Predicting Diabetes in PIMA Women

edX Capstone Project Submission

Kirtimay Pendse

6/23/2020

Introduction

Diabetes is a metabolic disorder defined as when one's blood glucose is too high (known as hyperglycemia) for a prolonged period of time. Glucose is an essential simple sugar widely consumed daily, and the hormone insulin helps in absorbing glucose from food and transforming it into energy; however, sometimes one's body doesn't make enough insulin or is unable to use it well, resulting in glucose staying in the bloodstream undigested and unable to reach the cells.[1]. This can cause health problems, especially diabetes. Around 9.5% -almost 30.5 million- of the United States population had diabetes in 2015 [2], and factors such as being overweight, being physically inactive, having a family history are linked with higher chances of developing diabetes. Due to several factors not discussed in this paper [3], diabetes is extremely prevalent in Native Americans, most notably within the Pima tribe- since the Pima tribe is a mostly homogenous group, Pima people have been the subject of several studies of diabetes.

This project is the final part of the HarvardX: PH125.9x Data Science: Capstone course[4], the last course for the Data Science Professional Certificate. This project is centered around predicting the presence of diabetes in Pima Indian women using data on factors such as age, body mass index, blood pressure etc. compiled together in the Pima Indians Diabetes dataset.

The dataset, loaded as 'pima_diabetes', is split into a training set containing 80% of the data and a test set containing 20% of the data for validation. This report is split into four sections: first, the objective and motivation behind the project is highlighted, then exploratory data analysis is conducted, following which the modeling approach to develop the diabetes prediction algorithm is presented. Finally, the modeling results are presented along with a discussion on the algorithm's performance and its limitations.

Objective

The dataset[5] is available on Kaggle and is originally sourced from the National Institute of Diabetes and Digestive and Kidney Diseases, a part of the Department of Health and Human Services. The objective of this analysis is to diagnostically predict whether or not a patient is diabetic, based on select diagnostic measurements included in the dataset (such as BMI, Age, Blood Pressure). There are 786 individuals in the dataset, all of whom are females of at least 21 years of age, and of Pima Indian heritage.

[1]https://www.niddk.nih.gov/health-information/diabetes

[2]Centers for Disease Control and Prevention. National diabetes statistics report, 2017. www.cdc.gov/diabetes/pdfs/data/statistics/national-diabetes-statistics-report.pdf

[3]more can be found at https://care.diabetesjournals.org/content/29/8/1866

[4]https://courses.edx.org/courses/course-v1:HarvardX+PH125.9x+1T2020/course/

[5]https://www.kaggle.com/ksp585/pima-indian-diabetes-logistic-regression-with-r

Methods and Analysis

Preparing the data

First, the dataset is downloaded and split into a train set and a test set. The train set is used to create the prediction algorithm, and then the algorithm is tested on the test set for a final validation.

```r
#Loading required packages
library(lubridate)
if(!require(ggthemes))
  install.packages("ggthemes", repos = "http://cran.us.r-project.org")
if(!require(scales))
  install.packages("scales", repos = "http://cran.us.r-project.org")
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
library(dplyr)
library(knitr)
library(ggplot2)
library(dslabs)
library(lubridate)
library(corrplot)
library(readr)

#Downloading the data
dl <- tempfile()
download.file("https://github.com/kirtimay/edX_Capstone/blob/master/cyo-diabetes/diabetes.csv", dl)
pima_diabetes <- read.csv("diabetes.csv",
col.names=c("pregnancies","glucose","bp","skin_thickness","insulin","bmi","dpf","age","outcome"))

#convert outcome to factor
pima_diabetes$outcome <- factor(pima_diabetes$outcome)
```

Description of Variables

As seen in the table, there are 9 variables in total. The response variable is 'outcome', which is a binary variable- 1 indicates that the patient is diabetic, and 0 indicates that they are not. The other 8 variables are predictors, and their descriptions are provided below.

It should be noted that the plasma glucose concentration was measured after a 2-hour glucose tolerance oral test, BMI is calculated as the patient's weight in kgs divided by their height in meters squared, and the DPF is a variable synthesizing family history of diabetes [6].

Variable	Class	Description
pregnancies	integer	No. of Pregnancies
glucose	integer	Plasma Glucose Concentration (mg/dL)
bp	integer	Diastolic BP (mm Hg)
skin_thickness	integer	Triceps Skin Thickness (mm)
insulin	integer	2 Hour Serum Insulin (uU/mL)
bmi	numeric	Body Mass Index
dpf	numeric	Diabetes Pedigree Function
age	integer	Age in Years
outcome	factor	Presence of Diabetes

[6] http://www.personal.kent.edu/~mshanker/personal/Zip_files/sar_2000.pdf

The pima_diabetes dataset was split into a training set (80% of data) and a test set (remaining 20% of data).

```r
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(y = pima_diabetes$outcome, times = 1, p = 0.2, list = FALSE)
train_set <- pima_diabetes[-test_index,]
test_set <- pima_diabetes[test_index,]
```

Exploratory Analysis

For the initial data exploration, the head() function was used to get a broad understanding of the data.

pregnancies	glucose	bp	skin_thickness	insulin	bmi	dpf	age	outcome
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1
5	116	74	0	0	25.6	0.201	30	0

The table above shows that there seems to be a lot of variation within all the variables, and that a value of 0 for skin_thickness and insulin seems to indicate some missing data. Summary statistics were then calculated to get a better understanding of the variables.

```
##   pregnancies       glucose          bp        skin_thickness    insulin
##  Min.   : 0.00   Min.   :  0   Min.   :  0.0   Min.   : 0.0   Min.   :  0.0
##  1st Qu.: 1.00   1st Qu.: 99   1st Qu.: 62.0   1st Qu.: 0.0   1st Qu.:  0.0
##  Median : 3.00   Median :117   Median : 72.0   Median :23.0   Median : 30.5
##  Mean   : 3.85   Mean   :121   Mean   : 69.1   Mean   :20.5   Mean   : 79.8
##  3rd Qu.: 6.00   3rd Qu.:140   3rd Qu.: 80.0   3rd Qu.:32.0   3rd Qu.:127.2
##  Max.   :17.00   Max.   :199   Max.   :122.0   Max.   :99.0   Max.   :846.0
##       bmi             dpf              age         outcome
##  Min.   : 0.0   Min.   :0.078   Min.   :21.0   0:500
##  1st Qu.:27.3   1st Qu.:0.244   1st Qu.:24.0   1:268
##  Median :32.0   Median :0.372   Median :29.0
##  Mean   :32.0   Mean   :0.472   Mean   :33.2
##  3rd Qu.:36.6   3rd Qu.:0.626   3rd Qu.:41.0
##  Max.   :67.1   Max.   :2.420   Max.   :81.0
```

In the summary statistics presented above, it's observed that the mean number of pregnancies is 3.85, which seems pretty high at first glance but is consistent with previous findings on Native American pregnancy rates and statistics [7]. The maximum value is 17, which is significantly higher than the 75th percentile value of 6. The mean glucose level is 121 mg/dL, which is towards the high end of the normal 70 to 130 mg/dL range [8] and the mean diastolic blood pressure is 69.1, which is well within a normal range. An average skin thickness of 20.5mm is within a normal range [9], and an average insulin of 127.2 μU/mL is within the normal range for an oral test conducted 2 hours after administration of glucose [10]. Interestingly, the mean BMI value of 32 seems to be very high, as the normal range of BMI is 18 to 24, and while a value of 67.1 is extremely high (the max value), it doesn't seem to be an outlier as BMIs have been measured in three figures before. Some concern arose here as the minimum value for glucose, bp, skin_thickness, and bmi are 0, which are not possible and there maybe some missing data to address before any modeling is done. The table also shows that from the 768 women in the Pima dataset, 500 tested negative for diabetes whereas 268 tested positive.

[7]https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2909384/
[8]https://www.diabetes.co.uk/diabetes_care/blood-sugar-level-ranges.html
[9]https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5083983/
[10]https://emedicine.medscape.com/article/2089224-overview

Missing Data

On first glance, it seems as if there isn't any missing data:

```r
sapply(pima_diabetes, function(x) sum(is.na(x)))
```

```
##    pregnancies        glucose             bp skin_thickness        insulin
##              0              0              0              0              0
##            bmi            dpf            age        outcome
##              0              0              0              0
```

However, some missing data is coded as 0's; except for 'outcome' and 'pregnancies', no variable should take a value of 0. Thus, these 0's are replaced using KNN imputation, which replaces these 0's with a value approximated by the values of points closest to it. First, the missing data rows are calculated:

```r
pima_miss <- pima_diabetes[,setdiff(names(pima_diabetes), c('outcome', 'pregnancies'))]
num_miss_features <- apply(pima_miss, 2, function(x) sum(x <= 0))
miss_features <- names(pima_miss)[ num_miss_features > 0]

missing_rows <- apply(pima_miss, 1, function(x) sum(x <= 0) >= 1)
sum(missing_rows)
```

```
## [1] 376
```

Then, the number of total 0's in each variable is ascertained:

```r
pima_miss[pima_miss <= 0] <- NA
pima_diabetes[, names(pima_miss)] <- pima_miss

data <- pima_diabetes
colSums(is.na(pima_diabetes))
```

```
##    pregnancies        glucose             bp skin_thickness        insulin
##              0              5             35            227            374
##            bmi            dpf            age        outcome
##             11              0              0              0
```

There are 227 and 374 0 values for skin_thickness and insulin respectively, which is a large proportion, and definitely needs to be addressed before the modeling process. The knnImputation function in the DMwR package is used:

```r
if(!require(DMwR)) install.packages("DMwR", repos = "http://cran.us.r-project.org")
library(DMwR)
pima_diabetes[,c(-8,-9)] <- knnImputation(pima_diabetes[,c(-8,-9)], k = 5)
```

To check if all 0 values were taken care of:

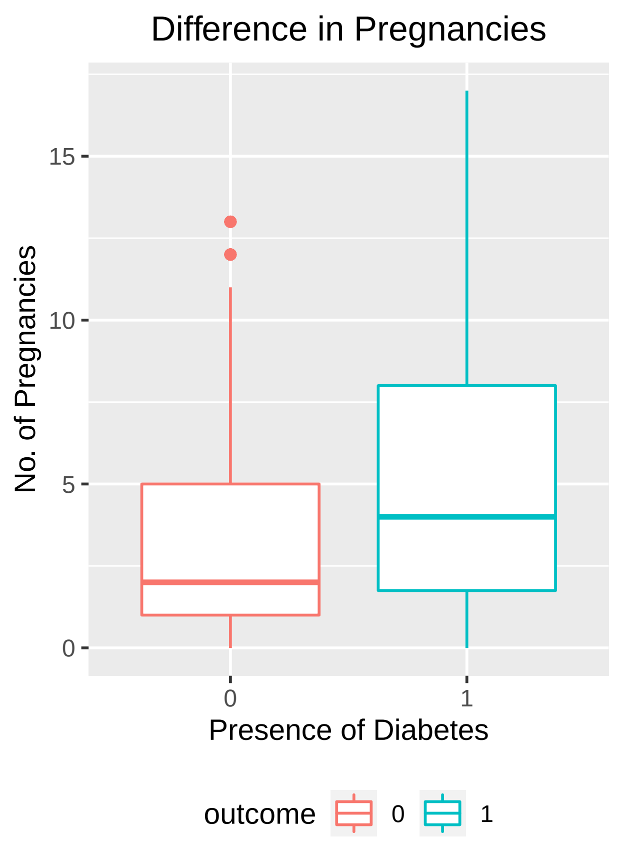
```r
colSums(is.na(pima_diabetes))
```

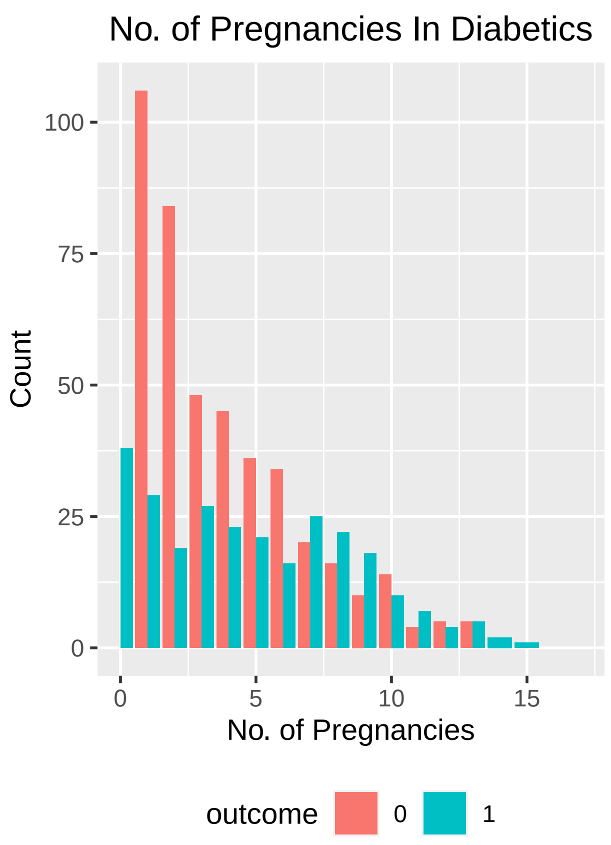
```
##    pregnancies        glucose             bp skin_thickness        insulin
##              0              0              0              0              0
##            bmi            dpf            age        outcome
##              0              0              0              0
```

Plots

The following plots look into each of the 8 predictor variables in detail, in the order they appear in the pima_diabetes dataset.

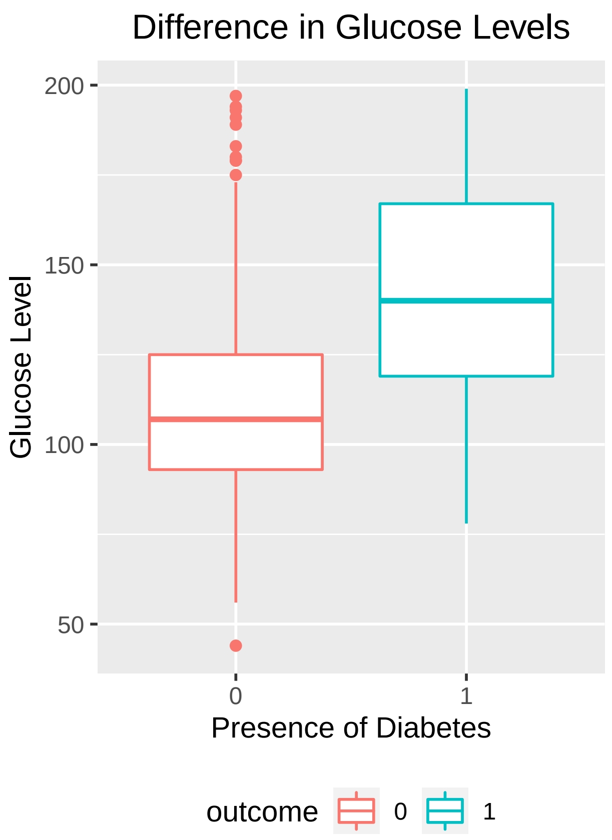
Pregnancies

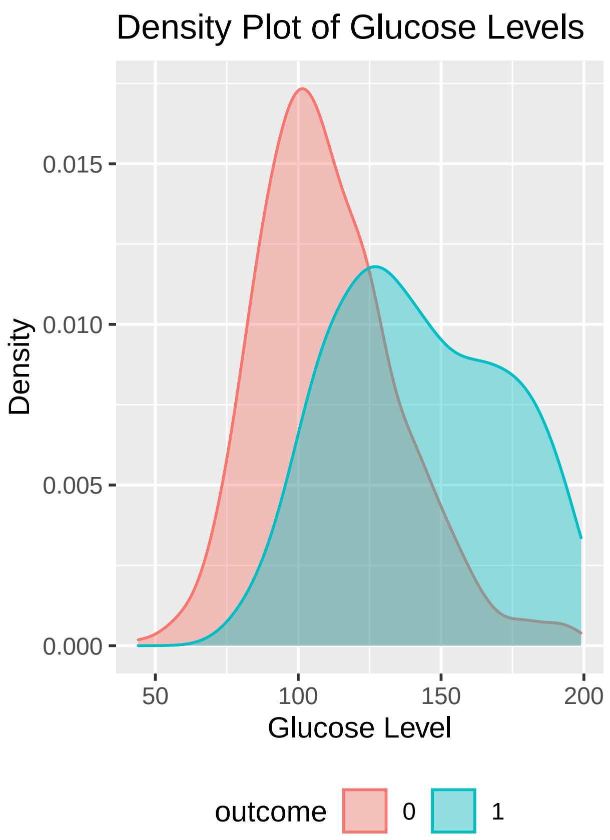




In the boxplot above, it's evident that diabetic women on average have had more pregnancies than non-diabetic women. The histogram shows that there isn't a correlation between the number of pregnancies in diabetic women.

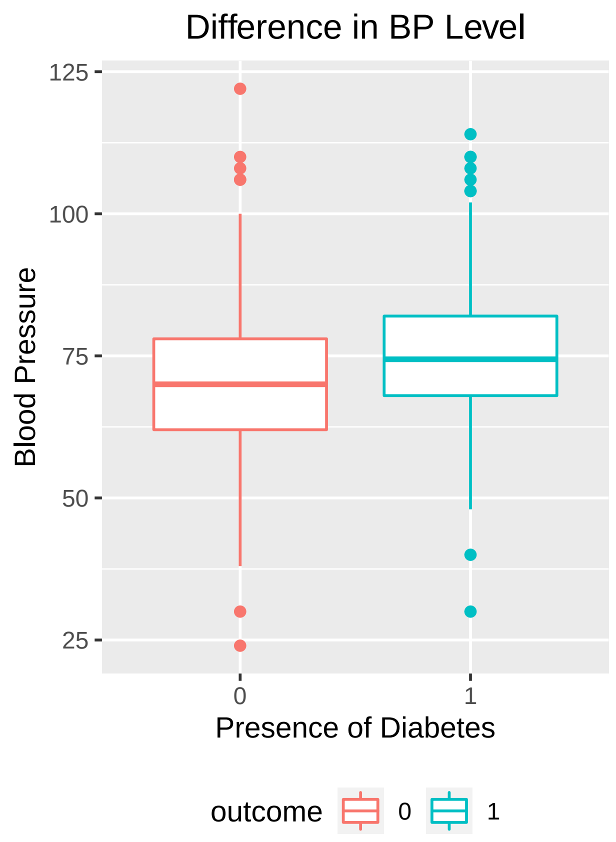
Glucose

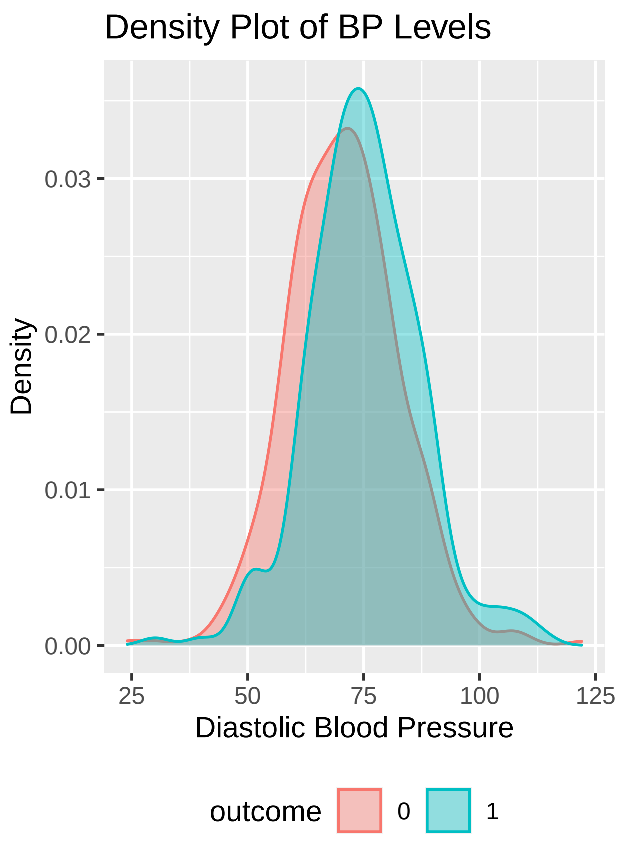




Glucose seems to be a key differentiator in diabetic and non-diabetic women- the average glucose level in diabetic women is ~140mg/dL, compared to ~110mg/dL in non-diabetic women. The density plot above also shows that while there is an overlap, diabetic women tend to have higher levels of glucose. Intuitively, this makes sense as diabetes is characterized by high blood sugar levels.

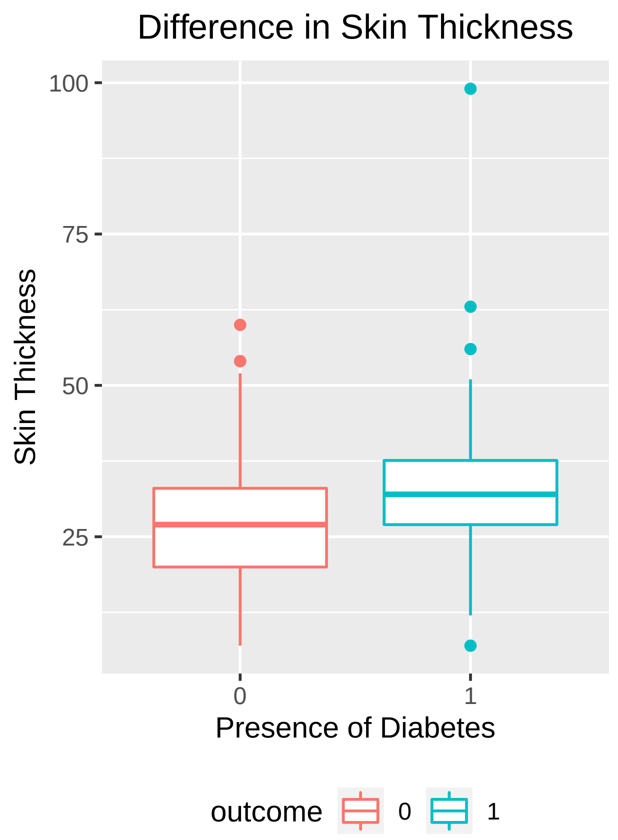
Blood Pressure

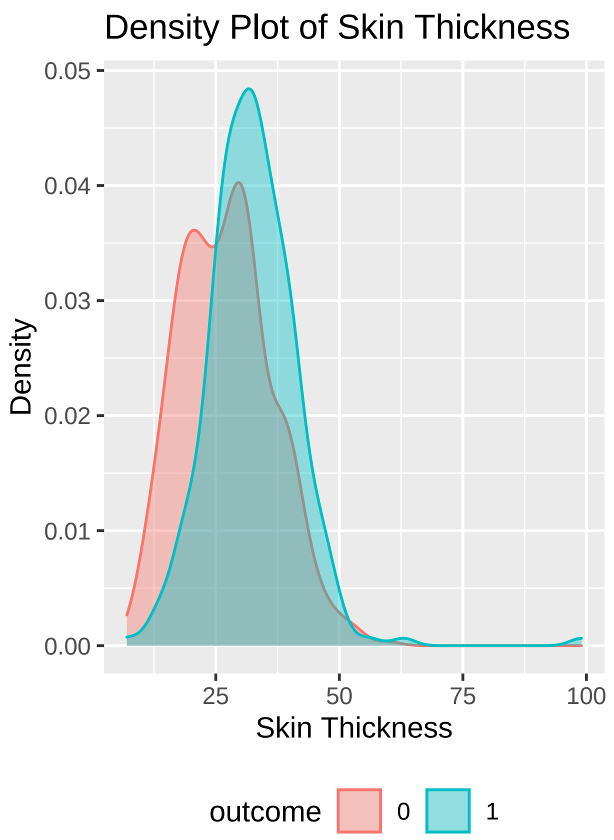




In the boxplot and density plot above, it can be seen that diabetic women have a very slightly higher blood pressure, but the difference doesn't seem to be very significant.

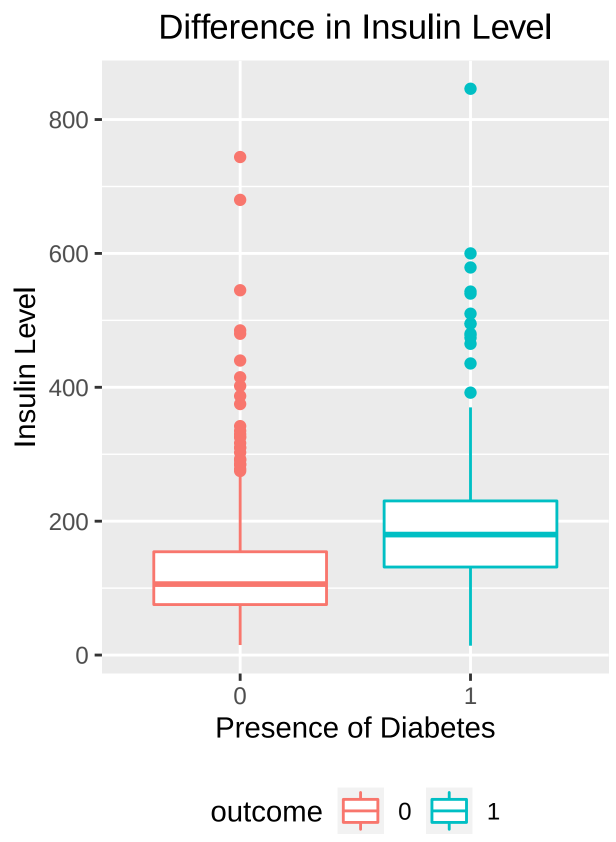
Skin Thickness

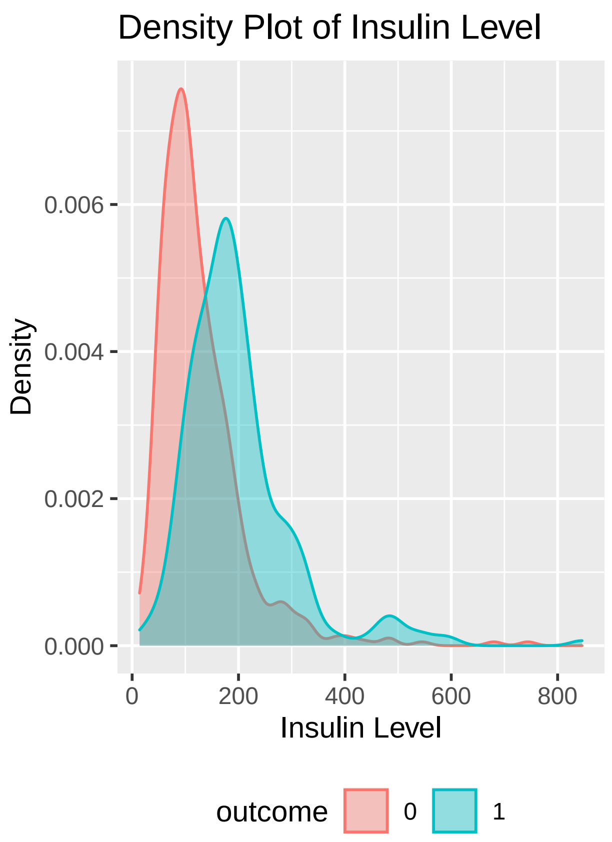




The plots above lead to a similar inference as with the blood pressure plots, and show that diabetic women have slightly thicker skin but not significantly.

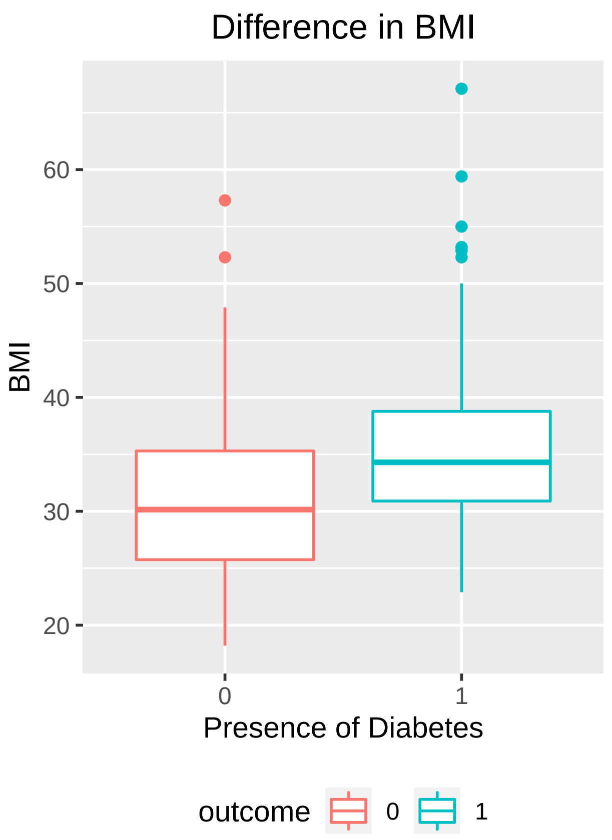
Insulin

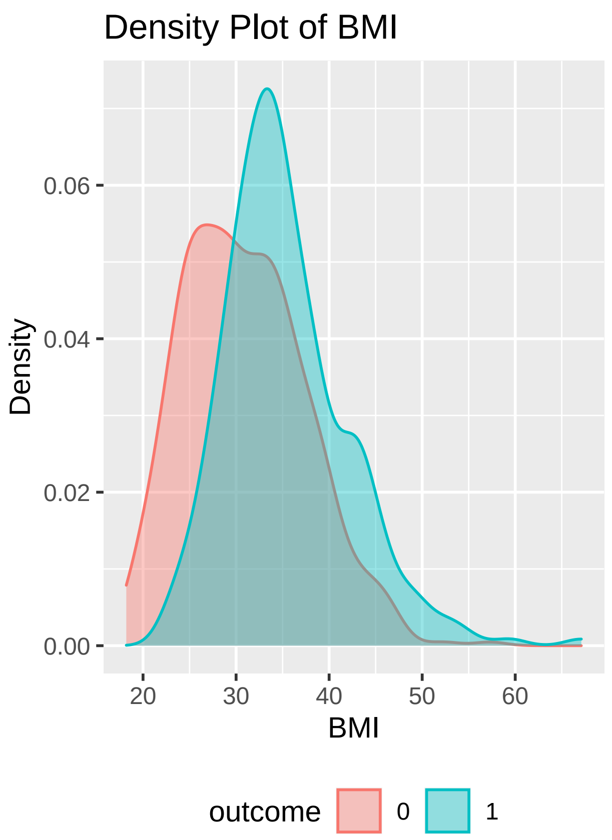




The boxplot above shows that diabetic women have an average insulin of ~180 uU/mL compared to ~110 uU/ml in non-diabetic women; the density plot also confirms that diabetic women tend to have slightly higher insulin levels.

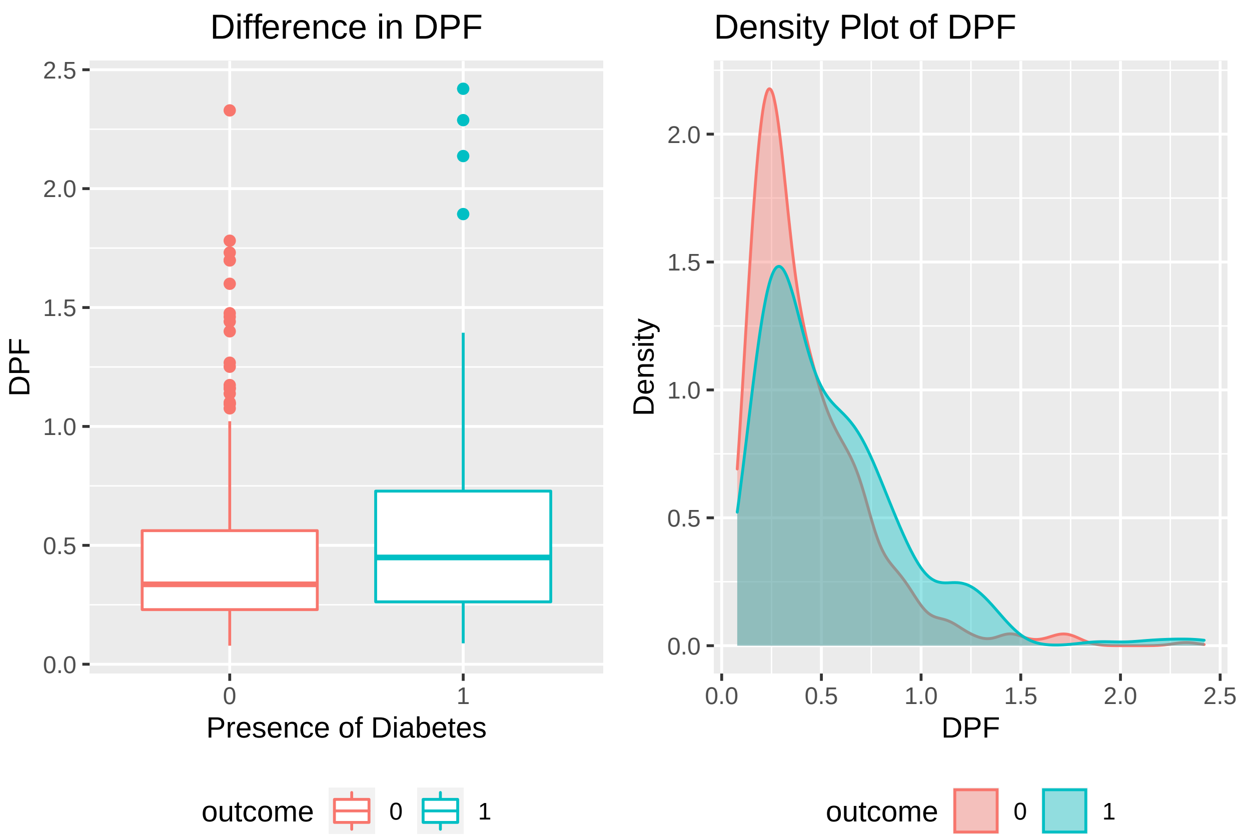
BMI





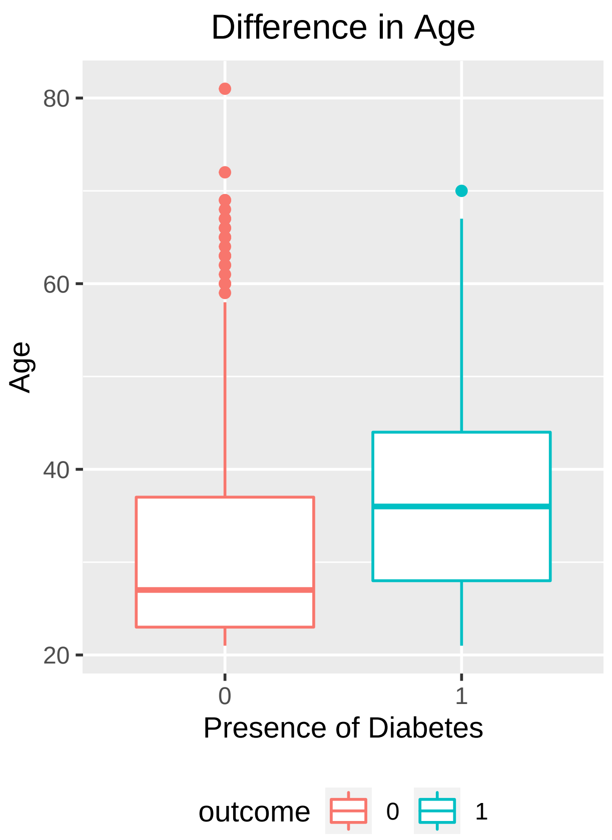
The boxplot above shows that diabetic women tend to have a higher BMI than non-diabetic women; the median for non-diabetic women is 30, which is well above the normal range of 18-24, and the median for diabetic women is ~34. While there is a difference in BMI for both groups, it seems as if Pima women in general tend to have a slightly higher BMI than the national average.

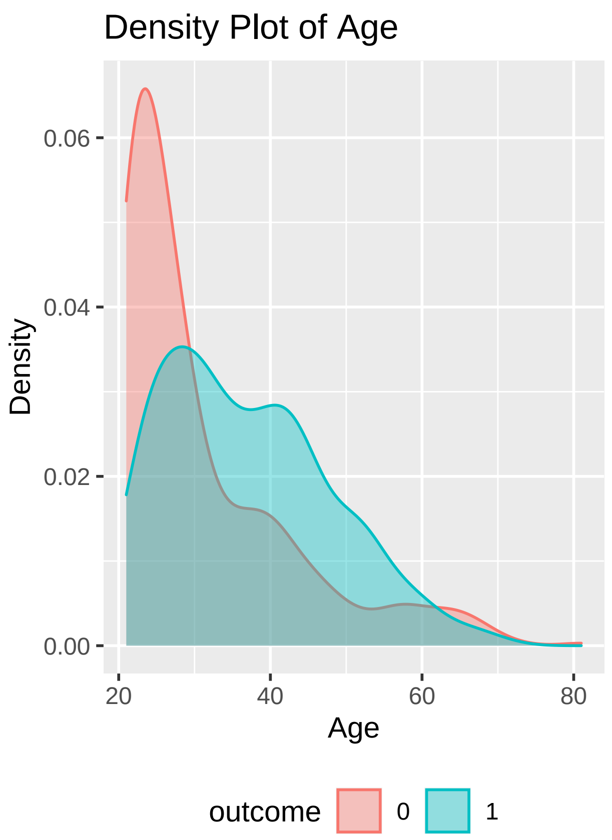
DPF



The DPF plots are interesting, as one would expect family history to increase one's chances of developing diabetes. However, the plots above show that there really isn't a significant difference.

Age



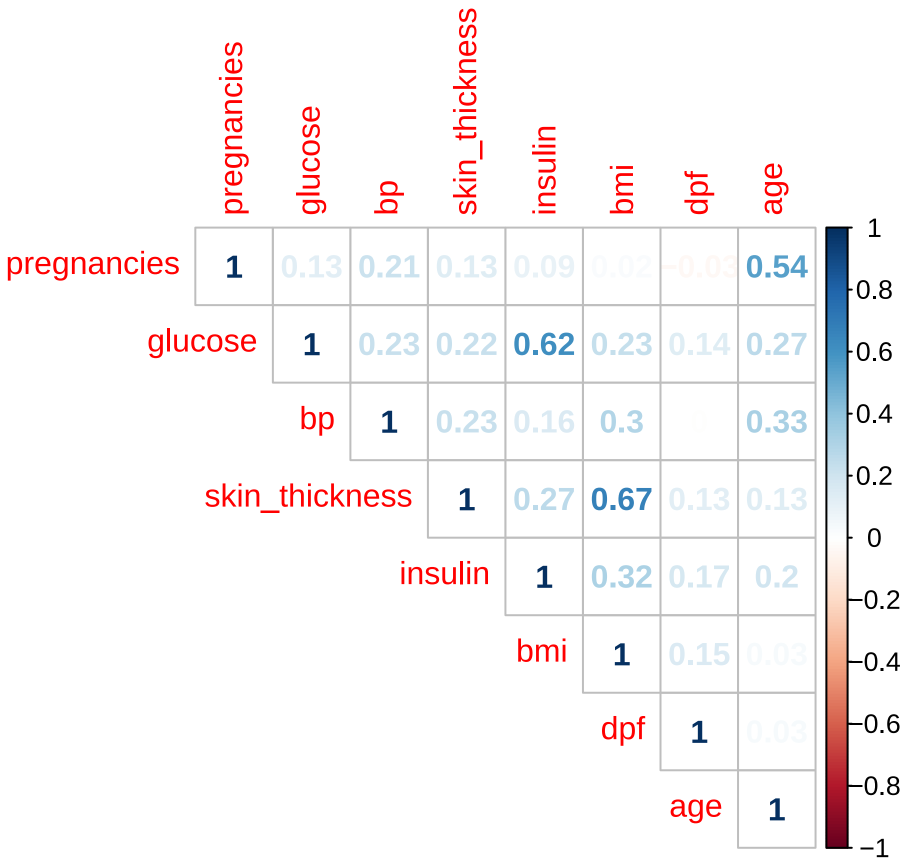


The plots above show that diabetic women tend to be slightly older, but this doesn't seem to be significant as it's possible that some of the younger women may develop diabetes later.

Correlation Matrix

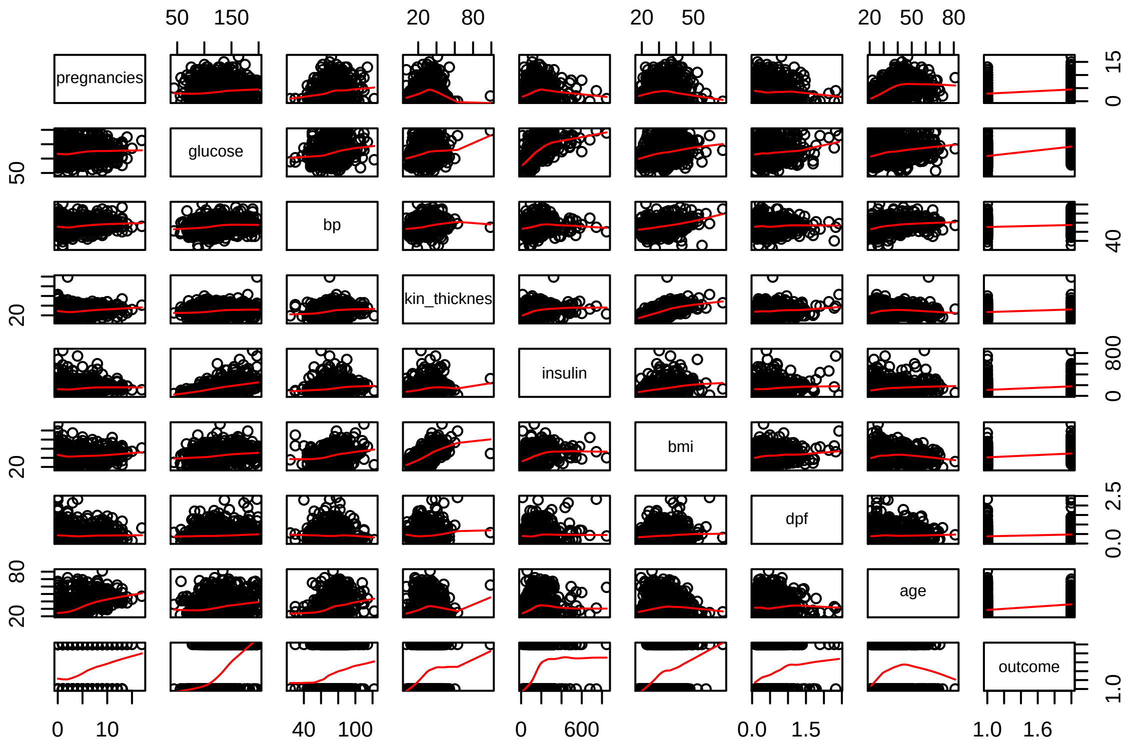
To see the correlations among the variables, a correlation matrix was created:

```r
corrplot(cor(pima_diabetes[, -9]), type = "upper", method = "number")
```



By a rule of thumb, a dataset with correlations above 0.70 can expect to have multicollinearity. While there is no value above 0.70, correlations between bmi and skin_thickness and insulin and glucose are pretty high. While no significant case of multi-multicollinearity is observed and a higher BMI is expected from individuals who have a high skin thickness, it's interesting to note that blood glucose is usually high in the absence of insulin and not it's presence. One reason could be that diabetic individuals may have high levels of insulin, but insulin doesn't work well and is unable to act on glucose, resulting in high levels of both.

The pair-wise plots below show all bivariate relationships:



Modeling

First, a model that randomly guesses whether a patient is diabetic or not is developed. This model doesn't serve any real purpose besides being a stepping stone and providing an accuracy which will become better via the following models: logistic regression and random forests. For both models, ROC curves are made to evaluate their performance on the training set, and then both models are used on the test set to make predictions as a final validation.

Results

Guessing Randomly

First, a baseline prediction was made by simply guessing the outcome. For each individual in the test set, the following code randomly guesses whether or not a person is diabetic by sampling from the vector (0,1):

```
set.seed(123, sample.kind = "Rounding")
guess <- sample(c(0,1), nrow(test_set), replace = TRUE)
mean(guess == test_set$outcome)
```

```
## [1] 0.487013
```

A low accuracy of 0.487 is obtained.

Logistic Regression

For the first model, the glm function is used to perform a logistic regression on 'outcome' using all the predictors.

```
fit_glm <- glm(outcome~.,data=train_set,family = binomial)
summary(fit_glm)
```

```
##
```

```
## Call:
## glm(formula = outcome ~ ., family = binomial, data = train_set)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.546  -0.735  -0.403   0.719   2.849
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -8.195665   0.803239  -10.20  < 2e-16 ***
## pregnancies     0.124033   0.035767    3.47  0.00052 ***
## glucose         0.037689   0.004289    8.79  < 2e-16 ***
## bp             -0.015073   0.005801   -2.60  0.00936 **
## skin_thickness  0.004449   0.007749    0.57  0.56585
## insulin        -0.001653   0.000968   -1.71  0.08777 .
## bmi             0.073743   0.016493    4.47  7.8e-06 ***
## dpf             0.985006   0.346170    2.85  0.00443 **
## age             0.016129   0.010318    1.56  0.11801
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 793.94  on 613  degrees of freedom
## Residual deviance: 575.17  on 605  degrees of freedom
## AIC: 593.2
##
## Number of Fisher Scoring iterations: 5
```

Then, the insignificant variables (i.e. age, insulin and skin_thickness) are dropped [11]:

```
fit_glm2 <- glm(formula = outcome ~ pregnancies + glucose + bmi + dpf,
                family = binomial,
                data = train_set)
summary(fit_glm2)
```

```
##
## Call:
## glm(formula = outcome ~ pregnancies + glucose + bmi + dpf, family = binomial,
##     data = train_set)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.801  -0.734  -0.418   0.754   2.895
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -8.20133    0.72213  -11.36  < 2e-16 ***
## pregnancies  0.14343    0.03007    4.77  1.8e-06 ***
## glucose      0.03556    0.00387    9.20  < 2e-16 ***
## bmi          0.06367    0.01503    4.23  2.3e-05 ***
## dpf          0.93963    0.33507    2.80    0.005 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

[11] bp was also dropped as it loses significance when in a model that excluded age, insulin and skin_thickness

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 793.94  on 613  degrees of freedom
## Residual deviance: 586.89  on 609  degrees of freedom
## AIC: 596.9
##
## Number of Fisher Scoring iterations: 5
```

All the variables in this model are significant, and have a positive relationship with 'outcome'- this makes sense, as all these variables had higher values in diabetic women as seen in the previous exploratory analysis.

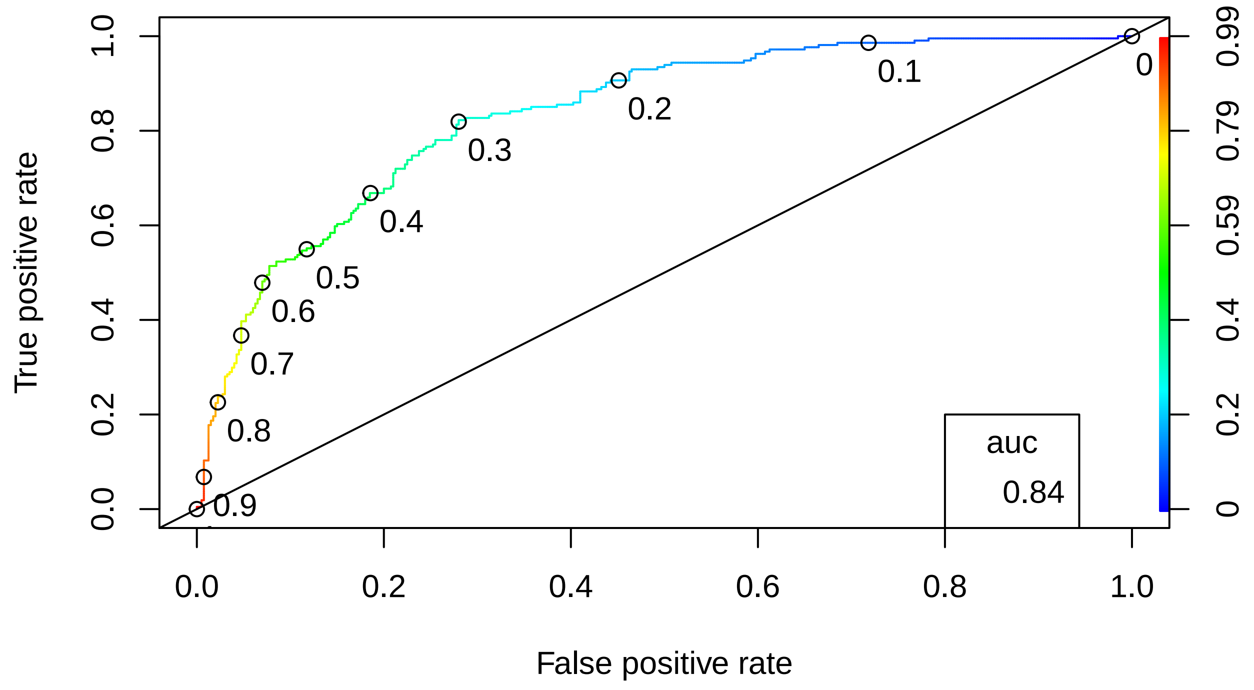
Using this model, predictions are made on the training set:

```
pred_glm <- predict(fit_glm2, type="response")
tapply(pred_glm, train_set$outcome, mean)
```

```
##        0        1
## 0.239494 0.552347
```

The average prediction for 0, or 'not diabetic' is 0.239, and the average prediction for 1, or 'diabetic' is 0.552.

The Receiver Operating Characteristic curve is used to determine the accuracy of a continuous variable for predicting a binary outcome, such as the one in the pima_diabetes dataset. The curve has 2 parameters- a false positive rate, and a true positive rate. The ROCR package allows for the following ROC curve to be made. Additionally, AUC (area under the curve), a measure of the area under the ROC curve is also calculated to evaluate the accuracy of the model's prediction.



An accuracy of ~84% is reported on the train_set. It is then tested on the test_set:

```
test_pred <- predict(fit_glm2, type="response", newdata = test_set)
test_table <- table(test_set$outcome, test_pred >0.5)
test_table %>% knitr::kable()
```

	FALSE	TRUE
0	90	10
1	23	31

```r
test_accuracy <- round(sum(diag(test_table))/sum(test_table),2)
print(test_accuracy)
```

```
## [1] 0.79
```

The accuracy of this model is 0.79.

Random Forest

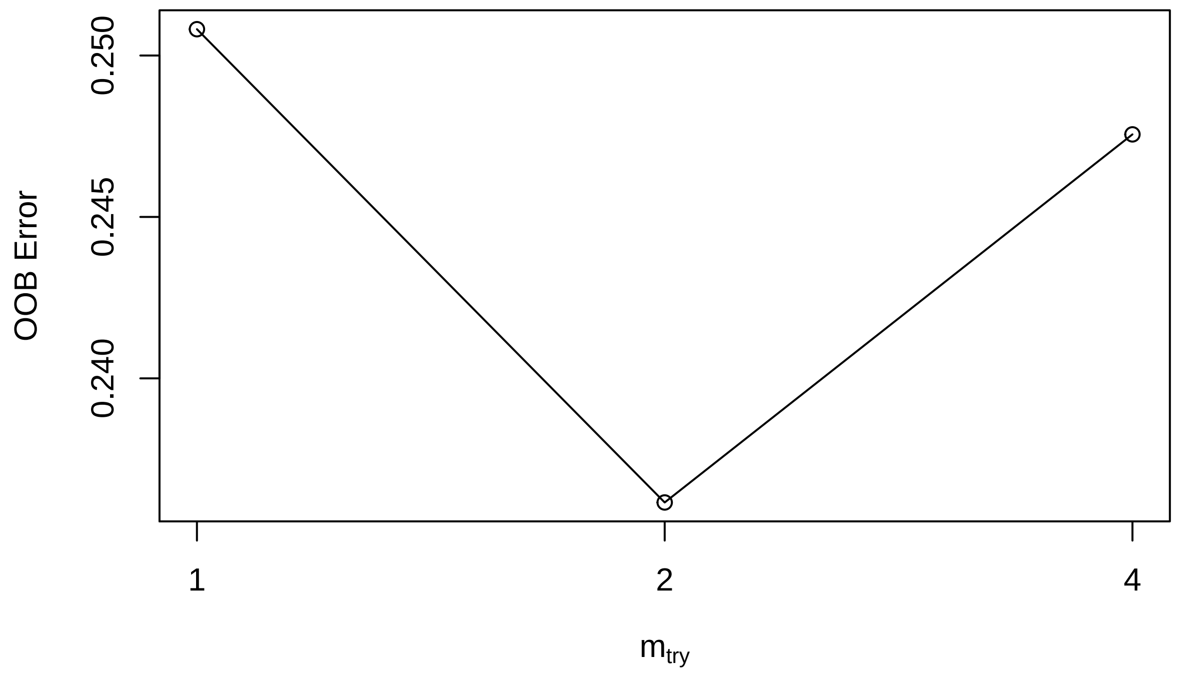
The process of random forests is about growing several 'decision trees', trained on different subsets of the dataset and different variables for each 'tree'- each 'tree' takes into consideration a random subset of predictors, and classifies the outcome variable. The 'random forest' then takes into account the outcome of the 'trees' and combines them to produce a final outcome. The mtry parameter sets the number of variables used to build each 'tree'. [12]

To select the best mtry, parameter tuning is done by plotting the OOB error (an overall sum of misclassification of negative and positive classes) against mtry values:

```r
library(randomForest)
set.seed(123, sample.kind = "Rounding")
fit_rf_tuning <- tuneRF(x = subset(train_set, select = -outcome),
              y = train_set$outcome,
              ntreeTry = 500,
              plot = TRUE,
              tunecontrol = tune.control(cross = 5))
```

```
## mtry = 2  OOB error = 23.62%
## Searching left ...
## mtry = 1     OOB error = 25.08%
## -0.062069 0.05
## Searching right ...
## mtry = 4     OOB error = 24.76%
## -0.0482759 0.05
```

[12] More can be found on https://towardsdatascience.com/understanding-random-forest-58381e0602d2



The graph above shows that a mtry value of 2 gives the lowest positive misclassification error, and is therefore selected as the best value for building the random forest model.

```
set.seed(123, sample.kind = "Rounding")
fit_rf <- randomForest(outcome~., data = train_set, importance = TRUE, mtry = 2)
fit_rf
```

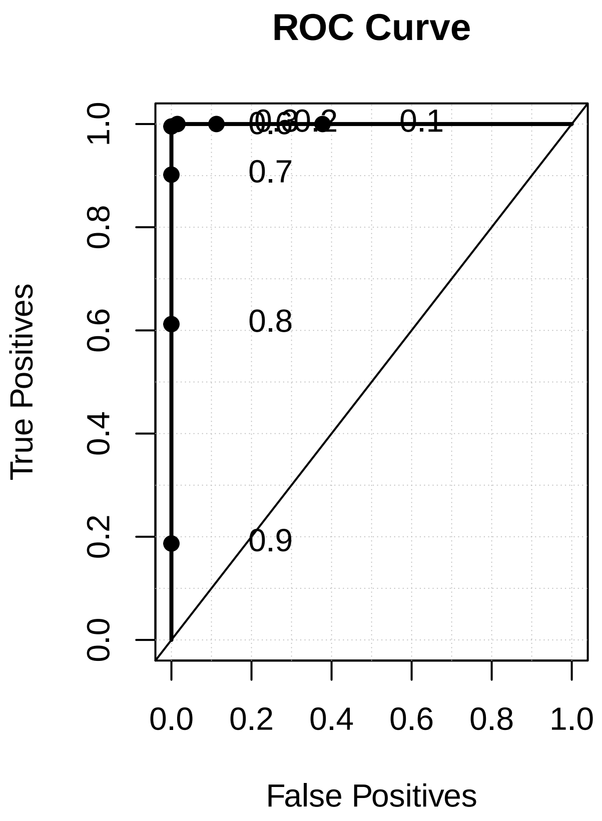
```
##
## Call:
##  randomForest(formula = outcome ~ ., data = train_set, importance = TRUE,      mtry = 2)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 25.24%
## Confusion matrix:
##     0   1 class.error
## 0 338  62    0.155000
## 1  93 121    0.434579
```

The accuracy of this model is ~75%. It is then evaluated on the test_set. First, the predictions and ROC curve are done for the train_set:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 400   0
##          1   0 214
##
##                Accuracy : 1
##                  95% CI : (0.994, 1)
##     No Information Rate : 0.651
##     P-Value [Acc > NIR] : <2e-16
```

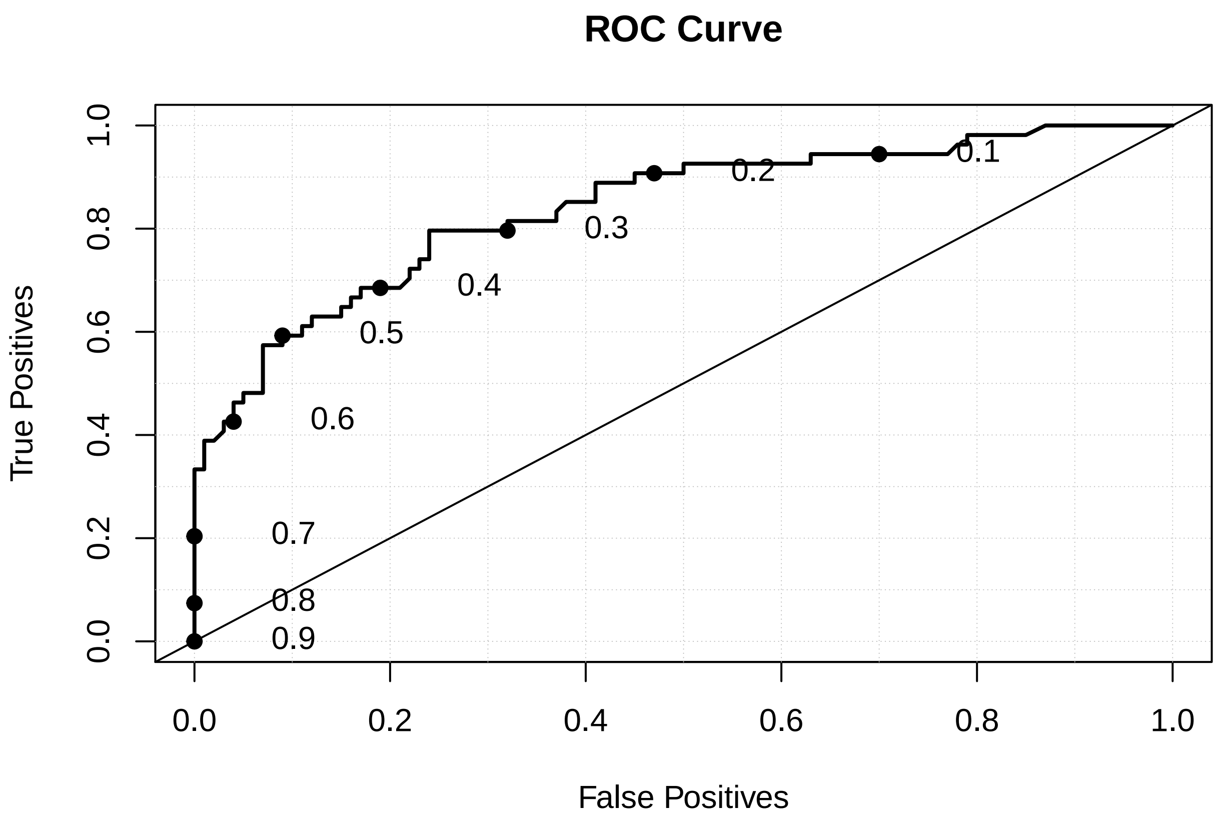
```
##
##                  Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.000
##             Specificity : 1.000
##          Pos Pred Value : 1.000
##          Neg Pred Value : 1.000
##              Prevalence : 0.651
##          Detection Rate : 0.651
##    Detection Prevalence : 0.651
##       Balanced Accuracy : 1.000
##
##        'Positive' Class : 0
##
```

```
##      Model Area     p.value binorm.area
## 1 Model  1    1 4.09836e-93          NA
```



Then, the predictions and ROC curve are done for the test_set:

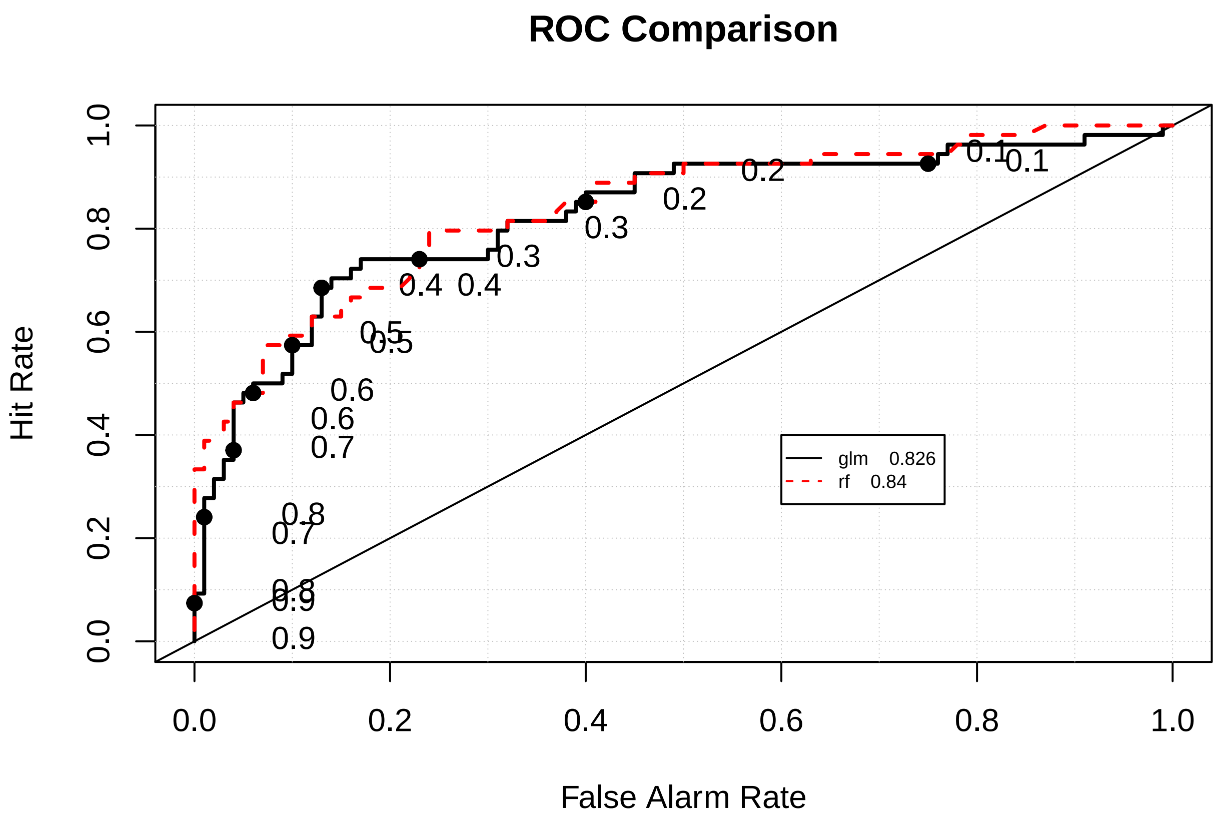
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 91  9
##          1 22 32
##
##                Accuracy : 0.799
##                  95% CI : (0.727, 0.859)
##     No Information Rate : 0.734
##     P-Value [Acc > NIR] : 0.0388
##
##                   Kappa : 0.532
##
##  Mcnemar's Test P-Value : 0.0311
##
##             Sensitivity : 0.805
##             Specificity : 0.780
##          Pos Pred Value : 0.910
##          Neg Pred Value : 0.593
##              Prevalence : 0.734
##          Detection Rate : 0.591
##    Detection Prevalence : 0.649
##       Balanced Accuracy : 0.793
##
##        'Positive' Class : 0
##
```



```
##      Model Area     p.value binorm.area
## 1 Model   1 0.84 1.82452e-12          NA
```

The accuracy of the RF model is 0.799. An ROC comparison of both models on the test_set is then done:

```
rocplot <- roc.plot(x = (test_set$outcome == "1"),
                     pred = cbind(test_pred, rf_pred2[,2]),
                     main = "ROC Comparison",
                     legend = T, leg.text = c("glm", "rf"))
```



The above plot shows that the random forest model is slightly preferable to the logistic regression model.

Conclusion

The objective of this project was to build an algorithm that can predict if an individual is diabetic or not using data in the Pima Indians Diabetes dataset. Two models were built, one using logistic regression that subsequently drops insignificant variables, and one using random forests. The random forest model seems to be marginally better, but the difference between the accuracy of the logistic regression model and random forests model (0.79 and 0.799 respectively) is negligible.

Limitations and Future Work

As mentioned earlier, there are a couple high correlations between the predictors, which could cause some multicollinearity issues which weren't addressed in this project. Additionally, missing data (0's) were hard to detect and were replaced using KNN imputation- an alternative could've been to use mean/median values instead. Furthermore, especially in the RF model, there may be an issue of overfitting as an accuracy of 1 is reported when applying the model to the train_set.

A more accurate prediction algorithm could be created using an SVM algorithm, and a classification trees model could be used in addition to random forests.

Appendix

Acknowledgments

I would like to thank Prof. Irizarry and the entire edX staff (especially the discussion forums) for giving me this opportunity to pursue the Data Science Professional Certificate. COVID-19 has been a challenging time for all, and this platform and project have allowed me to use my time productively.

Environment

```r
print("Operating System:")
```

```
## [1] "Operating System:"
```

```r
version
```

```
##                _
## platform       x86_64-apple-darwin15.6.0
## arch           x86_64
## os             darwin15.6.0
## system         x86_64, darwin15.6.0
## status
## major          3
## minor          6.1
## year           2019
## month          07
## day            05
## svn rev        76782
## language       R
## version.string R version 3.6.1 (2019-07-05)
## nickname       Action of the Toes
```