EdX_Harvard_Capstone_CYO_PIMA Indian Diabetes_ Prediction

Srividhya G Ammanur

11/18/2020

Executive Summary:

This project is for the data science professional certificate capstone, from HarvardX institution through the edX platform. The objective is to construct a machine learning model that learns from data by looking at a broad range of patterns and make inferences that efficiently predicts the outcome without human intervention. The data set utilized for this project is the PIMA Indian Diabetes Data Set.

Data Source:

Original Owners: National Institute of Diabetes and Digestive and Kidney Diseases. **Donor of database**: Vincent Sigillito (vgs@aplcen.apl.jhu.edu), who was part of Applied Physics Laboratory at Johns Hopkins. The Applied Physics Laboratory (APL) is a not-for-profit engineering research and development center founded in 1942 to assist the military with ballistics detonation. The data set was constructed from a larger database by the NIDDK.For this project the data set have been taken from the UCI Repository Of Machine Learning Databases.

The women behind the Dataset- A background:

All participants (768 Observations) in this dataset are women, at least 21 years old of PIMA Indian heritage. PIMA, are North American Indians who traditionally lived along the Gila and Salt rivers in Arizona, U.S., in what was the core area of the prehistoric Hohokam culture. The PIMA, who speak a Uto-Aztecan language and call themselves the "River People," are usually considered to be the descendants of the Hohokam. The PIMA Indian women have given a great gift to the humanity by volunteering for this research and donating their biological data for the larger good of the society. I express my sincere gratitude for their generosity. William Knowler, an NIH researcher since 1975, who is also recognized as one of the world's highly cited researchers in clinical medicine, biology and biochemistry, testified before congress, "This study has contributed much to the world's current understanding of the causes and consequences of Type 2 diabetes and its complications, for which we are indebted to this community".

Data Ethics:

It is important for the community of data scientists to be aware that all those who make investments in repositories of data whether it is heart disease, breast cancer research, or social media usage are not always the ones to benefit from their use. As Rebecca Lemov and Dan Bouk rightly said, "data are people too".

When data rises above the geographically defined locality and circumstances it becomes "big" and is fed into computers and algorithms to meet business goals. It therefore becomes obligatory, that we recognize the living people behind the data. Wherever feasible go beyond and create institutions and frameworks the benefit the contributors of data for data Science.

Project Methodology:

- Gathering data
- Preparing the data
- Choosing a model
- Training the model
- Evaluating the model
- Tuning the model
- Make Predictions

Load the dataset

setwd("C:\Users\agsri\OneDrive\Capstone") diabetes <- read.csv(file = 'Pima_Indian_Diabetes.csv')

```
setwd("C:\\Users\\agsri\\OneDrive\\Capstone")
diabetes <- read.csv(file = 'Pima_Indian_Diabetes.csv')
```

Load Libraries

```
## Warning: package 'ggcorrplot' was built under R version 4.0.2

## -- Attaching packages ------------------- tidyverse 1.3.0 --

## v tibble  3.0.1     v dplyr   0.8.5
## v tidyr   1.0.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
## v purrr   0.3.4

## -- Conflicts ---------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

## Warning: package 'GGally' was built under R version 4.0.2

## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2

##
## Attaching package: 'GGally'
```

```
## The following object is masked from 'package:dplyr':
##
##     nasa

## Warning: package 'caret' was built under R version 4.0.2

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

view the first and last 10 rows of the dataset

```r
head(diabetes)
```

```
##   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
```

```r
tail(diabetes)
```

```
##     Pregnancies Glucose BloodPressure SkinThickness Insulin  BMI
## 763           9      89            62             0       0 22.5
## 764          10     101            76            48     180 32.9
## 765           2     122            70            27       0 36.8
## 766           5     121            72            23     112 26.2
## 767           1     126            60             0       0 30.1
## 768           1      93            70            31       0 30.4
##     DiabetesPedigreeFunction Age Outcome
## 763                    0.142  33       0
## 764                    0.171  63       0
## 765                    0.340  27       0
## 766                    0.245  30       0
## 767                    0.349  47       1
## 768                    0.315  23       0
```

Summary Statistics of dataset

glance the data set to see if it is in tidy format

```
diabetes %>% as_tibble()
```

```
## # A tibble: 768 x 9
##    Pregnancies Glucose BloodPressure SkinThickness Insulin   BMI
##          <int>   <int>         <int>         <int>   <int> <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
##  7           3      78            50            32      88  31
##  8          10     115             0             0       0  35.3
##  9           2     197            70            45     543  30.5
## 10           8     125            96             0       0   0
## # ... with 758 more rows, and 3 more variables: DiabetesPedigreeFunction <dbl>,
## #   Age <int>, Outcome <int>
```

Summary of the data set

```
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
```

Structure of the data set

```r
str(diabetes)
```

```
## 'data.frame':    768 obs. of  9 variables:
##  $ Pregnancies             : int  6 1 8 1 0 5 3 10 2 8 ...
##  $ Glucose                 : int  148 85 183 89 137 116 78 115 197 125 ...
##  $ BloodPressure           : int  72 66 64 66 40 74 50 0 70 96 ...
##  $ SkinThickness           : int  35 29 0 23 35 0 32 0 45 0 ...
##  $ Insulin                 : int  0 0 0 94 168 0 88 0 543 0 ...
##  $ BMI                     : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
##  $ DiabetesPedigreeFunction: num  0.627 0.351 0.672 0.167 2.288 ...
##  $ Age                     : int  50 31 32 21 33 30 26 29 53 54 ...
##  $ Outcome                 : int  1 0 1 0 1 0 1 0 1 1 ...
```

Columns in the dataset

```r
colnames(diabetes)
```

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

#Rows in the dataset

```r
nrow(diabetes)
```

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

Average age of the women in the data set

```r
mean(diabetes$Age)
```

```
## [1] 33.24089
```

```r
mean(diabetes$Pregnancies)
```

```
## [1] 3.845052
```

```r
range(diabetes$BMI)
```

```
## [1]  0.0 67.1
```

```r
median(diabetes$Glucose)
```

```
## [1] 117
```

Check for any null values in the data

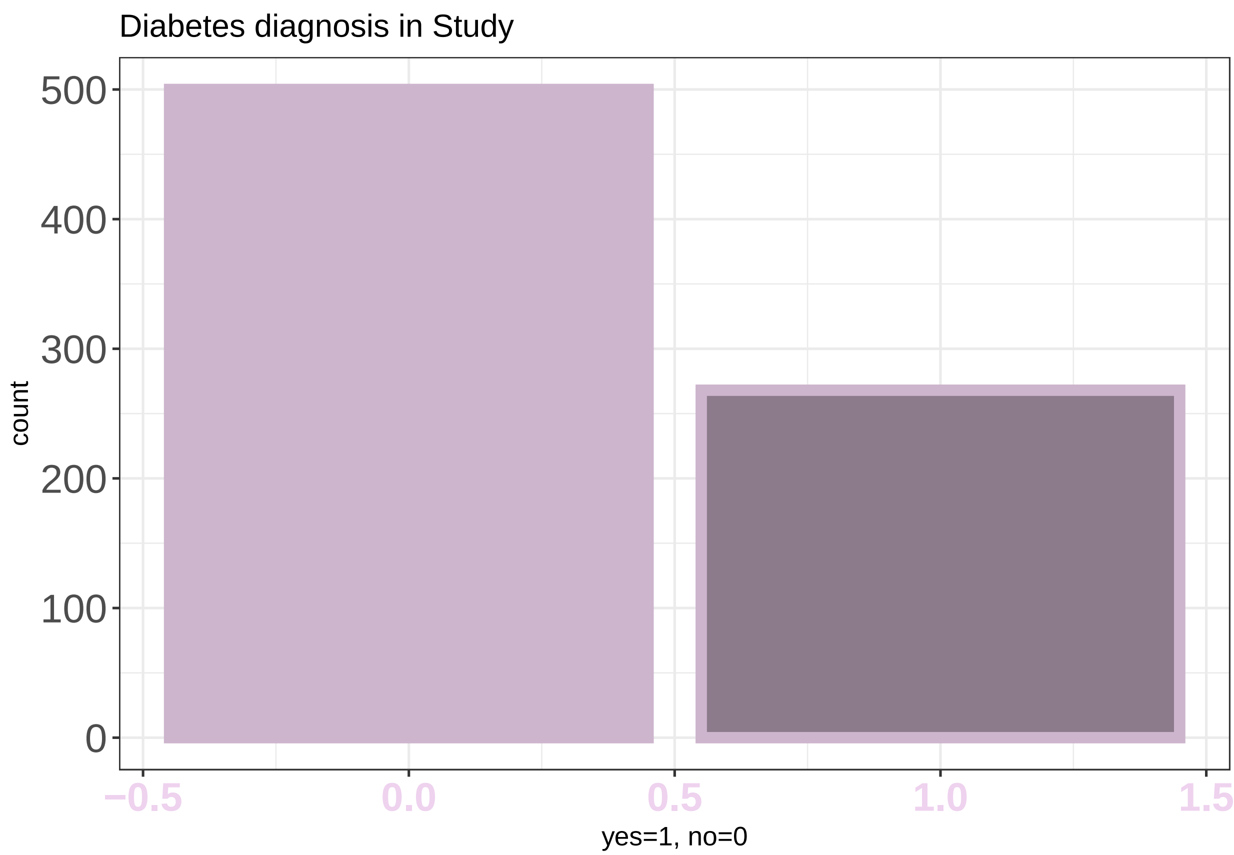
```r
colSums(is.na(diabetes))
```

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

DATA VISUALIZATION

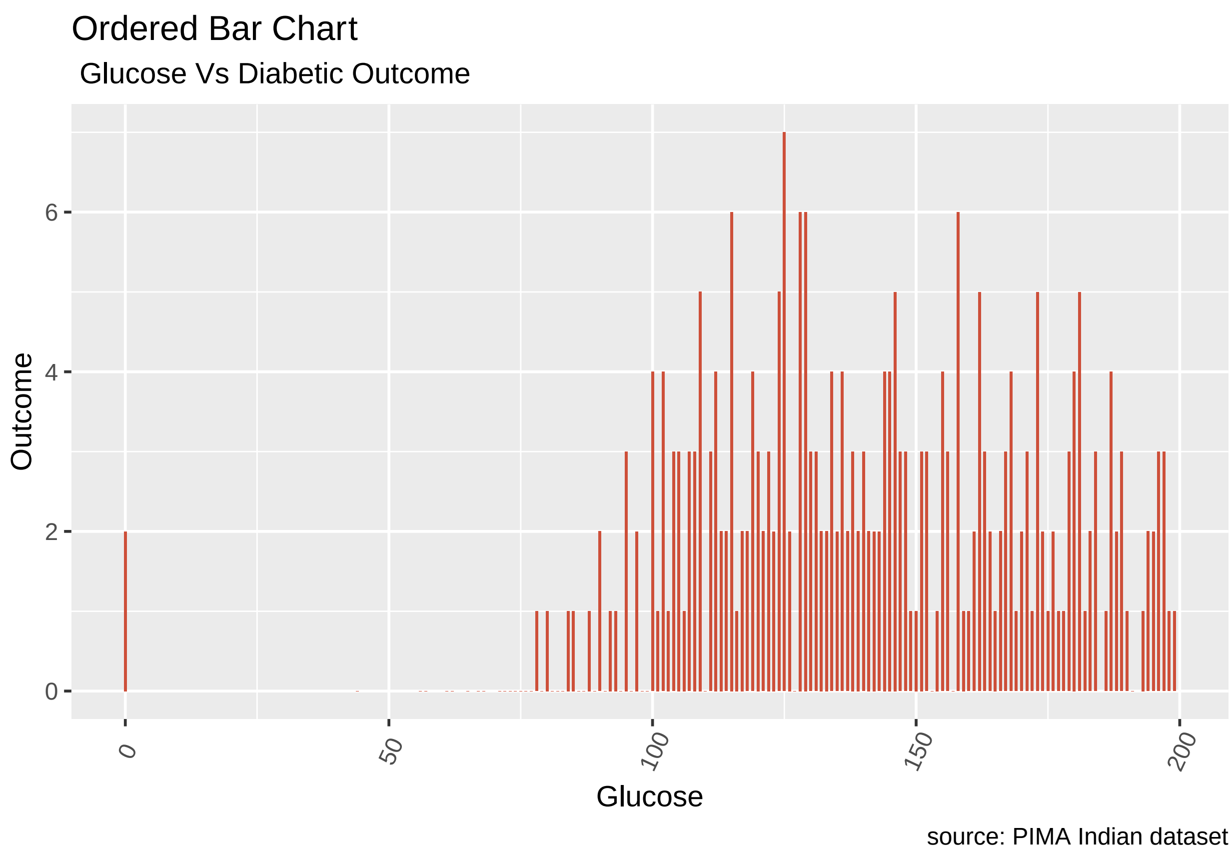
Exploring & understanding the data through visualizations

```r
options(repr.plot.width=8, repr.plot.height=7)

ggplot(data = diabetes) +
  geom_bar(stat = "count", mapping = aes(x = Outcome),
           color = 'thistle3', fill = c('thistle3','thistle4'), lwd = 2) +
  labs(x = "yes=1, no=0",title = "Diabetes diagnosis in Study") +
  theme_bw(base_size = 10) +
  theme(axis.text.x = element_text(size = 15,
                                   colour = 'thistle2', face = "bold"),
        axis.text.y = element_text(size = 15))
```



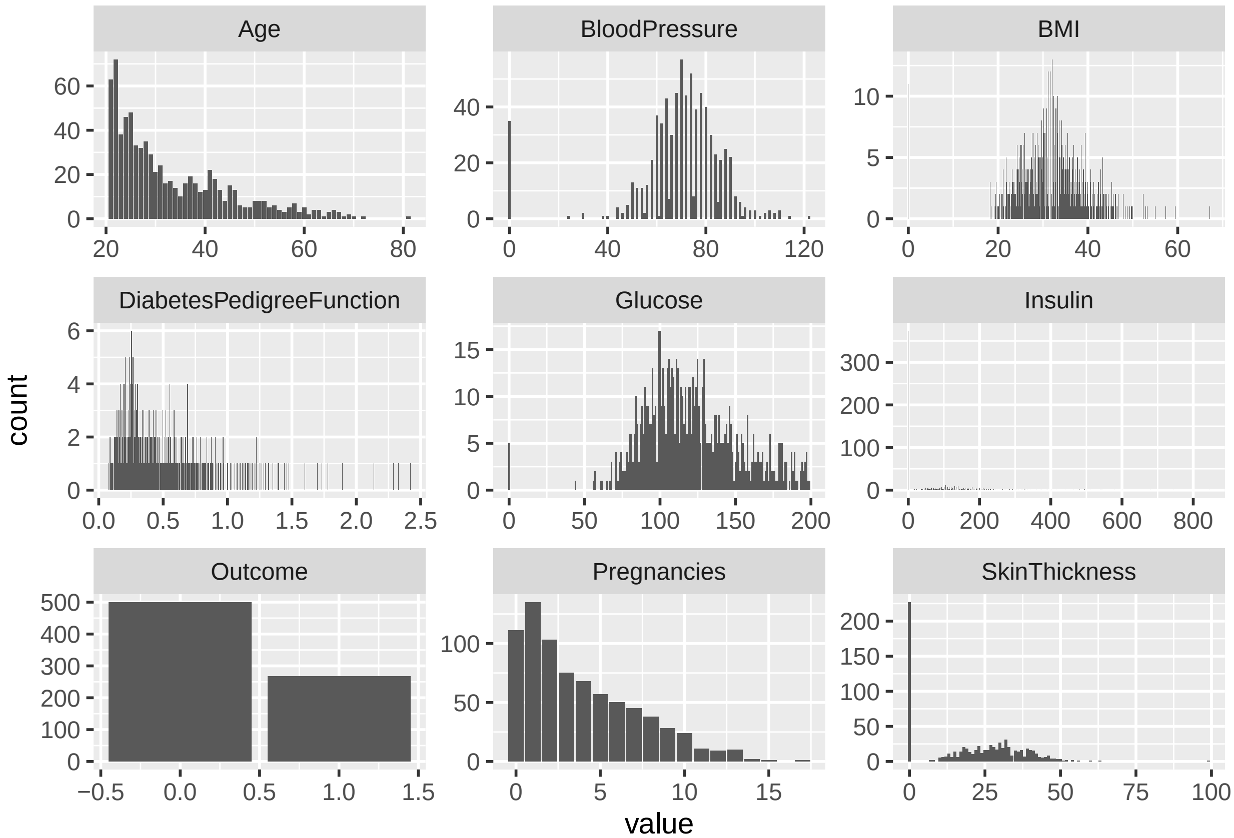
Draw plot on Glucose in blood and diabetic outcome.

```
ggplot(diabetes, aes(x=Glucose, y=Outcome)) +
  geom_bar(stat="identity", width=.5, fill="tomato3") +
  labs(title="Ordered Bar Chart",
       subtitle=" Glucose Vs Diabetic Outcome",
       caption="source: PIMA Indian dataset") +
  theme(axis.text.x = element_text(angle=65, vjust=0.6))
```



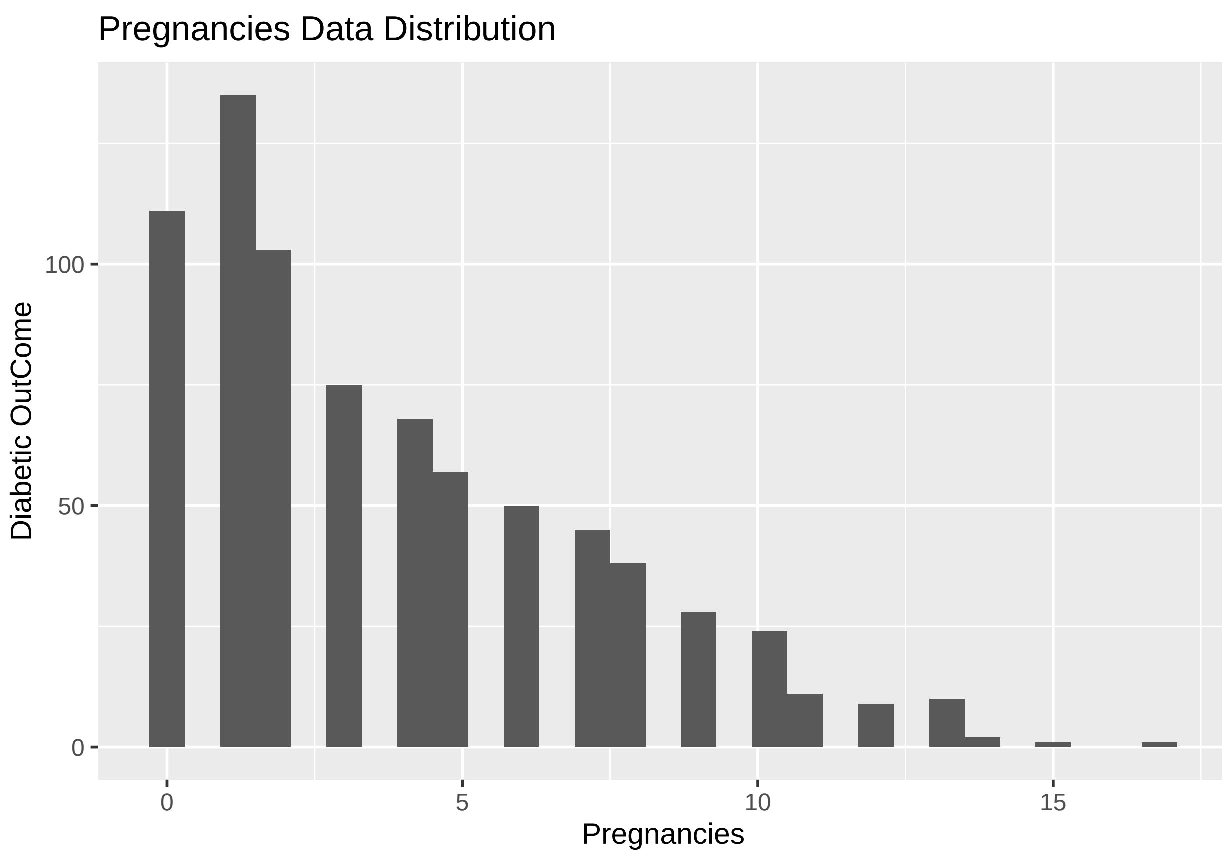
#Histogram of all the variables in the data set.

```
diabetes %>%
  gather() %>%
  ggplot(aes(value)) +
  facet_wrap(~ key, scales = "free") +
  geom_bar()
```

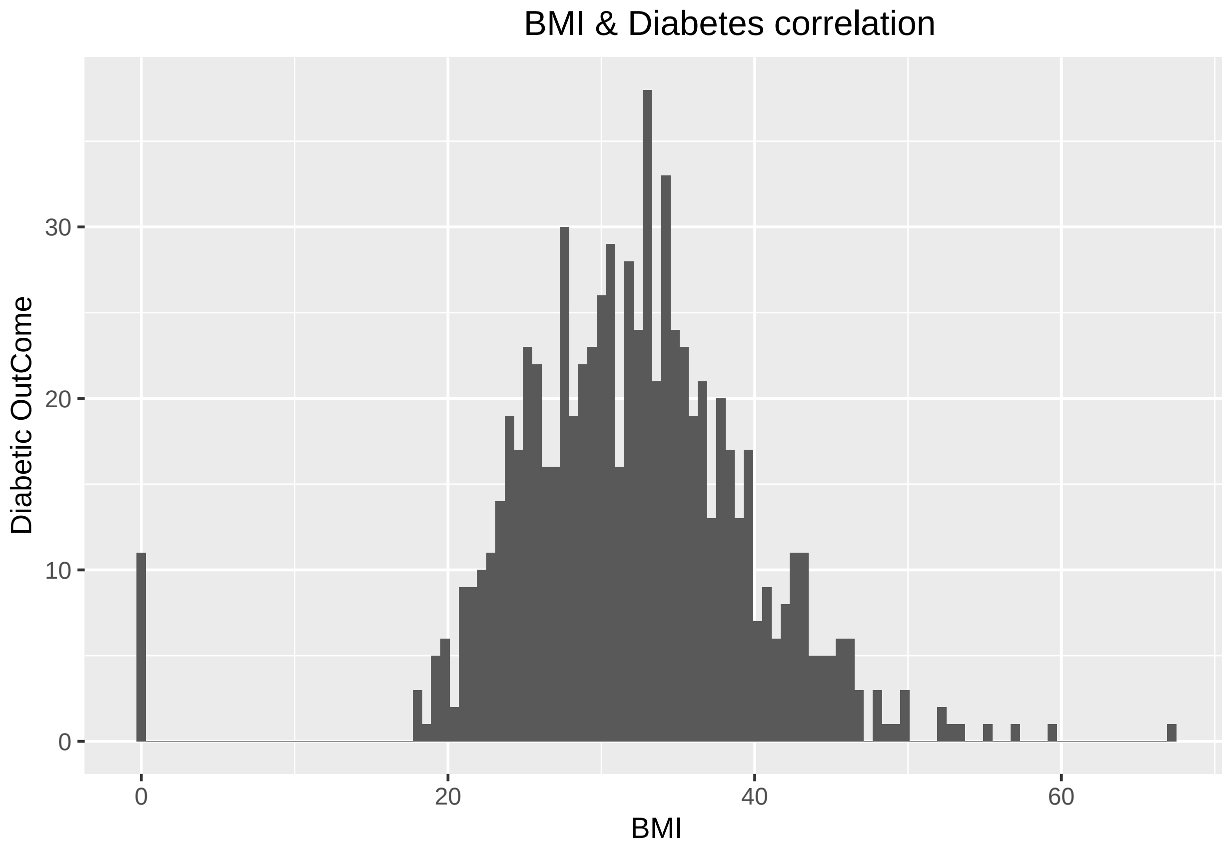


Plot each variables with diabetic outcome distribution

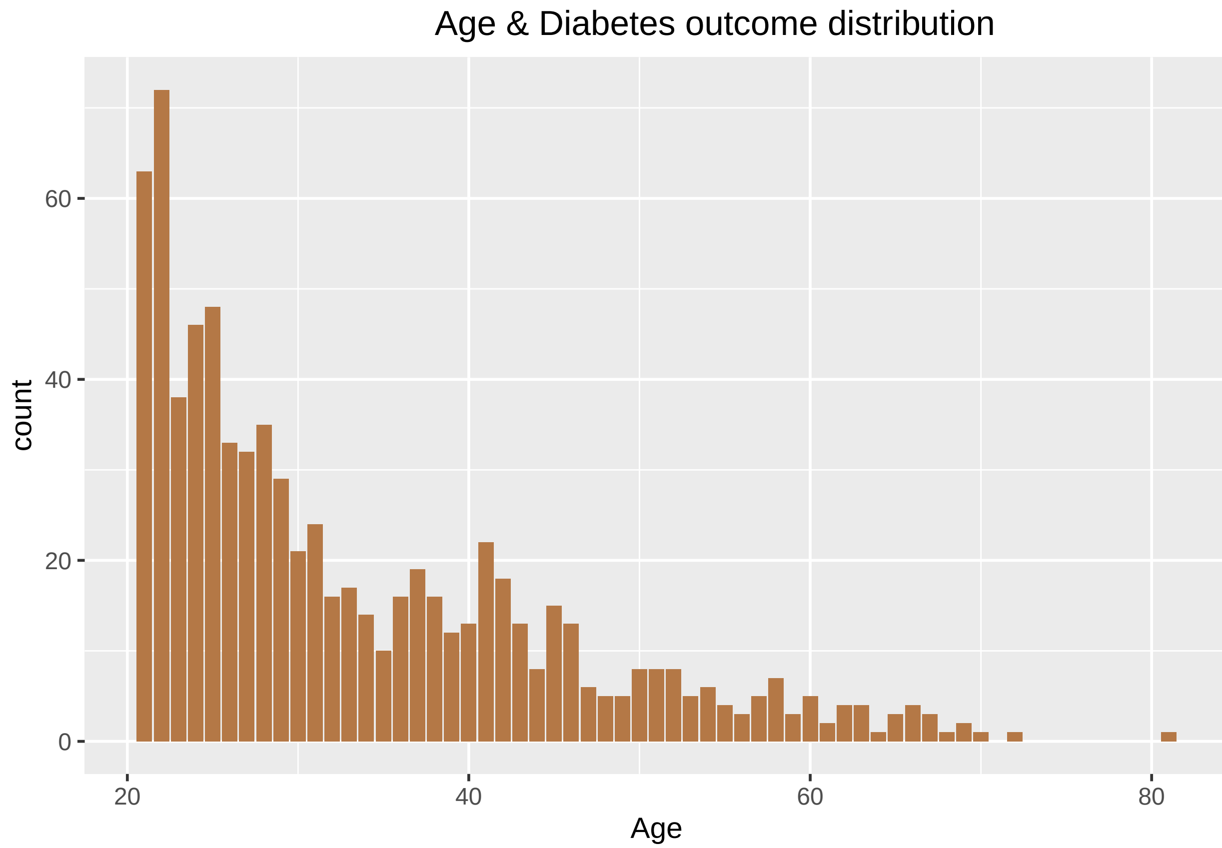
```
# Plot pregnancy vs outcome distribution
ggplot(data = diabetes,aes(x = Pregnancies)) +
  geom_histogram(binwidth = 0.6,aes(fill = Outcome),position = "dodge") +
  ggtitle("Pregnancies Data Distribution") + ylab("Diabetic OutCome") +
  theme_gray() +
  theme_update(plot.title = element_text(hjust = 0.6))
```



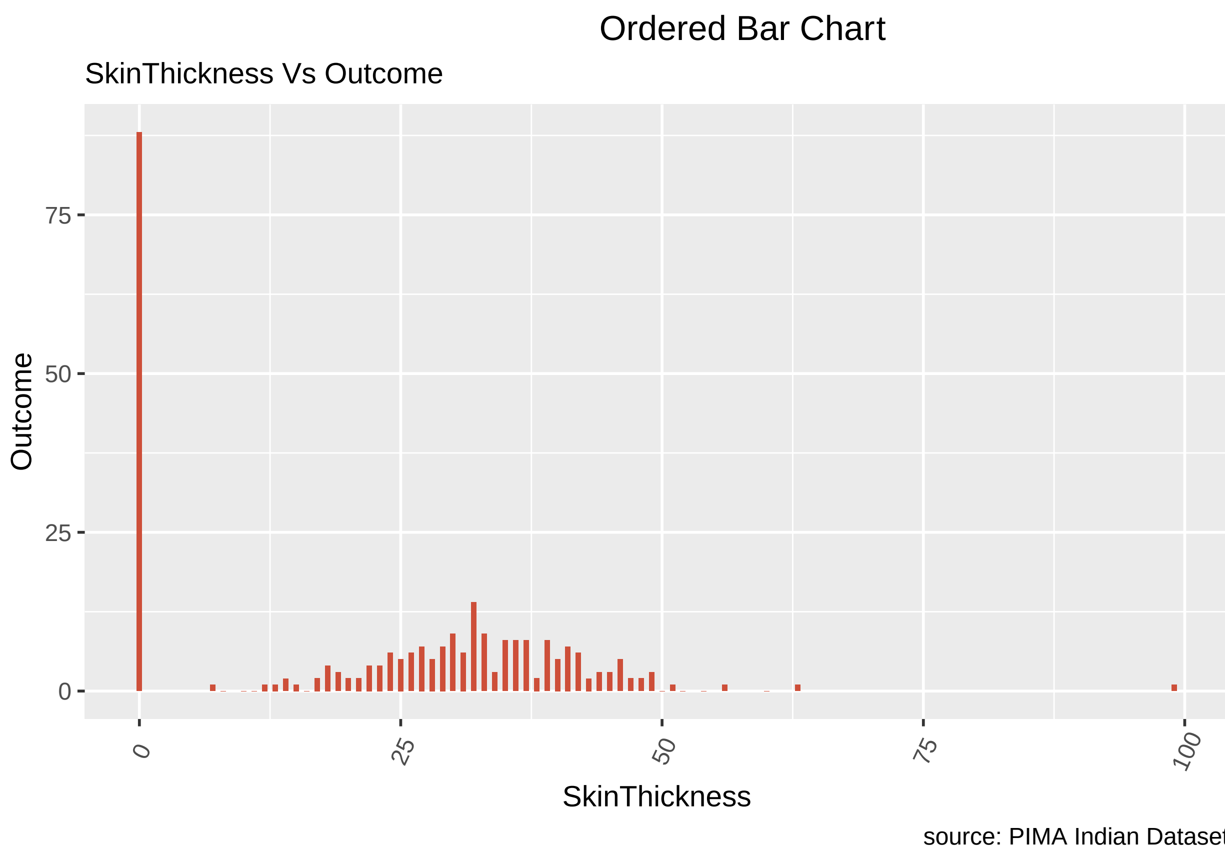

```r
# Plot BMI vs outcome distribution
ggplot(data = diabetes,aes(x = BMI)) +
  geom_histogram(binwidth = 0.6,aes(fill = Outcome),position = "dodge") +
  ggtitle("BMI & Diabetes correlation") + ylab("Diabetic OutCome") +
  theme_gray() +
  theme_update(plot.title = element_text(hjust = 0.6))
```




```
# plot Age vs outcome distribution
ggplot(aes(x = Age), data = diabetes) +
  geom_bar(fill='#b47846')+ ggtitle("Age & Diabetes outcome distribution")
```




```r
# plot Skinthickness vs outcome distribution
ggplot(diabetes, aes(x=SkinThickness, y=Outcome)) +
  geom_bar(stat="identity", width=.5, fill="tomato3") +
  labs(title="Ordered Bar Chart",
       subtitle="SkinThickness Vs Outcome",
       caption="source: PIMA Indian Dataset") +
  theme(axis.text.x = element_text(angle=65, vjust=0.6))
```



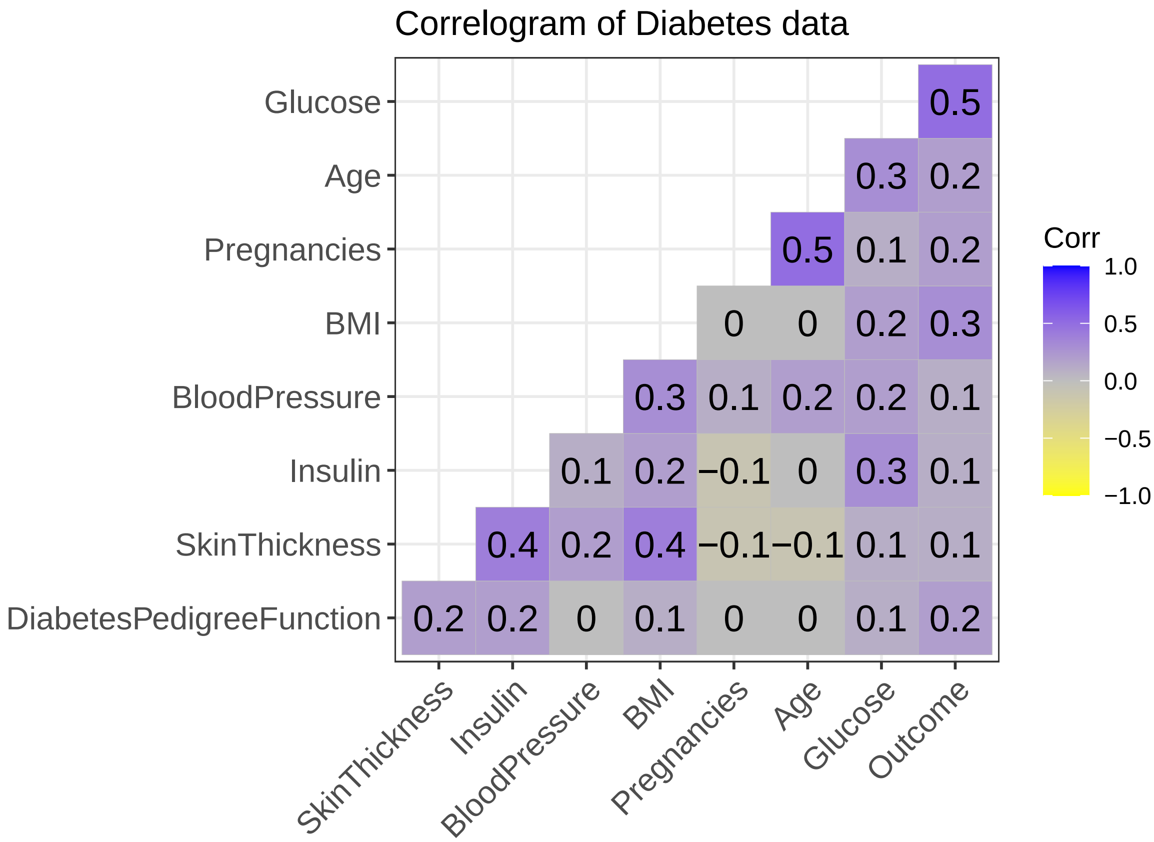
#Proptional table of the diabetes dataset

```
prop.table(table(diabetes$Outcome))
```

```
##
##         0         1
## 0.6510417 0.3489583
```

Correlation between attributes

```
corr<-round(cor(diabetes),1)
ggcorrplot(corr, hc.order = TRUE,
           type = "lower",
           lab = TRUE,
           lab_size = 5,
           method="square",
           colors = c("Yellow", "Grey", "blue"),
           title="Correlogram of Diabetes data",
           ggtheme=theme_bw)
```



#A correlation measures the relationship between two variables. A -ve correlation indicates that if one variable increases the other decreases. A negative correlation exists between pregnancy and insulin and skin thickness. Also age and skin thickness are negetively correlated. Age, pregnancy with diabetes pedigree, are not correlated. Simiarly BMI is also not correlated with pregnancy & age. They all show a 0 in the correlogram. The rest of the variables have correlation with values close to 0. The variables show the most correlation are the following:

1. BMI & Diabetes pedigree function
2. Blood Pressure & Insulin
3. Pregnancy & blood pressure
4. Glucose & skin thickness
5. Glucose & pregnancy
6. Glucose & diabetes pedigree function

In order to build a model and train it lets Split data into training set and test data set

```
set.seed(2017)

trainIndex <- createDataPartition(diabetes$Outcome,     p = .8, list = FALSE,times = 1)


diabetes$Outcome <- as.factor(diabetes$Outcome)
```

```r
diabetes.train <- diabetes[trainIndex,]
diabetes.test <- diabetes[-trainIndex,]
```

Training data Proportion

```r
prop.table(table(diabetes.train$Outcome))
```

```
##
##         0         1
## 0.6569106 0.3430894
```

Test data Proportion

```r
prop.table(table(diabetes.test$Outcome))
```

```
##
##        0        1
## 0.627451 0.372549
```

Model1: RANDOM FOREST MODEL:

Build a Random Forest Model.

#Individual decisions trees are combined to make a random forest.Each decision tree is the building block of the random forest model. Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

#Fit Random Forest Model in training set data # Train the model using random forest algorithm.

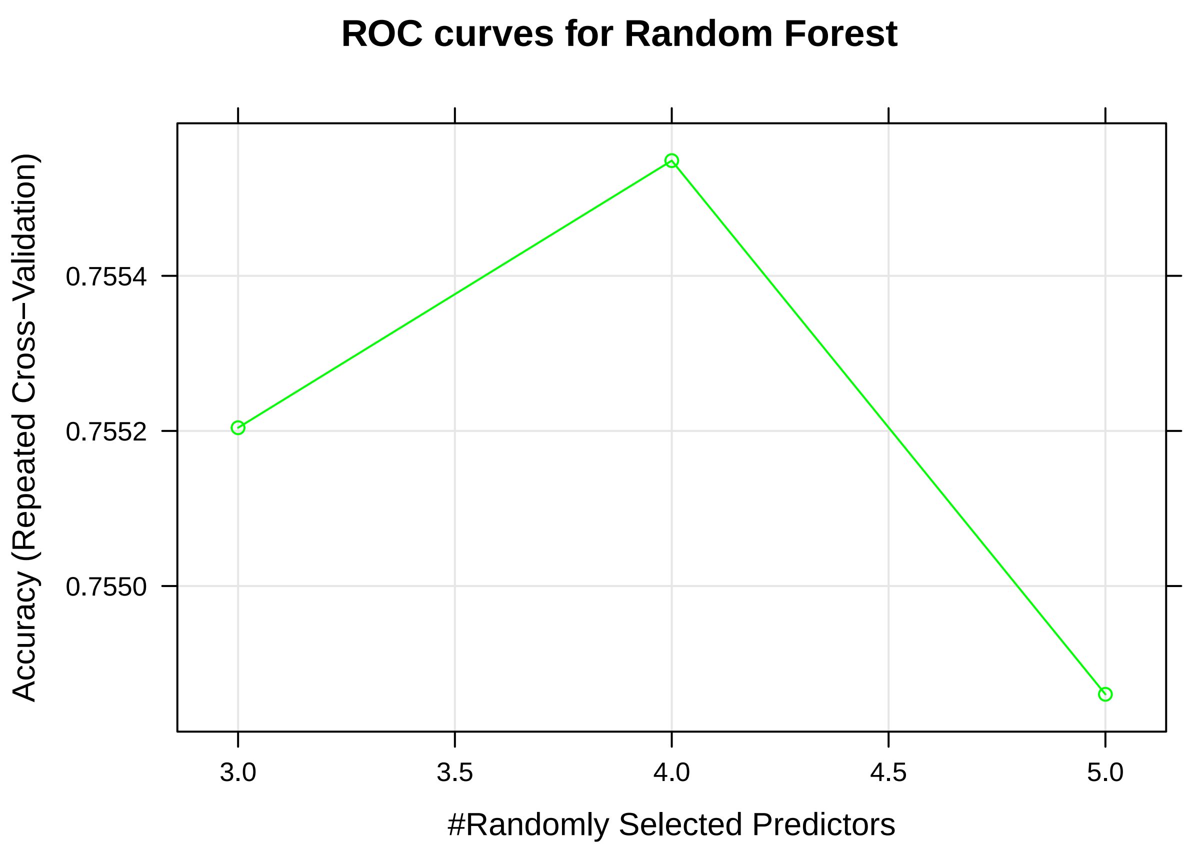
```r
control <- trainControl(
  method = "repeatedcv",
  number = 20,
  repeats = 20
)
# Performance Parameters Setting
grid <- expand.grid(mtry  = c(3,4,5))

model.Random.Forest <- train(Outcome ~ ., data = diabetes.train,
  method = "rf", tuneGrid = grid,trControl = control)

model.Random.Forest
```

```
## Random Forest
##
## 615 samples
```

```
##   8 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (20 fold, repeated 20 times)
## Summary of sample sizes: 584, 585, 583, 585, 584, 585, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   3     0.7552041  0.4362251
##   4     0.7555486  0.4380512
##   5     0.7548604  0.4375340
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 4.
```

Plotting ROC curve

```r
plot(model.Random.Forest, main = "ROC curves for Random Forest", col='Green')
```



Predict the outcome on the test data

```r
predict.Random.Forest <- predict(model.Random.Forest,diabetes.test)
predict.Random.Forest
```

```
##   [1] 0 1 1 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 1 0 1 1 0 0 1 1 0 1
##  [38] 1 1 1 1 1 0 1 1 1 1 1 0 0 1 1 1 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 1 0
##  [75] 1 1 0 0 0 1 1 0 1 1 1 0 0 1 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0
## [112] 0 1 0 1 0 1 0 0 1 1 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 1 0 0 1 0 0 0 1 0
## [149] 0 0 1 1 0
## Levels: 0 1
```

Confusion Matrix

```r
confusionMatrix(predict.Random.Forest,diabetes.test$Outcome)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 79 14
##          1 17 43
##
##                Accuracy : 0.7974
##                  95% CI : (0.7249, 0.858)
##     No Information Rate : 0.6275
##     P-Value [Acc > NIR] : 4.322e-06
##
##                   Kappa : 0.5712
##
##  Mcnemar's Test P-Value : 0.7194
##
##             Sensitivity : 0.8229
##             Specificity : 0.7544
##          Pos Pred Value : 0.8495
##          Neg Pred Value : 0.7167
##              Prevalence : 0.6275
##          Detection Rate : 0.5163
##    Detection Prevalence : 0.6078
##       Balanced Accuracy : 0.7887
##
##        'Positive' Class : 0
##
```

Accuracy is 79% the percentage of correctly classified instances out of all instances.

Sensitivity is 82% which is the true positive rate. It is the number instances from the positive class that actually predicted diabetes outcome correctly.

Specificity is 75% which is the true negative rate. Is the number of instances from the negative class that actually predicted diabetes outcome correctly.

##Model2: LOGISTIC REGRESSION

Build a Logistic Regression Model based on variables

```
model_glm<-glm(Outcome~Pregnancies+Glucose+BMI+SkinThickness+Insulin+DiabetesPedigreeFunction+Age,data=d
summary(model_glm)
```

```
##
## Call:
## glm(formula = Outcome ~ Pregnancies + Glucose + BMI + SkinThickness +
##     Insulin + DiabetesPedigreeFunction + Age, family = binomial,
##     data = diabetes.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4915  -0.7566  -0.4448   0.7685   2.7520
##
## Coefficients:
##                           Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -8.522945   0.781303 -10.909  < 2e-16 ***
## Pregnancies               0.117029   0.035605   3.287  0.00101 **
## Glucose                   0.034151   0.004088   8.354  < 2e-16 ***
## BMI                       0.083839   0.016970   4.940  7.8e-07 ***
## SkinThickness             0.002600   0.007609   0.342  0.73257
## Insulin                  -0.001228   0.001035  -1.186  0.23556
## DiabetesPedigreeFunction  0.696008   0.329683   2.111  0.03476 *
## Age                       0.004259   0.010097   0.422  0.67316
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 790.97  on 614  degrees of freedom
## Residual deviance: 600.13  on 607  degrees of freedom
## AIC: 616.13
```

```
##
## Number of Fisher Scoring iterations: 5
```

#Test model on the test data

```r
predict_test <- predict(model_glm,newdata = diabetes.test,type = "response")

predict_test
```

```
##           2           5           9          22          27          29
## 0.056185761 0.800102880 0.686114407 0.319754341 0.727438069 0.534731612
##          31          34          43          51          53          65
## 0.378948192 0.059500896 0.146060817 0.052225398 0.076230366 0.331323201
##          66          75          80          81          84          87
## 0.144577490 0.070577603 0.119733829 0.098864295 0.064173446 0.535652118
##          88         107         110         112         118         132
## 0.208838937 0.047861557 0.136093991 0.643587631 0.131585148 0.599750467
##         135         143         144         148         150         153
## 0.062577753 0.172136595 0.358975069 0.259668801 0.061256722 0.835132750
##         160         162         170         176         179         184
## 0.960743187 0.311564100 0.168453624 0.866591704 0.740948802 0.052331901
##         187         188         193         194         205         212
## 0.783749799 0.436246095 0.667034277 0.911503724 0.303052206 0.643164456
##         217         236         237         247         248         251
## 0.309709860 0.868118359 0.873395360 0.446130537 0.758213241 0.312043458
##         252         257         260         261         267         271
## 0.227354619 0.224307582 0.869566689 0.763842545 0.497260836 0.724785144
##         284         296         304         308         309         316
## 0.658861254 0.699819542 0.665815363 0.151689157 0.333696540 0.208610586
##         325         335         336         343         348         349
## 0.232309270 0.051779490 0.771635912 0.005129434 0.117942215 0.062986623
##         350         351         354         356         358         359
## 0.017657260 0.252277323 0.071251393 0.756346902 0.803838573 0.343307692
##         364         374         375         376         381         388
## 0.691307764 0.178667448 0.381662932 0.781826461 0.179670982 0.520222256
##         391         392         395         403         409         410
## 0.120755403 0.871498659 0.687872228 0.496937265 0.909490394 0.740249555
##         411         421         426         427         432         434
## 0.341955075 0.454487175 0.803461556 0.006508378 0.108985579 0.239407292
##         450         466         470         472         473         479
## 0.169493548 0.104178301 0.852023158 0.320813430 0.296941641 0.304339796
##         480         487         498         502         504         509
## 0.363800220 0.433679512 0.071343356 0.142325562 0.260100275 0.105095465
##         512         513         514         516         522         527
## 0.129742119 0.124897208 0.080670176 0.566526217 0.288128461 0.034852706
##         546         549         551         559         569         584
## 0.872644008 0.604571584 0.136693154 0.628508903 0.535462636 0.350699886
##         585         590         591         592         597         599
## 0.284885147 0.019523159 0.805059356 0.273170089 0.108514586 0.692216785
##         607         608         610         612         618         620
## 0.863857982 0.040210866 0.069224099 0.713769699 0.018176839 0.176203044
##         624         638         643         644         654         662
## 0.196896858 0.128535652 0.502364324 0.111333765 0.180605016 0.959397681
```

```
##         665         667         668         672         673         674 
## 0.348465240 0.532522141 0.285433365 0.083435678 0.160307600 0.791702281 
##         681         682         683         688         692         695 
## 0.017894359 0.835559098 0.228205843 0.111029553 0.909445534 0.047168847 
##         705         708         709         720         723         740 
## 0.142966474 0.230032075 0.764155031 0.252417596 0.372921539 0.226513485 
##         749         756         765 
## 0.819805592 0.471312112 0.350828147
```

```r
predict_test <- ifelse(predict_test > 0.5,1,0)
```

Confusion Matrix

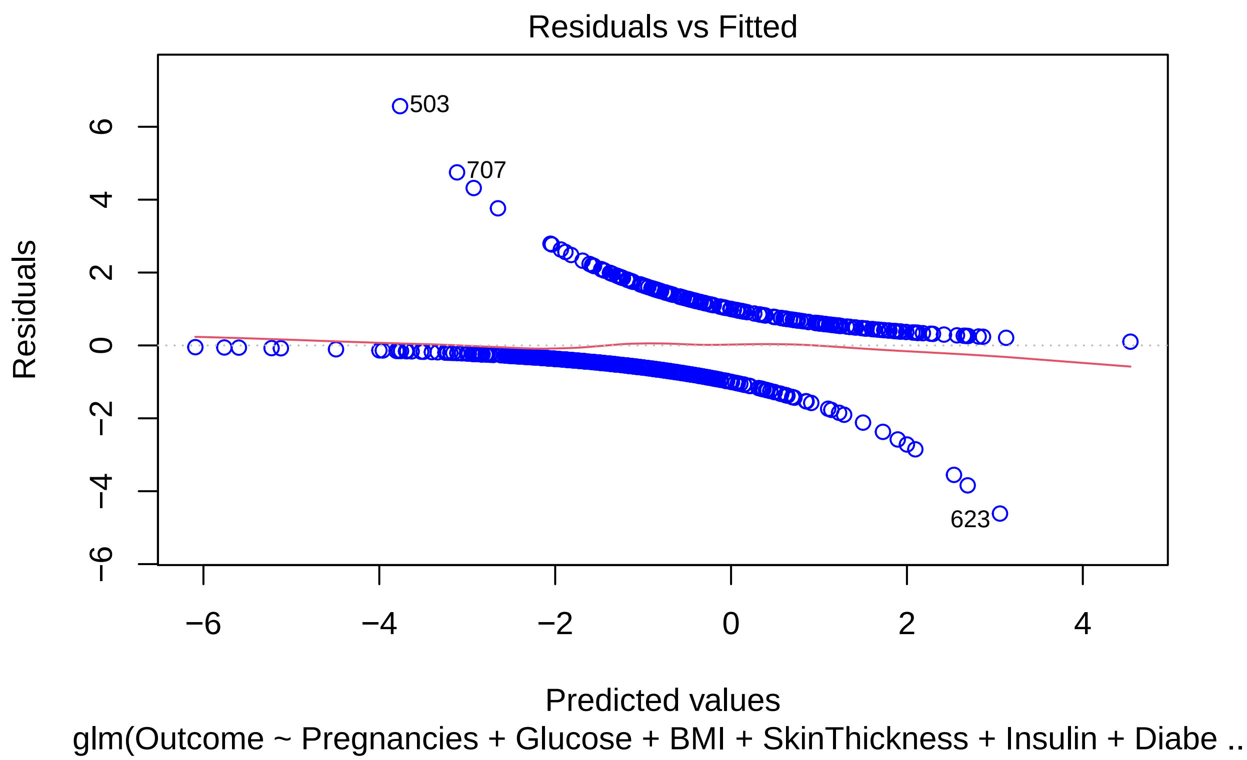
```r
confusionMatrix(factor(diabetes.test$Outcome), factor(predict_test))
```

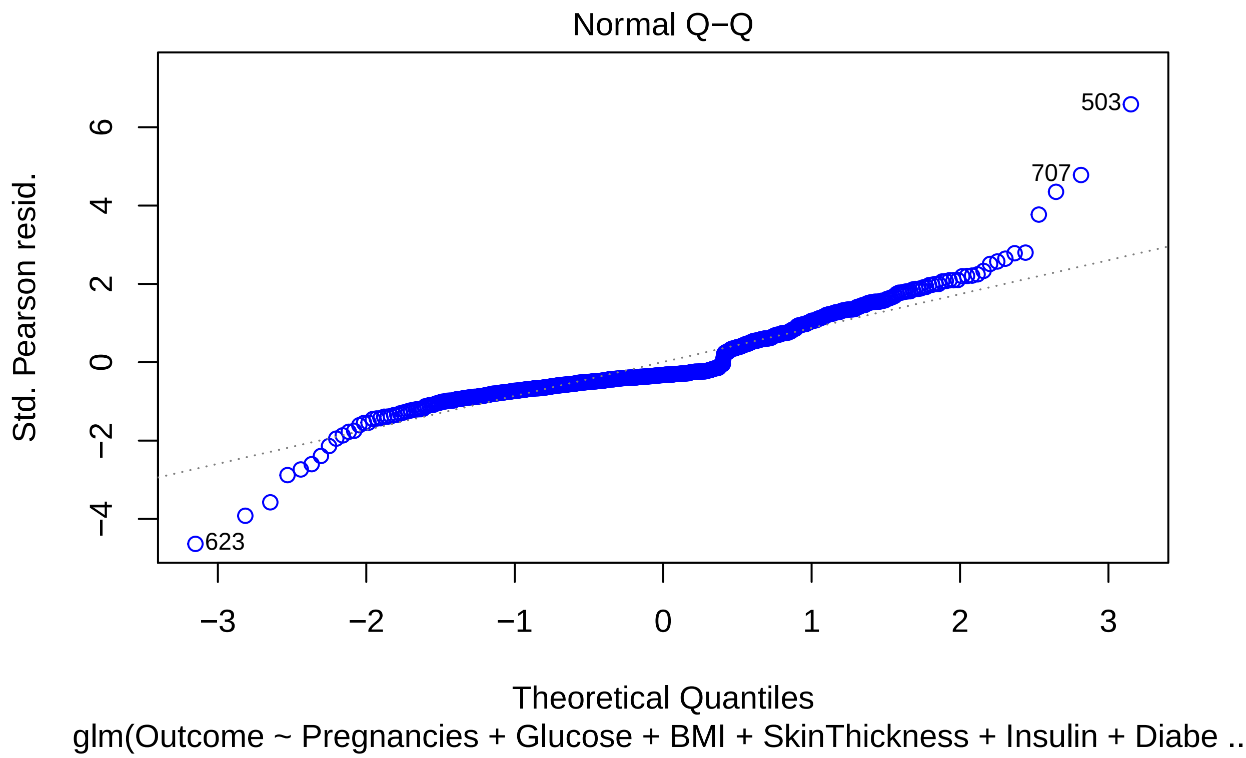
```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction  0  1
##          0 83 13
##          1 17 40
##                                           
##                Accuracy : 0.8039          
##                  95% CI : (0.7321, 0.8636)
##     No Information Rate : 0.6536          
##     P-Value [Acc > NIR] : 3.3e-05         
##                                           
##                   Kappa : 0.5745          
##                                           
##  Mcnemar's Test P-Value : 0.5839          
##                                           
##             Sensitivity : 0.8300          
##             Specificity : 0.7547          
##          Pos Pred Value : 0.8646          
##          Neg Pred Value : 0.7018          
##              Prevalence : 0.6536          
##          Detection Rate : 0.5425          
##    Detection Prevalence : 0.6275          
##       Balanced Accuracy : 0.7924          
##                                           
##        'Positive' Class : 0              
## 
```

Applying the Logistic Regression Algorithm, the Accuracy is 80%, sensivitity is 83% and specificity is 75%

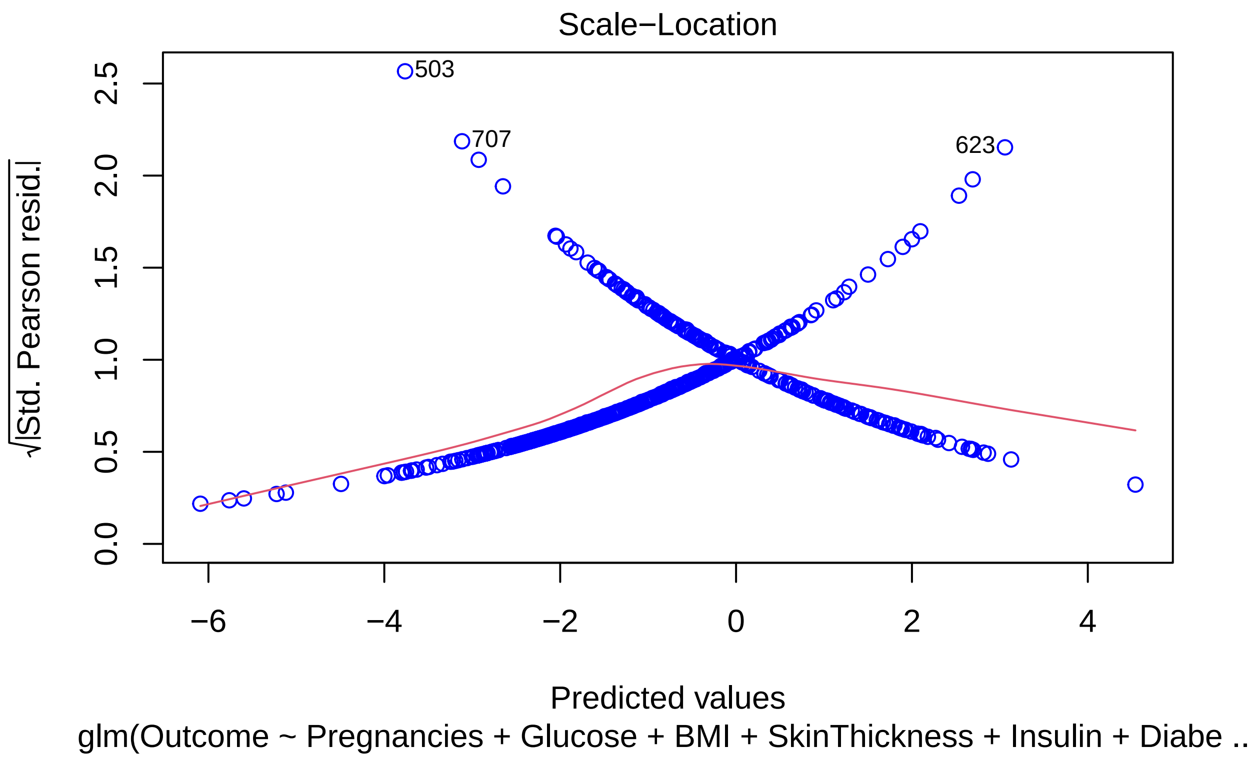
Plotting ROC curve of Logistic Regression Model

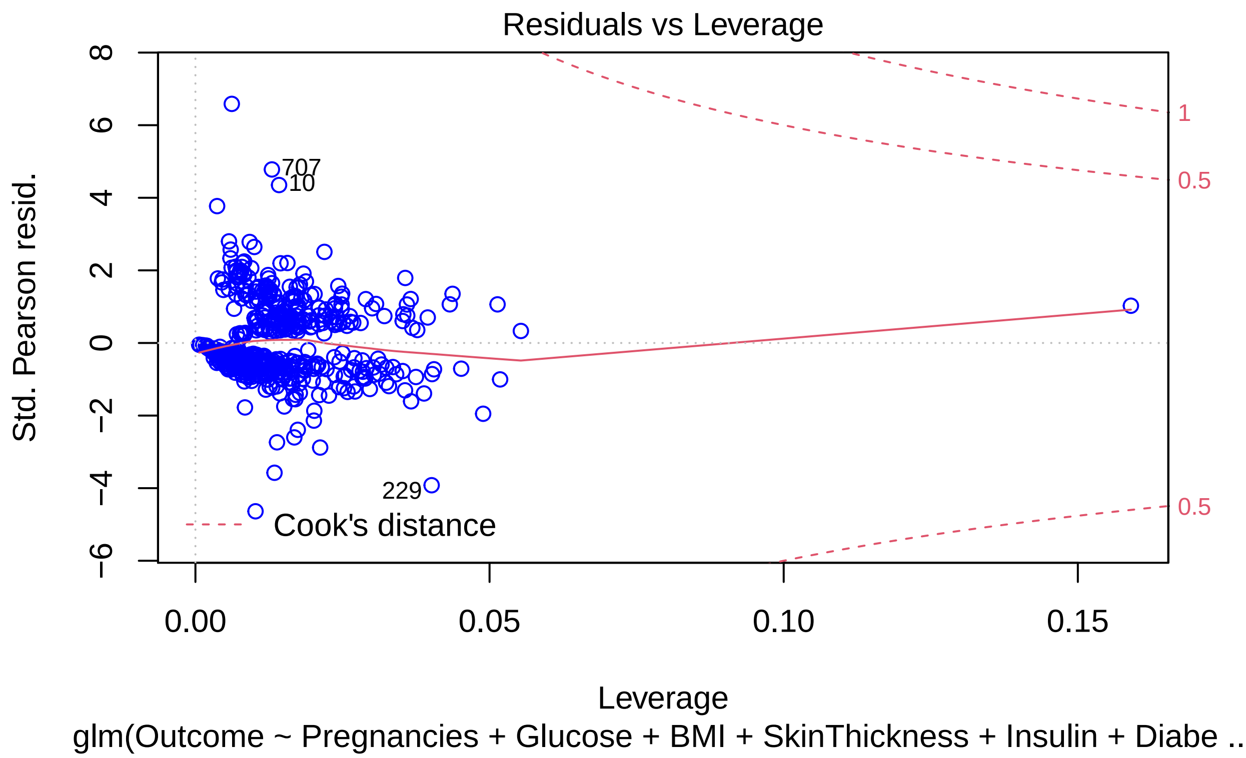
```
plot(model_glm, col='blue')
```





Normal Q−Q
Std. Pearson resid.
6
4
2
0
−2
−4
503
707
623
−3
−2
−1
0
1
2
3
Theoretical Quantiles
glm(Outcome ~ Pregnancies + Glucose + BMI + SkinThickness + Insulin + Diabe ...


Scale−Location
503
707
623
√|Std. Pearson resid.|
0.0
0.5
1.0
1.5
2.0
2.5
−6
−4
−2
0
2
4
Predicted values
glm(Outcome ~ Pregnancies + Glucose + BMI + SkinThickness + Insulin + Diabe ...



The Z score is a test of statistical significance that helps you decide whether or not to reject the null hypothesis. It also gives you an idea of how far from the mean a data point is. Looking at the summary, we can see which variables are significant by comparing the p-values. P-values with '***' next to them are significant and play a role in whether a subject has diabetes or not.

Plot the results of the two machine learning models

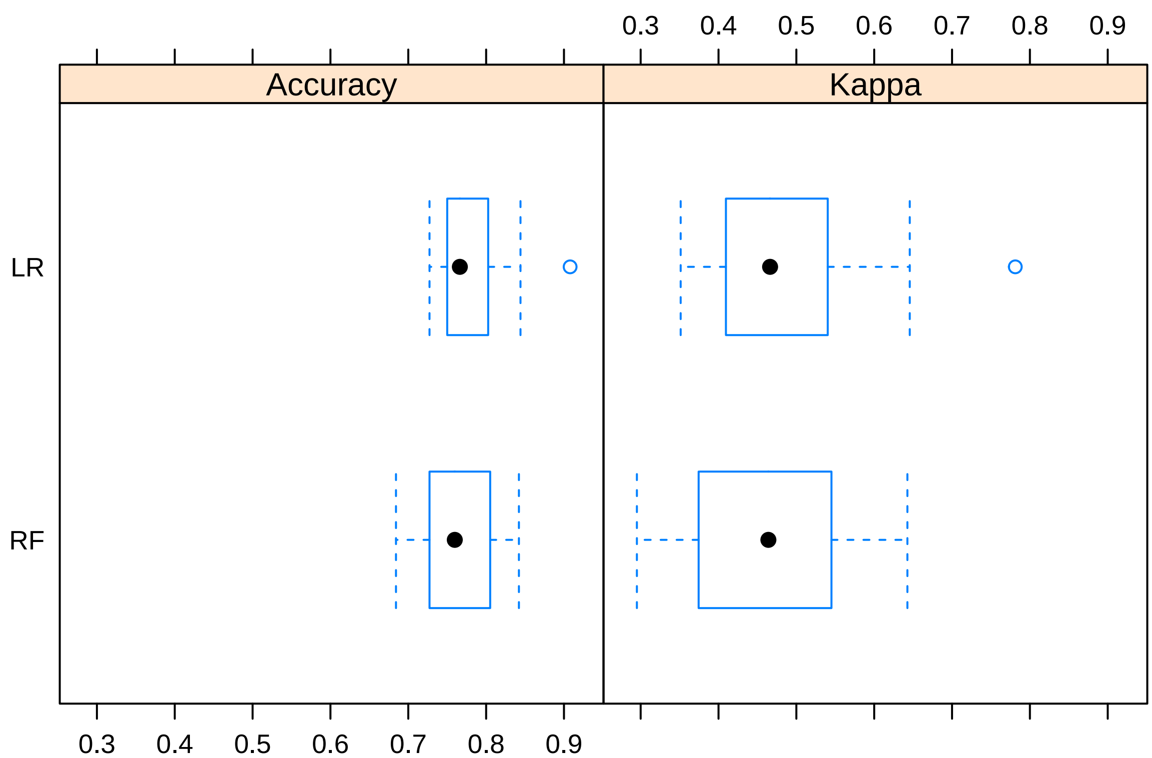
```
# prepare training scheme
control <- trainControl(method="repeatedcv", number=10, repeats=3)
set.seed(7)
fit.rf <- train(Outcome~., data=diabetes, method="rf", trControl=control)
set.seed(7)
fit.glm <- train(Outcome~., data=diabetes, method="glm", trControl=control)
# collect resamples
results <- resamples(list(LR=fit.glm, RF=fit.rf))
```

Summarize the distributions

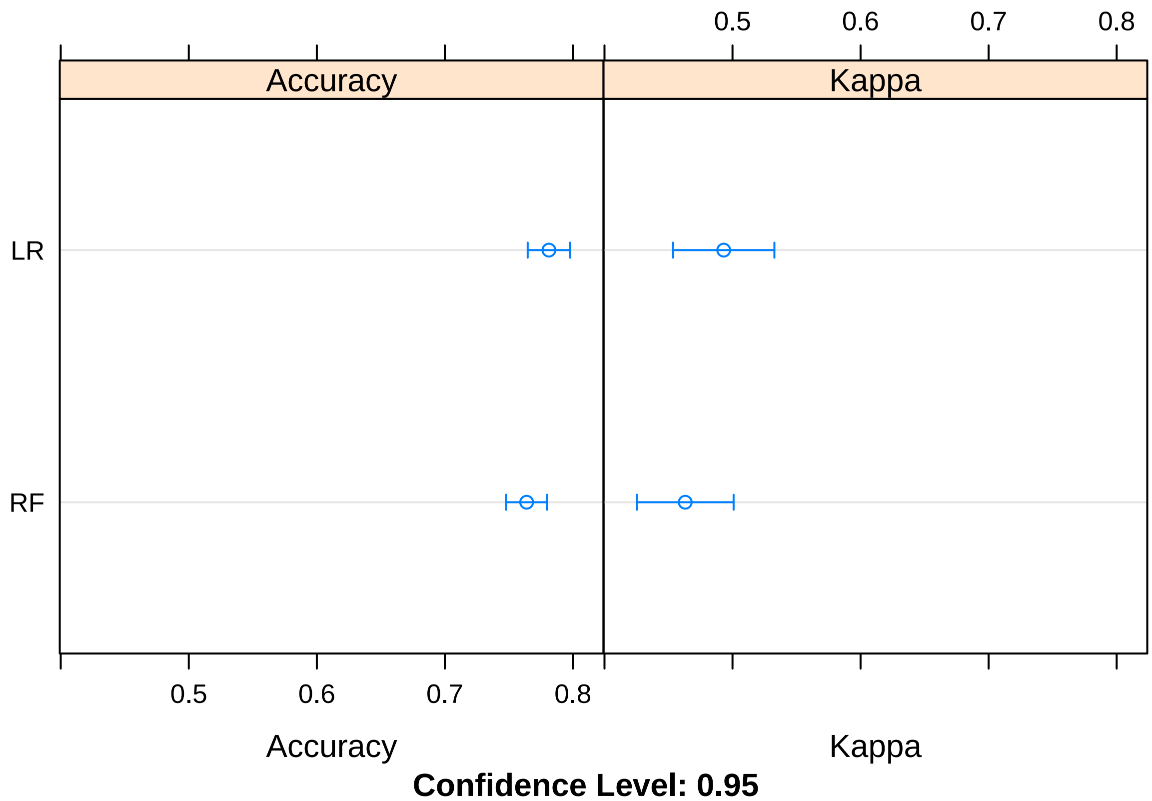
```r
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: LR, RF
## Number of resamples: 30
##
## Accuracy
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## LR 0.7272727 0.7508117 0.7662338 0.7812657 0.8000256 0.9078947    0
## RF 0.6842105 0.7305195 0.7597403 0.7638528 0.8019481 0.8421053    0
##
## Kappa
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## LR 0.3513839 0.4168485 0.4662541 0.4931161 0.5391907 0.7812500    0
## RF 0.2951613 0.3778304 0.4640696 0.4630809 0.5447483 0.6426332    0
```

```r
# boxplots of results
bwplot(results)
```

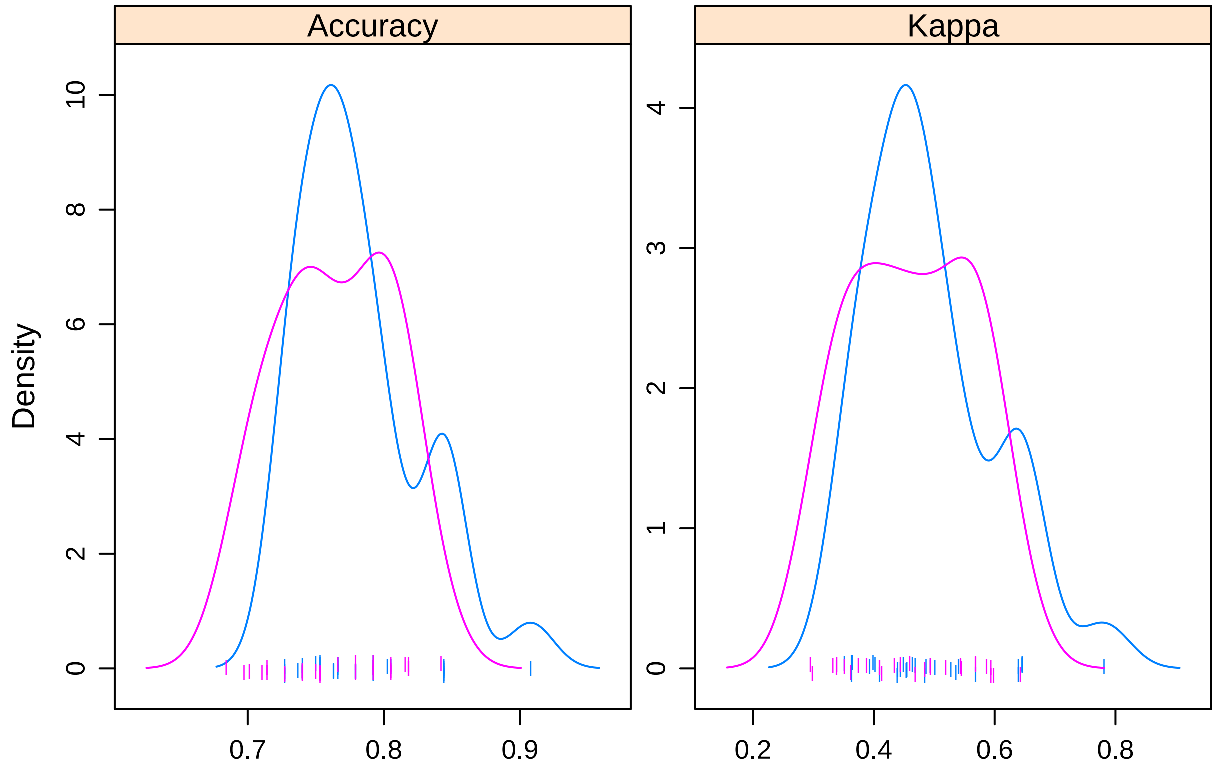


```r
# dot plots of results
dotplot(results)
```



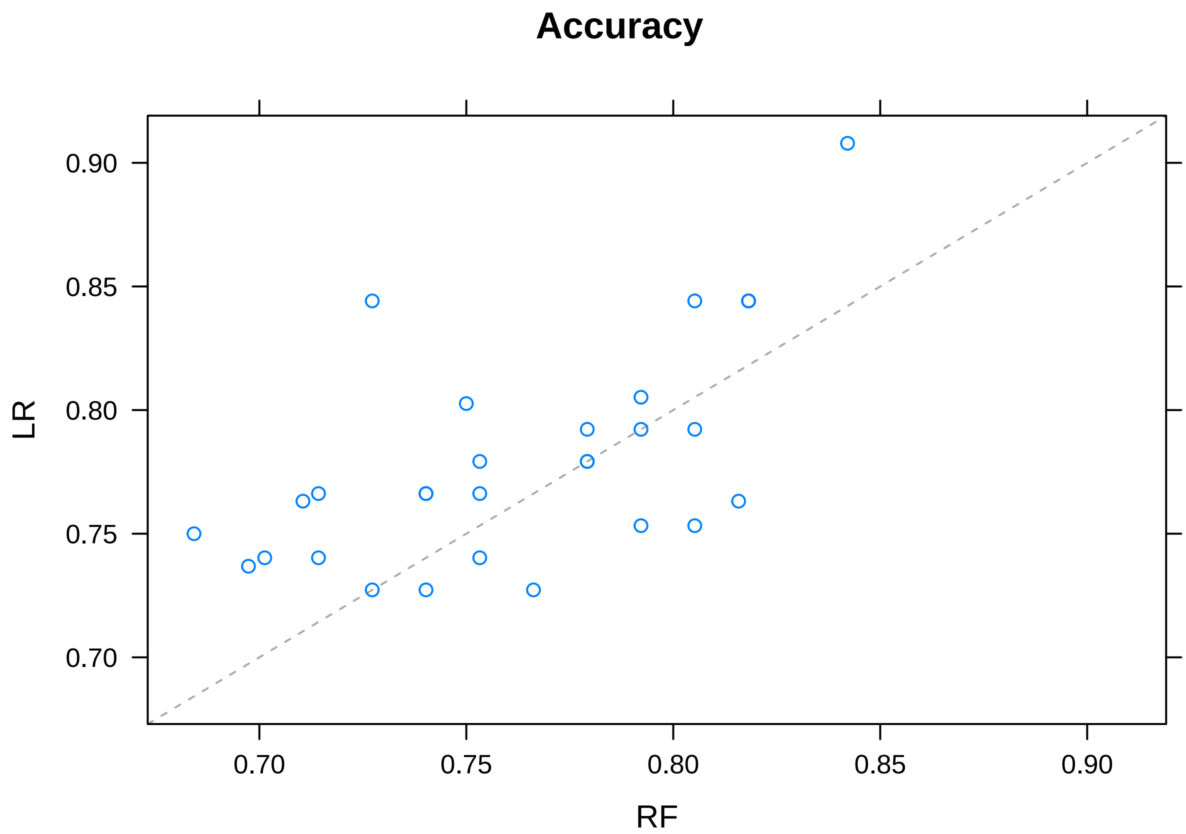
The distributions are summarized in terms of the percentiles. The distributions are summarized as box plots and finally the distributions are summarized as dot plots.

```r
# density plots of accuracy
scales <- list(x=list(relation="free"), y=list(relation="free"))
densityplot(results, scales=scales, pch = "|")
```



xyplot plots to compare models

```
xyplot(results, models=c("LR", "RF"))
```



Difference in model predictions

```r
diffs <- diff(results)

# summarize p-values for pair-wise comparisons
summary(diffs)
```

```
##
## Call:
## summary.diff.resamples(object = diffs)
##
## p-value adjustment: bonferroni
## Upper diagonal: estimates of the difference
## Lower diagonal: p-value for H0: difference = 0
##
## Accuracy
##    LR      RF
## LR         0.01741
## RF 0.01599
##
## Kappa
##    LR      RF
## LR         0.03004
```

```
## RF 0.07278
```

Results:

*Context: The goal of the project was to predict diabetes among PIMA women using variables such as BMI, Glucose, Pregnancies,Pedigree,Age, Family, Insulin, Skin Thickness.

Problem: The problem was to predict diabetes and the Machine Learning Models in this project were able to effectively predict the outcome.

Solution: We can interpret from the p-values of our Models that BMI and glucose are the biggest factors in determining whether members of the PIMA Indian have diabetes. But these two alone are not sufficient enough to accurately predict the outcome.

Findings: Accuracy is 79% the percentage of correctly classified instances out of all instances using Random Forest Model and applying the Logistic Regression Algorithm, the Accuracy is 80%.

Limitations: The sample size being not large enough to validate accurate predictions could be considered a limitation. A data set of at least 100,000 or even a million observations would be ideal for accurate predictions. The sample data comprising of only women, is also a limitation to universally apply the prediction algorithms.

Conclusions:

Based on the concepts learned in the data science course series, all aspects of building an effetive model to predict the outcome has been implemented. However, the larger the data set more accurate the predicted outcome would be. Patterns were established using data exploration and validation. The project shows us what are the most important factors that influence a person to have diabetes. Predictive models improve prediction performance but they don't provide outstanding results. Maybe other Machine learning models can be tried to see how it influences the results. The patterns identified using Data exploration methods were validated using the modeling techniques employed.Based on the above modelling, Logistic Regression is the best predictive model to determine if there is a possibility of diabetic outcome in a person.

#Reference:

1. University of Chicago Press Journals, https://www.journals.uchicago.edu/doi/full/10.1086/693853?mobileUi=0&
2. https://fderyckel.github.io/machinelearningwithr/logistic.html
3. http://www.joyofdata.de/blog/illustrated-guide-to-roc-and-auc/
4. https://github.com/joyofdata/joyofdata-articles/tree/master/roc-auc
5. https://machinelearningmastery.com/compare-the-performance-of-machine-learning-algorithms-in-r/ 6.https://www.kaggle.com/nileshvarshney1/pima-indians-diabetes