Exploring Statistical Machine Learning Models

Tahsin Azad

2023-11-24

Intro

Diabetes is a crucial medical issue today. For this final project, we will be observing a diabetes dataset and doing classification, by classifying whether a female over the age of 21 from the Pima Indian heritage has diabetes or not given other factors. This dataset is from kaggle:

https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database

Data Description

The first step will be to load the dataset and take a look at the data description.

```
diabetes <- read_csv("diabetes.csv")
```

```
## Rows: 768 Columns: 9
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (9): Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, D...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
diabetes<- as_tibble(diabetes)
head(diabetes)
```

```
## # A tibble: 6 x 9
##   Pregnancies Glucose BloodPressure SkinThickness Insulin   BMI
##         <dbl>   <dbl>         <dbl>         <dbl>   <dbl> <dbl>
## 1           6     148            72            35       0  33.6
## 2           1      85            66            29       0  26.6
## 3           8     183            64             0       0  23.3
## 4           1      89            66            23      94  28.1
## 5           0     137            40            35     168  43.1
## 6           5     116            74             0       0  25.6
## # i 3 more variables: DiabetesPedigreeFunction <dbl>, Age <dbl>, Outcome <dbl>
```

Descriptive Statistics:

```r
summary(diabetes)
```

```
##   Pregnancies        Glucose      BloodPressure    SkinThickness
##  Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.: 0.00
##  Median : 3.000   Median :117.0   Median : 72.00   Median :23.00
##  Mean   : 3.845   Mean   :120.9   Mean   : 69.11   Mean   :20.54
##  3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:32.00
##  Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00
##     Insulin           BMI        DiabetesPedigreeFunction      Age
##  Min.   :  0.0   Min.   : 0.00   Min.   :0.0780           Min.   :21.00
##  1st Qu.:  0.0   1st Qu.:27.30   1st Qu.:0.2437           1st Qu.:24.00
##  Median : 30.5   Median :32.00   Median :0.3725           Median :29.00
##  Mean   : 79.8   Mean   :31.99   Mean   :0.4719           Mean   :33.24
##  3rd Qu.:127.2   3rd Qu.:36.60   3rd Qu.:0.6262           3rd Qu.:41.00
##  Max.   :846.0   Max.   :67.10   Max.   :2.4200           Max.   :81.00
##     Outcome
##  Min.   :0.000
##  1st Qu.:0.000
##  Median :0.000
##  Mean   :0.349
##  3rd Qu.:1.000
##  Max.   :1.000
```

Length of dataset

```r
nrow(diabetes)
```

```
## [1] 768
```

Checking for missing values:

```r
sum(is.na(diabetes))
```

```
## [1] 0
```

There are no missing values. Moreover, we noticed that there are no categorial variables besides the Outcome variable, which is what we will be classifying for.

One observation is that some medical variables have "0" which does not make sense in this context. This would not make sense for Glucose, BloodPressure, SkinThickness, and BMI. We can check how many have zero.

```r
sum(diabetes$Glucose == 0)
```

```
## [1] 5
```

```r
sum(diabetes$BloodPressure == 0)
```

```
## [1] 35
```

```r
sum(diabetes$SkinThickness == 0)
```

```
## [1] 227
```

```r
sum(diabetes$BMI == 0)
```

```
## [1] 11
```

Since we have 768 rows, we will opt to drop those rows where the variable is 0 for Glucose, BloodPressure, and BMI since they are a small amount. However for SkinThickness, intead we will place the mean of SkinThickness.

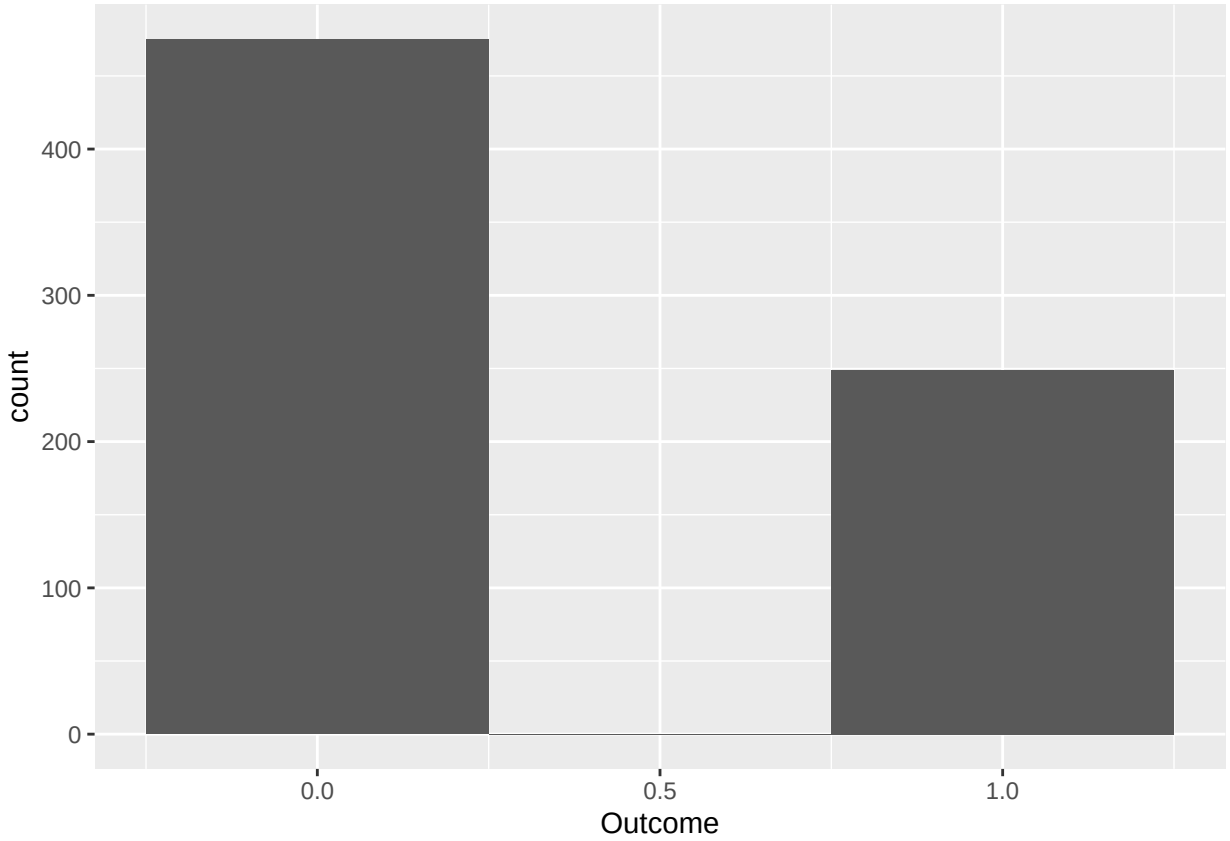
```r
new_diabetes <-diabetes %>%
  filter(Glucose != 0, BloodPressure != 0, BMI != 0)
skin <- diabetes$SkinThickness
mean_skin <- mean(skin[skin != 0], na.rm = TRUE)
new_diabetes$SkinThickness[new_diabetes$SkinThickness == 0] <- mean_skin
```

Exploratory Data Analysis and Visualization

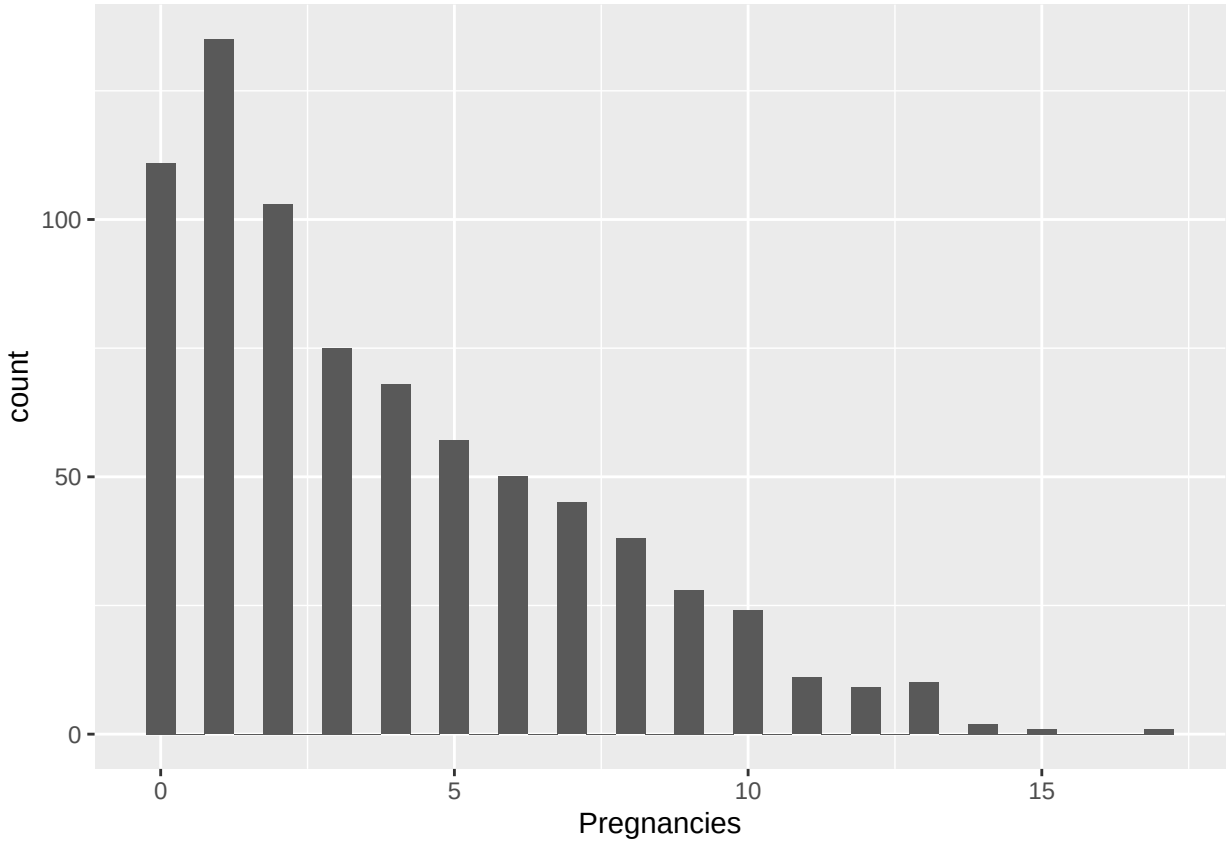
The following will show different visualizations for variables in this dataset:

Noticing the Outcome variable, we'll look at a histogram for it. 0 represents "no the patient does not have diabetes" and 1 represents "yes, the patient has diabetes". Also a histogram for pregnancies.

```r
ggplot(data = new_diabetes) + geom_histogram(mapping = aes(x = Outcome), binwidth = 0.5)
```

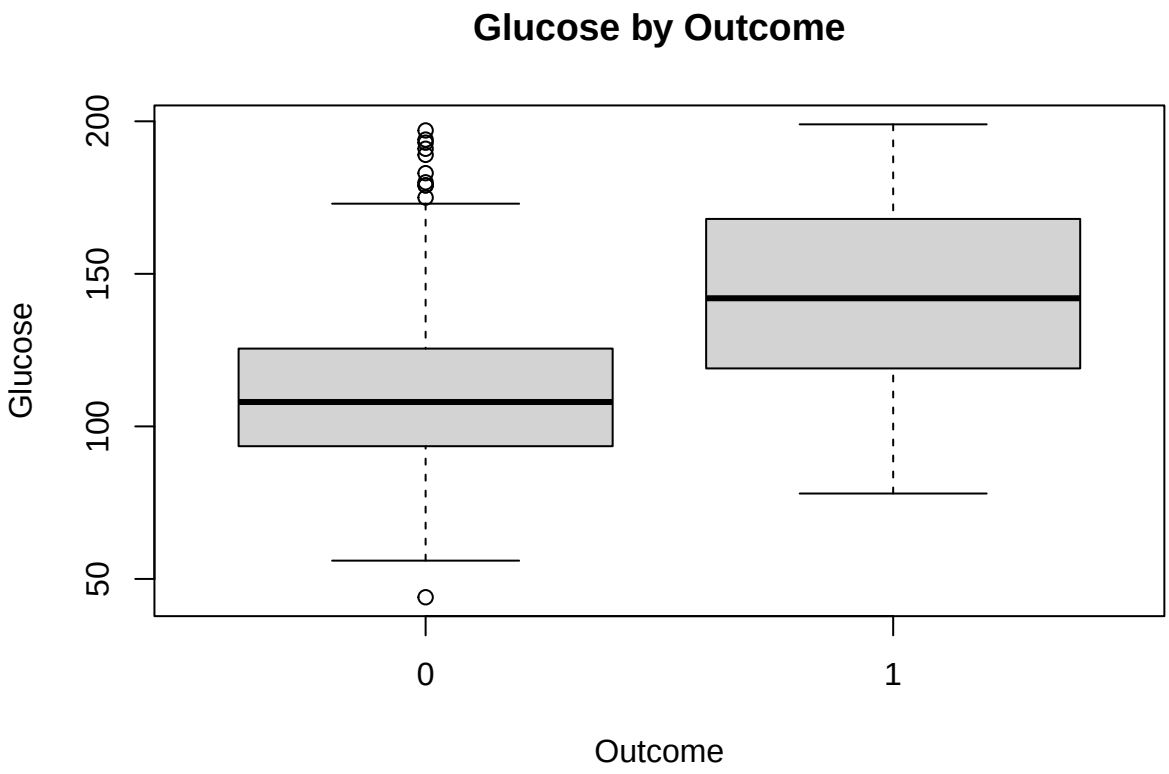



```
ggplot(data = diabetes) + geom_histogram(mapping = aes(x = Pregnancies), binwidth = 0.5)
```

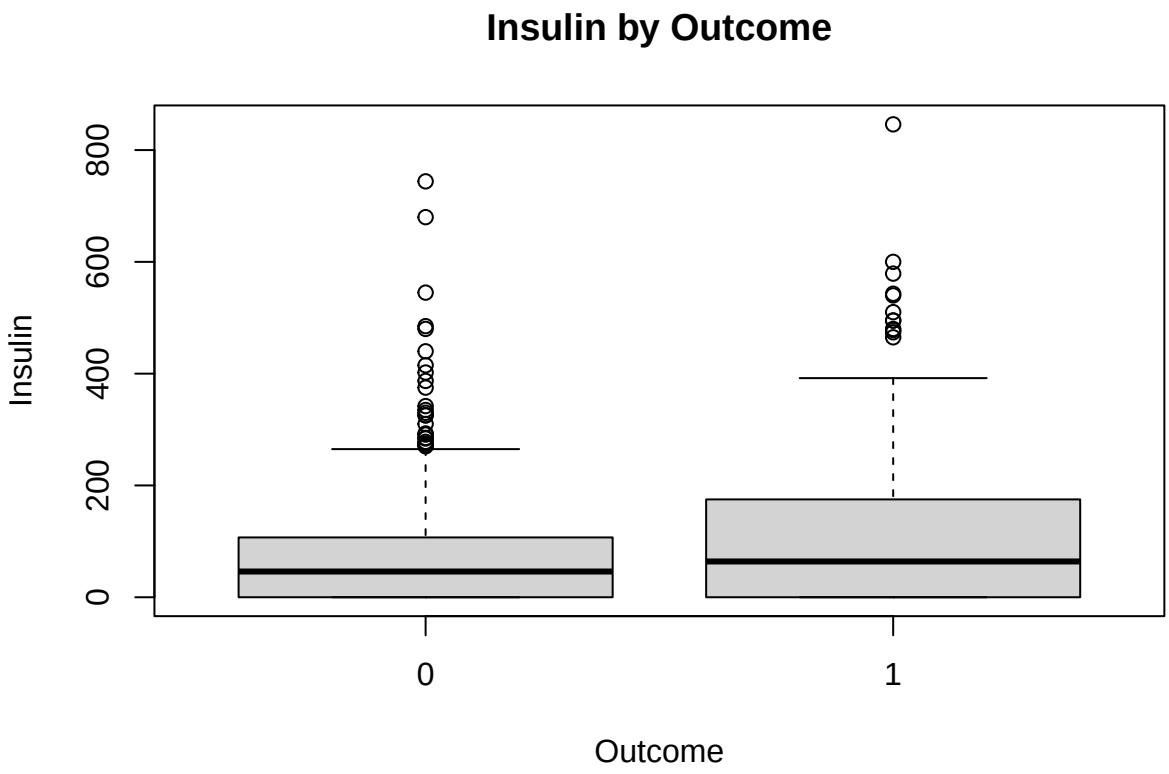


Boxplots:

```
boxplot(new_diabetes$Glucose ~ new_diabetes$Outcome, main="Glucose by Outcome", xlab="Outcome", ylab="Gl
```

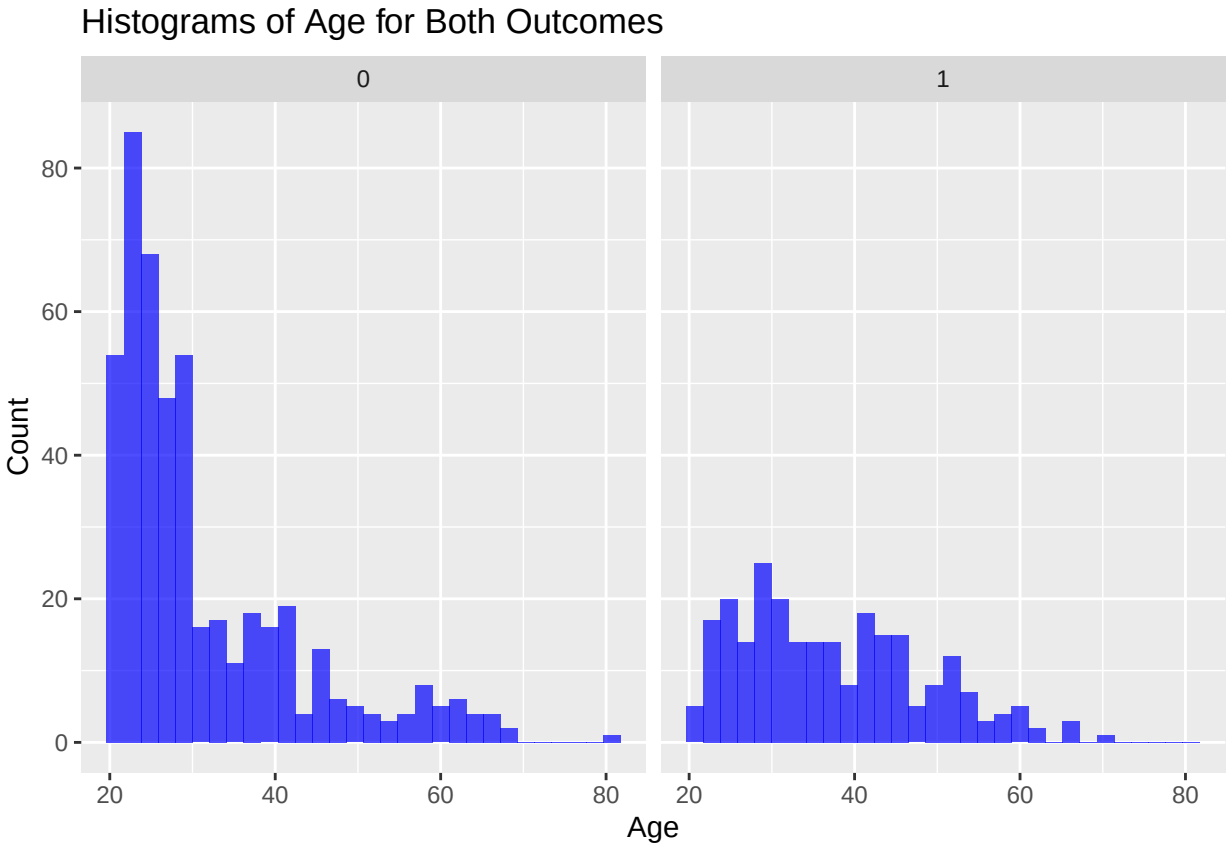



```
boxplot(new_diabetes$Insulin ~ new_diabetes$Outcome, main="Insulin by Outcome", xlab="Outcome", ylab="In
```



Histogram for age:

```
ggplot(new_diabetes, aes(x = Age)) +
  geom_histogram(bins = 30, fill = "blue", alpha = 0.7) +
  facet_grid(. ~ Outcome) + labs(x = "Age", y = "Count", title = "Histograms of Age for Both Outcomes")
```



We can observe that younger people generally do not have diabetes, and that older people do have it slightly more.

Due to the various different ranges between the variables, they will be normalized:

```
cop_diabetes<- new_diabetes
features <- setdiff(names(cop_diabetes), "Outcome")
preProcValues <- preProcess(cop_diabetes[, features], method = c("range"))
diabetes_normalized <- predict(preProcValues, cop_diabetes[, features])
diabetes_normalized$Outcome <- cop_diabetes$Outcome

head(diabetes_normalized)
```

```
## # A tibble: 6 x 9
##   Pregnancies Glucose BloodPressure SkinThickness Insulin   BMI
##         <dbl>   <dbl>         <dbl>         <dbl>   <dbl> <dbl>
## 1      0.353    0.671         0.490         0.304   0     0.315
## 2      0.0588   0.265         0.429         0.239   0     0.172
## 3      0.471    0.897         0.408         0.241   0     0.104
## 4      0.0588   0.290         0.429         0.174   0.111 0.202
## 5      0        0.6           0.163         0.304   0.199 0.509
## 6      0.294    0.465         0.510         0.241   0     0.151
## # i 3 more variables: DiabetesPedigreeFunction <dbl>, Age <dbl>, Outcome <dbl>
```

Normalization ensures that there is less data redundancy.

Correlation between variables below (since Outcome is categorical, Point-Biserial method is used):

```
sapply(diabetes_normalized[, c("Pregnancies", "Glucose", "BloodPressure", "SkinThickness", "Insulin", "
```

```
##              Pregnancies                  Glucose            BloodPressure
##                -0.2244172                -0.4883842                -0.1667034
##            SkinThickness                  Insulin                      BMI
##                -0.2166611                -0.1454881                -0.2993753
## DiabetesPedigreeFunction                      Age
##                -0.1849474                -0.2457412
```

Problem formulation and discussion of statistical algorithms

Discussion

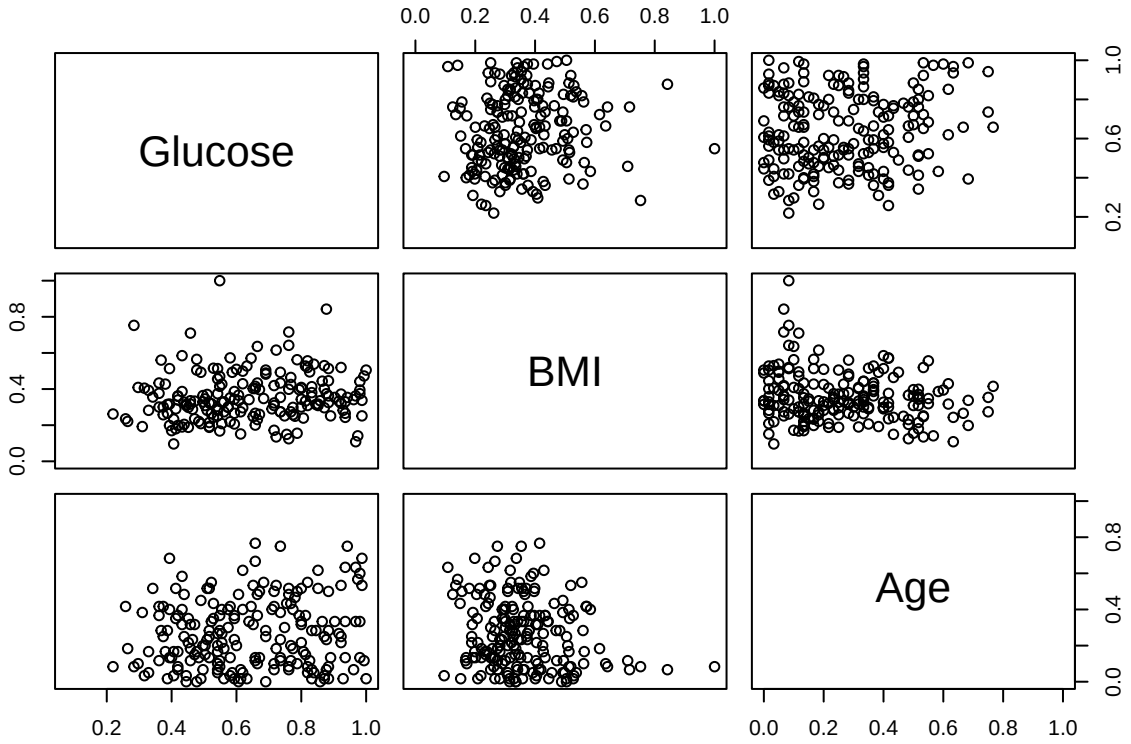
The objective which we will try to reach is predicting whether a female over the age of 21 from the Pima Indian heritage has diabetes or not. This is important because diabetes is a health-related issue and figuring out which factors effect having diabetes is crucial. Based on the correlation matrix and the previous EDA, it seems the strongest factors are Glucose, BMI, and Age. Since this is a binary classification, we will start with choosing a logistic regression model after splitting the data into a training and testing split (based on 80-20).

```
set.seed(123)
partition <- createDataPartition(diabetes_normalized$Outcome, p = 0.80, list = FALSE)
train.diabetes.norm <- diabetes_normalized[partition, ]
test.diabetes.norm  <- diabetes_normalized[-partition, ]

log.model <-glm(Outcome ~ Glucose + BMI + Age, data = train.diabetes.norm , family = binomial)
summary(log.model)$coefficients
```

```
##              Estimate Std. Error    z value     Pr(>|z|)
## (Intercept) -5.241799  0.4357575 -12.029166 2.496712e-33
## Glucose      5.240214  0.6132506   8.544980 1.285580e-17
## BMI          4.654975  0.8173768   5.695017 1.233597e-08
## Age          1.568920  0.5305880   2.956946 3.107030e-03
```

By observing the p values we can see the significance. The null hypothesis is that these values have no significant influence over the Outcome of diabetes. Since all of these have extremely small P-values, this tells us that the all the values, Glucose, BMI, and Age are significant estimators in this logistic regression model.For every one unit change in Glucose, the log odds of having diabetes, versus not having diabetes, increases by 5.240214, holding other variables fixed. For every one unit change in BMI, the log odds of having diabetes, versus not having diabetes, increases by 4.654975, holding other variables fixed.For every one unit change in Age, the log odds of having diabetes, versus not having diabetes, increases by 1.568920, holding other variables fixed. # Checking for outliers Having chosen the variables, we will also check for outliers:

```
pairs(~Glucose + BMI + Age, data = train.diabetes.norm, col = train.diabetes.norm$Outcome)
```



Between the variables, there are some outliers noticed.

Logistic model on test data and confusion matrix

```
set.seed(123)
train.diabetes.norm.copy = train.diabetes.norm
test.diabetes.norm.copy = test.diabetes.norm

pred.train <- predict(log.model, test.diabetes.norm.copy, type = "response")
test.diabetes.norm.copy.mutated <- test.diabetes.norm.copy %>%
  mutate(PredictedOutcome = as.factor(ifelse(pred.train <= 0.5, "No", "Yes")))

conf_log <- table(pred=test.diabetes.norm.copy.mutated$PredictedOutcome, true=test.diabetes.norm$Outcome
conf_log
```

```
##      true
## pred   0  1
##   No  80 28
##   Yes  9 27
```

Interpretation of the confusion matrix: 80 believed to have diabetes correctly had it, while 28 believed to have diabetes did not. 27 correctly identified to not have diabetes while 9 were identified to have diabetes when they did not (False Positive). 74.31% were identified correctly.

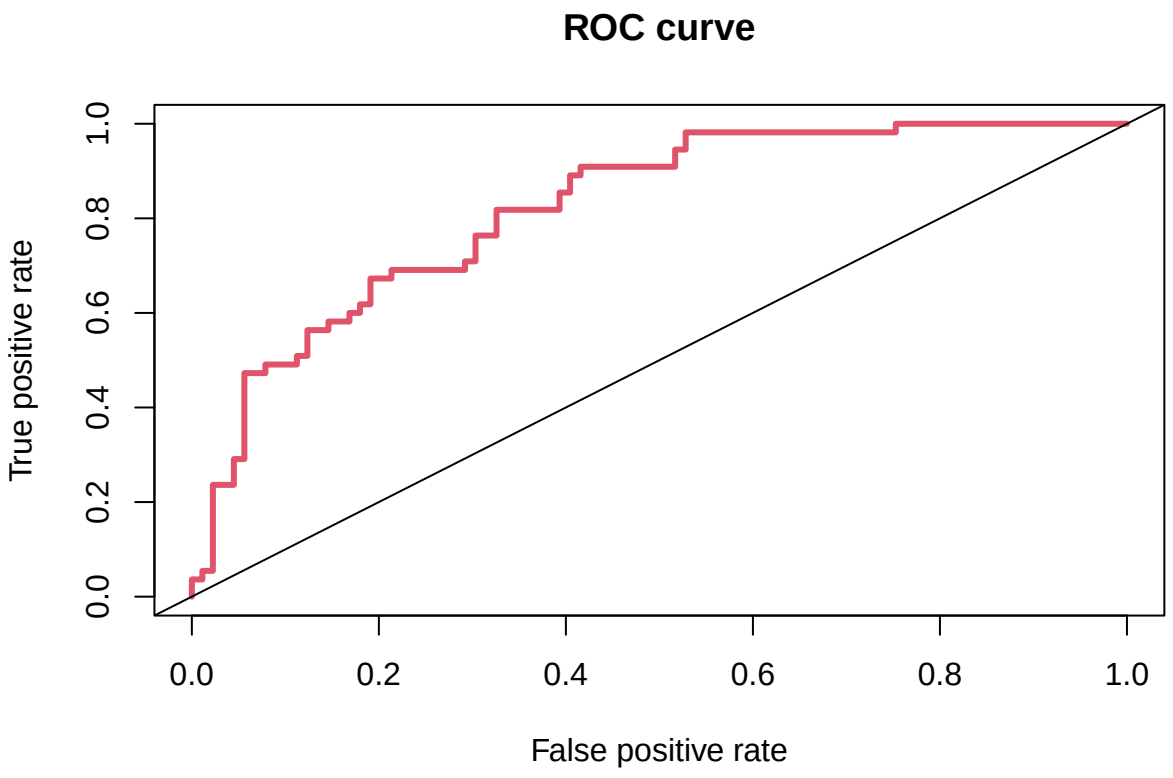
True positive rate is defined as the true positive divided by (true positive + false negative). In this problem it is 0.75. False positive rate is defined as the false positive divided by (false positive + true negative). In this problem it is 0.2593.

This suggest that the although model is good and accurate at predicting positive "yes" for diabetes, it does get some mistakes, around 1/4.

ROC Curve and AUC (Performace Metrics)

We wil observe ROC Curve and AUC (area under curve) to have a better glance at the performance metrics.

```
set.seed(123)
predic <- prediction(pred.train, test.diabetes.norm.copy$Outcome)
for_roc <- performance(predic, measure="tpr", x.measure="fpr")
plot(for_roc, col=2, lwd=3, main="ROC curve")
abline(0,1)
```



Judging by the plot, the ROC curve is considered moderately good, and could be better potentially. The Area under the curve (AUC) gives a better outlook:

```
auc = performance(predic, "auc")@y.values
auc
```

```
## [[1]]
## [1] 0.8216547
```

This confirms that it is moderately good. For AUC, the closer to 1.0, the better.

Checking for overfitting

Now is a good time to check for overfitting. We will do this by comparing the AUC for the test data and train data. Ideally, the AUC for both should be relatively close to each other. We have the AUC for the test data already.

```
pred.traina <- predict(log.model, train.diabetes.norm.copy, type = "response")
predic1e <- prediction(pred.traina, train.diabetes.norm.copy$Outcome)
auc32 = performance(predic1e, "auc")@y.values
auc32
```

```
## [[1]]
## [1] 0.8263982
```

Both values are extremely close, so it is likely there is very little overfitting.

Cross validation

Cross validation will be used to evaluate the model.

```
set.seed(123)
control <- trainControl(method = "cv", number = 10)
new_diabetes$Outcome <- as.factor(new_diabetes$Outcome)
log_cv <- train(Outcome ~ Glucose + BMI + Age,data = new_diabetes, family = binomial(), method = "glm",
log_cv
```

```
## Generalized Linear Model
##
## 724 samples
##   3 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 651, 651, 652, 652, 651, 652, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.7678653  0.457678
```

With a 10-fold cross validation, we can see that there is a 76.79% accuracy and a kappa which is moderate. This shows that the model is good, but there could be more done to improve.

Testing with different models

We will also compare between different model types, instead of just logistic regression. Since we are classifying a binary result, we can use a K-Nearest Neighbor, Decision Tree, Random Forest, and Support Vector Machine model, then compare between them all and discuss the pros and cons surrounding it.

K-NN

First is the K-NN. We will normalize the data for this model.

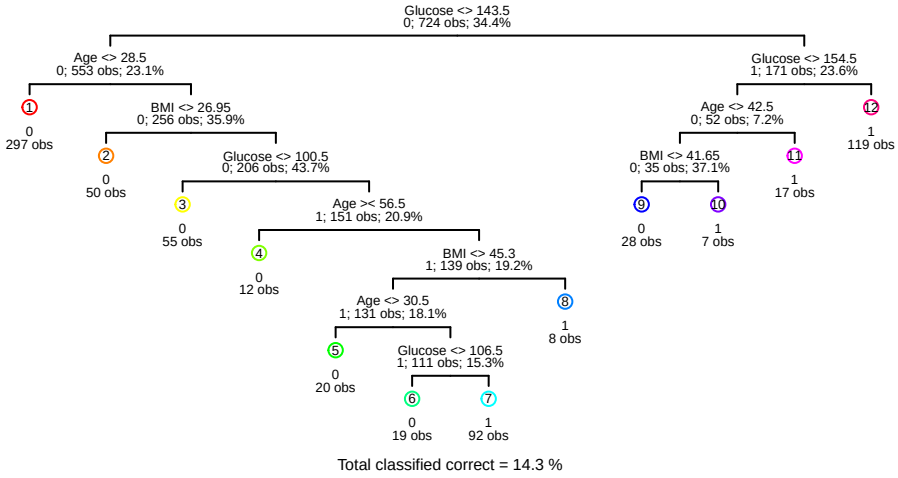
```
set.seed(123)

x_train_norm <- train.diabetes.norm.copy[, -which(names(test.diabetes.norm) == "Outcome")]
y_train_norm <- train.diabetes.norm.copy$Outcome
x_test_norm <- test.diabetes.norm.copy[, -which(names(test.diabetes.norm) == "Outcome")]
y_test_norm <- test.diabetes.norm.copy$Outcome

pred.ytrain <- knn(train = x_train_norm, test = x_test_norm, cl = y_train_norm, k = 5)
conf_knn <- table(Predicted = pred.ytrain, Actual = y_test_norm)
conf_knn
```

```
##           Actual
## Predicted  0  1
##         0 75 27
##         1 14 28
```

Decision Tree

```
set.seed(123)
tree.decision = rpart(Outcome ~ Glucose + BMI + Age, data = new_diabetes)
draw.tree(tree.decision, nodeinfo=TRUE,cex = 0.4)
title("Decision Tree")
```

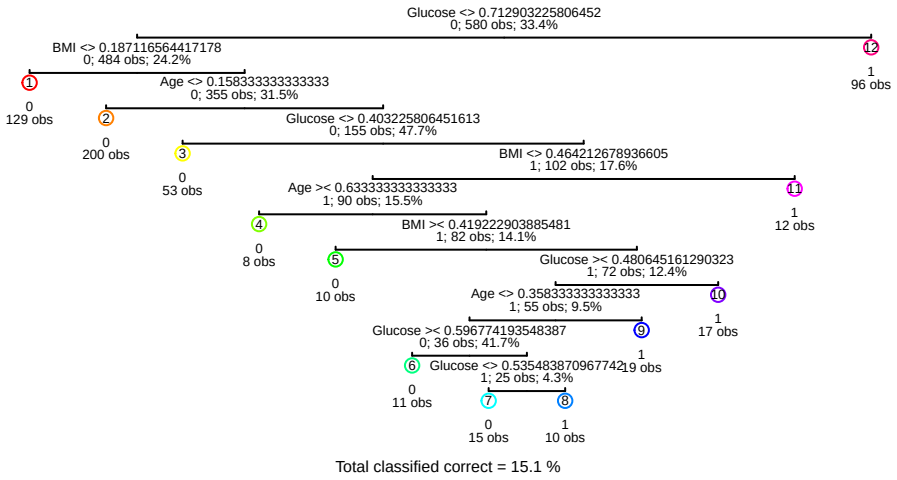
Decision Tree



From this decision tree model we can see how the outcome of diabetes is classified based on Glucose, Age, and BMI. Glucose is the most defining factor. We will also use the train test split to determine evaluate the performance.

On the training data:

```
set.seed(123)
train.diabetes.norm.copy$Outcome <- as.factor(train.diabetes.norm.copy$Outcome)
train.tree <- rpart(Outcome ~ Glucose + BMI + Age, data = train.diabetes.norm.copy, method = "class")
draw.tree(train.tree, nodeinfo=TRUE,cex = 0.4)
title("Decision Tree on Training data")
```



Then we predict on the test data and compute the confusion matrix

```
set.seed(123)
pred.tree = predict(train.tree, test.diabetes.norm.copy, type="class")
confus.tree = table(pred.tree, test.diabetes.norm.copy$Outcome)
confus.tree
```

```
##
## pred.tree  0  1
##         0 79 21
##         1 10 34
```

Random Forest

```
set.seed(123)
train.random <- randomForest(Outcome ~ Glucose + BMI + Age, data=train.diabetes.norm.copy, importance=TR
train.random
```

```
##
## Call:
##  randomForest(formula = Outcome ~ Glucose + BMI + Age, data = train.diabetes.norm.copy,      importa
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 1
```

```
##
##         OOB estimate of  error rate: 27.59%
## Confusion matrix:
##     0   1 class.error
## 0 318  68   0.1761658
## 1  92 102   0.4742268
```

```r
set.seed(123)
pred.random = predict(train.random, test.diabetes.norm.copy, type="class")
confus.random = table(pred.random, test.diabetes.norm.copy$Outcome)
confus.random
```

```
##
## pred.random  0  1
##           0 76 20
##           1 13 35
```

Comparison of models and evaluation

We will compare the different models and evaluate them, and disucss the strenghts and weaknesses amongst them and finally choose a model.

```
## Confusion Matrix for Logistic Regression:
```

```
##      true
## pred   0  1
##   No  80 28
##   Yes  9 27
```

```
## Accuracy for Logistic Regression:
```

```
## [1] 0.7430556
```

```
## Confusion Matrix for k-Nearest Neighbors:
```

```
##          Actual
## Predicted  0  1
##         0 75 27
##         1 14 28
```

```
## Accuracy for k-Nearest Neighbors:
```

```
## [1] 0.7152778
```

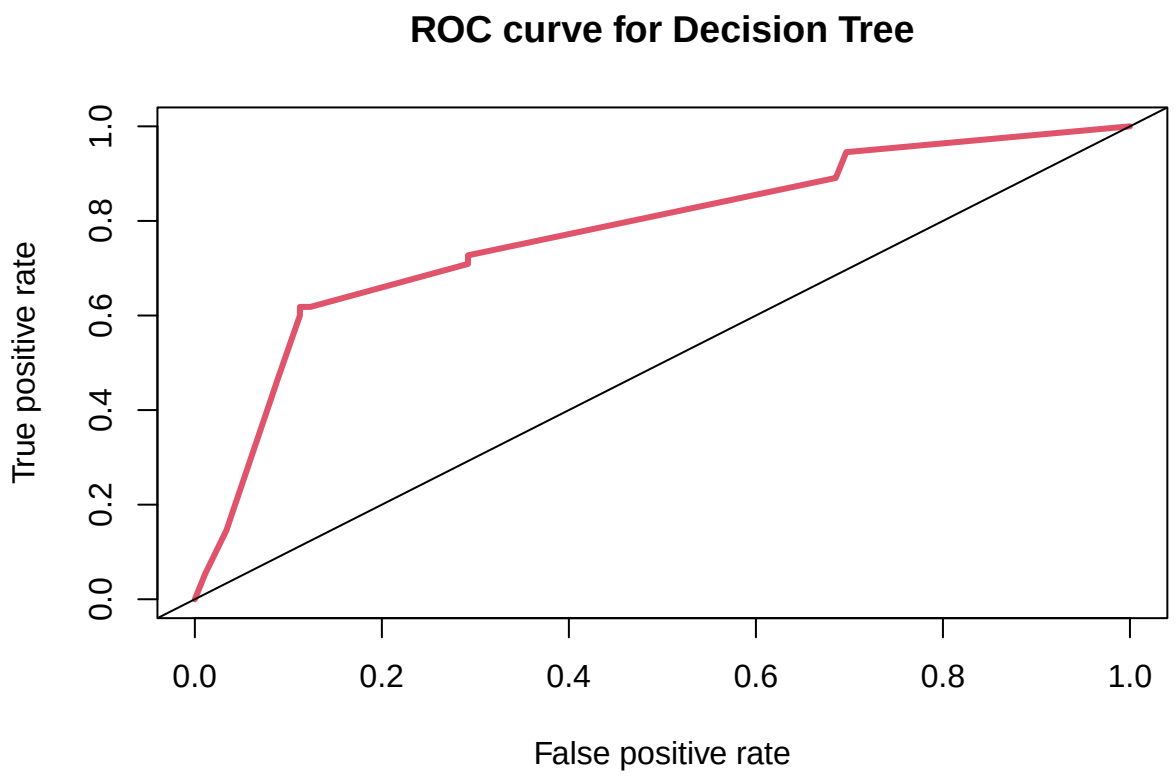
```
## Confusion Matrix for Decision Tree:
```

```
##
## pred.tree  0  1
##         0 79 21
##         1 10 34
```

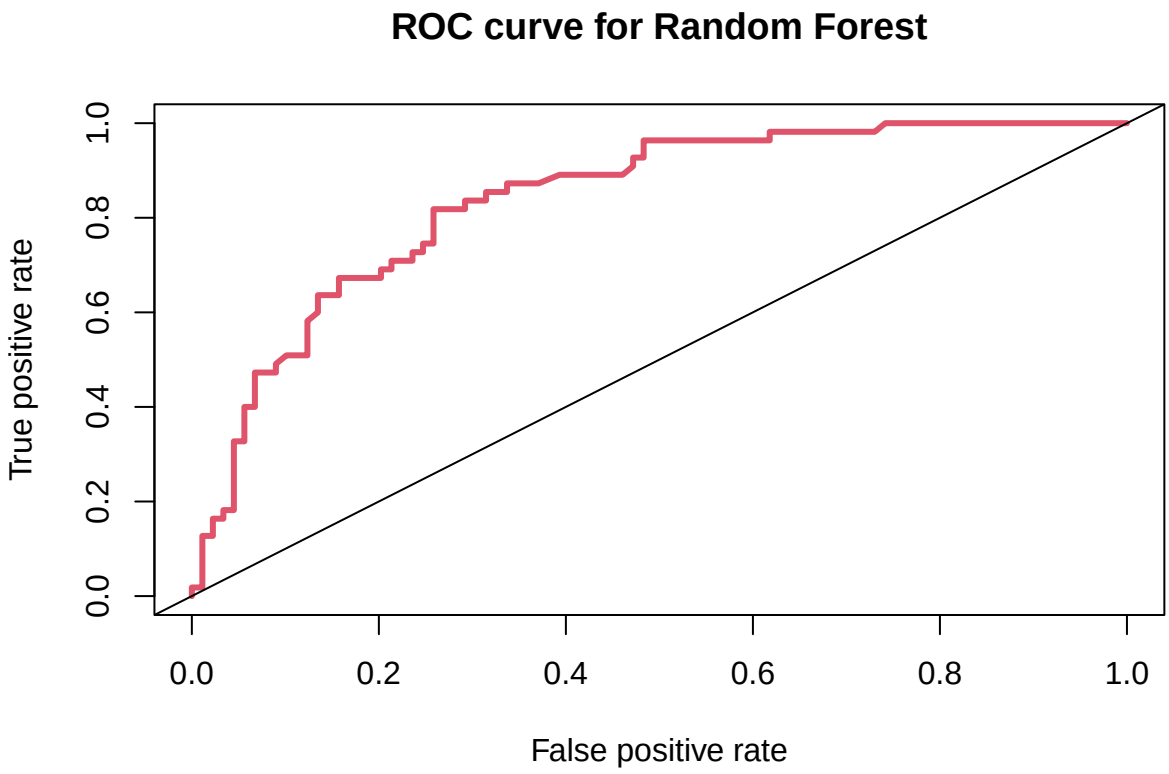
```
## Accuracy for Decision Tree:

## [1] 0.7847222

## Confusion Matrix for Random Forest:

##
## pred.random  0  1
##           0 76 20
##           1 13 35

## Accuracy for Random Forest:

## [1] 0.7708333
```

Since Decision Tree and Random Forest have the highest accuracy, we check the ROC and AUC (Area under the curve) are observed for them:

```
set.seed(123)
train.tree1 <- rpart(Outcome ~ Glucose + BMI + Age, data = train.diabetes.norm.copy, method = "class")
prob <- predict(train.tree1, test.diabetes.norm.copy, type = "prob")[, 2]
actual_outcomes <- ifelse(test.diabetes.norm.copy$Outcome == '1', 1, 0)
predic_random <- prediction(prob, actual_outcomes)
for_roc <- performance(predic_random, measure="tpr", x.measure="fpr")
plot(for_roc, col=2, lwd=3, main="ROC curve for Decision Tree")
abline(0,1)
```



```r
set.seed(123)
prob1 <- predict(train.random, test.diabetes.norm.copy, type = "prob")[,2]
actual_outcomes1 <- ifelse(test.diabetes.norm.copy$Outcome == '1', 1, 0)
predic_random1 <- prediction(prob1, actual_outcomes1)
for_roc1 <- performance(predic_random1, measure="tpr", x.measure="fpr")
plot(for_roc1, col=2, lwd=3, main="ROC curve for Random Forest")
abline(0,1)
```




```r
set.seed(123)
auc1 = performance(predic_random, "auc")@y.values
cat("Area under Curve for Decision Tree:", format(auc1, digits = 4), "\n")
```

```
## Area under Curve for Decision Tree: 0.7744
```

```r
set.seed(123)
auc2 = performance(predic_random1, "auc")@y.values
cat("Area under Curve for Random Forest:", format(auc2, digits = 4), "\n")
```

```
## Area under Curve for Random Forest: 0.838
```

The Area under the curve for the Random Forest is closer to 1.0, which means it is a better model. Moreover, it is also slightly more accurate. This seems to be the best model for classifying whether a female over the age of 21 from the Pima Indian heritage has diabetes, even better than the Logistic Model.

Discussion about models

The following will discuss the different models and why the final choice for Random Forest was made. Although in this project, the Logistic regression model came out relatively good, there are some issues. Logistic regression assumes that there is a lienar relationship between the outcome diabetes and the log odds of the other variables Glucose, BMI, and Age. It is very possible that there were non-linear variables.

For the K-Nearest Neighbor model, it had the lowest accuracy. This may be due to the choice of the k value of 5. For improvement of this model, a graph showcasing the accuracies based on the k value would have been a better method to proceed by. Moreover, it could be that the data itself had high variance.

The Decision Tree model had a decent accuracy and overall is a good model. Improvement could be made had there been more importance stressed toward overfitting in this project, since Decision Tree models are susceptible easily to overfitting.

The Random Forest Tree model seemed to be the best and this may be for multiple reasons. The RF model reduces overfitting by having a plethora of trees and works in a more complex way compared to the Decision Tree model. The high accuracy as well as AUC is proof of this.