Diabetes Logistic Regression

Kaia Lindberg

12/11/2021

Set up

```
# Import libraries
library(tidyverse)
library(ROCR)
```

```
# Load data
Data <- read.csv("diabetes2.csv", header=T)
print(head(Data))
```

```
##   Pregnancies Glucose BloodPressure SkinThickness Insulin  BMI
## 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
##   DiabetesPedigreeFunction Age Outcome
## 1                    0.627  50       1
## 2                    0.351  31       0
## 3                    0.672  32       1
## 4                    0.167  21       0
## 5                    2.288  33       1
## 6                    0.201  30       0
```

The data has 768 rows and 9 columns. One of these columns (Outcome) will be our response variable for our logistic regression and the others are potential predictors.

```
# Check dimensions of data
print(dim(Data)) # 768 rows and 9 columns
```

```
## [1] 768   9
```

```
# Display names of all columns
print(colnames(Data))
```

```
## [1] "Pregnancies"              "Glucose"
## [3] "BloodPressure"            "SkinThickness"
## [5] "Insulin"                  "BMI"
## [7] "DiabetesPedigreeFunction" "Age"
## [9] "Outcome"
```

Before we go any further with our analysis we will split our data into a training and test set. We've chosen a random seed of "123" for reproducibility. We will do all further analysis, visualization, and model building using our training data and then use our test data to evaluate our model's performance on unseen data. Since our data isn't particularly large (768 rows), we chose a 60/40 split so that we would have more data for training and thus have more stable model results.

```
# Randomly split data into two halves
set.seed(123) # For reproducability
sample<-sample.int(nrow(Data), floor(.60*nrow(Data)), replace = F)
train<-Data[sample, ] # Training data
test<-Data[-sample, ] # Test data
```

Data Cleaning

Before creating our linear models we needed to make sure that our data was clean. First, we checked for missing values. At first glance there did not appear to be any missing values.

```
# Check for missing values in each column
colSums(is.na(train))
```

```
##              Pregnancies                  Glucose            BloodPressure
##                        0                        0                        0
##            SkinThickness                  Insulin                      BMI
##                        0                        0                        0
## DiabetesPedigreeFunction                      Age                  Outcome
##                        0                        0                        0
```

Next, we created a summary of each variable in our data including the mean and five number summary.

```
# Summary of columns
print(summary(train))
```

```
##   Pregnancies        Glucose      BloodPressure    SkinThickness
##  Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00
##  1st Qu.: 1.000   1st Qu.:100.0   1st Qu.: 62.00   1st Qu.: 0.00
##  Median : 3.000   Median :119.5   Median : 72.00   Median :23.00
##  Mean   : 3.846   Mean   :122.0   Mean   : 68.95   Mean   :20.91
##  3rd Qu.: 6.000   3rd Qu.:141.0   3rd Qu.: 80.00   3rd Qu.:33.00
##  Max.   :17.000   Max.   :199.0   Max.   :110.00   Max.   :99.00
##     Insulin            BMI        DiabetesPedigreeFunction      Age
##  Min.   :  0.00   Min.   : 0.00   Min.   :0.0880           Min.   :21.00
##  1st Qu.:  0.00   1st Qu.:27.50   1st Qu.:0.2437           1st Qu.:24.00
##  Median : 39.00   Median :32.10   Median :0.3690           Median :29.00
##  Mean   : 82.21   Mean   :32.19   Mean   :0.4753           Mean   :33.29
##  3rd Qu.:130.00   3rd Qu.:36.73   3rd Qu.:0.6145           3rd Qu.:40.25
##  Max.   :846.00   Max.   :67.10   Max.   :2.3290           Max.   :70.00
##     Outcome
##  Min.   :0.0000
##  1st Qu.:0.0000
##  Median :0.0000
##  Mean   :0.3609
##  3rd Qu.:1.0000
##  Max.   :1.0000
```

This summary shows us the distribution of each variable. For example, we can see that the patients in the dataset range from age 21 to 70 with a median of 29 and mean of 33.3. Since we mean age is greater than the median age we can tell that the age distribution is skewed right. From this summary of the data we also observe some non-sensical values. For example some participants have a recorded BMI of 0 or an insulin value of 0, which cannot occur logically. Although there were no missing values above, we suspect that 0 was zero was recorded in place of missing data. The only variables for which a zero makes sense are pregnancies and outcome. For any other variable we'll fill zero with unknown and then evaluate the best way to handle unknowns. We will do this for both the train and test sets even though we won't be using the test data until later.

```
# List of columns where zero is non-sensical (i.e. zero indicates unknown)
# All predictor columns other than pregnancies
zero_unknown_cols <-c("Glucose", "BloodPressure", "SkinThickness",
                      "Insulin", "BMI", "DiabetesPedigreeFunction",
                      "Age")

# Replace zeros with unknowns
train[,zero_unknown_cols] <- replace(train[,zero_unknown_cols], train[,zero_unknown_cols]==0,NA)
test[,zero_unknown_cols] <- replace(test[,zero_unknown_cols], test[,zero_unknown_cols]==0,NA)
```

Now that we have encoded our unknown values we checked what percent of observations were unknown for each variable.

```
# Check for missing values in each column again
round(colSums(is.na(train))/dim(train)[1],2)
```

```
##              Pregnancies                  Glucose            BloodPressure
##                     0.00                     0.01                     0.05
##            SkinThickness                  Insulin                      BMI
##                     0.30                     0.48                     0.01
## DiabetesPedigreeFunction                      Age                  Outcome
##                     0.00                     0.00                     0.00
```

We observed that a large portion of some columns have missing data, especially Insulin with 48% missing and SkinThickness with 30%. Now that we've identified the missing values we needed to handle them before we continue with our analysis. We removed variables with over 25% missing (Insulin and Skin Thickness) because we didn't think they would be reliable predictors with so much missing data and we didn't want to drop nearly 50% of our data.

```
# Remove variables with high percent missing
train <- select(train, -c('SkinThickness', 'Insulin'))
test <- select(test, -c('SkinThickness', 'Insulin'))
```

For the remaining data we chose to remove any unknowns. Before doing so we checked what percent of the data would be dropped. Seeing that this was only about 6% we thought that was reasonable given that it would give us a complete dataset so we went ahead and dropped any rows with missing values.

```
# Check what percent of our test data we will drop
round(sum(!complete.cases(train))/dim(train)[1],2)
```

```
## [1] 0.06
```

```
# Drop rows with missing data from our train and test data
train <- train[complete.cases(train), ]
test <- test[complete.cases(test), ]
```

Our outcome variable (Diabetes) was originally labeled as 0/1. We converted this to a factor and labeled the outcomes so that R treated it as a categorical response.

```
# Create diabetes factor column with labels from outcome column
train$Outcome<-factor(train$Outcome)
levels(train$Outcome) <- c("No", "Yes")
```

```r
# For the test data too
test$Outcome<-factor(test$Outcome)
levels(test$Outcome) <- c("No", "Yes")
```

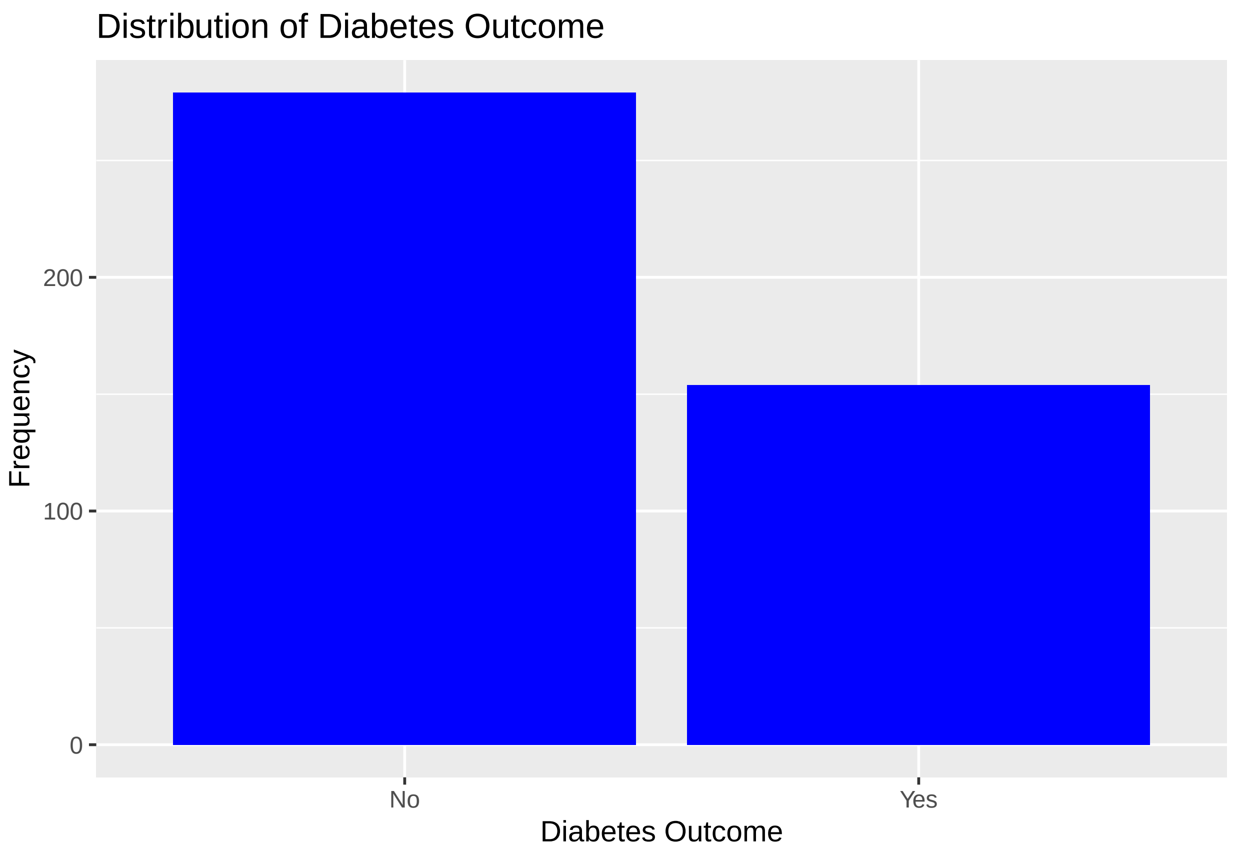
Analysis and Visualization

Distribution of Reponse Variable

```r
# Calculate proportion of patients with diabetes
outcome_prop <- round(prop.table(table(train$Outcome)),2)
outcome_prop
```

```
##
##   No  Yes
## 0.64 0.36
```

```r
# Bar plot of distribution
ggplot(train, aes(x=Outcome)) +
  geom_bar(fill="blue") +
  labs(x="Diabetes Outcome", y="Frequency", title="Distribution of Diabetes Outcome")
```

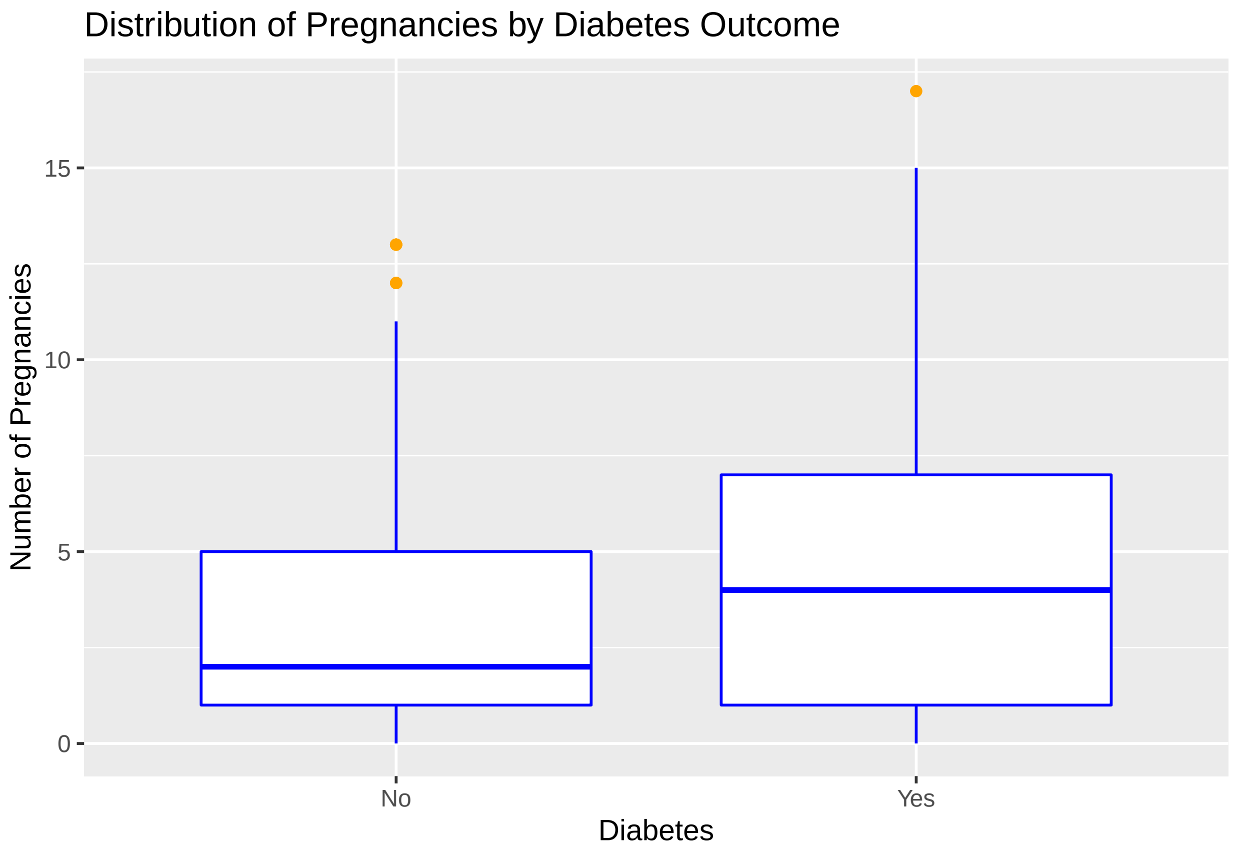


About 36% of patients in this data set have diabetes while 64% do not. More don't have diabetes, but this is not a huge imbalance.

Visualizaling Response and Potential Predictors

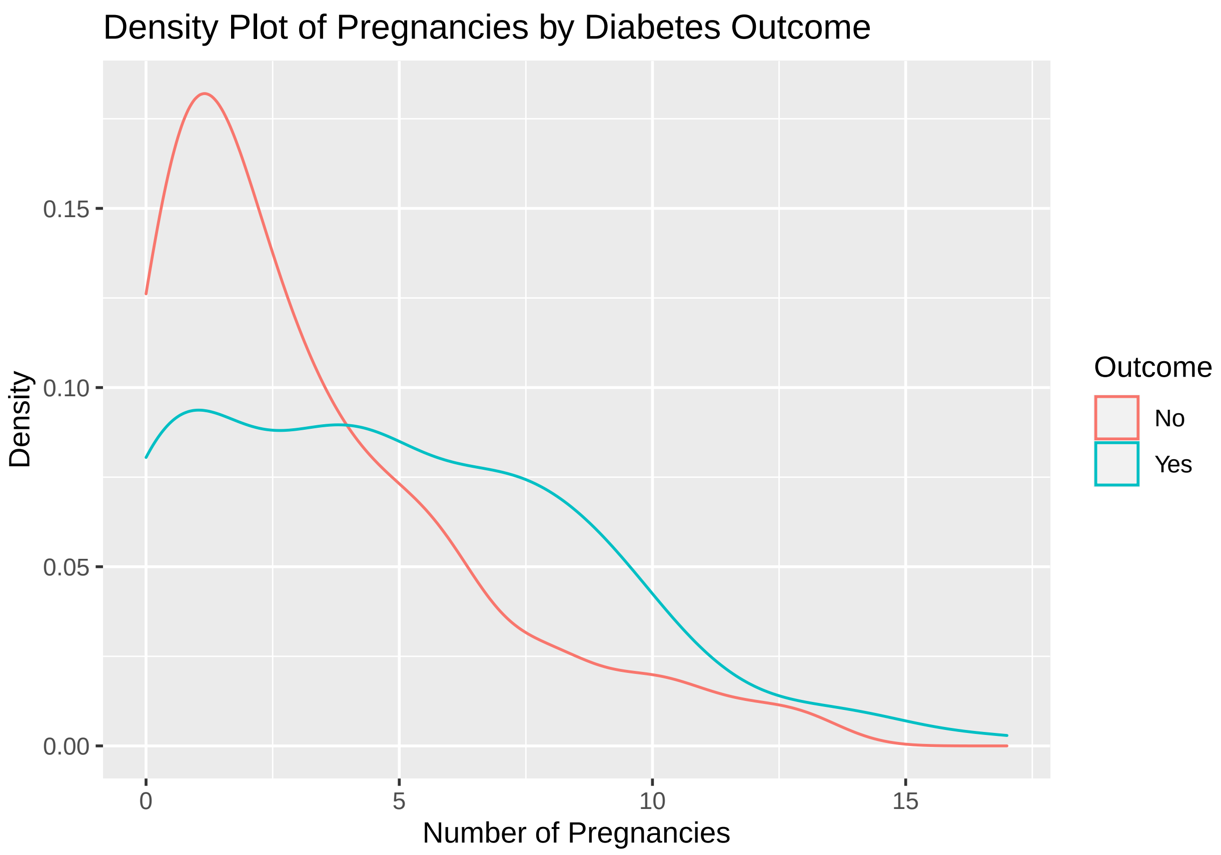
Pregnancies

```
# Side by side box plot
ggplot(train, aes(x=Outcome, y=Pregnancies))+
  geom_boxplot(color = "blue", outlier.color = "orange") +
  labs(x="Diabetes", y="Number of Pregnancies", title="Distribution of Pregnancies by Diabetes Outcome")
```



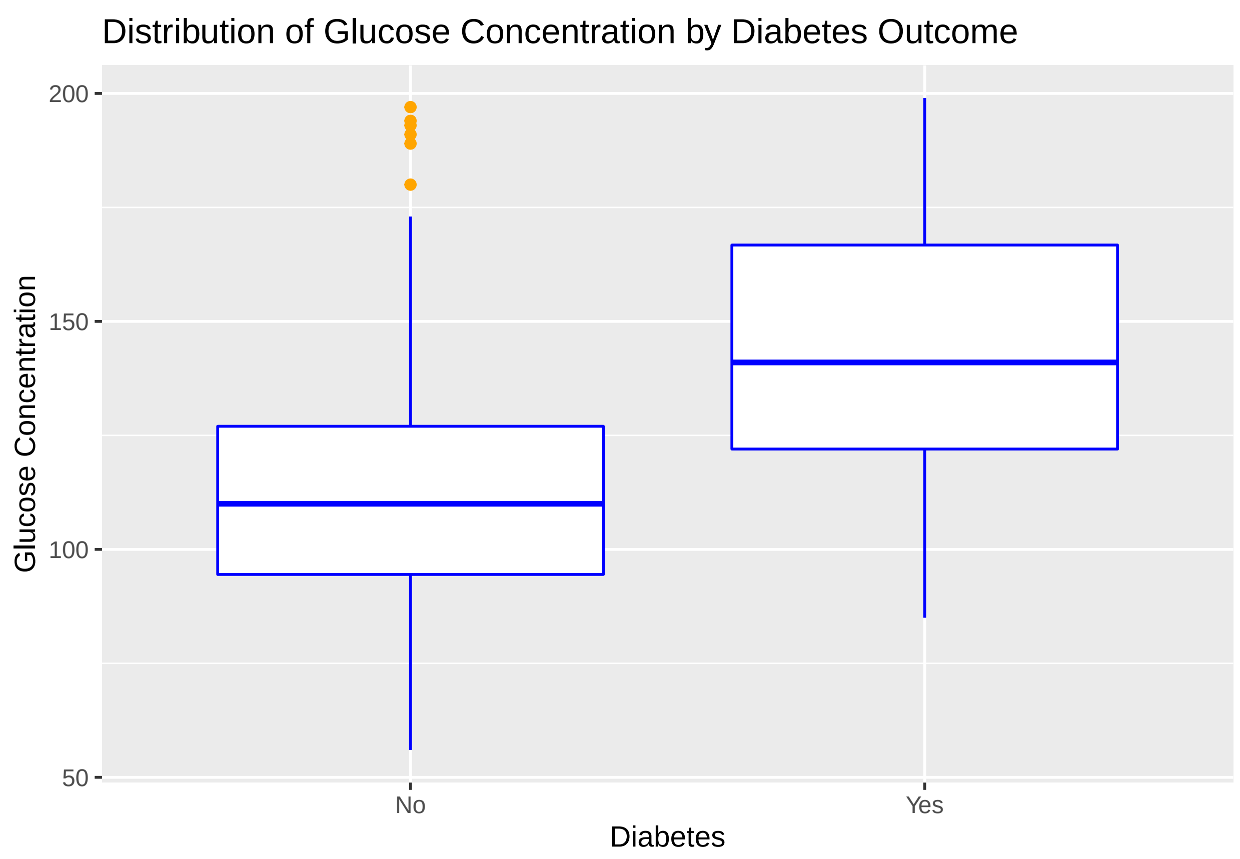
There is a fair amount of overlap in these box plots, but they do suggest that patients with diabetes tend to have higher number of pregnancies. We also observe this in the density plots where patients without diabetes are more likely to have had small numbers of or no pregnancies.

```
# Density plot
ggplot(train, aes(x=Pregnancies, color=Outcome)) +
  geom_density() +
  labs(x="Number of Pregnancies", y="Density", title="Density Plot of Pregnancies by Diabetes Outcome")
```



### Glucose

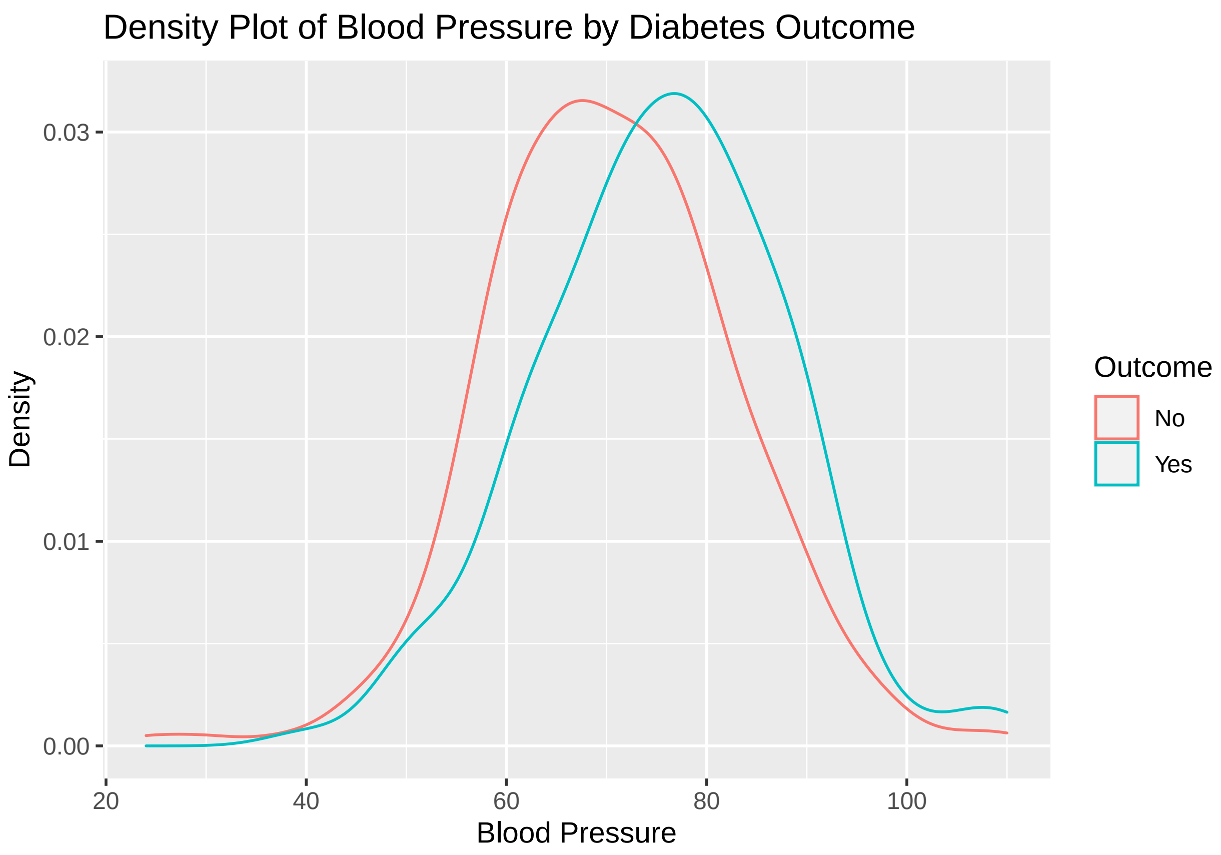
```
# Side by side box plot
ggplot(train, aes(x=Outcome, y=Glucose))+
  geom_boxplot(color = "blue", outlier.color = "orange") +
  labs(x="Diabetes", y="Glucose Concentration", title="Distribution of Glucose Concentration by Diabete
```



These box plots (especially from Q1 to Q3) have little overlap so we suspect that glucose will be predictive in identifying which patients have diabetes. From this visual we can see that patients with diabetes tend to have higher glucose concentrations.

Blood Pressure

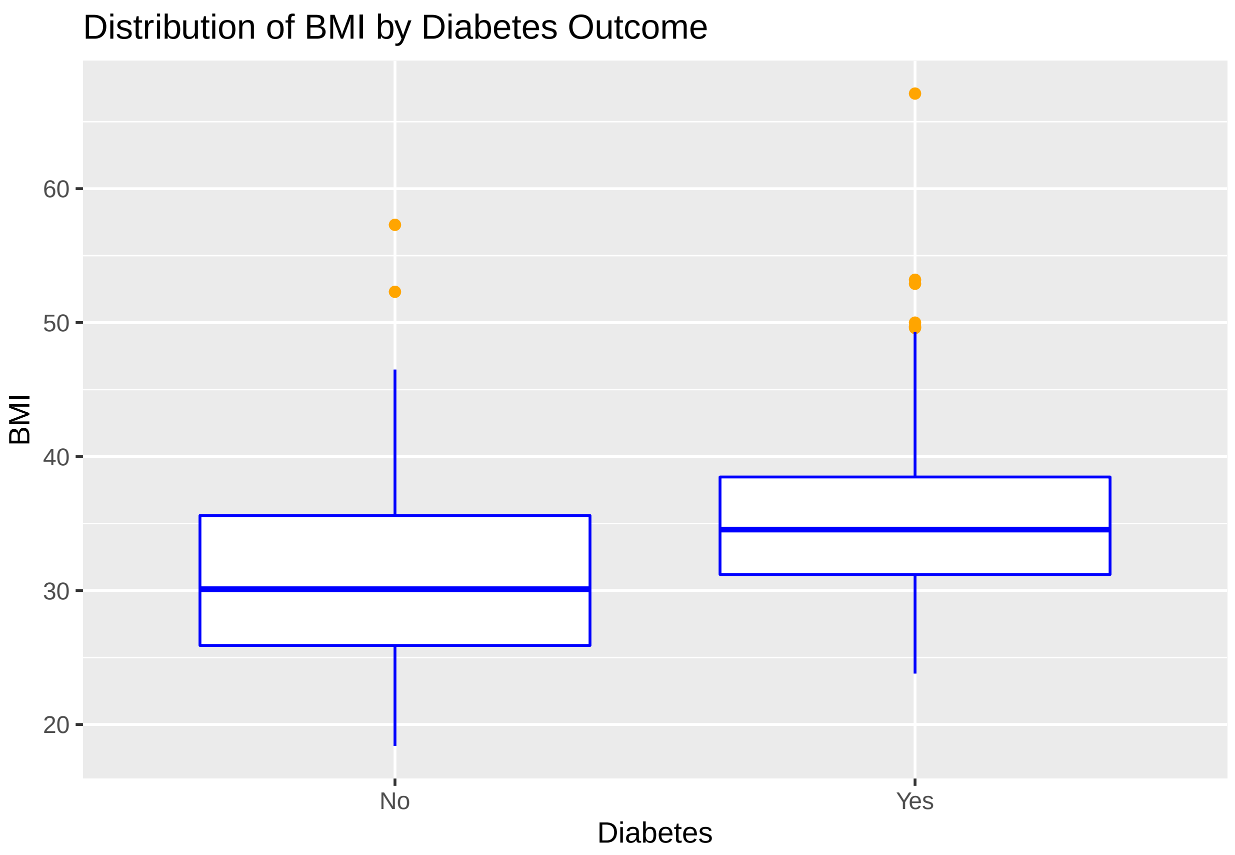
```
# Density plot
ggplot(train, aes(x=BloodPressure, color=Outcome)) +
  geom_density() +
  labs(x="Blood Pressure", y="Density", title="Density Plot of Blood Pressure by Diabetes Outcome")
```



There is quite a bit of overlap in the density plots of blood pressure for those that do and do not have diabetes. While there is a lot of overlap we do observe that those with diabetes tend to have slightly higher blood pressure than those that do not.

BMI

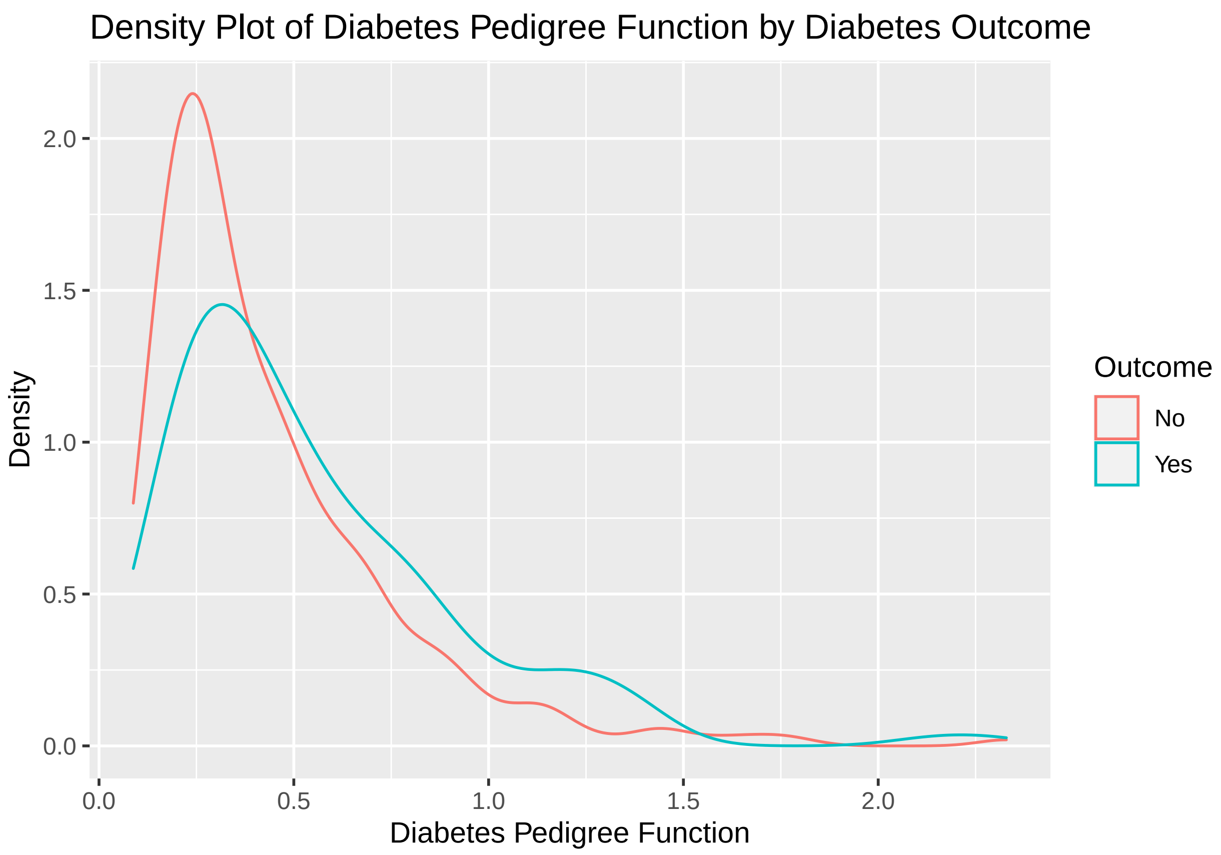
```
# Side by side box plot
ggplot(train, aes(x=Outcome, y=BMI))+
  geom_boxplot(color = "blue", outlier.color = "orange") +
  labs(x="Diabetes", y="BMI", title="Distribution of BMI by Diabetes Outcome")
```



Patients with diabetes tend to have higher BMIs than those without.

Diabetes Pedigree Function

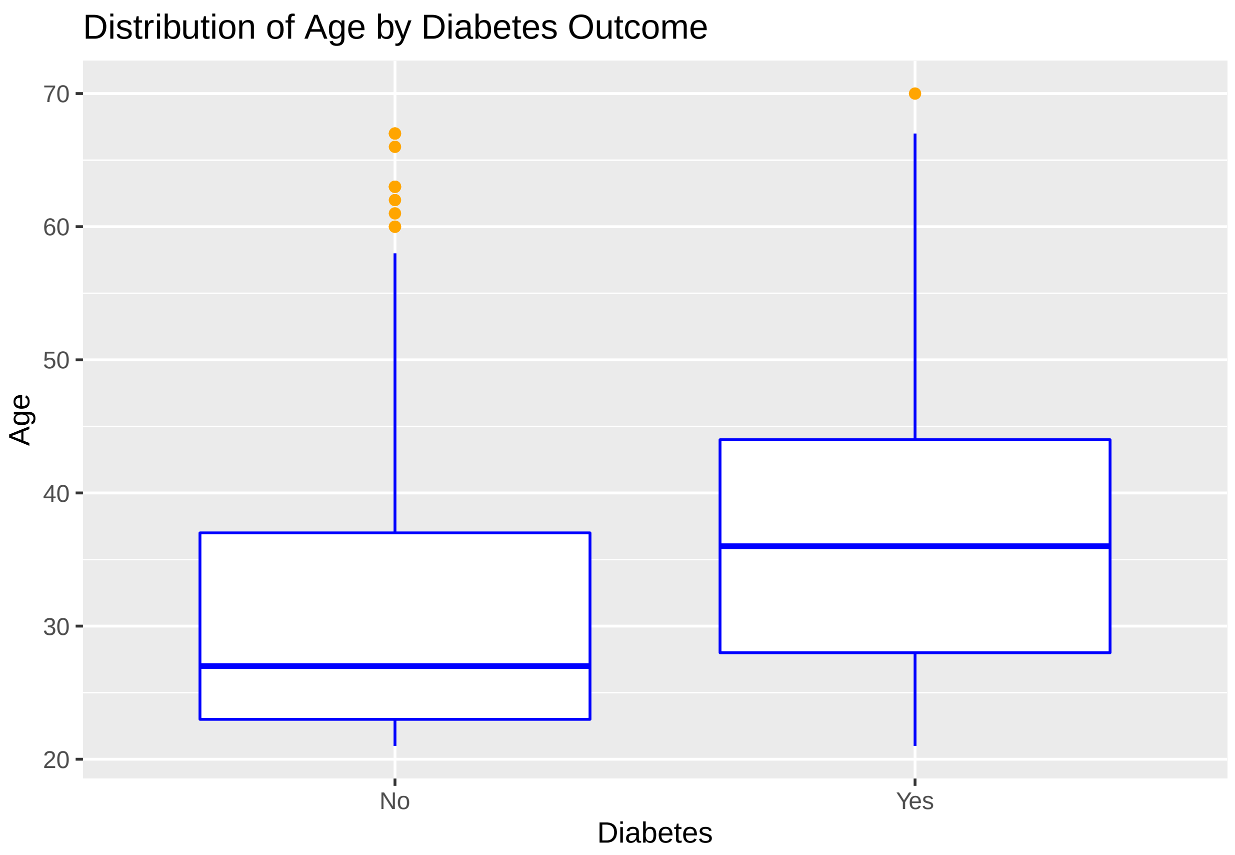
```
# Density plot
ggplot(train, aes(x=DiabetesPedigreeFunction, color=Outcome)) +
  geom_density() +
  labs(x="Diabetes Pedigree Function", y="Density", title="Density Plot of Diabetes Pedigree Function by
```



Those with lower Diabetes Pedigree Functions (less family history of diabetes) appear less likely to have diabetes.

Age

```
# Side by side box plot
ggplot(train, aes(x=Outcome, y=Age))+
  geom_boxplot(color = "blue", outlier.color = "orange") +
  labs(x="Diabetes", y="Age", title="Distribution of Age by Diabetes Outcome")
```

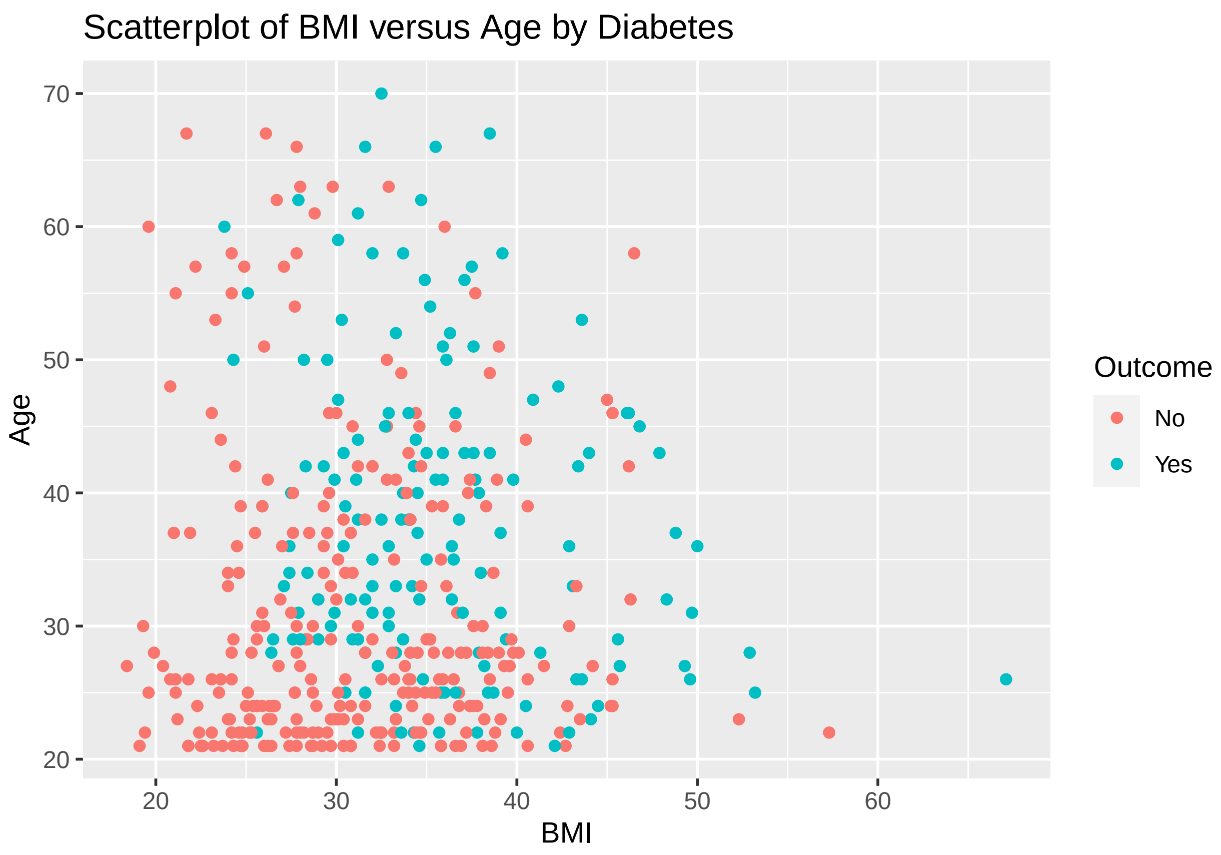


The median age of patients with diabetes is higher than those that do not have diabetes.

Multiple Predictors and Response

Next, we created scatter plots between pairs of potential predictors and colored the dots with the diabetes outcome. From this scatter plot of BMI versus age there does seem to be a relationship with diabetes where the women with diabetes tend to be higher BMIs and older as well.

```r
# Scatter plot of potential predictors (colored by response)
# bmi and blood pressure colored by diabetes
ggplot(train, aes(x=BMI, y=Age, color=Outcome)) +
  geom_point() +
  labs(x="BMI", y="Age", title="Scatterplot of BMI versus Age by Diabetes")
```



Logistic Regression Model

Now that we had cleaned and visualized our data we could start fitting our models. For logistic regression we wanted to predict which patients have diabetes. ## Fit Initial Model For our initial logistic regression model we included all of our potential predictors (after dropping those with higher percent missing values), which included Pregnancies, Glucose, Blood Pressure, BMI, Diabetes Pedigree Function, and Age.

```
# Fit initial regression model
full <- glm(Outcome~., family="binomial", data=train)
summary(full)
```

```
##
## Call:
## glm(formula = Outcome ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6120  -0.7134  -0.4053   0.7003   2.1501
##
## Coefficients:
##                             Estimate Std. Error z value Pr(>|z|)
## (Intercept)                -9.013998   1.021340  -8.826  < 2e-16 ***
## Pregnancies                 0.097572   0.041461   2.353   0.0186 *
## Glucose                     0.034536   0.004768   7.243 4.38e-13 ***
## BloodPressure              -0.009317   0.010998  -0.847   0.3969
## BMI                         0.095598   0.020321   4.704 2.55e-06 ***
## DiabetesPedigreeFunction  0.687768   0.367994   1.869   0.0616 .
## Age                       0.023859   0.012792   1.865   0.0622 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 563.66  on 432  degrees of freedom
## Residual deviance: 405.32  on 426  degrees of freedom
## AIC: 419.32
##
## Number of Fisher Scoring iterations: 5
```

Based on the summary of this initial logistic regression model we observe that Glucose and BMI and highly significant in predicting who has diabetes. These two variables had the largest differences in the density and box plots above so it is not surprising that they are highly predictive of diabetes. Pregnancies is also significant at a 5% significance level. The other predictors, Blood Pressure, Diabetes Pedigree Function, and Age do not add as much value given all of the other predictors are fit in the model. This does not mean that these three variables are not related to diabetes, it only means that they may not add value given that the other variables are also included in the model.

Test Hypothesis on Subset of Parameters

We will test whether we can drop all three of these insignificant predictors. For this test our null hypothesis will be $H_0 : \beta_{BloodPressure} = \beta_{DiabetesPredigreeFunction} = \beta_{Age} = 0$ and the alternative is H_A : at least one β in H_0 is non-zero.

```r
reduced <- glm(Outcome~Pregnancies + Glucose + BMI, family="binomial", data=train)
summary(reduced)
```

```
##
## Call:
## glm(formula = Outcome ~ Pregnancies + Glucose + BMI, family = "binomial",
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1698  -0.7208  -0.4239   0.7293   2.2191
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -8.641263   0.871351  -9.917  < 2e-16 ***
## Pregnancies  0.130098   0.035287   3.687 0.000227 ***
## Glucose      0.036172   0.004618   7.833 4.77e-15 ***
## BMI          0.088533   0.018969   4.667 3.05e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 563.66  on 432  degrees of freedom
## Residual deviance: 413.21  on 429  degrees of freedom
## AIC: 421.21
##
```

```
## Number of Fisher Scoring iterations: 4
```

All of the predictors in this three variable model are statistically significant. Let's test whether we can use this reduced model by calculating the difference in deviance between this reduced model and our full model.

```
# Test statistic
test_stat <- reduced$deviance - full$deviance
# P-value
p_value <- 1 - pchisq(test_stat, 3)
p_value
```

```
## [1] 0.04843538
```

This p-value of 0.048 is less than our significance level of 0.05 so we reject the null hypothesis and conclude that at least one of the predictors that we dropped is useful in predicting diabetes. Looking back at the standard errors of our full model, we observed that Age and Diabetes Pedigree Function had p-values of 0.06, just above our cutoff of 0.05 so perhaps these two variables should remain in our model.

```
reduced2 <- glm(Outcome~Pregnancies + Glucose + BMI + DiabetesPedigreeFunction + Age, family="binomial"
summary(reduced2)
```

```
##
## Call:
## glm(formula = Outcome ~ Pregnancies + Glucose + BMI + DiabetesPedigreeFunction +
##     Age, family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5909  -0.7191  -0.4074   0.7067   2.0991
##
## Coefficients:
##                           Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -9.358662   0.943930  -9.915  < 2e-16 ***
## Pregnancies               0.094695   0.041101   2.304   0.0212 *
## Glucose                   0.034116   0.004724   7.221 5.15e-13 ***
## BMI                       0.089850   0.019121   4.699 2.61e-06 ***
## DiabetesPedigreeFunction  0.694142   0.365518   1.899   0.0576 .
## Age                       0.021475   0.012424   1.728   0.0839 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 563.66  on 432  degrees of freedom
## Residual deviance: 406.04  on 427  degrees of freedom
## AIC: 418.04
##
## Number of Fisher Scoring iterations: 4
```

In this 5 variable model we observe that neither Age nor Diabetes Pedigree Function are statistically significant. Perhaps we should keep only Diabetes Pedigree Function?

```
reduced3 <- glm(Outcome~Pregnancies + Glucose + BMI + DiabetesPedigreeFunction, family="binomial", data
summary(reduced3)
```

```
##
## Call:
## glm(formula = Outcome ~ Pregnancies + Glucose + BMI + DiabetesPedigreeFunction,
##     family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6860  -0.7325  -0.4124   0.7016   2.1362
##
## Coefficients:
##                           Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -8.905831   0.893225  -9.970  < 2e-16 ***
## Pregnancies               0.131453   0.035510   3.702 0.000214 ***
## Glucose                   0.035819   0.004656   7.693 1.44e-14 ***
## BMI                       0.086965   0.018988   4.580 4.65e-06 ***
## DiabetesPedigreeFunction  0.734972   0.363732   2.021 0.043317 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 563.66  on 432  degrees of freedom
## Residual deviance: 409.02  on 428  degrees of freedom
## AIC: 419.02
##
## Number of Fisher Scoring iterations: 4
```

Alas, in this four variable model all of the predictors are statistically significant. We will confirm this result by testing whether we can drop Age and Blood Pressure from our model. For this test our null hypothesis will be $H_0 : \beta_{BloodPressure} = \beta_{Age} = 0$ and the alternative is H_A : at least one β in H_0 is non-zero.

```r
# Test statistic
test_stat2 <- reduced3$deviance - full$deviance
# P-value
p_value2 <- 1 - pchisq(test_stat2, 2)
p_value2
```

```
## [1] 0.157691
```

The p-value of 0.16 is not less than our significance level of 0.05 so we fail to reject the null hypothesis. We don't have enough evidence to say that any of the two variables (Age and Blood Pressure) we dropped in our new reduced model have a non-zero coefficient and thus we can use our reduced model with just Pregnancies, Glucose, BMI, and Diabetes Pedigree Function to predict diabetes.

Test Whether Model is Useful

Next, we'll test whether our reduced model is useful in predicting diabetes. In other words, can we drop all coefficients from our model? We'll start from our reduced model given the results from our hypothesis test above. For this test our null hypothesis will be $H_0 : \beta_{Pregnancies} = \beta_{Glucose} = \beta_{BMI} = \beta_{DiabetesPedigreeFunction} = 0$ and the alternative is H_A : at least one β in H_0 is non-zero.

```r
# Test statistic
test_stat3 <- reduced3$null.deviance - reduced$deviance
# P-value
p_value3 <- 1 - pchisq(test_stat3, 4)
p_value3
```

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

The p-value for this hypothesis is 0 so we reject the null hypothesis and conclude that at least one of predictors has a non-zero coefficient and thus this logistic regression model is useful in estimating the odds of developing diabetes.

Interpreting Model's Coefficients

```r
reduced3
```

```
##
## Call:  glm(formula = Outcome ~ Pregnancies + Glucose + BMI + DiabetesPedigreeFunction,
##     family = "binomial", data = train)
##
## Coefficients:
##              (Intercept)               Pregnancies                   Glucose
##                 -8.90583                   0.13145                   0.03582
##                      BMI  DiabetesPedigreeFunction
##                  0.08697                   0.73497
##
## Degrees of Freedom: 432 Total (i.e. Null);  428 Residual
## Null Deviance:       563.7
## Residual Deviance: 409   AIC: 419
```

Our estimated logistic regression equation is: $log(\frac{\hat{\pi}}{1-\hat{\pi}}) = -8.90583 + 0.13145 * Pregnancies + 0.03582 * Glucose + 0.08697 * BMI + 0.73497 * DiabetesPedigreeFunction$. All three of these predictors have positive coefficients, suggesting that for larger number of pregnancies, higher glucose concentration, higher BMI, and higher diabetes pedigree function the odds of having diabetes goes up, with all other variables held constant.

The estimated coefficient for pregnancies is 0.13145. This means that for each additional pregnancies the log odds that the woman will have diabetes increases by 0.13145 while controlling for glucose concentration and BMI. In other words, for each one unit increase in pregnancies the odds that the woman has diabetes is multiplied by 1.140481, while holding the other variables constant.

```r
exp(0.13145)
```

```
## [1] 1.140481
```

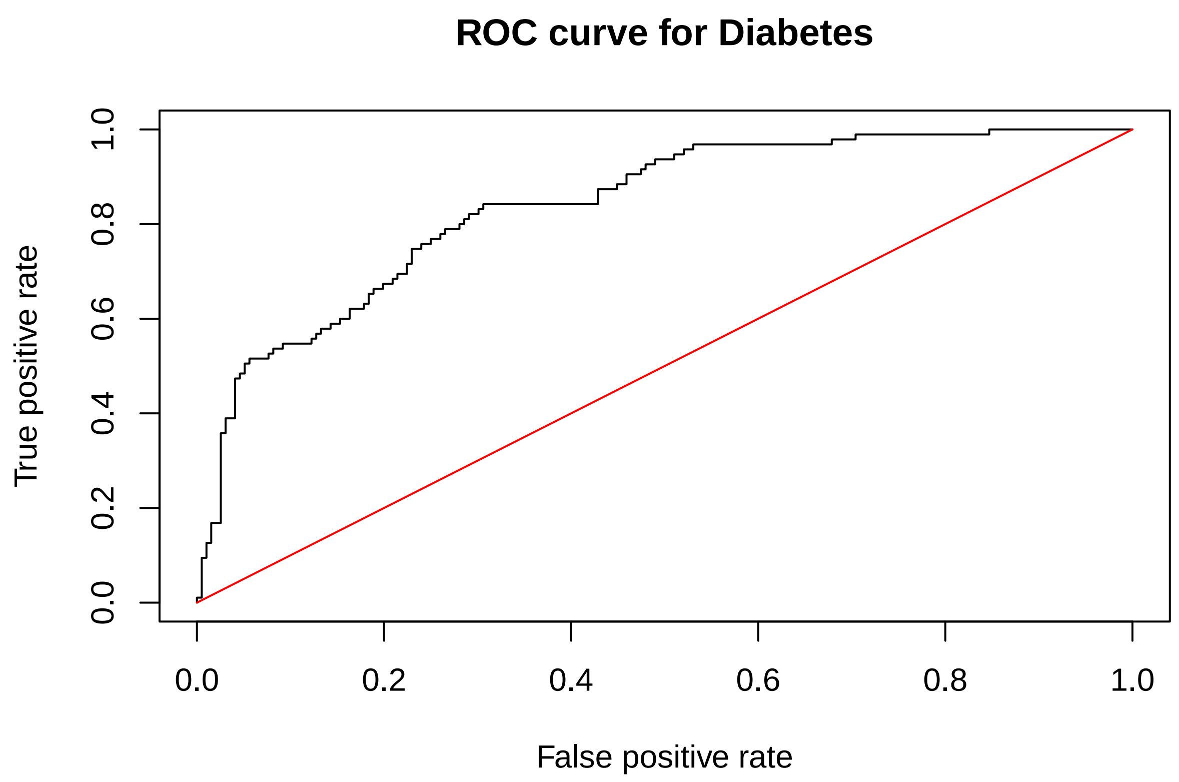
TODO: interpretations of other coefficients.

Model Evaluation

Use model built on training data to estimate the probabilities for observations in the unseen test data set.

Plot ROC curve

```r
# Predicted diabetes rate
preds <- predict(reduced3, newdata=test, type="response")
# Transform input data
rates <- prediction(preds, test$Outcome)
# Store true positive and false positive rates
roc_result <- performance(rates, measure="tpr", x.measure="fpr")
# Plot roc curve and overly diagonal (random)
plot(roc_result, main = "ROC curve for Diabetes")
lines(x=c(0,1), y=c(0,1), col="red")
```



The ROC curve lies above the straight diagonal line, suggesting that this model identifies/classifies people who have diabetes better than random.

Calculate AUC

```
auc <- performance(rates, measure="auc")
auc@y.values
```

```
## [[1]]
## [1] 0.8386144
```

The value of AUC for the ROC curve above is 0.8386144. Since this value is greater than 0.5 the model does better than random for classifying who develops diabetes.

Create Confusion Matrix

TODO QUESTION: what do we want to use for a threshold? I just did 0.5 for now, but maybe in predicting diabetes, due to the health implications, we may be more concerned about false negatives (classify someone as not having diabetes when they do have it) so we've chose a threshold lower than 0.5. With this lower threshold we'll have a lower false negative rate (as desired), but as a result would have a higher false positive rate.

```
threshold = 0.5 # Define threshold
table(test$Outcome, preds>threshold)
```

```
##
##       FALSE TRUE
##   No    174   22
##   Yes    43   52
```

```r
# Accuracy
(174+52)/(174+22+43+52)
```

```
## [1] 0.7766323
```

```r
# TPR
52/(43+52)
```

```
## [1] 0.5473684
```

```r
# TNR
174/(174+22)
```

```
## [1] 0.8877551
```

At a threshold of 0.5, the models' overall accuracy is 78% with a true positive rate of 55% and a true negative rate of 89%.