Medical Appointment Analysis

Nirmal Sai Swaroop Janapaneedi

26/05/2020

1. Introduction

We'll try to evaluate the accuracy of the **Medical Appointment No-Show** dataset which is available on Kaggle.com.

As part of this exercise, we'll also review the complete dataset and try to identify some pattern based on the data available.

This Report includes:

- Data Analysis and Visualization
- Split the data into Train(90%) and Test(10%) set
- Training Models
- Compare Accuracy with Test data set
- Combining of all train models results into Ensemble to predict the Optimal accuracy by decrease any variance, if any.

Intial Setup:

- Loading required Libraries
- Creating dataset from CSV file
- Clean-up data
- Defining functions to plot graph

```r
#load Libraries if not available
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(summarytools))
  install.packages("summarytools", repos = "http://cran.us.r-project.org")
if(!require(gbm))
  install.packages("gbm", repos = "http://cran.us.r-project.org")
if(!require(devtools))
  install.packages("devtools", repos = "http://cran.us.r-project.org")
```

```r
# Read the dataset from csv file using the read_csv function
ds_apptnoshow <- read_csv("https://raw.githubusercontent.com/varora9/EDX_Project/master/Kaggle
                col_names = c("PatientId","AppointmentID","Gender","ScheduledDay","Appointment
                "Neighbourhood","Scholarship","Hipertension","Diabetes","Alcoholism",
                "Handcap","SMS_received","No-show"))
# remove the duplicate header row
ds_apptnoshow <- ds_apptnoshow[-1,]

# Covert columns which has numeric data from chr into double. This is required
for training th ds_apptnoshow$Age = as.double(ds_apptnoshow$Age)
ds_apptnoshow$Scholarship = as.double(ds_apptnoshow$Scholarship)
ds_apptnoshow$Hipertension = as.double(ds_apptnoshow$Hipertension)
ds_apptnoshow$Diabetes = as.double(ds_apptnoshow$Diabetes)
ds_apptnoshow$Alcoholism = as.double(ds_apptnoshow$Alcoholism)
ds_apptnoshow$Handcap = as.double(ds_apptnoshow$Handcap)
ds_apptnoshow$SMS_received = as.double(ds_apptnoshow$SMS_received)

#Rename No-Show Column Name
names(ds_apptnoshow)[names(ds_apptnoshow) == 'No-show'] <- 'NoShow'

# Defining function to generate graph based on columnname
plotGraphbasedonColumnname <- function(title, colname){
  ds_apptnoshow %>% ggplot(aes(get(colname))) +
    geom_bar(stat='count',colour="darkgreen", fill="lightgreen") +
    geom_text(aes(label = scales::percent((..count..)/sum(..count..))),
              stat = "count", vjust = -0.5,
    colour="black" ) + labs(title = title,
         x = colname,
         y = "Count")
}


# Defining function to generate graph based on columnname in
facet plotGraphbasedonNoShow <- function(title, colname){
  cols<- c('lightgreen','darkred')
  ds_apptnoshow %>% ggplot(aes(NoShow, fill=NoShow)) +
    geom_bar(stat='count',colour="darkgreen") +
    facet_grid(~get(colname)) +
    geom_text(aes(label =
    scales::percent((..count..)/sapply(PANEL,
                              FUN=function(x) sum(count[PANEL == x])))),
            stat = "count", vjust = -0.5, hjust= 0.5, colour="black" ) +
    scale_fill_manual(name="Bars",values=cols) +
    labs(title = title,
         x = "No-Show",
         y = "Count")
}
```

2. Data Analysis and Visualization

Data Dictionary:

- PatientId - Identification of a patient
- AppointmentID - Identification of each appointment
- Gender - Male or Female .
- ScheduledDay - The day of the actual appointment, when they have to visit the doctor.
- AppointmentDay - The day someone registered the appointment, this is before appointment of course.
- Age - Patient’s Age.
- Scholarship - True of False . Data is stored in binary format (1,0)
- Hipertension - True or False. Data is stored in binary format (1,0).
- Alcoholism - True or False. Data is stored in binary format (1,0).
- Handcap - Data is stored in the range of 1 to 4.
- SMS_received - 1 or more messages sent to the patient.
- No-show - Yes or No. Yes = Didn’t show-up, No = Show-up.

Observation - Scholarship is a broad topic, consider reading this article.

a. Quick glance on the Appointment No-Show Dataset

```
#Print Top 6 rows of the dataset
head(ds_apptnoshow)
```

```
## # A tibble: 6 x 14
##   PatientId AppointmentID Gender ScheduledDay AppointmentDay   Age Neighbourhood
##   <chr>     <chr>         <chr>  <chr>        <chr>          <dbl> <chr>
## 1 29872499~ 5642903       F      2016-04-29T~ 2016-04-29T00~    62 JARDIM DA PE~
## 2 55899777~ 5642503       M      2016-04-29T~ 2016-04-29T00~    56 JARDIM DA PE~
## 3 42629622~ 5642549       F      2016-04-29T~ 2016-04-29T00~    62 MATA DA PRAIA
## 4 86795121~ 5642828       F      2016-04-29T~ 2016-04-29T00~     8 PONTAL DE CA~
## 5 88411864~ 5642494       F      2016-04-29T~ 2016-04-29T00~    56 JARDIM DA PE~
## 6 95985133~ 5626772       F      2016-04-27T~ 2016-04-29T00~    76 REPÚBLICA
## # ... with 7 more variables: Scholarship <dbl>, Hipertension <dbl>,
## #   Diabetes <dbl>, Alcoholism <dbl>, Handcap <dbl>, SMS_received <dbl>,
## #   NoShow <chr>
```

b. Quick glance of the Datset’s Summary

```
#summary of Dataset
summary(ds_apptnoshow)
```

```
##   PatientId          AppointmentID         Gender            ScheduledDay
##  Length:110527      Length:110527      Length:110527      Length:110527
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##  AppointmentDay          Age           Neighbourhood        Scholarship
##  Length:110527      Min.   : -1.00   Length:110527      Min.   :0.00000
##  Class :character   1st Qu.: 18.00   Class :character   1st Qu.:0.00000
##  Mode  :character   Median : 37.00   Mode  :character   Median :0.00000
##                     Mean   : 37.09                      Mean   :0.09827
##                     3rd Qu.: 55.00                      3rd Qu.:0.00000
##                     Max.   :115.00                      Max.   :1.00000
##   Hipertension       Diabetes         Alcoholism         Handcap
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.0000   Min.   :0.00000
##  1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.00000
##  Median :0.0000   Median :0.00000   Median :0.0000   Median :0.00000
##  Mean   :0.1972   Mean   :0.07186   Mean   :0.0304   Mean   :0.02225
##  3rd Qu.:0.0000   3rd Qu.:0.00000   3rd Qu.:0.0000   3rd Qu.:0.00000
##  Max.   :1.0000   Max.   :1.00000   Max.   :1.0000   Max.   :4.00000
##   SMS_received      NoShow
##  Min.   :0.000   Length:110527
##  1st Qu.:0.000   Class :character
##  Median :0.000   Mode  :character
##  Mean   :0.321
##  3rd Qu.:1.000
##  Max.   :1.000
```

c. Medical Appointment No-Show dataset contain 110527 rows and 14 columns. d.

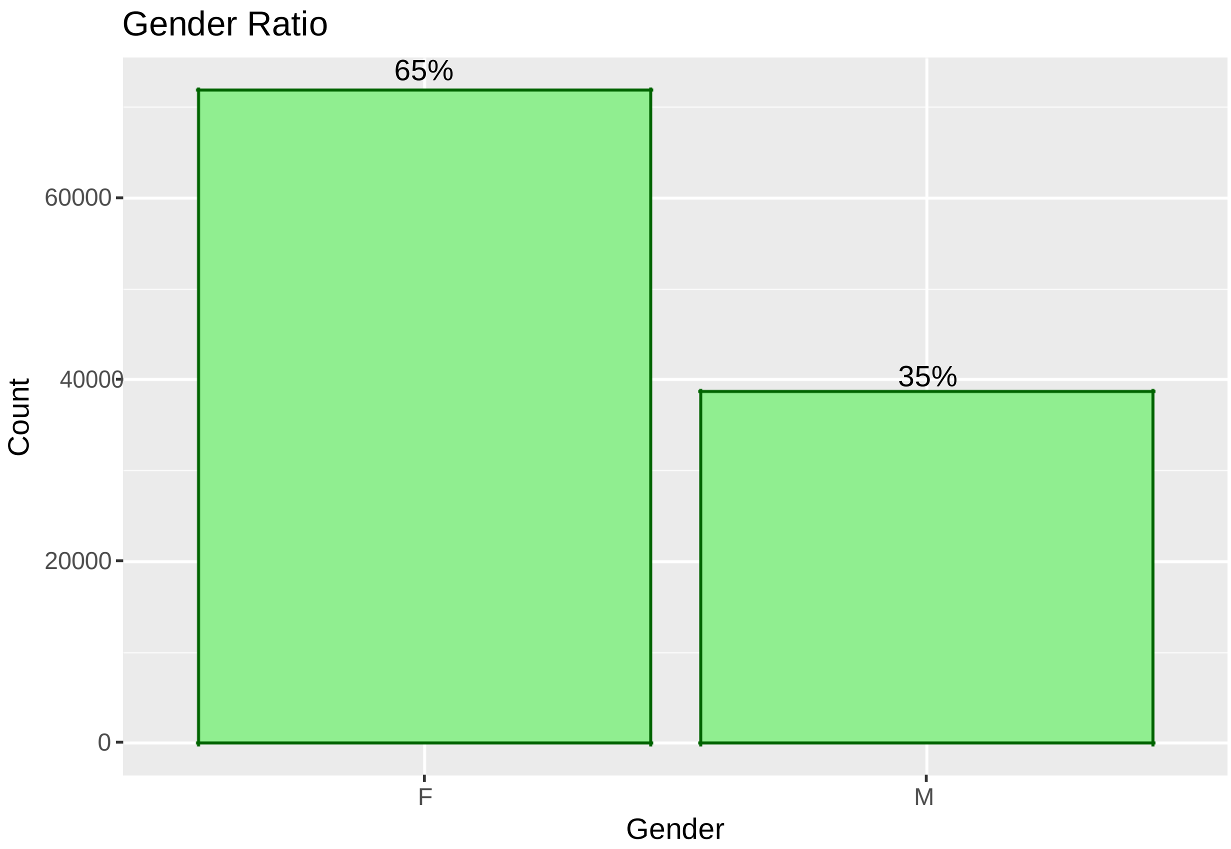
Total count of unique Patients - 62299

e. Data segregation based on Gender. Females patients are significantly higher than Male patients.

```
# Print Dataset no. based on Gender
ds_apptnoshow %>% group_by(Gender) %>% summarize(Gender_Count= n())
```

```
## # A tibble: 2 x 2
##   Gender Gender_Count
##   <chr>         <int>
## 1 F             71840
## 2 M             38687
```

```
# plot graph to depict Gender %
ds_apptnoshow %>% ggplot(aes(Gender)) +
  geom_bar(stat='count', fill="lightgreen", color="darkgreen") +
  geom_text(aes(label = scales::percent((..count..)/sum(..count..))),
            stat = "count", vjust = -0.45, hjust = 0.5,
  colour="black" ) + labs(title = "Gender Ratio",
       x = "Gender",
       y = "Count")
```



f. Multiple patients have taken appointment multiple times and here is quick glance of the top 10 appointments Maximum appointments taken by a Patient is **88**.

```
# No. of appointment taken by same patients
AppointCnt_Per_Patient <- ds_apptnoshow %>%
  group_by(PatientId) %>%
  summarize(AppointCnt = n())

# Top 10 Rows based on highest appointment count
top_n(AppointCnt_Per_Patient %>% arrange(desc(AppointCnt)),10)
```

```
## Selecting by AppointCnt

## # A tibble: 10 x 2
##   PatientId     AppointCnt
##   <chr>              <int>
##  1 822145925426128      88
##  2 99637671331          84
##  3 26886125921145       70
##  4 33534783483176       65
##  5 258424392677         62
##  6 6264198675331        62
##  7 75797461494159       62
##  8 871374938638855      62
##  9 66844879846766       57
## 10 872278549442         55
```

g. 3.2% Age data in dataset is incorrect where Age is documented as less than or equal to 0. Here is the quick review of the data.

```
# Review Age Data and fount incorrect Age data based
on Gender. ds_apptnoshow %>% filter(Age <= 0) %>%
  group_by(Gender) %>% summarize(AgeErrorCount = n() )
```

```
## # A tibble: 2 x 2
##   Gender AgeErrorCount
##   <chr>          <int>
## 1 F               1722
## 2 M               1818
```

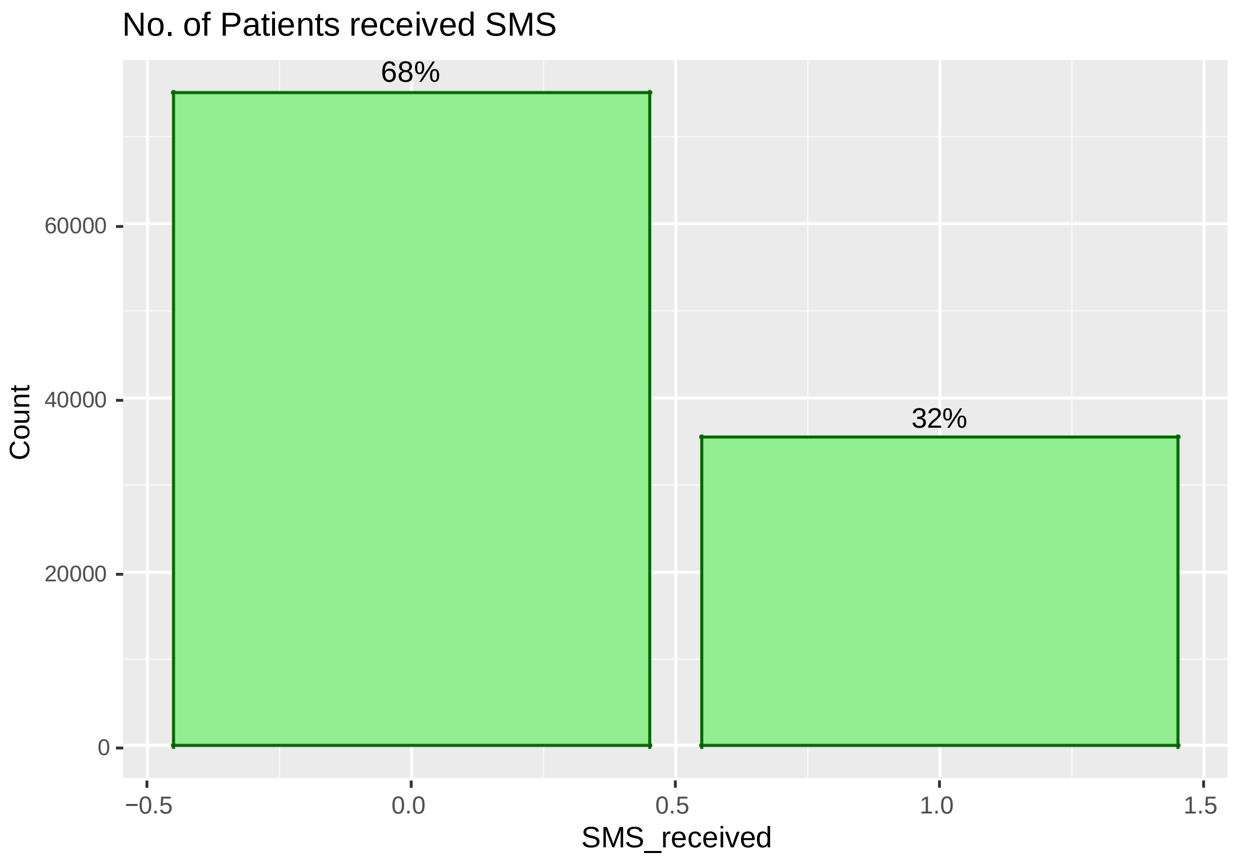
h. Only 32% Patients received SMS of their appointments and out of that 28% patients didn’t Show-Up.

Recommendation: Doctor’s office must need to look into this % to improve **No-Show** appointment rate.

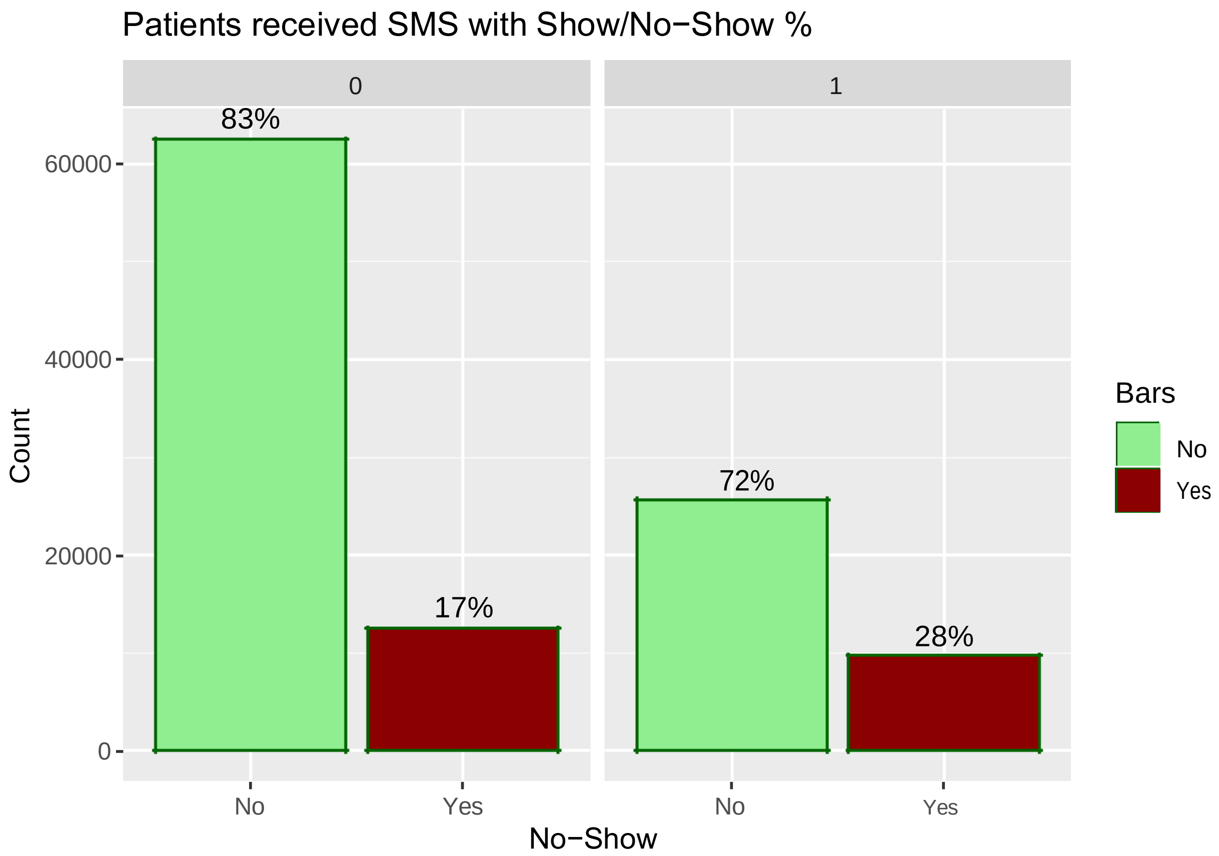
```
# SMS Received by Patients of the appointments in Percentage
# 0 = No, 1 = Yes
summarytools::freq(ds_apptnoshow$SMS_received, order = "freq")[1:2, 4]
```

```
##        0        1
## 67.89744 32.10256
```

```
#Plot a graph to depict the no. of patients and No-Show by SMS Received
plotGraphbasedonColumnname("No. of Patients received SMS","SMS_received")
```



```
plotGraphbasedonNoShow("Patients received SMS with Show/No-Show %", "SMS_received")
```



i. ~20% of the Patients didn't Show-Up post taking the appointments. Probably, reminders would have helped them.

```
# No Show Frequency in Percentage
# 0 = Show Up, 1 = Didn't Show up
summarytools::freq(ds_apptnoshow$NoShow, order = "freq")[1:2, 4]
```

```
##       No      Yes
## 79.80674 20.19326
```

j. Maximum Appointments observed on Wednesdays, followed by Tuesdays and minimal on Saturdays.

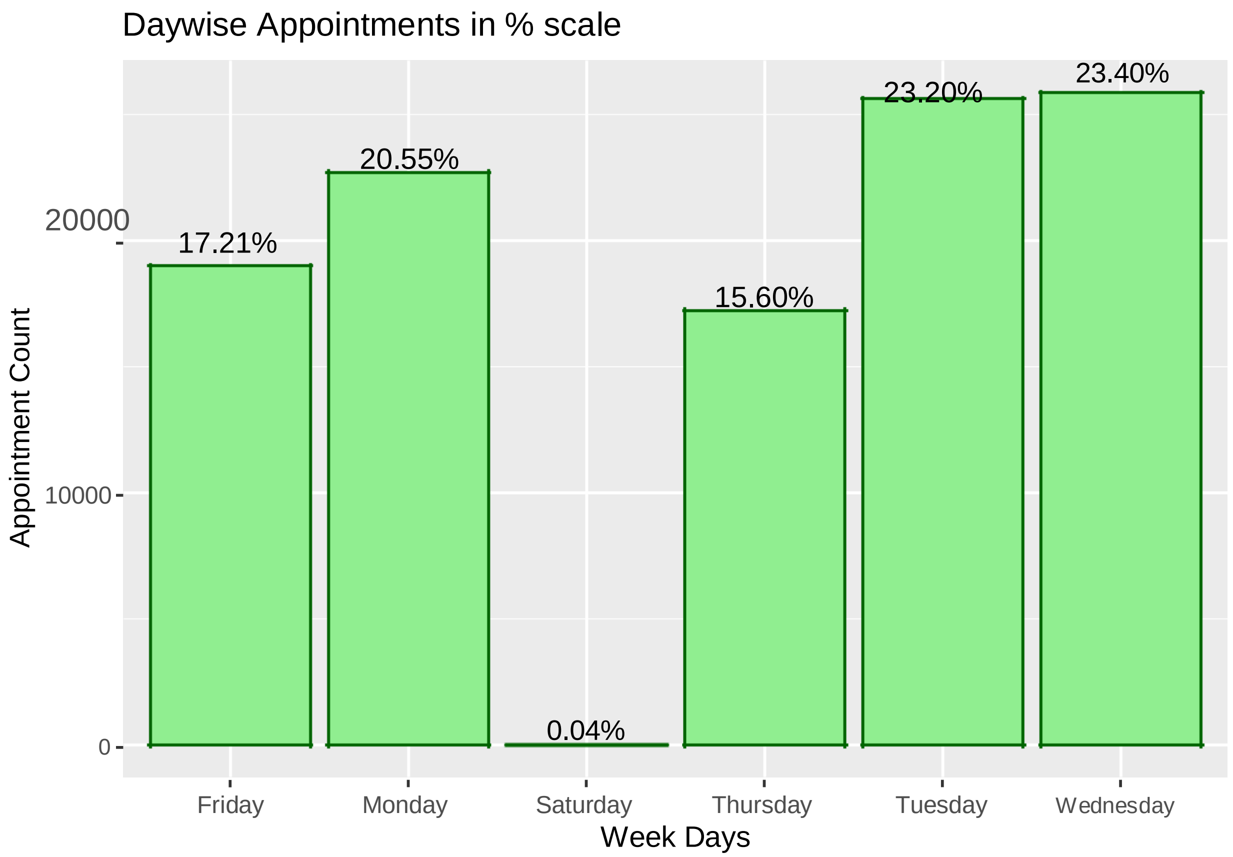
```
# Adding Appointment Day in the main Dataset
ds_apptnoshow <- ds_apptnoshow %>%
  mutate(apptdayoftheweek = weekdays(as.Date(AppointmentDay)),
         apptday = as.numeric(format(AppointmentDay, format = "%u")))
```

```
## Warning: NAs introduced by coercion
```

```
# Appointment Count on day
wise ds_apptnoshow %>%
  group_by(apptdayoftheweek, apptday) %>%
  summarize(daywiseAppts=n()) %>% arrange(apptday)
  %>% select(apptdayoftheweek, daywiseAppts)
```

```
## # A tibble: 6 x 2
## # Groups:   apptdayoftheweek [6]
##   apptdayoftheweek daywiseAppts
##   <chr>                   <int>
## 1 Friday                  19019
## 2 Monday                  22715
## 3 Saturday                   39
## 4 Thursday                17247
## 5 Tuesday                 25640
## 6 Wednesday               25867
```

```
#Plot Appointment Graph Day wise with %
scale ds_apptnoshow %>%
  ggplot(aes(apptdayoftheweek)) +
  geom_bar(stat='count',fill="lightgreen", color="darkgreen") +
  geom_text(aes(label = scales::percent((..count..)/sum(..count..))),
            stat = "count", vjust = -0.25, colour="black" ) +
  labs(title = "Daywise Appointments in % scale",
       x = "Week Days",
       y = "Appointment Count")
```

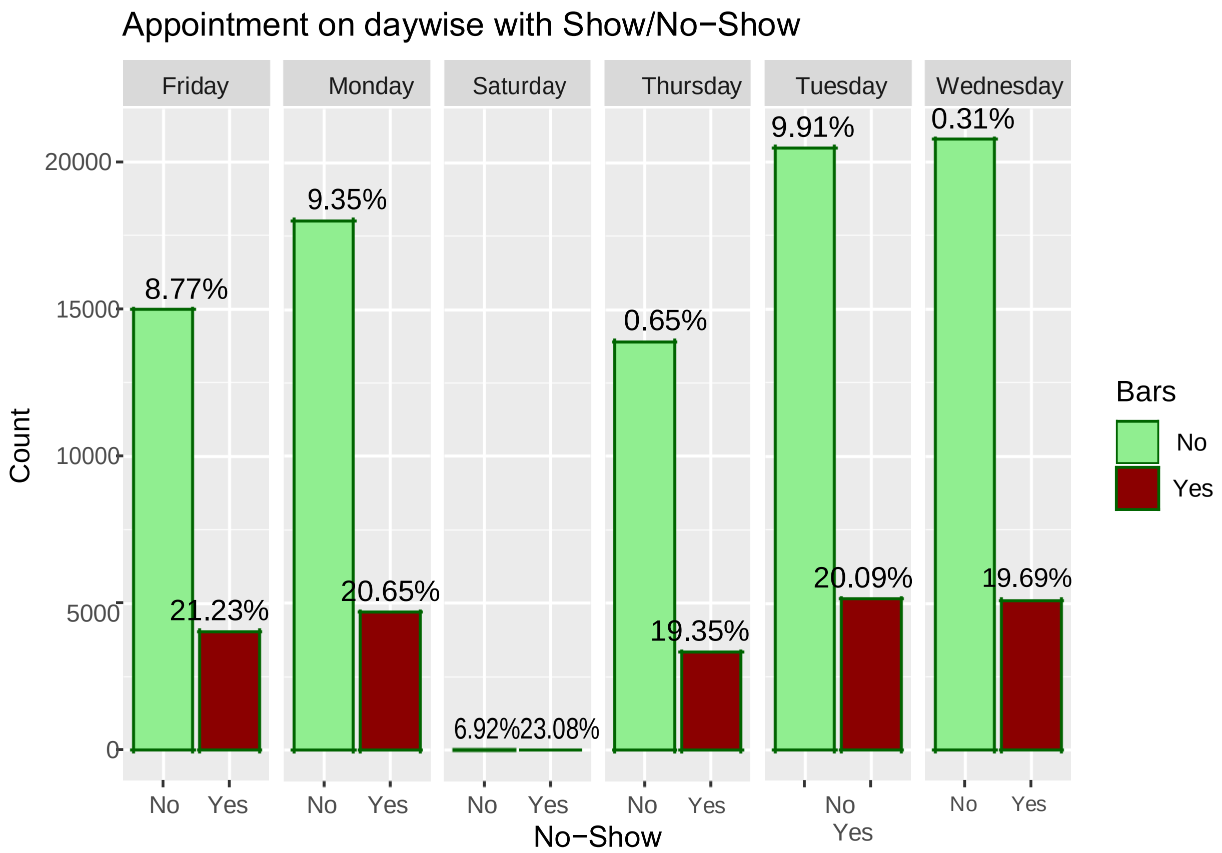


k. % of No-Shows observed same on Mondays and Fridays but total count of Patients on Mondays are higher than Fridays.

l. Overall percentage of No-Shows throughout week days is 19% or above.

m. No-Shows % on Saturdays is higher than other days but the Patients count is significantly low.

```
# Appointment on day wise with Show/No-Show
plotGraphbasedonNoShow("Appointment on daywise with Show/No-Show ", "apptdayoftheweek")
```

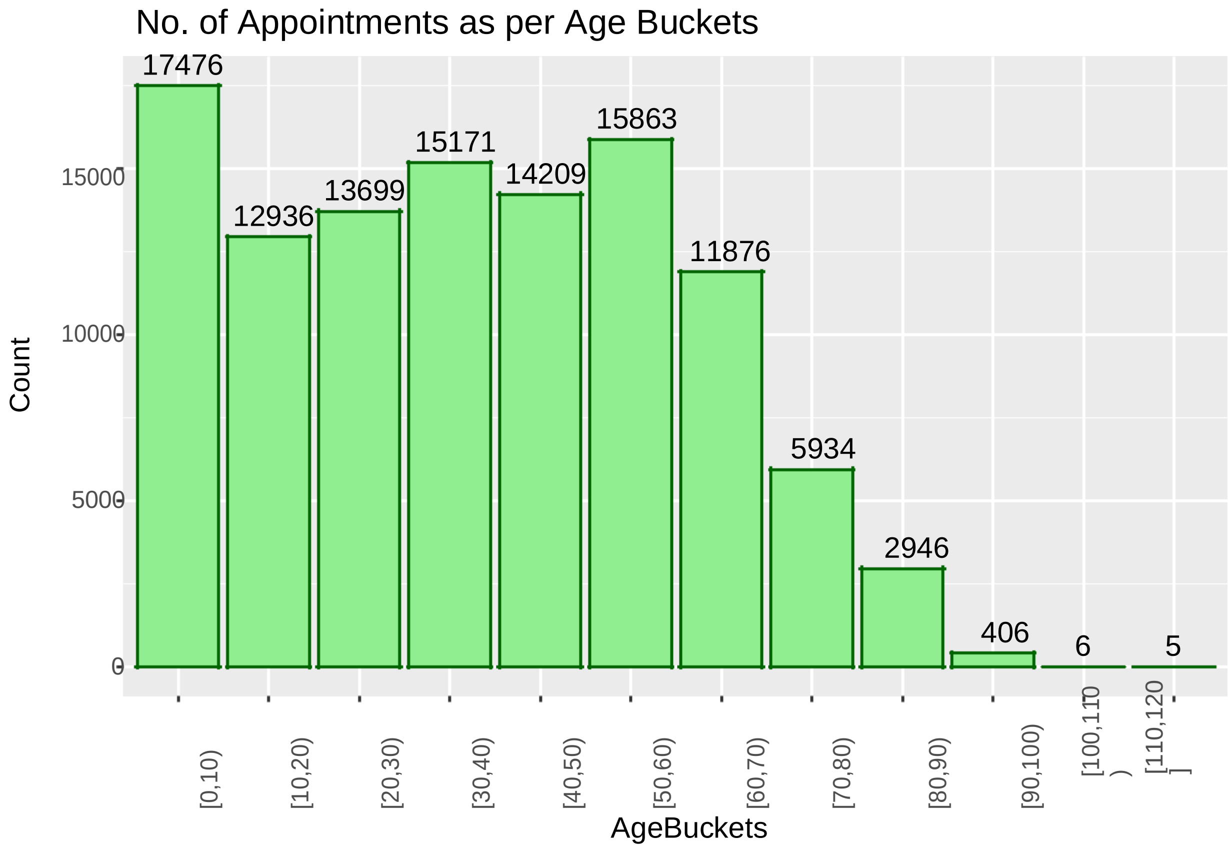


```
# Data Clean-up - Set Age to 0 where it is stored as
<0 ds_apptnoshow[ds_apptnoshow$Age <0,]$Age <- 0

# Create the Age Buckets to depict Age Pattern
ds_apptnoshow$AgeBuckets <- cut(ds_apptnoshow$Age, c(seq(0, 120,
                      10)), include.lowest=TRUE, right = FALSE)
```

n. Maximum no. of Patients observed in the Age Bucket of 0-10 followed by 50-60.

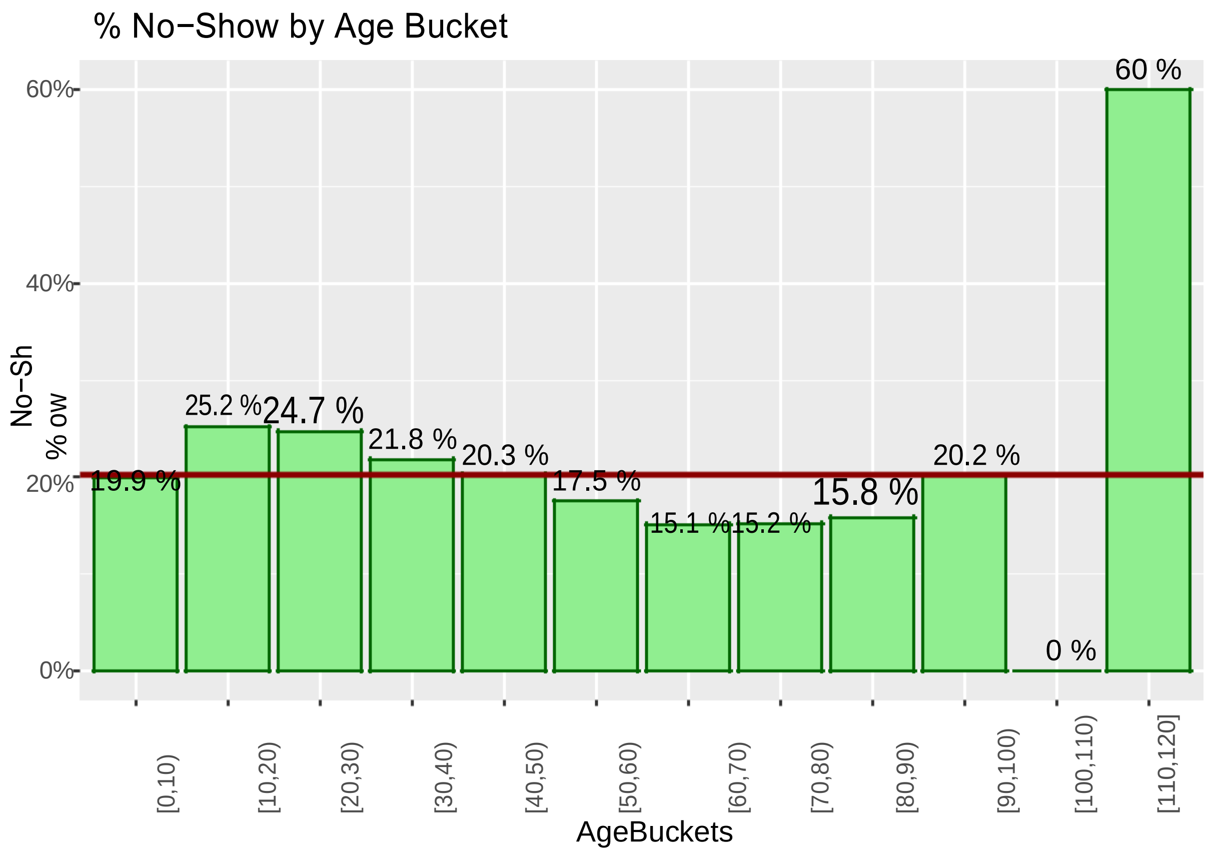
```
#Plot a graph to depict the no. of patients and No-Show by
Age Buckets ds_apptnoshow %>%
        group_by(AgeBuckets) %>% summarise(Total=n()) %>%
        ggplot(aes(AgeBuckets, Total)) +
        geom_bar(stat='identity',colour="darkgreen", fill="lightgreen") +
        geom_text(aes(label = Total),  vjust = -0.45) +
        theme(axis.text.x = element_text(angle = 90, hjust = 1))  +
        labs(title = "No. of Appointments as per Age Buckets",
             x = "AgeBuckets",
             y = "Count")
```



o. Maximum % of No-Show Patients observed in the Age Bucket of 110-120 but the no.of patients in that is significantly low hence we can ignore that No-Show ratio.

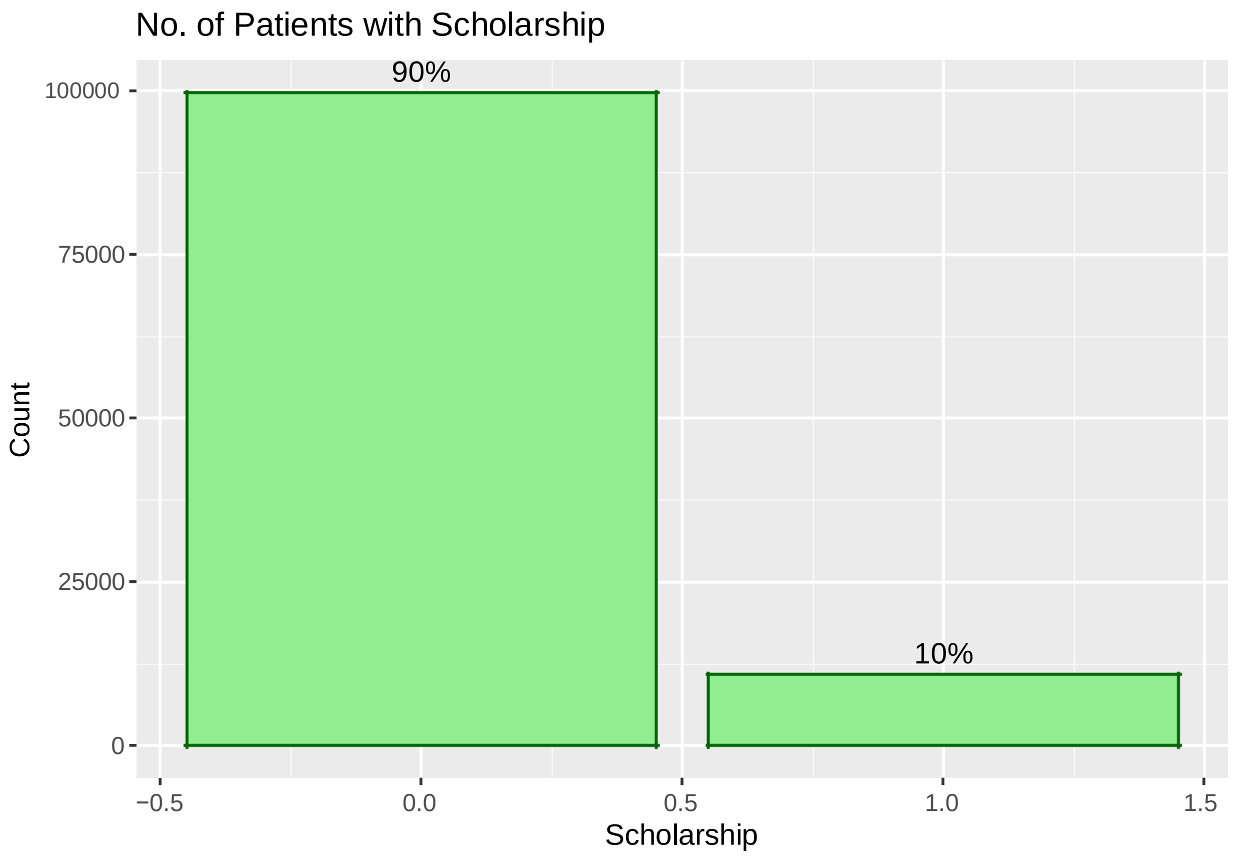
Patients in age bucket of 10-20 has maximum rate of No-Show followed by 20-30.

```
    ds_apptnoshow      %>%
    group_by(AgeBuckets) %>% summarise(percentNoShow=round(sum(NoShow=='Yes')/n(),3)) %>%
    ggplot(aes(AgeBuckets, percentNoShow)) +
    geom_bar(stat='identity',colour="darkgreen", fill="lightgreen") +
    geom_text(aes(label = paste(percentNoShow*100, "%")),    vjust = -0.45) +
    theme(axis.text.x = element_text(angle = 90, hjust = 1))+
    scale_y_continuous(labels = scales::percent) +
    labs(title = "% No-Show by Age Bucket",
         x = "AgeBuckets",
         y = "% No-Show") +
  geom_hline(yintercept = mean(ds_apptnoshow$NoShow=='Yes'), col='darkred', size=1)
```

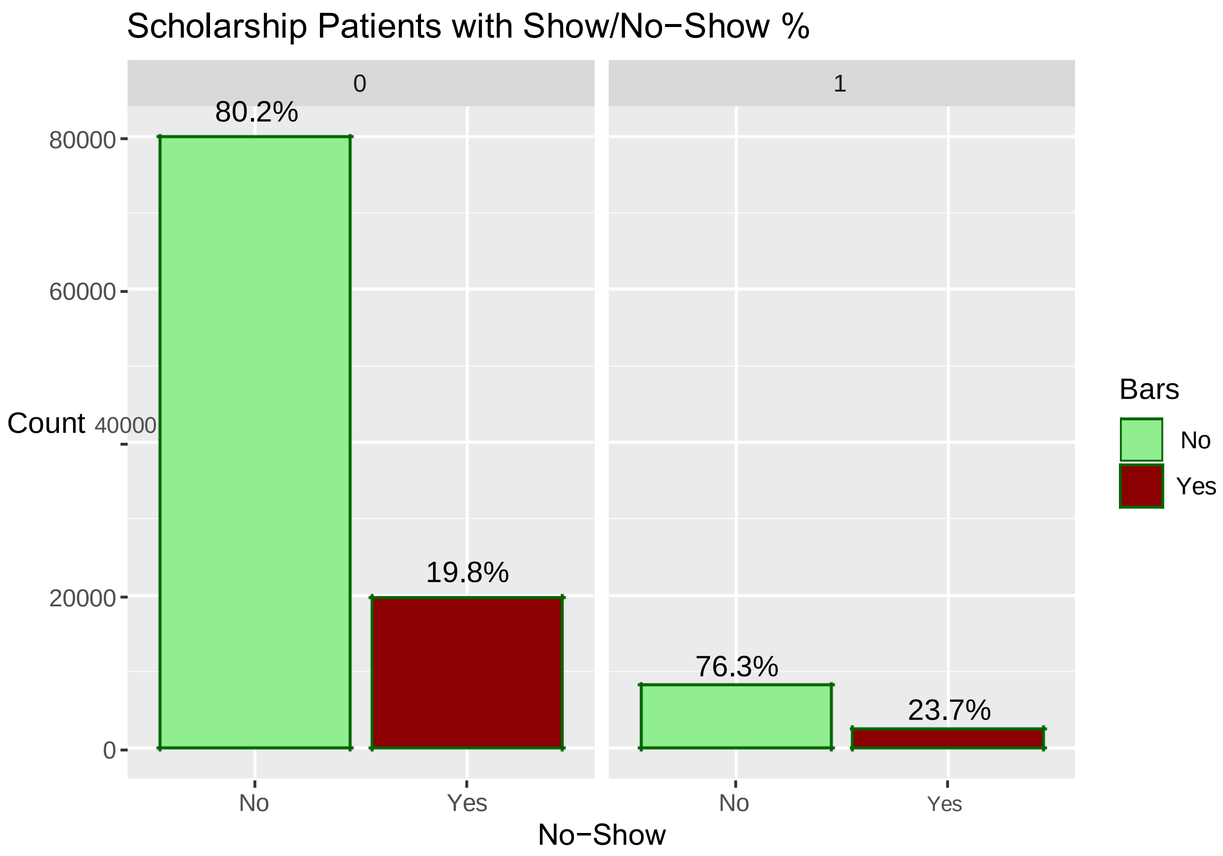


p. 10% of the total Patients count has taken social aid from government for their illness expense and out of the that 24% of the Patients didn't Show-Up as per their Scheduled Appointments.

```
#Plot a graph to depict the no. of patients and No-Show by Scholarship
plotGraphbasedonColumnname("No. of Patients with Scholarship", "Scholarship")
```

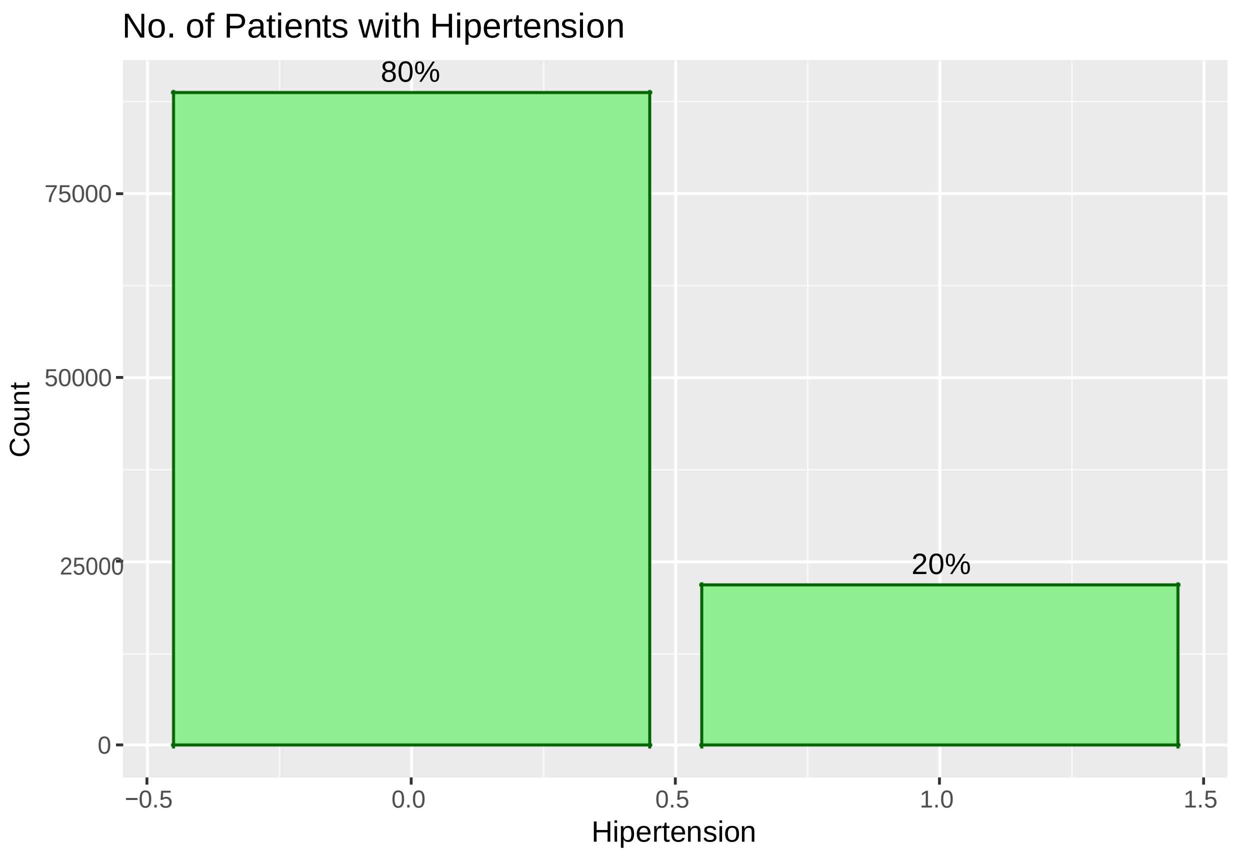


```
plotGraphbasedonNoShow("Scholarship Patients with Show/No-Show %","Scholarship")
```

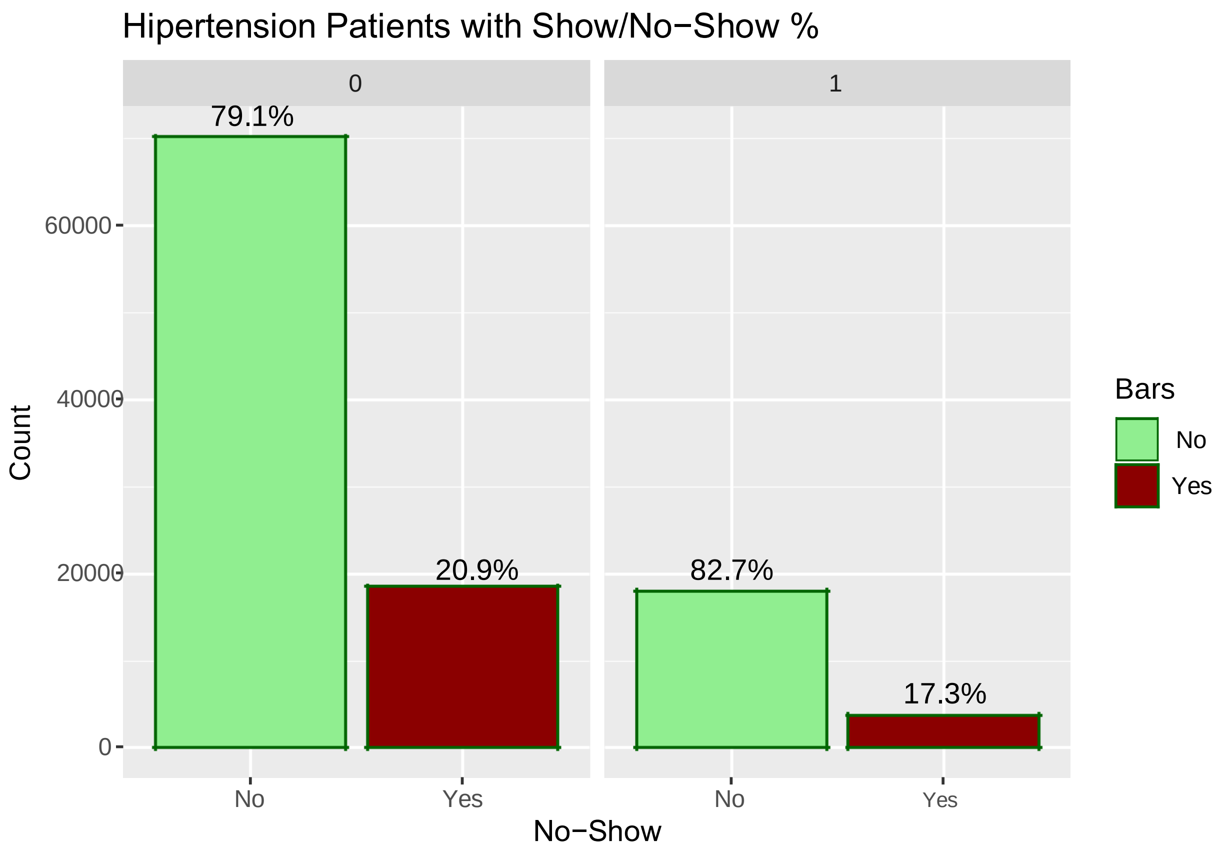


q. 20% of the total Patients observed with Hipertension and out of the that 17% of the Patients didn't Show-Up as per their Scheduled Appointments.

```
#Plot a graph to depict the no. of patients and No-Show by Hipertension
plotGraphbasedonColumnname("No. of Patients with Hipertension","Hipertension")
```

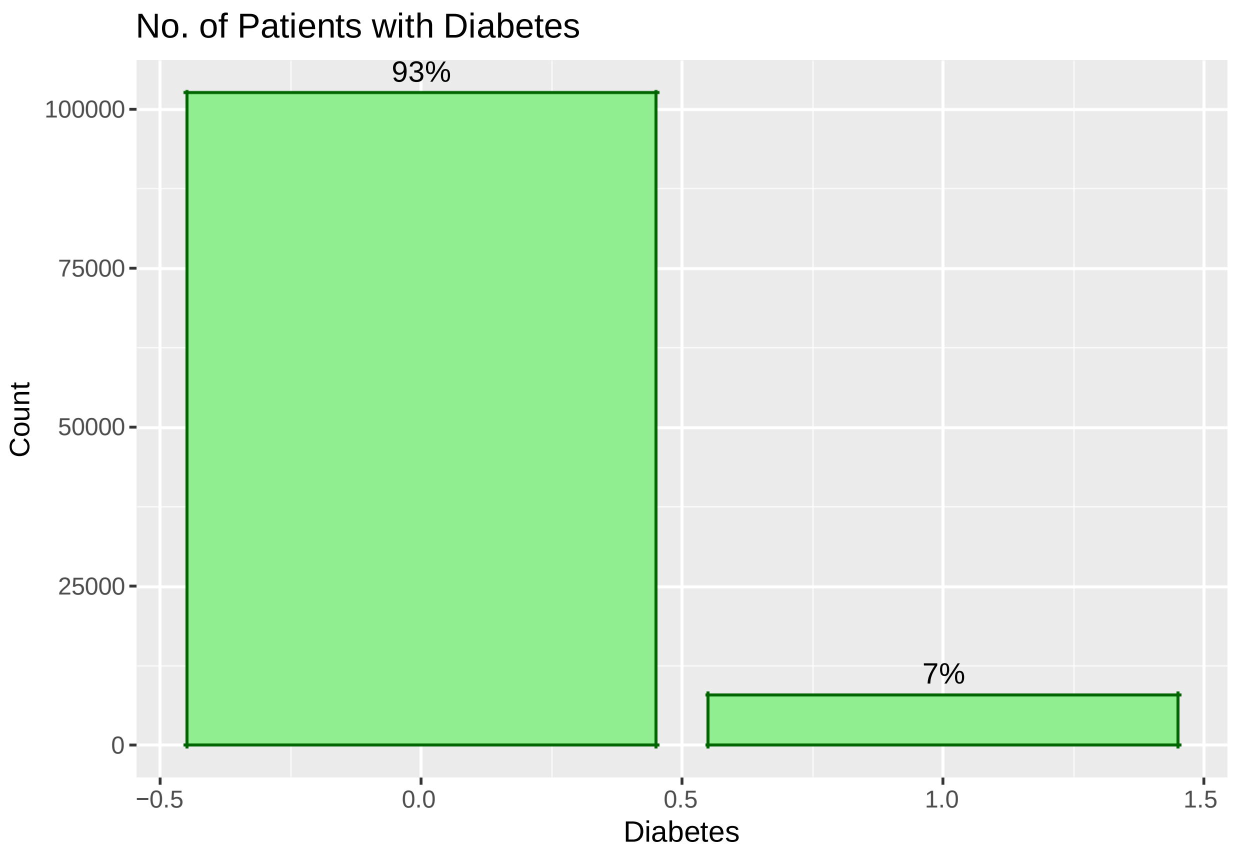


```
plotGraphbasedonNoShow("Hipertension Patients with Show/No-Show %","Hipertension")
```

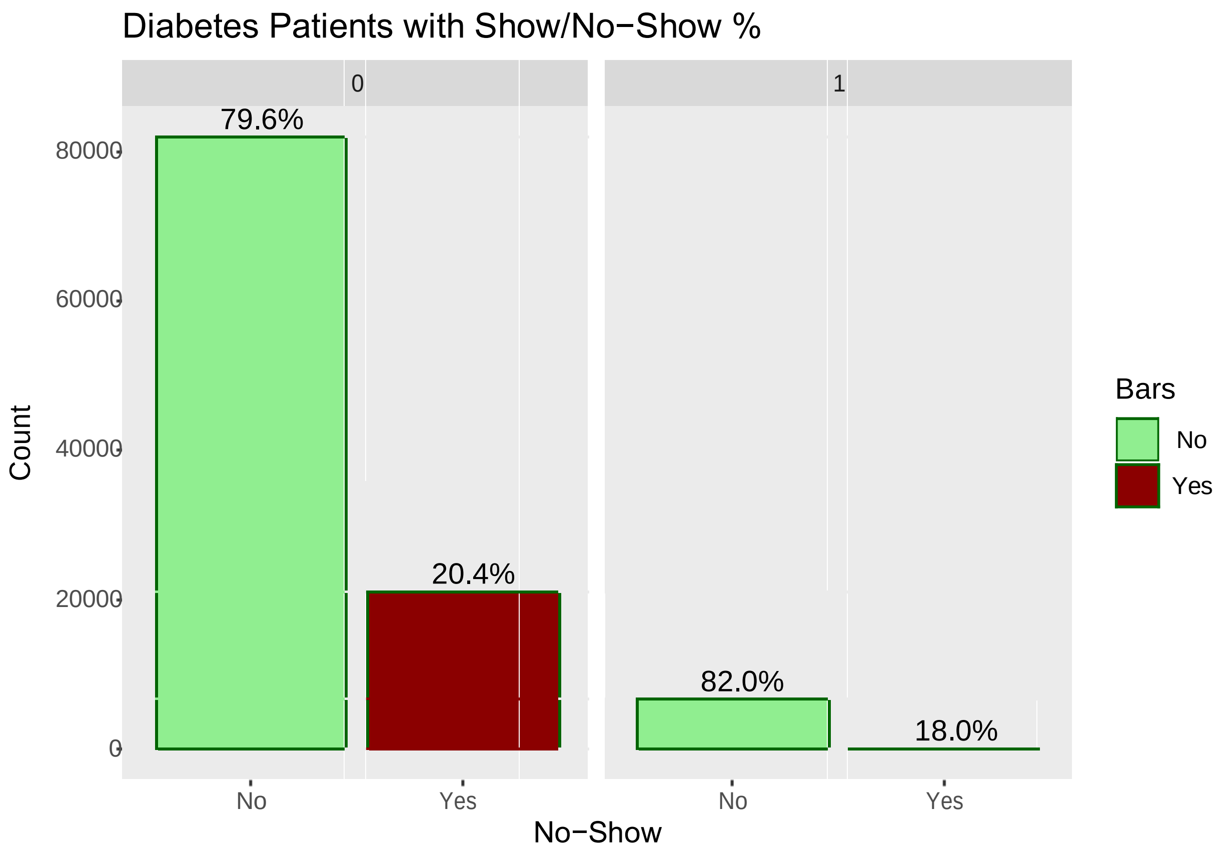


r. 7% of the total Patients observed with Diabetes and out of the that 18% of the Patients didn’t Show-Up as per their Scheduled Appointments.

```
#Plot a graph to depict the no. of patients and No-Show by Diabetes
plotGraphbasedonColumnname("No. of Patients with Diabetes","Diabetes")
```

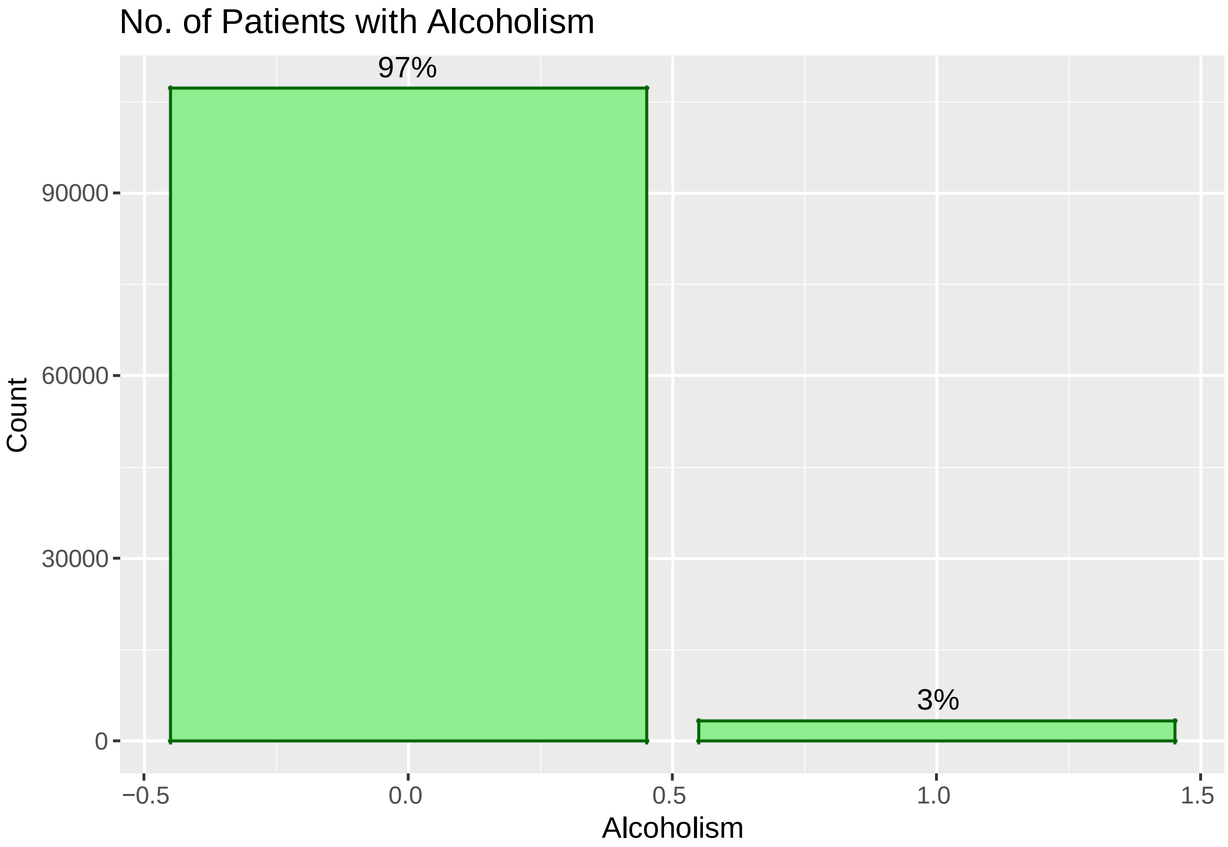


```
plotGraphbasedonNoShow("Diabetes Patients with Show/No-Show %","Diabetes")
```

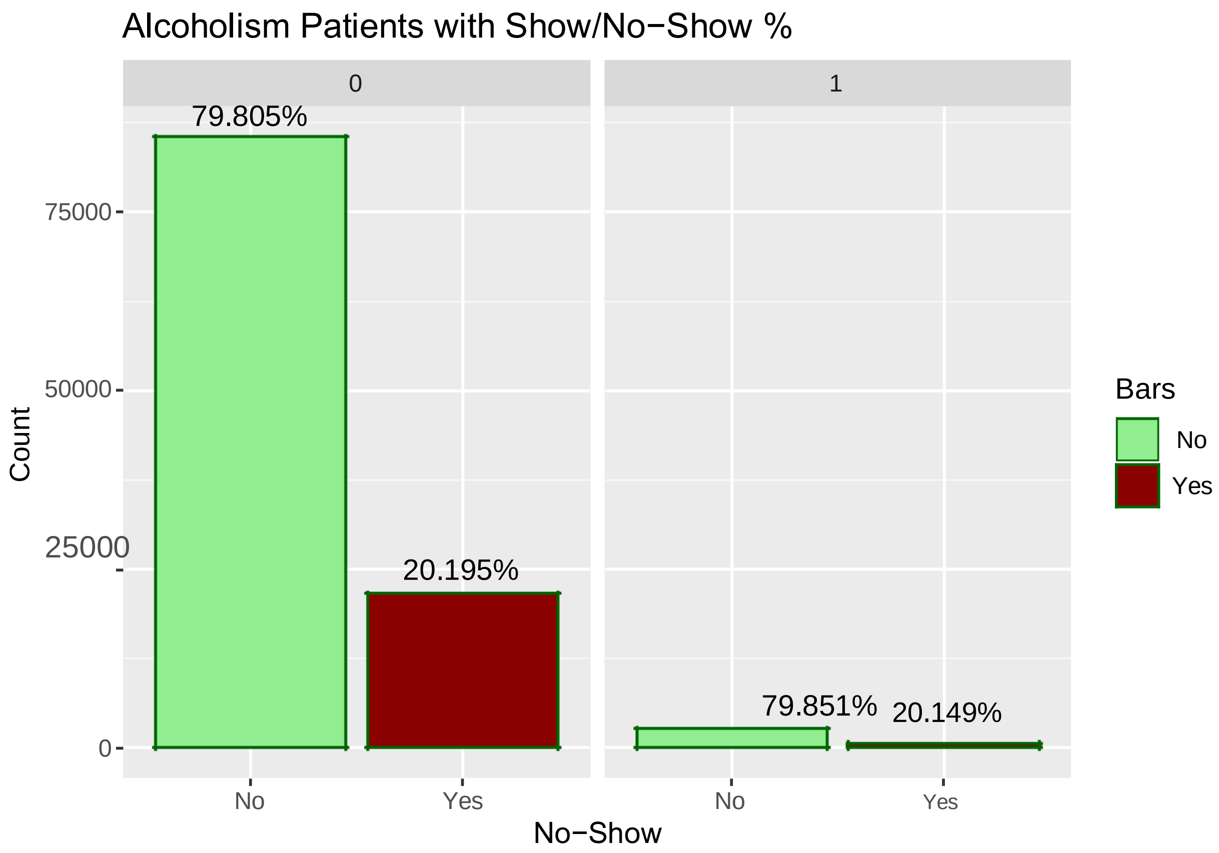


s. 3% of the total Patients observed with Acholism and out of the that 20% of the Patients didn’t Show-Up as per their Scheduled Appointments.

```r
#Plot a graph to depict the no. of patients and No-Show by Alcoholism
plotGraphbasedonColumnname("No. of Patients with Alcoholism","Alcoholism")
```

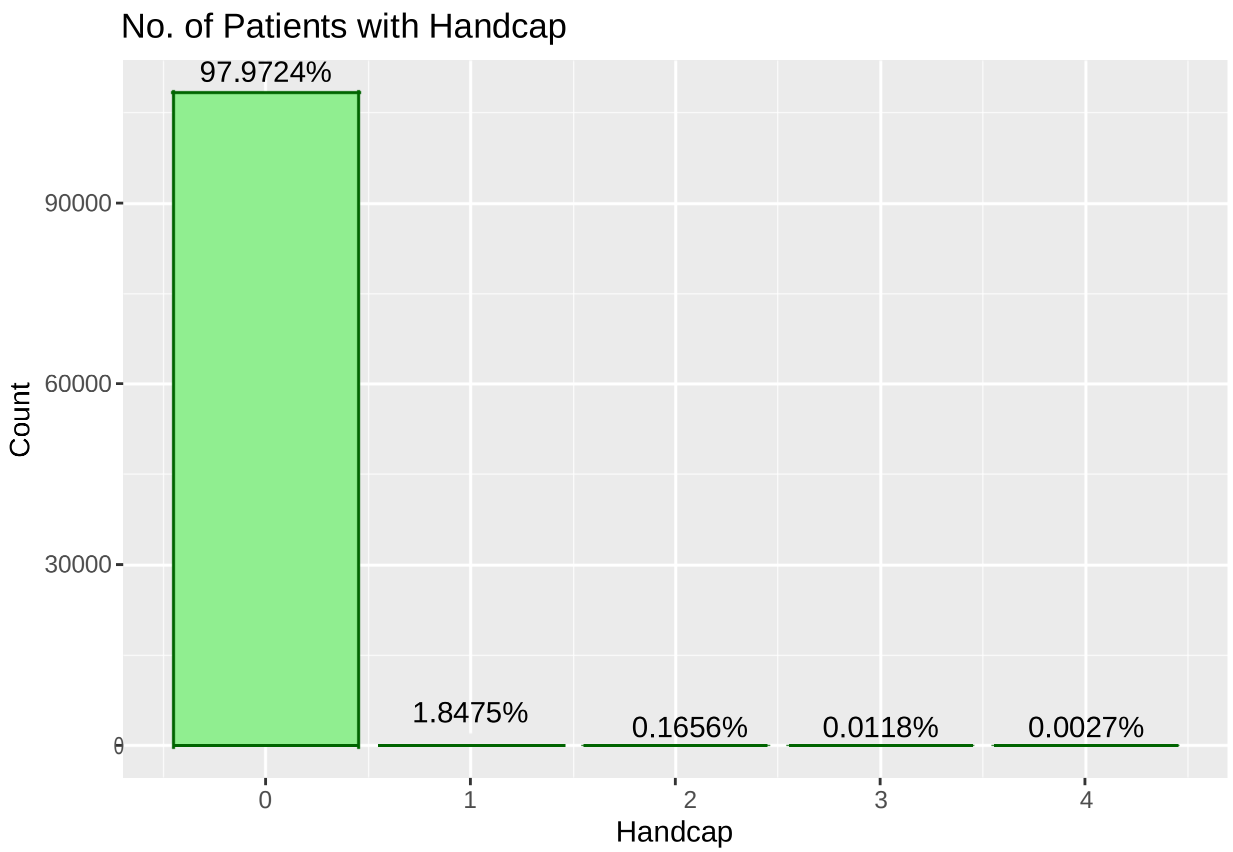


```
plotGraphbasedonNoShow("Alcoholism Patients with Show/No-Show %","Alcoholism")
```

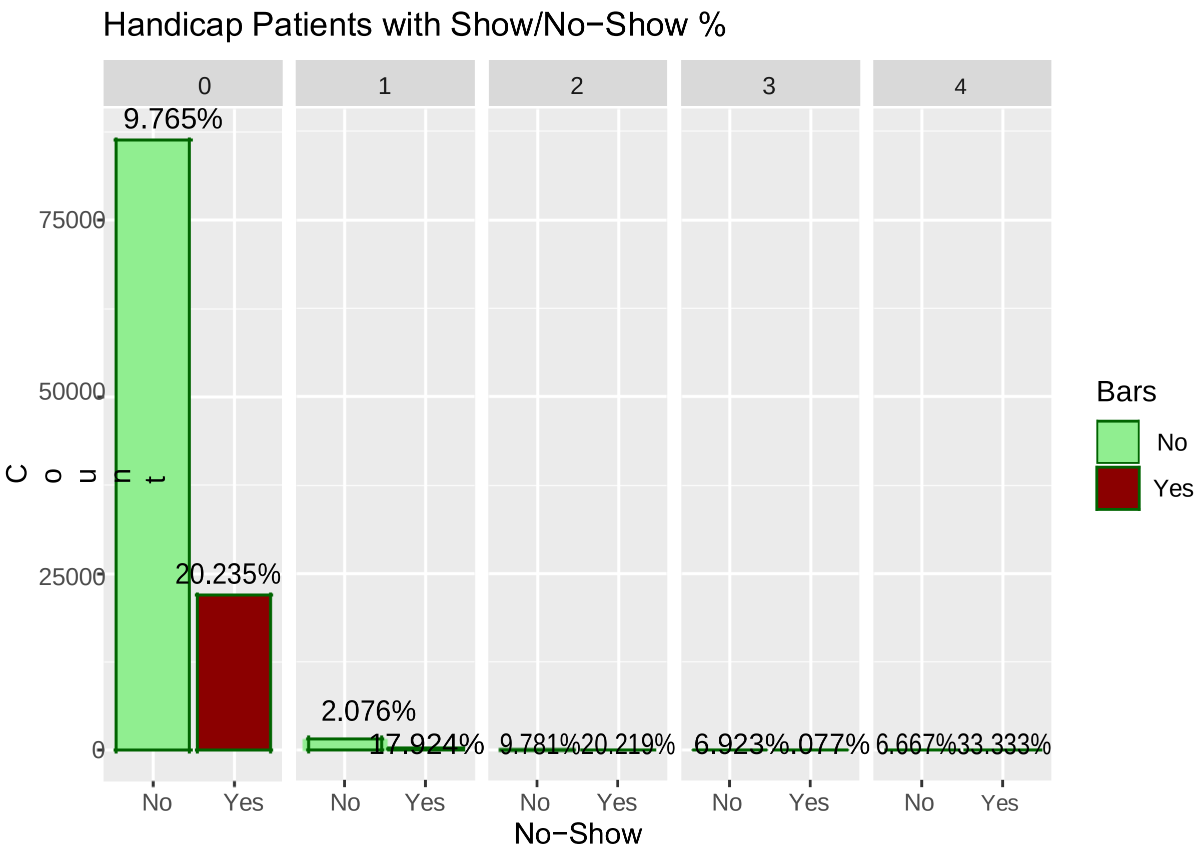


t. 2% of the total Patients observed with Handicap and average 23% of the Patients didn't Show-Up as per their Scheduled Appointments.

```
#Plot a graph to depict the no. of patients and No-Show by Handcap
plotGraphbasedonColumnname("No. of Patients with Handcap", "Handcap")
```



```r
plotGraphbasedonNoShow("Handicap Patients with Show/No-Show %","Handcap")
```



```
#Data Cleanup
# Converting No-Show data in binary format in column NoShow; 1 = Yes
(Didn't Show-Up), #0 = No (Show-Up)
ds_apptnoshow$NoShow <- ifelse(ds_apptnoshow$NoShow =='Yes',
1, 0) ds_apptnoshow$NoShow <- as.factor(ds_apptnoshow$NoShow)

# Removing some columns (non relevant) from dataset to shorten the
# overall execution time to train the data.
ds_apptnoshow <- ds_apptnoshow %>% select(-PatientId,-AppointmentID,
                               -ScheduledDay,-AppointmentDay,
                               -Age, -Neighbourhood,apptdayoftheweek,
                               -AgeBuckets, -apptdayoftheweek, -apptday)
```

Quick view of the Dataset columns post cleanup.

```
names(ds_apptnoshow)
```

```
## [1] "Gender"       "Scholarship" "Hipertension" "Diabetes"    "Alcoholism"
## [6] "Handcap"      "SMS_received" "NoShow"
```

Top 6 rows of the Dataset post cleanup

```
head(ds_apptnoshow)
```

```
## # A tibble: 6 x 8
##   Gender Scholarship Hipertension Diabetes Alcoholism Handcap SMS_received
##    <chr>       <dbl>        <dbl>    <dbl>      <dbl>   <dbl>        <dbl>
  ##1F               0            1        0          0       0            0
  ##2M               0            0        0          0       0            0
  ##3F               0            0        0          0       0            0
## 4  F               0            0        0          0       0            0
## 5  F               0            1        1          0       0            0
## 6  F               0            1        0          0       0            0
## # ... with 1 more variable: NoShow <fct>
```

3. Training Models

Note: Execution of Models takes approx 15-20 mins.

Splitting Appointment NoShow Dataset into Train (90%) / Test (10%) dataset

```
set.seed(1)
test_index <- createDataPartition(ds_apptnoshow$NoShow, times = 1, p = 0.1, list = FALSE)
train <- ds_apptnoshow[-test_index,]
test <- ds_apptnoshow[test_index,]
```

- Train dataset contains **99474** rows and **8** columns.
- Test dataset contains **11053** rows and **8** columns.

Model 1: Latent Dirichlet Allocation (LDA) Model

Linear Discriminant Analysis (LDA) is a dimensionality reduction technique which is commonly used for the supervised classification problems.

```
set.seed(1)
#set.seed(1, sample.kind = "Rounding") # if using R 3.6 or later
#Linear Discriminant Analysis (LDA)
train_lda <- train(NoShow ~ ., data=train, method
= "lda") #to display Accuracy of the trained
dataset train_lda$results
```

```
##   parameter  Accuracy Kappa  AccuracySD KappaSD
## 1      none 0.7985919   0 0.001428771     0
```

```
#predict Accuracy by comparing it with test dataset
lda_preds <- predict(train_lda, test)
test_accuracy <- mean(lda_preds == test$NoShow)
#Add a result in Model Result table
model_results <- tibble(Models="LDA",
                        Trained_Set_Accuracy = max(train_lda$results$Accuracy),
                        Test_Set_Accuracy = test_accuracy)
```

Model 2: Quadratic Discriminant Analysis (QDA) Model

A quadratic classifier is used in machine learning and statistical classification to separate measure-ments of two or more classes of objects or events by a quadric surface.

```
set.seed(1)
#set.seed(1, sample.kind = "Rounding")  # if using R 3.6 or later
#Quadratic Discriminant Analysis (QDA)
train_qda <- train(NoShow ~ ., data=train, method = "qda")
#to display Accuracy of the trained  dataset
train_qda$results
```

```
##   parameter  Accuracy      Kappa AccuracySD     KappaSD
## 1      none 0.7893205 0.03722797 0.001892571 0.003574995
```

```
#predict Accuracy by comparing it with test dataset
qda_preds <- predict(train_qda, test)
test_accuracy <-mean(qda_preds == test$NoShow)
#Add a result in Model Result table
model_results <- bind_rows(model_results,
                           tibble(Models="QDA",
                          Trained_Set_Accuracy = max(train_qda$results$Accuracy),
                          Test_Set_Accuracy = test_accuracy))
```

Model 3: Linear Regression Model (GLM) Model

Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables, they are considering and the number of independent variables being used.

```
set.seed(1)
#set.seed(1, sample.kind = "Rounding")      # if using R 3.6 or later
#Linear Regression Model
train_glm <- train(NoShow ~ ., data=train, method
= "glm") #to display Accuracy of the trained
dataset train_glm$results
```

```
##   parameter  Accuracy Kappa  AccuracySD KappaSD
## 1      none 0.7985919     0 0.001428771       0
```

```
#predict Accuracy by comparing it with test dataset
glm_preds <- predict(train_glm, test)
test_accuracy <- mean(glm_preds == test$NoShow)
#Add a result in Model Result table
model_results <- bind_rows(model_results,
                                tibble(Models="GLM",
                              Trained_Set_Accuracy = max(train_glm$results$Accuracy),
                              Test_Set_Accuracy = test_accuracy))
```

Model 4: Gradient Boosting Machine Model (GBM) Model

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees.

```
set.seed(1)
#set.seed(1, sample.kind = "Rounding")# if using R 3.6 or later
# gradient boosting machine
train_gbm <- train(NoShow~., data=train, method="gbm",
verbose=FALSE) #to display Accuracy of the trained dataset
train_gbm$results
```

```
##   shrinkage interaction.depth n.minobsinnode n.trees Accuracy        Kappa
## 1       0.1                 1             10      50 0.7985919 0.0000000000
## 4       0.1                 2             10      50 0.7985919 0.0000000000
## 7       0.1                 3             10      50 0.7986017 0.0008750556
## 2       0.1                 1             10     100 0.7985919 0.0000000000
## 5       0.1                 2             10     100 0.7985951 0.0002789547
## 8       0.1                 3             10     100 0.7986355 0.0020063561
## 3       0.1                 1             10     150 0.7985919 0.0000000000
## 6       0.1                 2             10     150 0.7986104 0.0007288995
## 9       0.1                 3             10     150 0.7986508 0.0025548720
##    AccuracySD      KappaSD
## 1 0.001428771 0.0000000000
## 4 0.001428771 0.0000000000
## 7 0.001446302 0.0012403323
## 2 0.001428771 0.0000000000
## 5 0.001438391 0.0008088975
## 8 0.001416280 0.0011429484
## 3 0.001428771 0.0000000000
## 6 0.001434812 0.0012539552
## 9 0.001415043 0.0005384473
```

```
#predict Accuracy by comparing it with test dataset
gbm_preds <- predict(train_gbm, test)
test_accuracy <- mean(gbm_preds == test$NoShow)
#Add a result in Model Result table
model_results <- bind_rows(model_results,
                           tibble(Models="GBM",
                           Trained_Set_Accuracy = max(train_gbm$results$Accuracy),
                           Test_Set_Accuracy = test_accuracy))
```

Model 5: TREE (RPART) Model

Decision tree learning is one of the predictive modeling approaches used in statistics, data mining and machine learning. It uses a decision tree (as a predictive model) to go from observations about an item to conclusions about the item’s target value.

```
# Convert all objects of train dataset into factor to train the data with
tree model train$Gender <- as.factor(train$Gender)
train$Scholarship <- as.factor(train$Scholarship)
train$Hipertension <- as.factor(train$Hipertension)
train$Diabetes <- as.factor(train$Diabetes)
train$Alcoholism <- as.factor(train$Alcoholism)
train$Handcap <- as.factor(train$Handcap)
train$SMS_received <- as.factor(train$SMS_received)
train$NoShow <- as.factor(train$NoShow)

# Convert all objects of test dataset into factor for predicting a tree value
test$Gender <- as.factor(test$Gender)
test$Scholarship <- as.factor(test$Scholarship)
test$Hipertension <- as.factor(test$Hipertension)
test$Diabetes <- as.factor(test$Diabetes)
test$Alcoholism <- as.factor(test$Alcoholism)
test$Handcap <- as.factor(test$Handcap)
test$SMS_received <- as.factor(test$SMS_received)
test$NoShow <- as.factor(test$NoShow)

set.seed(1)
#set.seed(1, sample.kind = "Rounding")# if using R 3.6 or later
# Tree Based Model
train_rpart <- train(NoShow ~ ., data=train, method = "rpart")
#to display Accuracy of the trained dataset
train_rpart$results
```

```
##             cp  Accuracy       Kappa  AccuracySD      KappaSD
## 1 0.000000e+00 0.7985798 0.002537126 0.001347480 0.0009813338
## 2 3.111465e-05 0.7986224 0.002352783 0.001414693 0.0010219446
## 3 6.222930e-05 0.7986082 0.001659357 0.001414702 0.0013308642
```

```
#predict Accuracy by comparing it with test dataset
rpart_preds <- predict(train_rpart, test)
test_accuracy <- mean(rpart_preds == test$NoShow)
#Add a result in Model Result table
model_results <- bind_rows(model_results,
                           tibble(Models="RPART",
                          Trained_Set_Accuracy = max(train_rpart$results$Accuracy),
                          Test_Set_Accuracy = test_accuracy))
```

4. Result

Summary of all above models with Accuracy on the Trained and Test dataset.

```
#to display consolidated accuracy of trained and test
data set model_results %>% knitr::kable()
```

Models	Trained_Set_Accuracy	Test_Set_Accuracy
LDA	0.7985919	0.7980639
QDA	0.7893205	0.7901927
GLM	0.7985919	0.7980639
GBM	0.7986508	0.7984258
RPART	0.7986224	0.7983353

Ensemble method

In statistics and machine learning, ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone.

```
#Using Ensemble method to combine the predicted values to decrease variance
ensemble <- cbind(glm = glm_preds == "0", lda = lda_preds == "0",
                  qda = qda_preds == "0", gbm = gbm_preds
                  == "0", rpart = rpart_preds =="0")

#predict Accuracy by comparing it with test dataset
ensemble_preds <- ifelse(rowMeans(ensemble) > 0.5, "0", "1")
mean(ensemble_preds == test$NoShow)
```

```
## [1] 0.7984258
```

5. Conclusion

Average accuracy acheived is ~80% from all models but is not an excellent result.

Doctors’ office can decrease the No-Show rate by sending the SMS reminders to all of their patient and it can help in an improving the predictability of the patients.
Data seems to be limited and doesn’t have enough information to improve the results.

Data can be improved by adding more weights/factors that may be able to help in improving the overall accuracy for No-Show.