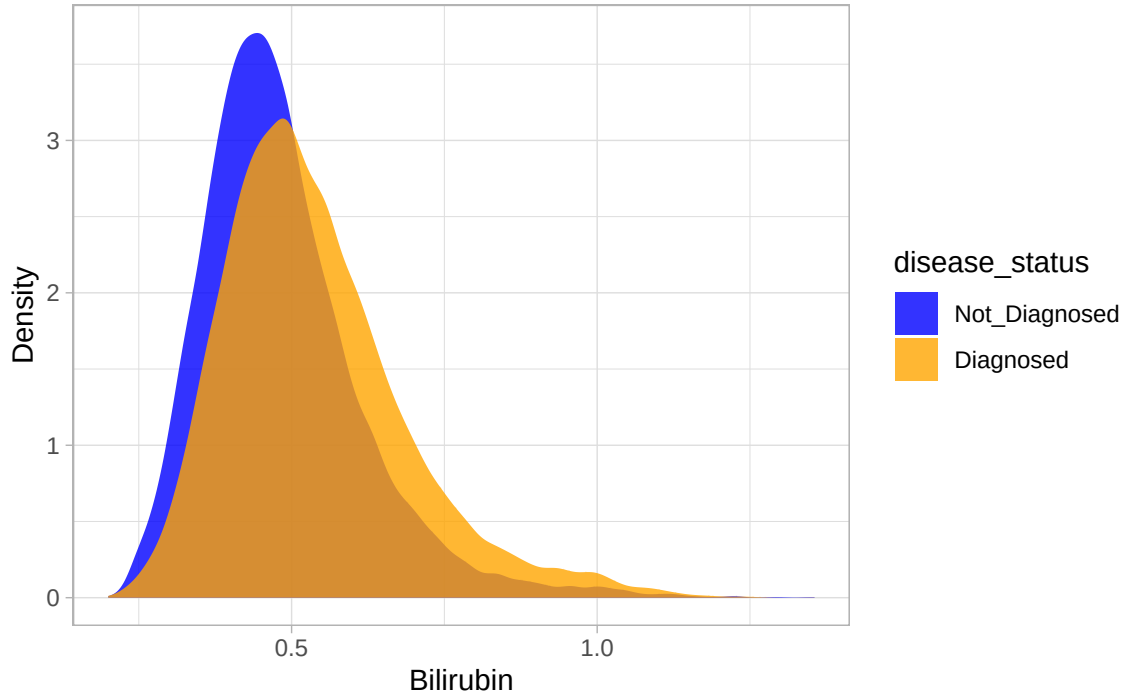
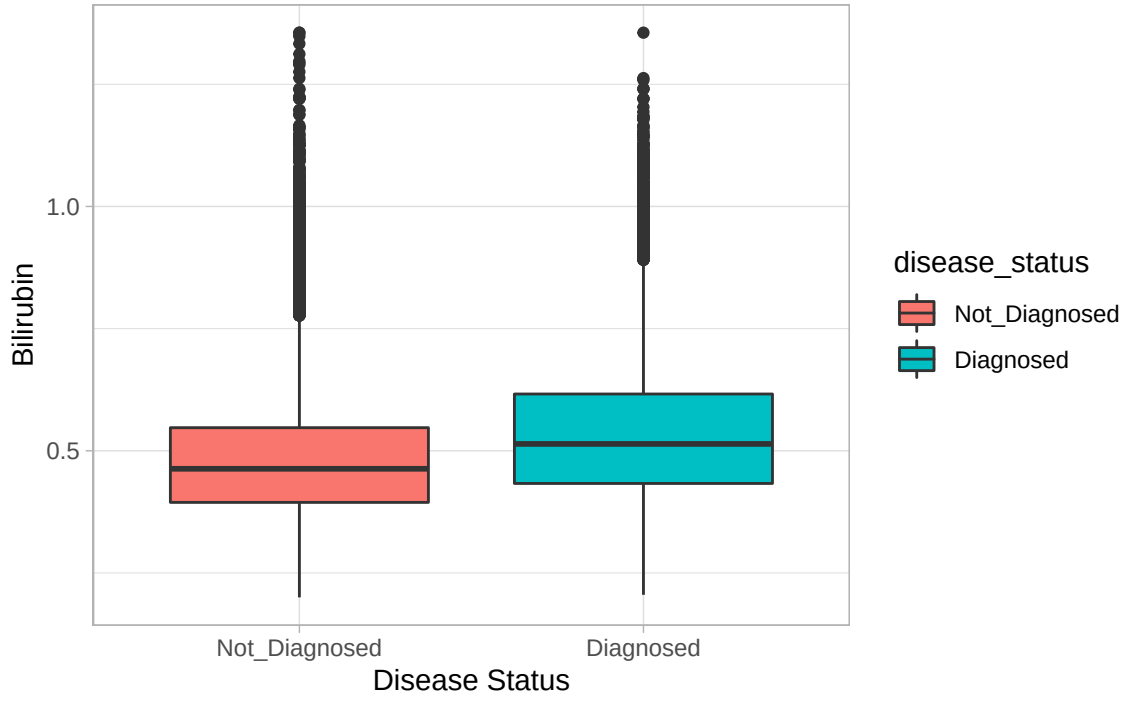


Figure 11. Box plot of bilirubin on Disease Status



The plot of the level of bilirubin is positively skewed (Figure 9). Further, we note that many individuals diagnosed with the disease have higher bilirubin level than those not diagnosed with the disease (Figure 10 and Figure 11). This slight difference in distribution between these two(2) groups would be important in machine learning model development.

Figure 12. Distribution of Creatinine Level

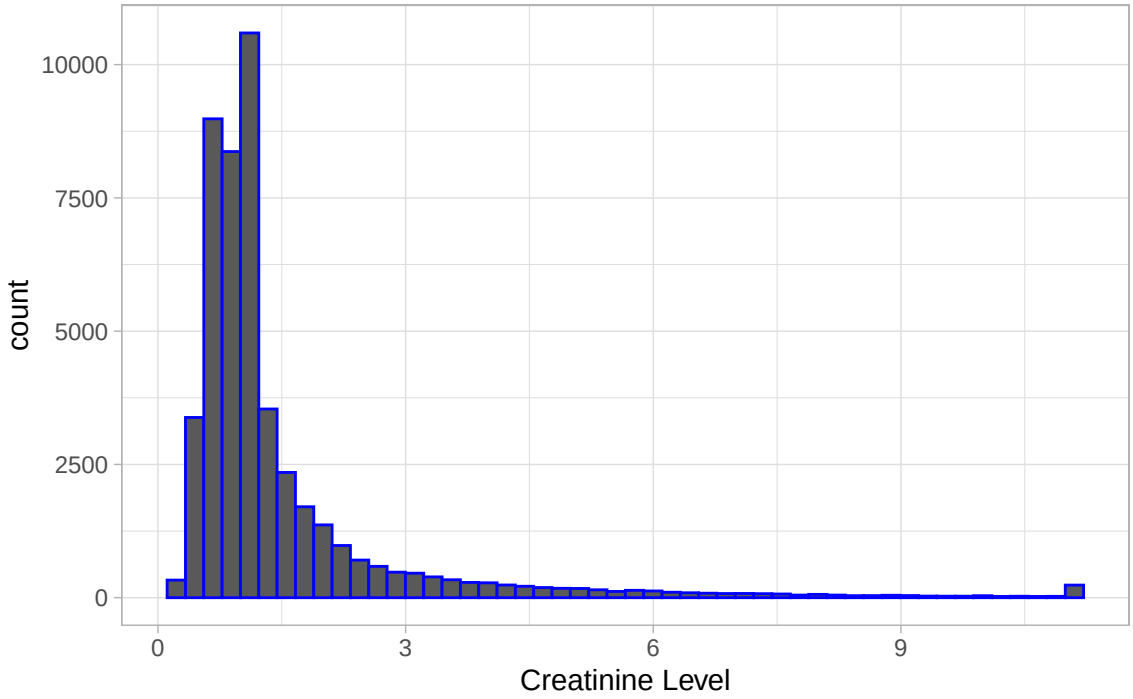


Figure 13. Density plot of Creatinine Level on Disease Status

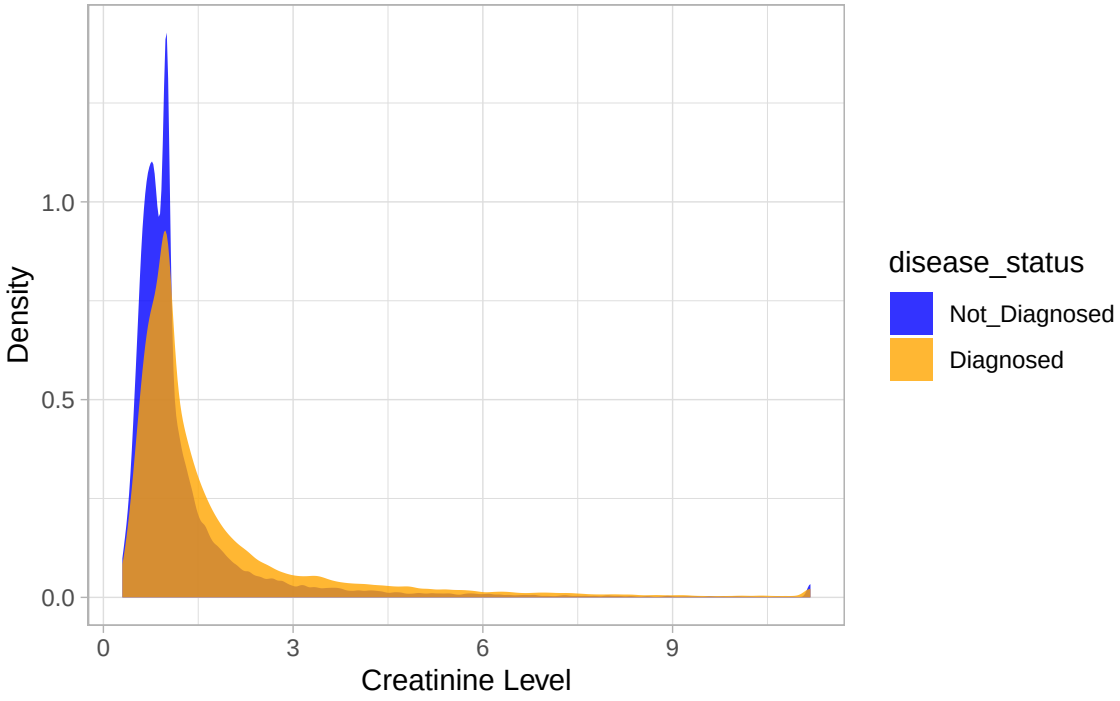
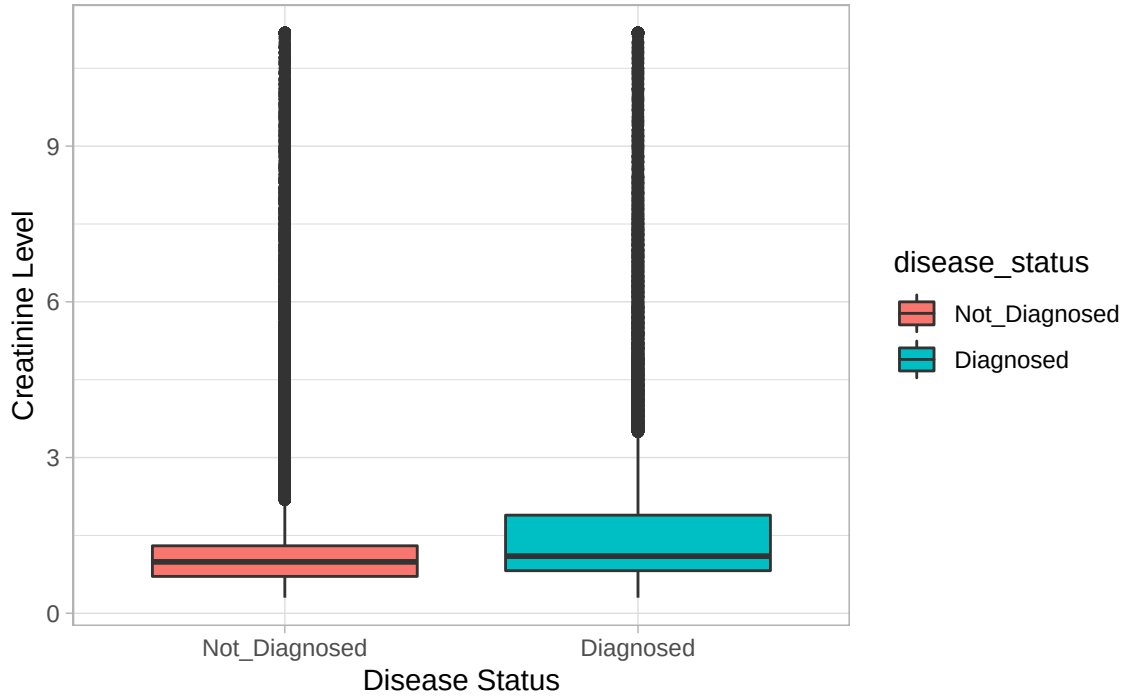


Figure 14. Box plot of Creatinine Level on Disease Status



The plot of the creatinine level shows very high positive skewness (Figure 12). We note from the density plot (Figure 13) and box plot (Figure 14) that there exist a wider interquartile range for individuals who were diagnosed with the disease than those who were not diagnosed. It is therefore expected that machine learning algorithm such as the decision tree algorithm would likely differentiate nodes based on creatinine level. More on the decision tree algorithm would be examined in later sections.

Figure 15. Distribution of Blood Urea Nitrogen

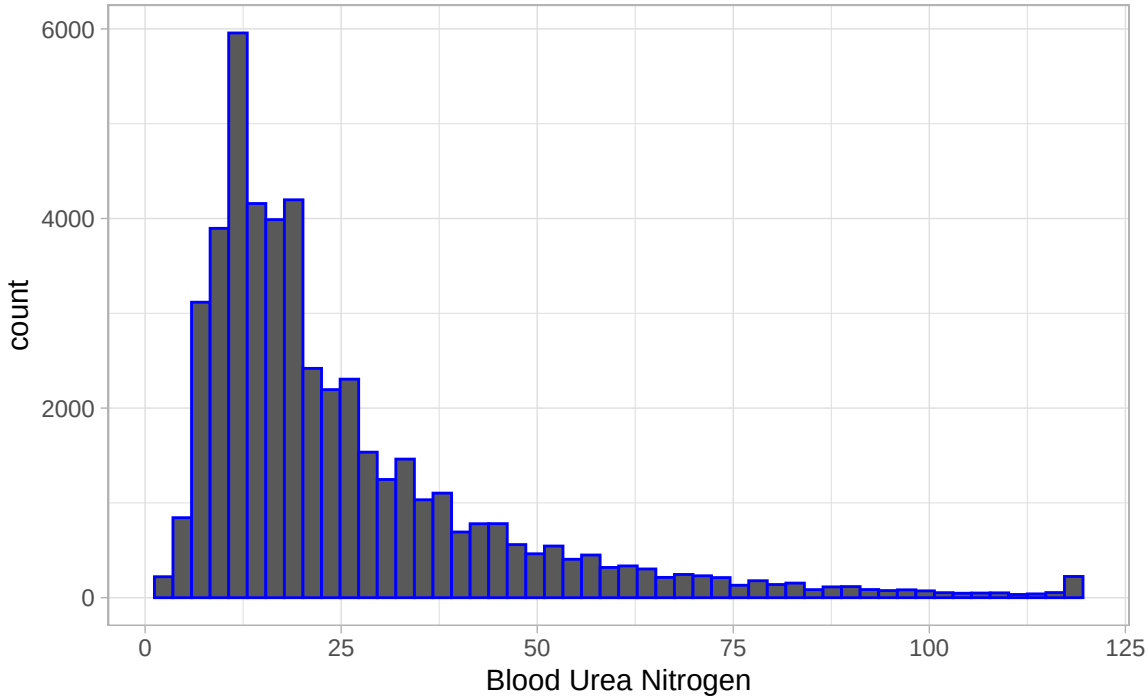


Figure 16. Density plot Blood Urea Nitrogen on Disease Status

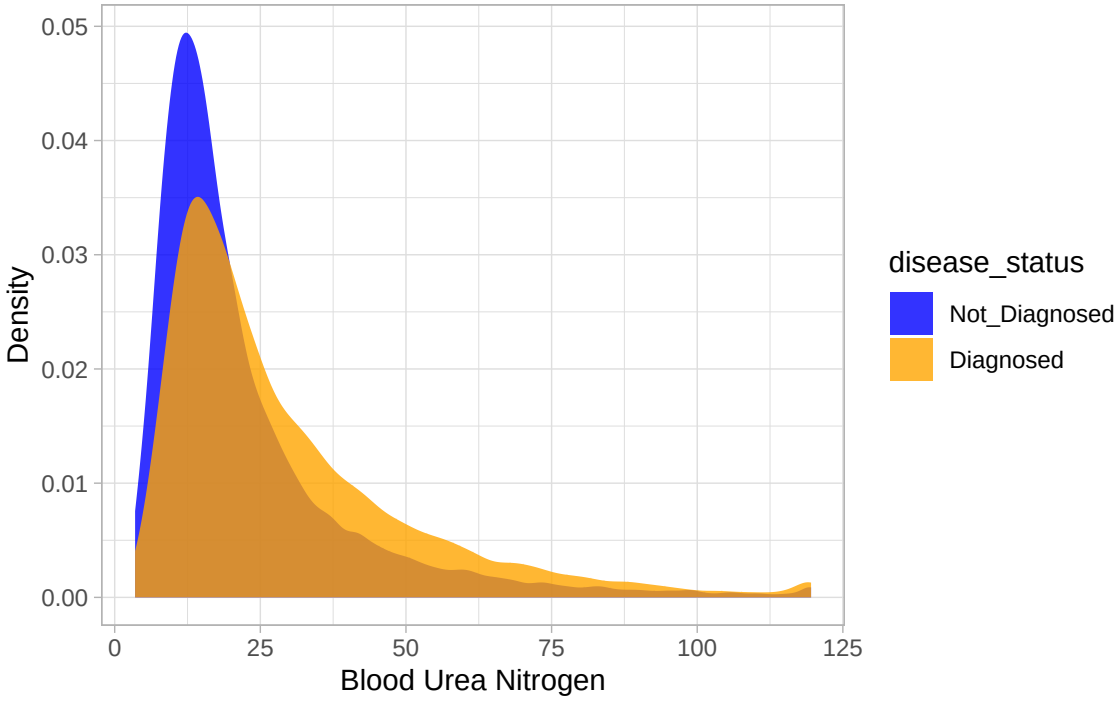
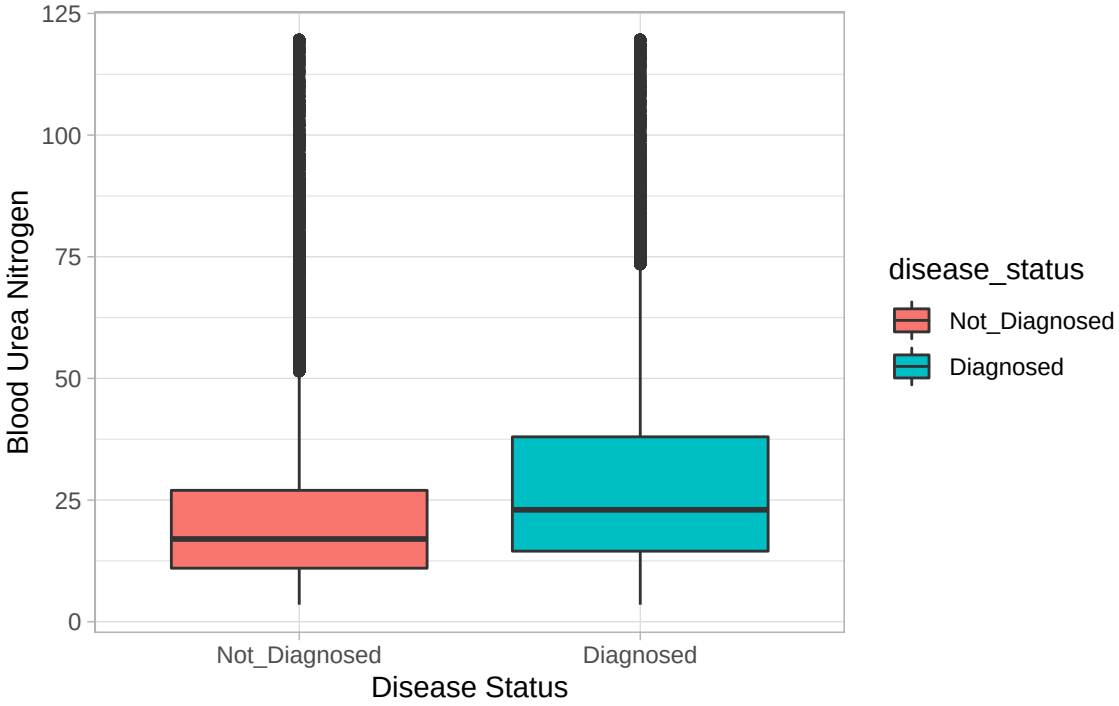


Figure 17. Box plot of Blood Urea Nitrogen on Disease Status



The blood urea nitrogen of patients is also positively skewed (Figure 16). As noted by the summary statistics, more individuals diagnosed with the disease have greater levels of blood urea nitrogen than those not diagnosed with the disease (Figure 17 and Figure 18). Numerically, individuals not diagnosed are more than those diagnosed. Therefore, it is likely there would be more weight attached to predicting individuals within the narrow inter-quartile range as not being diagnosed of the disease.

Figure 18. Distribution of Age

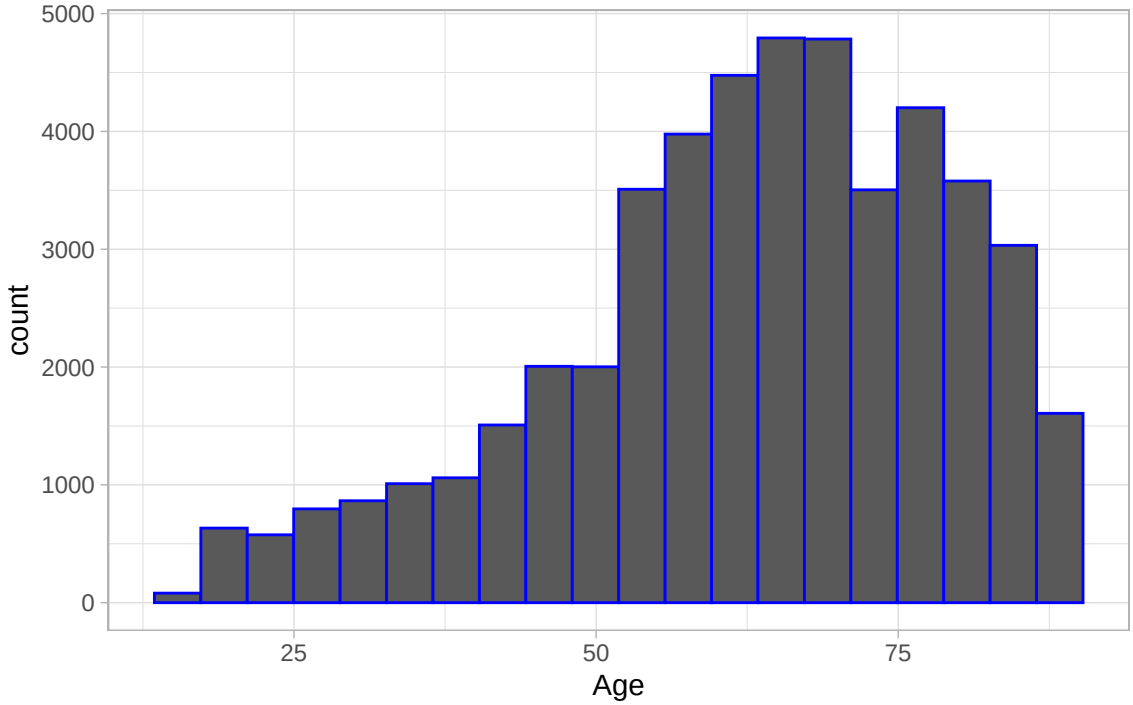


Figure 19. Density plot of Age on Disease Status

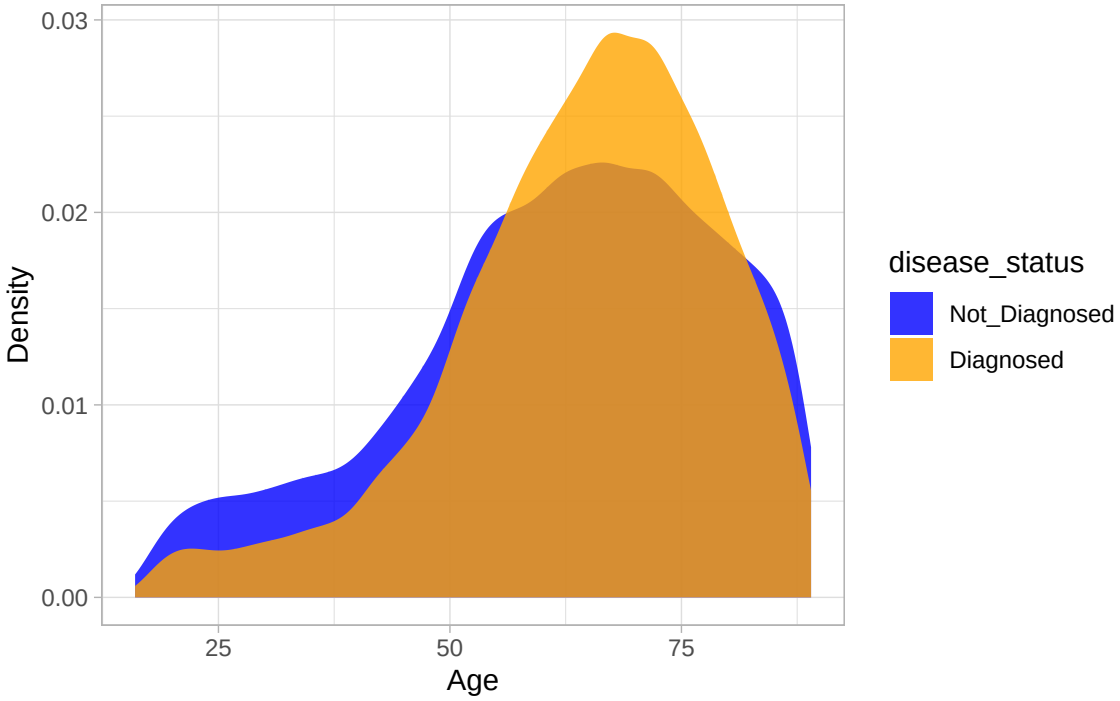
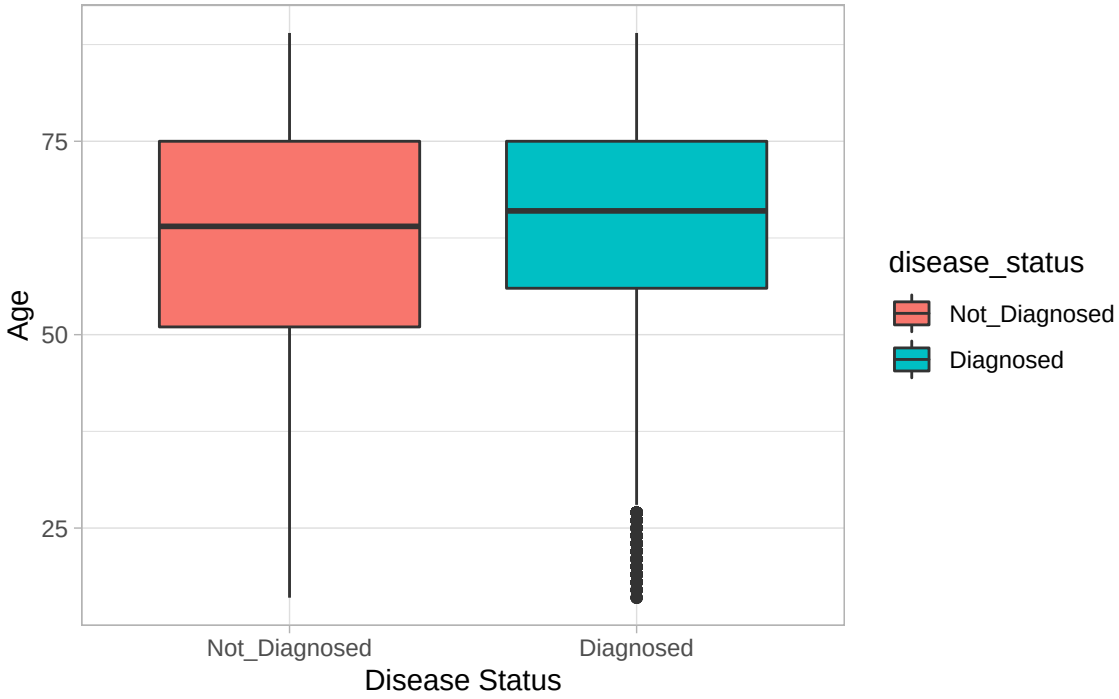


Figure 20. Box plot of Age on Disease Status



The plot of the distribution of age shows negative skewness (Figure 18), different from the other variables. Again, the inter-quartile range of the diagnosed group is narrower than that of the not diagnosed group (Figure 19 and Figure 20). Whiles there are no obvious outliers within the not diagnosed group, individuals below the age of 27 years are considered outliers within the diagnosed group.

Figure 21. Distribution of Platelets Count

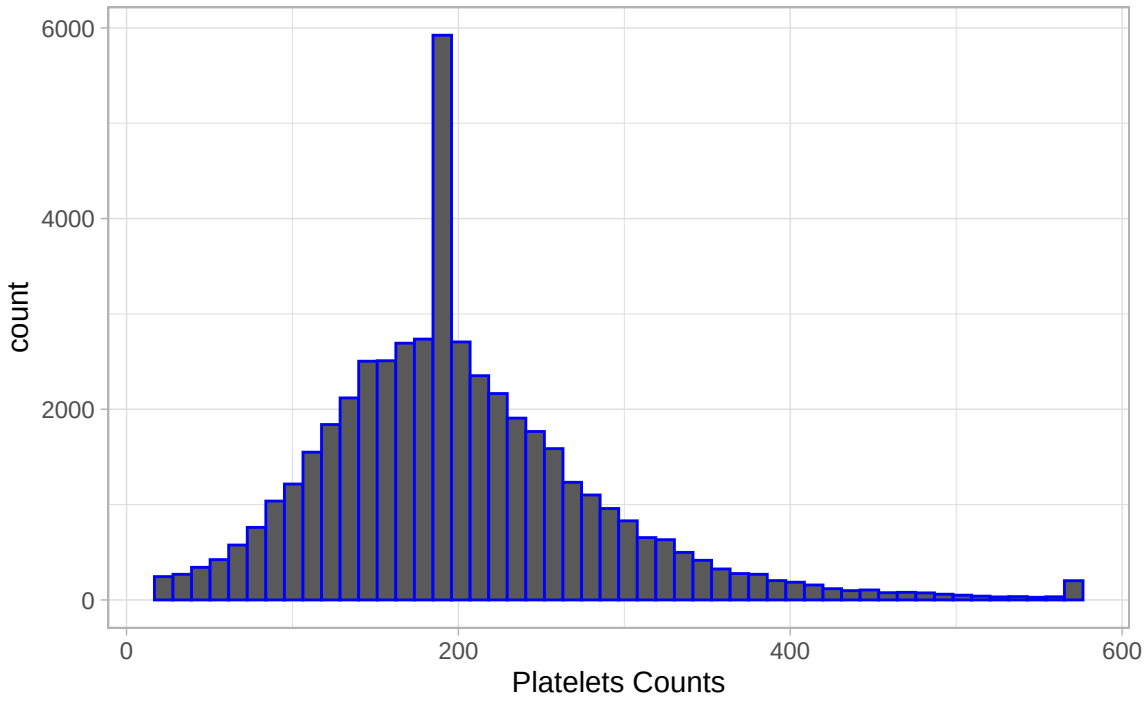


Figure 22. Density plot of Platelets Count on Disease Status

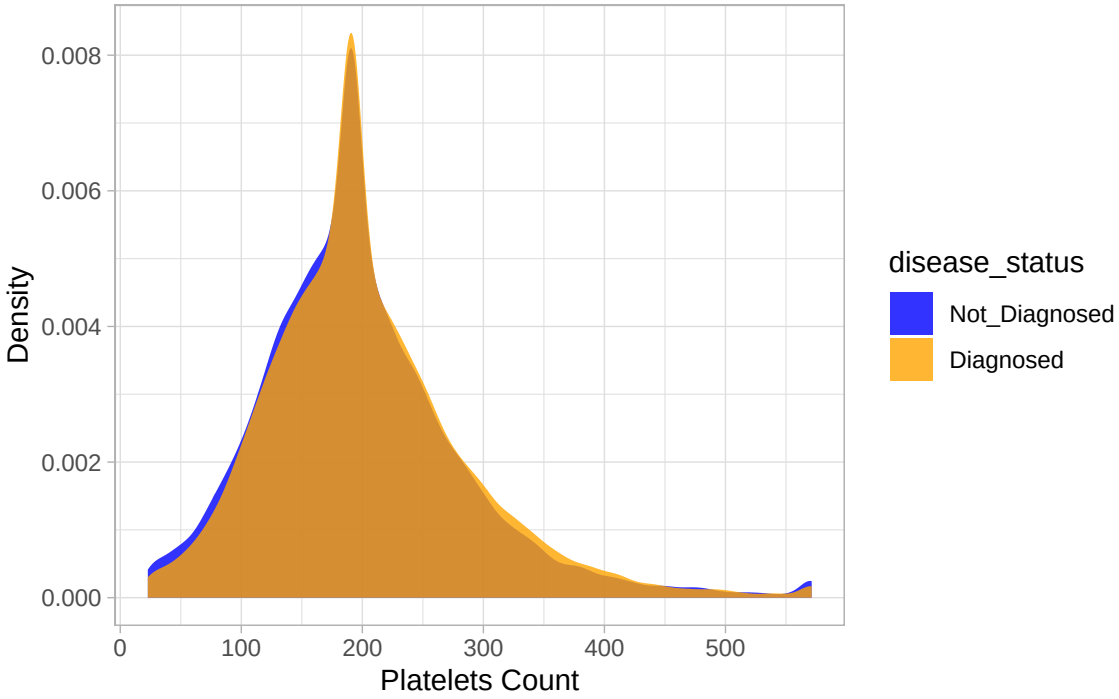
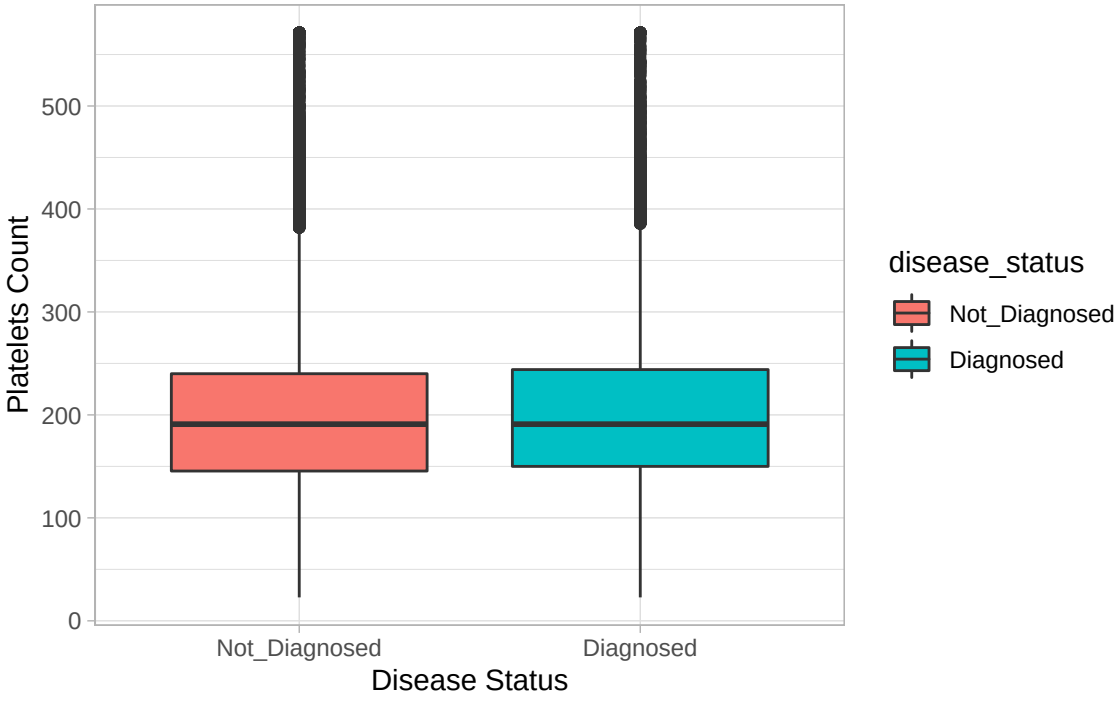


Figure 23. Box plot of Platelets Count on Disease Status



The plot of platelet shows an obvious unimodal group within a positively skewed distribution (Figure 21). The density plot does not show much difference between individuals diagnosed with the disease and those not diagnosed with the disease (Figure 22). However, the box plot shows that the inter-quartile range of those diagnosed with the disease is slightly higher than those not diagnosed with the disease (Figure 23).

2.4. Pre-processing

2.4.1. Min-Max Scaling

The first pre-processing method employed in the study is scaling. Specifically, this report used the Min-Max Scaling. Scaling is usually important in machine learning to reduce the potential bias inherent that can be caused by the range of specific input features, particularly their effects on distance-based algorithms. Scaling is therefore crucial because it compresses the range of input features into common boundaries, making each feature contribute almost similarly to final model variation or distance. Though non-tree-based algorithms are not dependent on scaling, it is necessary to use the same state of the data on all algorithms to ensure uniformity in comparison. The MinMax scaling used in the report compressed each continuous input feature within the boundary of 0 and 1 inclusive. The formula for the Min-Max scale is given below:

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

where x is the observation of a specified variable , x_{min} is the minimum value of the variable and x_{max} is the maximum value of the variable.

From the formula above, we note that the minimum and maximum values of the input features would contribute much to the scaling. However, different values within the minimum and maximum values are uniquely compressed and thus, the Min-Max is likely to produce superior results, particularly in distance-based algorithms.

2.4.2. Binary Encoding

There were two(2) categorical variables in the dataset; gender and Region. Both variables had two(2) categories each. Therefore, the presence of one category specify the absence of the other category and vice versa. Therefore, gender and region were converted into binary dummy variable, where 1 represent the presence of one category and 0 represent the presence of the other category. Binary encoding is necessary because the supervised learning models are applied to numerical input variables.

2.5. Performance Metrics

In this study, we examine the performance of the various algorithms using Accuracy, Balanced Accuracy and F_1-score.

2.5.1. Accuracy

Accuracy measures the percentage of correct predictions out of the total number of predictions. In our case, it shows the capability of the algorithm in rightly classifying whether an individual was diagnosed with the disease or not. The accuracy metric is given by the formula below:

$$Accuracy = \frac{TruePositive + TrueNegative}{TruePositive + FalsePositive + TrueNegative + FalseNegative}$$

2.5.2. Balanced Accuracy

The balanced accuracy metric measures the performance of the model by considering the proportion of each class. It is computed as the arithmetic average of specificity and sensitivity. The balanced accuracy provides better performance than the overall accuracy metric within an imbalanced dataset. Balanced accuracy is given by the formula below:

$$BalancedAccuracy = \frac{Sensitivity + Specificity}{2}$$

The balanced accuracy is a composite index because it combines two(2) metrics: Sensitivity and Specificity. Sensitivity measures the proportion of true positive to the sum of true positive and false negative whiles specificity measures the proportion of true negative to the sum of true negative and false positive.

2.5.3. F1-Score

The F_1-score is a composite metric that is computed as the harmonic mean of recall and precision. F_1-score is an important metric where both recall and precision are equally crucial in a classification problem. In disease diagnostics, recall tends to be more important than precision because it is more costly to misdiagnose a healthy individual as unhealthy than the other way round. However, in this study, we are more interested in prediction accuracy since the final model would not be the golden standard for actual diagnosis. The F1-score is given by the formula below:

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

2.6. Machine Learning Models

In this section, we would examine the machine learning models used in the report. The models to be considered include the baseline (guessing), logistics regression, K-nearest neighbor (KNN) : a typical distance-based model, QDA : a generative model, decision tree and random forest : both tree-based models.

2.6.1. Cross-Validation

In estimating the models, 5 fold cross-validations were used. Cross-validation provides better estimation of the performance of our model. In the 5-fold cross validation, the training set is split into 5 smaller sets. In each iteration, the model uses 4 (k-1) of the folds for training. The caret package was used to implement the cross-validation in applicable models. The cross-validation function used in the model estimation is provided by the code below:

```
#Set default cross_validation for models
control = trainControl(method = "cv",
                       number = 5,
                       classProbs = TRUE,
                       summaryFunction = twoClassSummary,
                       savePredictions = T)
#5 fold cross-validation; twoClassSummary for binary classification;
# savePredictions = T to obtain predictions
#classProbs = TRUE to save probabilities for further analysis if need be.
```

2.6.2.1. Model 1: Baseline Model (Guessing the Prediction) The baseline model for this report is guessing the prediction of the test set. In other words, how well do one perform if one guess the disease status? The guessing algorithm ignores the variations of the input variables but only took into consideration the probability of individuals actually diagnosed within the training set. The code for generating the baseline model is given below.

```
#Model 1: Guessing
 # set random seed for reproducibility for R 3.6 or later
set.seed(105, sample.kind = "Rounding")
p <- mean(train$disease_status == "Diagnosed") #Find probability of guessing Diagnosed
n <- length(test$disease_status) # Find the length of the test set
y_hat <- sample(c("Not_Diagnosed", "Diagnosed"),
                n,
                replace = TRUE,
                prob=c(1-p, p)) %>%
  factor(levels = levels(test$disease_status)) ##Generate prediction in the guessing model
cm_guessing <- confusionMatrix(y_hat,
                               test$disease_status,
                               positive = 'Diagnosed',
                               mode = 'everything')  ##Generate confusion Matrix of the model
#The relevant metrics for this report are accuracy, balanced accuracy and F1-Score
# accuracy = cm_guessing$overall['Accuracy']
# balanced_accuracy = cm_guessing$byClass['Balanced Accuracy']
# F1_score = cm_guessing$byClass['F1']
```

The probability associated with individuals diagnosed with the disease is 0.4017. Predicting the outcome using the baseline model is obtained by sampling the number of observations of the test set, 12000 with replacement, considering probabilities from the train set. The performance of the model are as follows: Accuracy = 0.5276; Balanced Accuracy = 0.5081 and $F_1 = 0.4103$

2.6.2.2. Model 2: Logistics Regression Logistics regression solves the classification problem by estimating the probability of an event occurring or not occurring. Thus, as required the dependent variable should have binary outcome; in our case, the binary outcome is whether the individual has been diagnosed of the disease or not. The logistics regression uses the cut-off of 0.5; where one class is predicted when the probability is above 0.5 and the other class is predicted when the probability is below 0.5.

```
# Model 2: Logistics Regression
## Fit the logistics regression using the caret model
getModelInfo("glm")$glm$parameters  # Ascertain whether there are parameters to be tuned.
```

```
##   parameter     class     label
## 1 parameter character parameter
```

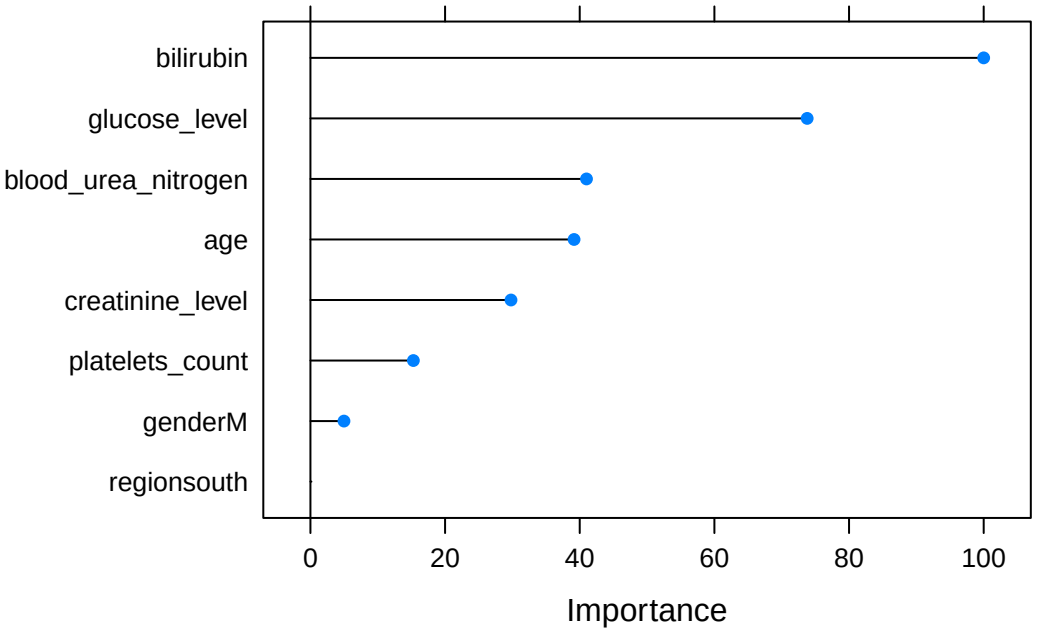
```
# there are no parameters to tune
# set random seed for reproducibility for R 3.6 or later
set.seed(105, sample.kind = "Rounding")
train_lr <- train(disease_status ~ .,
                  data = train,
                  method = "glm",
                  family = "binomial",
                  trControl = control) #Fit the model

varImp(train_lr)
```

```
## glm variable importance
##
##                     Overall
```

```
## bilirubin           100.000
## glucose_level        73.783
## blood_urea_nitrogen  41.006
## age                  39.159
## creatinine_level     29.792
## platelets_count      15.291
## genderM               4.982
## regionsouth           0.000
```

```
plot(varImp(train_lr), main = "Figure 24. Plot of Variable Importance of Logistics Model")
```

Figure 24. Plot of Variable Importance of Logistics Model



The variable importance of the logistics regression shows that the top 3 predictors of disease diagnosis are bilirubin, glucose level and blood urea nitrogen. Further, we note that the region variable has virtually no predictive power. The code for obtaining the statistical effects of the feature variables on the outcome variable is shown below.

```
##Summary Results of coefficients
summary(train_lr)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3743  -0.9629  -0.7681   1.2105   2.1457
##
## Coefficients:
```

```
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -2.65823    0.05311 -50.048  < 2e-16 ***
## glucose_level        1.66628    0.06270  26.574  < 2e-16 ***
## bilirubin            2.90325    0.08199  35.408  < 2e-16 ***
## creatinine_level     1.06328    0.09048  11.751  < 2e-16 ***
## blood_urea_nitrogen  1.18967    0.07661  15.530  < 2e-16 ***
## age                  0.71468    0.04794  14.907  < 2e-16 ***
## platelets_count      0.43788    0.06379   6.865 6.66e-12 ***
## genderM             -0.06722    0.01982  -3.391 0.000696 ***
## regionsouth          0.03349    0.01956   1.712 0.086818 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 64673  on 47999  degrees of freedom
## Residual deviance: 60682  on 47991  degrees of freedom
## AIC: 60700
##
## Number of Fisher Scoring iterations: 4
```

The summary results of the logistics regression shows that all things being equal, the negative coefficient of genderM indicates that males have lower statistical tendency of being diagnosed with diabetes than females. Again, all feature variables apart from region are statistically significant at 1% level of significance, confirming the little to no importance of region in determining disease status.

```
##Prediction and confusion Matrix
lr_preds <- predict(train_lr, test) # Obtain prediction
cm_lr <- confusionMatrix(lr_preds,
                         test$disease_status,
                         positive = 'Diagnosed',
                         mode = 'everything') #Store performance results
#The relevant metrics for this report are accuracy, balanced accuracy and F1-Score
# accuracy = cm_lr$overall['Accuracy']
# balanced_accuracy = cm_lr$byClass['Balanced Accuracy']
# F1_score = cm_lr$byClass['F1']
```

The logistics regression had the following performance metrics: Accuracy = 0.6474; Balanced Accuracy = 0.596 and $F_1 = 0.4328$

2.6.2.3. KNN The K-nearest neighbour is a typical distance-based model that would employ in the analysis. As with every distance-based model, knn is affected by the range of the feature variables and thus was one of the main reasons for the pre-processing scaling technique. The knn model used in this study employed the Euclidean distance metric in establishing neighbourhoods. Neighbourhoods in the knn model are created by minimizing the distance between similar observations whiles maximizing the distance between dissimilar observations in relation to the outcome variable. Members of same neighbourhood vote to indicate disease status. The knn parameter to be tuned was the number of neighbours, k.

```
##Model 3: KNN
getModelInfo("knn")$knn$parameters #Ascertain whether there are parameters to be tuned.
```

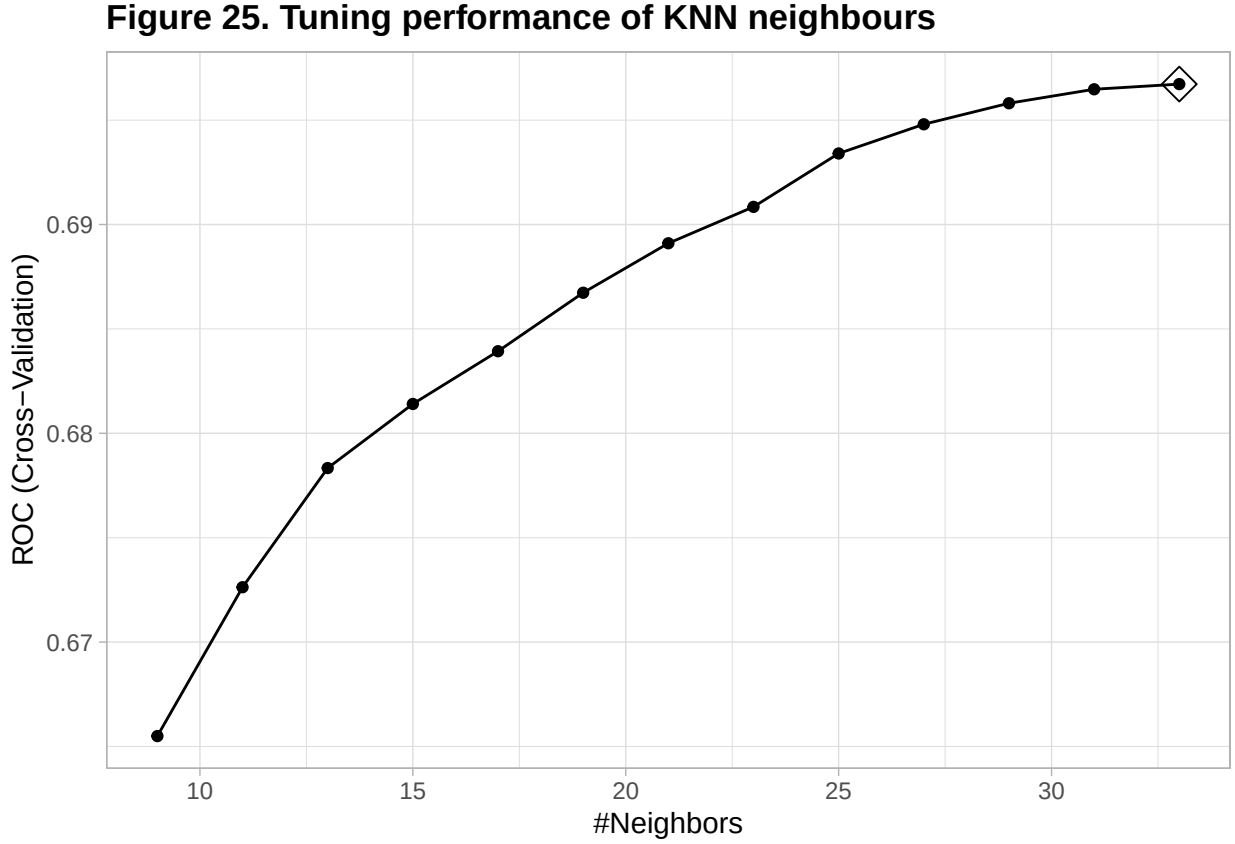
```
##   parameter   class      label
## 1         k numeric #Neighbors
```

```
#the parameter to be tuned is number of neighbours (k)

set.seed(105, sample.kind = "Rounding") # set random seed for reproducibility for R 3.6 or later
train_knn <- train(disease_status ~ .,
                   data = train,
                   method = "knn",
                   trControl = control,
                   tuneGrid = data.frame(k = seq(9, 33, 2))) #Fit the model with tuning of k

train_knn$bestTune  # the best tuned knn
```

```
##     k
## 13 33
```

We note that the best model had 33 neighbours. The plot of the choice of neighbours for knn is shown below. From the plot, we note that as the number of neighbors increases, performance increases but at a decreasing rate. Thus, further increases may add little to performance at an expense of probably adding noise.

```
ggplot(train_knn, highlight = TRUE) +
  theme_light() +
  labs(title = "Figure 25. Tuning performance of KNN neighbours") +
  theme(plot.title = element_text(face = "bold"))
```



The performance of the model is obtained from the code below.

```
knn_preds <- predict(train_knn, test) # Obtain prediction
cm_knn <- confusionMatrix(knn_preds,
                          test$disease_status,
                          positive = 'Diagnosed',
                          mode = 'everything') #Generate confusion Matrix of the model
#The relevant metrics for this report are accuracy, balanced accuracy and F1-Score
# accuracy = cm_knn$overall['Accuracy']
# balanced_accuracy = cm_knn$byClass['Balanced Accuracy']
# F1_score = cm_knn$byClass['F1']
```

The KNN model had the following performance metrics: Accuracy = 0.6601; Balanced Accuracy = 0.622 and $F_1 = 0.5032$

2.6.2.4. Model 4: Generative Model - QDA In this section, we examine the effect of the Quadratic Discriminant Analysis (QDA) model on our training data. The QDA is an example of generative models. Other generative models include Linear Discriminant Analysis (LDA) and Naïve Bayes. The Naïve Bayes is the fundamental generative model based on the Bayes' theorem. The LDA model allows feature covariance matrices for classes to be the same whiles the QDA allows for differences in feature covariance matrices. Thus, this flexibility of QDA over LDA also posses a drawback where there exists high dimensionality. There are no parameters to be tuned in the QDA estimation. From the data exploration, the features do not tend to follow a normal distribution. It is also clear that the dummy binary features are not normally distributed. That notwithstanding, we estimate the QDA for comparison with the baseline and other models.

The code for fitting the QDA is shown below.

```
##Model 4: QDA (Generative Model)
getModelInfo("qda")$qda$parameters  #Ascertain whether there are parameters to be tuned.
```

```
##   parameter     class     label
## 1 parameter character parameter
```

```
#there are no parameters to be tuned.

set.seed(105, sample.kind = "Rounding") # set random seed for reproducibility for R 3.6 or later
train_qda <- train(disease_status ~ .,
                   method = "qda",
                   trControl = control,
                   data = train) # fit the model

#plot the model showing the best prediction
qda_preds <- predict(train_qda, test) # Obtain prediction
cm_qda <- confusionMatrix(qda_preds,
                          test$disease_status,
                          positive = 'Diagnosed',
                          mode = 'everything') #Generate confusion Matrix of the model
#The relevant metrics for this report are accuracy, balanced accuracy and F1-Score
# accuracy = cm_qda$overall['Accuracy']
# balanced_accuracy = cm_qda$byClass['Balanced Accuracy']
# F1_score = cm_qda$byClass['F1']
```

The performance of QDA are as follows: Accuracy = 0.644; Balanced Accuracy = 0.5944 and $F_1 = 0.4357$

2.6.2.5. Model 5: Decision Tree The decision tree model uses decision rules created from the feature variables to classify whether an individual is diagnosed or not. Decision trees are non-parametric machine learning algorithms that recursively uses the feature the best divides decisions at nodes. Nodes are connected to each other by branches.

```
##Model 5: Decision Tree
getModelInfo("rpart")$rpart$parameters #Ascertain whether there are parameters to be tuned
```

```
##   parameter   class                label
## 1        cp numeric Complexity Parameter
```

```
# the parameter to be tuned is cp.

# set random seed for reproducibility for R 3.6 or later
set.seed(105, sample.kind = "Rounding")
#Fit the decision tree by tuning cp
train_dtree <- train(disease_status ~.,
                     data = train,
                     method = 'rpart',
                     tuneGrid = data.frame(cp = seq(0, 0.05, 0.002)))
train_dtree$bestTune # the best tuned is cp = 0.002
```

```
##      cp
## 2 0.002
```

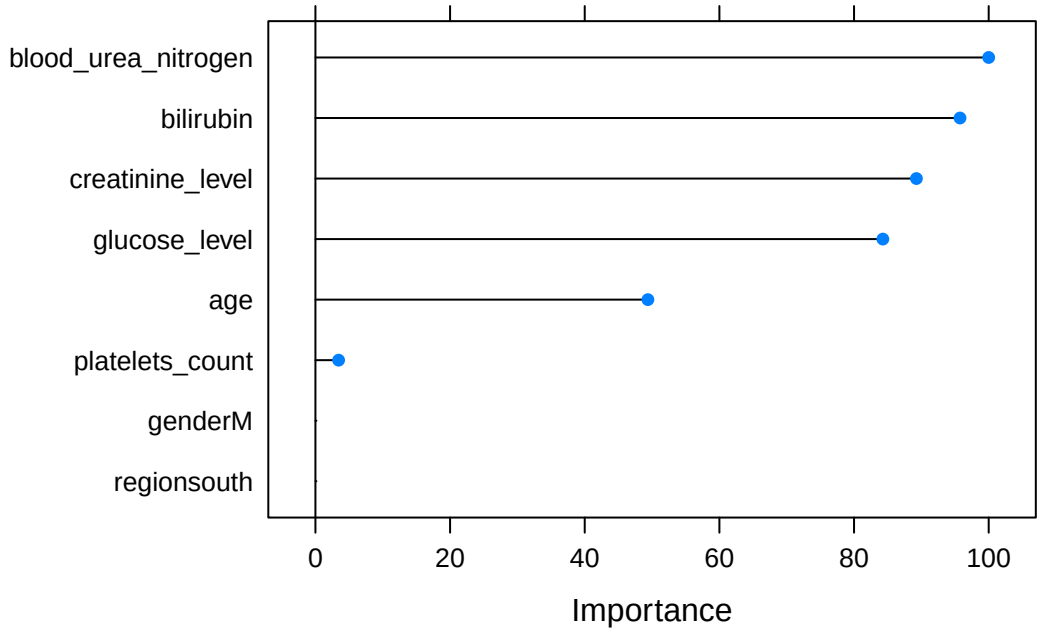
The tuning parameter in the decision tree is cp. The complexity parameter (cp) is a stopping parameter that indicates the minimum improvement needed at each node of the model. In other words, it ensures that splits are made at nodes that meet its criteria and it prunes out any that does not. The deeper the tree, the more complex the decision matrix becomes. Variable importance from this decision tree is shown and plotted below.

```
##Variable Importance
varImp(train_dtree)
```

```
## rpart variable importance
##
##                     Overall
## blood_urea_nitrogen 100.000
## bilirubin            95.744
## creatinine_level     89.268
## glucose_level        84.284
## age                  49.383
## platelets_count       3.443
## regionsouth           0.000
## genderM               0.000
```

```
plot(varImp(train_dtree), main = "Figure 26. Plot of Variable Importance of Decision Tree Model")
```

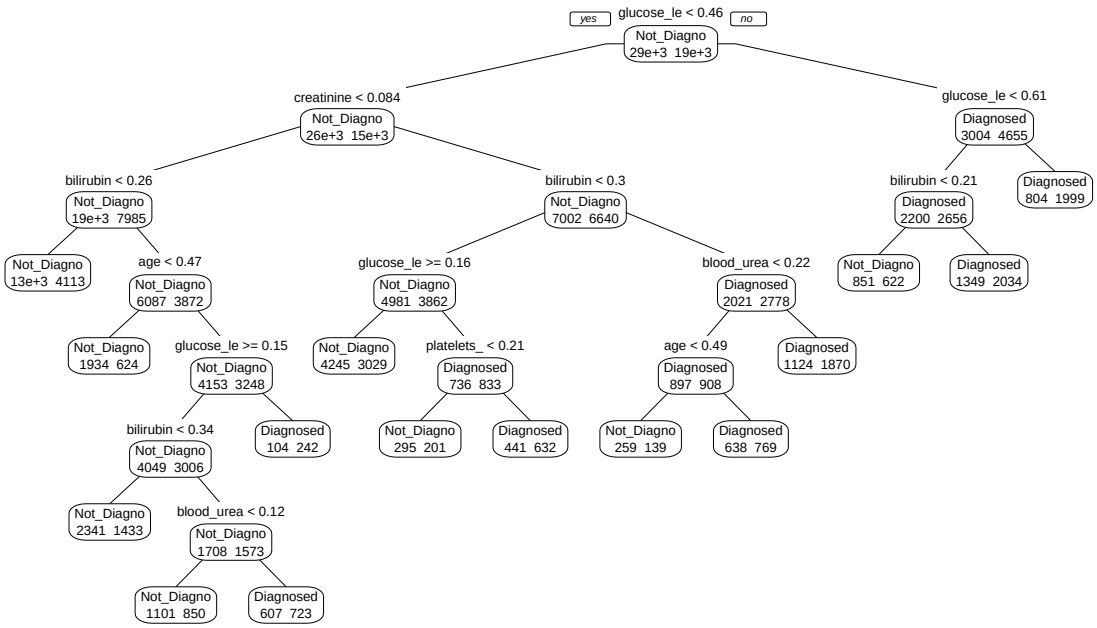
Figure 26. Plot of Variable Importance of Decision Tree Model



The variable importance of the decision tree shows that the top 3 predictors of disease diagnosis are blood urea nitrogen, bilirubin and creatinine level. Further, we note that the region and Gender variables had virtually no predictive power. Next, we examine the decision tree. The code for generating the decision tree is also shown below.

```
#Plot the final decision tree
prp(train_dtree$finalModel, type = 1, extra = 1, split.font = 1, varlen = -10,
    main = "Figure 27. Classification Tree for Disease Diagnostics")
```

Figure 27. Classification Tree for Disease Diagnostics



The parent node is split at the glucose level with MinMax value of 0.46. The decision tree helps determine the set of decision rules that can predict whether individuals were diagnosed or not. For instance, the parent node that indicates glucose level of above 0.46 MinMax value and which branches into node of glucose level of above MinMax value of 0.61 predicts diabetic diagnosis.The code for generating prediction and assessing model performance is shown below.

```
dtree_preds <- predict(train_dtree, test) #Obtain prediction
cm_dtree <- confusionMatrix(dtree_preds,
                            test$disease_status,
                            positive = 'Diagnosed',
                            mode = 'everything') #Obtain confusion Matrix and Performance
#The relevant metrics for this report are accuracy, balanced accuracy and F1-Score
# accuracy = cm_dtree$overall['Accuracy']
# balanced_accuracy = cm_dtree$byClass['Balanced Accuracy']
# F1_score = cm_dtree$byClass['F1']
```

The decision tree had the following performance: Accuracy = 0.6575; Balanced Accuracy = 0.6193 and F_1 = 0.4994

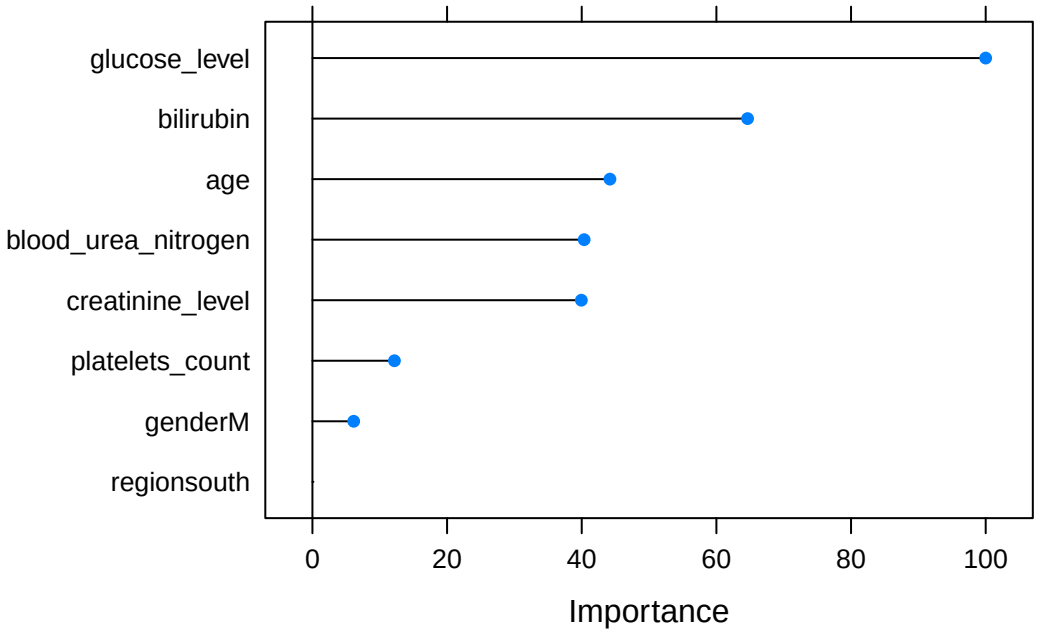
2.6.2.6. Model 6: Random Forest The random forest algorithm is an ensemble of many decision trees. It randomly uses features to build numerous decision trees where prediction is made by the contribution of each tree. Therefore, there is the likelihood that the decisions made by this uncorrelated forest of trees would be more stable than that made by a single tree. The parameter to be tuned in the random forest is mtry. This parameter connotes the number of variables randomly sampled at each split.

```r
##Model 6: Random Forest
# set.seed(105)
# set random seed for reproducibility for R 3.6 or later
set.seed(105, sample.kind = "Rounding")
train_rnf <- train(disease_status ~ .,
                   data = train,
                   method = "rf",
                   tuneGrid = data.frame(mtry = seq(1:5)),
                   trControl = control,
                   ntree = 500,
                   importance = TRUE) #Fit the model by tuning mtry parameter with 500 trees.
##Set importance = TRUE to obtain variable importance
```

The best random forest under tuning had an mtry of 2. The random forest was fit with 500 random trees. Plot of variable importance is shown below.

```r
##Ascertain Variable Importance
plot(varImp(train_rnf), main = "Figure 28. Plot of Variable Importance of Random Forest Model")
```

Figure 28. Plot of Variable Importance of Random Forest Model




```r
varImp(train_rnf)
```

```
## rf variable importance
##
##                     Importance
## glucose_level          100.000
## bilirubin               64.634
## age                     44.190
```

```
## blood_urea_nitrogen     40.358
## creatinine_level        39.941
## platelets_count         12.181
## genderM                  6.129
## regionsouth              0.000
```

The variable importance of the random forest shows that the top 3 predictors of disease diagnosis are glucose level, bilirubin and age. Further, we note that the region variable has virtually no importance in prediction.

```
rnf_preds <- predict(train_rnf, test) ## Obtain prediction
cm_rnf <- confusionMatrix(rnf_preds,
                          test$disease_status,
                          positive = 'Diagnosed',
                          mode = 'everything') #Obtain confusion Matrix
#The relevant metrics for this report are accuracy, balanced accuracy and F1-Score
# accuracy = cm_rnf$overall['Accuracy']
# balanced_accuracy = cm_rnf$byClass['Balanced Accuracy']
# F1_score = cm_rnf$byClass['F1']
```

The random forest had the following performance: Accuracy = 0.6688; Balanced Accuracy = 0.6358 and F_1 = 0.5317

3. Results

The performance of all the models is given in the table below. *Code for compiling all results into table not shown using chunk options for brevity*

Method	Accuracy	Balanced_Accuracy	F1_Score
Model 1: Baseline(Guessing)	0.5276	0.5081	0.4103
Model 2: Logistics Regression	0.6474	0.5960	0.4328
Model 3: KNN	0.6601	0.6220	0.5032
Model 4: QDA	0.6440	0.5944	0.4357
Model 5: Decision Tree	0.6575	0.6193	0.4994
Model 6: Random Forest	0.6688	0.6358	0.5317

The baseline model for this classification problem had an accuracy of 0.5276, balanced accuracy of 0.5081 and F_1-Score of 0.4103. The table shows that all other models performed better than this baseline model of guessing. The random forest was the best model, achieving the highest value on all measuring performance metrics. Apart from the baseline model, the random forest had the lowest difference between the balanced accuracy and the accuracy. The random forest achieved an accuracy of 0.6688, balanced accuracy of 0.6358 and F_1-Score of 0.5317.

An accuracy of 0.6688 means that the model was able to correctly predict about 66.88% of target labels on test set. In simple terms, given hundred (100) patients, the random forest model was able to detect about 67 conditions correctly. The balanced accuracy of 0.6358 means that the model was able to classify about 63.58% of patients correctly where the imbalance in the target variable is considered. Lastly, the F_1-Score of 0.5317 means that attributing the positive class to diagnosed individuals, the model apportions more than 50% importance to the harmonic mean of recall and precision.

The random forest was selected as the best model. Thus, the three(3) most important predictors are glucose level, bilirubin and age.

4. Conclusion

In this report, we examined some supervised learning algorithms for binary classification problem of disease diagnosis. The data for the analysis had 60000 observations which was divided into train set and test set in the ratio, 0.8:0.2. There were 9 columns, comprising 8 features and the target variable, disease_status. The train set revealed that about 40% were diagnosed with diabetes whiles 60% were not diagnosed. The exploratory data analysis shown that feature variables were skewed. Again, the plots show that there exists overlap of values of observations for each of the feature variables, an indication that similar sets of values predict both binary outcomes.

The following machine learning models were considered: baseline(guessing), logistics regression, knn, Quadratic Discriminant Analysis (QDA), decision tree and random forest. The random forest was the best model, achieving an accuracy of 0.6688, balanced accuracy of 0.6358 and F_1-Score of 0.5317. Though the algorithm is not the golden standard in medical diagnosis, in the extreme case where the needed robust diagnostic tests are unavailable, a potential impact would be to use the random forest to provide some level of insights which from this report, has proven to be better than guessing. This may be helpful especially in a country where health resources are scarce.

That notwithstanding, we acknowledge some limitations. First, primary examination tests are standard tests and not unique to diabetes diagnosis. Therefore, more robust data such as image data of patient cells can generate better input variables to detect diagnosis. However, this form of data may not be publicly available owing to ethical considerations. Also, diabetes is part of a group of non-communicable diseases that usually occur together. The no-mention of whether particular observations have had other conditions such as cirrhosis, hypertension, pancreas failure, etc. certainly reduces the ability of obtaining very high accuracy in modelling. Last but not least, the data pertained to a very small geographical location on the globe. Therefore, results may not be globally applied since there could be different underlying factors in different part of the world.

Disease diagnosis tend to be an important aspect of the medical field. Future research could be explored on publicly available and certified image data. There should also be more collaborative efforts between health institutions and academia in establishing concrete and broad metrics for medical machine learning. This could reduce bottlenecks inherent in acquiring quality data for model training. Additionally, a more comprehensive study that covers many geographical locations would be helpful for worldwide applicability.

Reference Irizarry, R.A. (2022).Introduction to Data Science: Data Analysis and Prediction Algorithms with R. *The coding in this report was inspired by the course as contained in Irizarry(2022).*

Mayo Clinic (2022). Creatinine tests. Retrieved from https://www.mayoclinic.org/tests-procedures/creatinine-test/about/pac-20384646

Xie Y. (2020). Chunk options and package options. https://yihui.org/knitr/options

Xie Y (2022). tinytex: Helper Functions to Install and Maintain TeX Live, and Compile LaTeX Documents. R package version 0.40, https://github.com/rstudio/tinytex.*tinytex was used as the background latex language in knitting*