Final Project

Linh Tran

5/7/2021

Import data

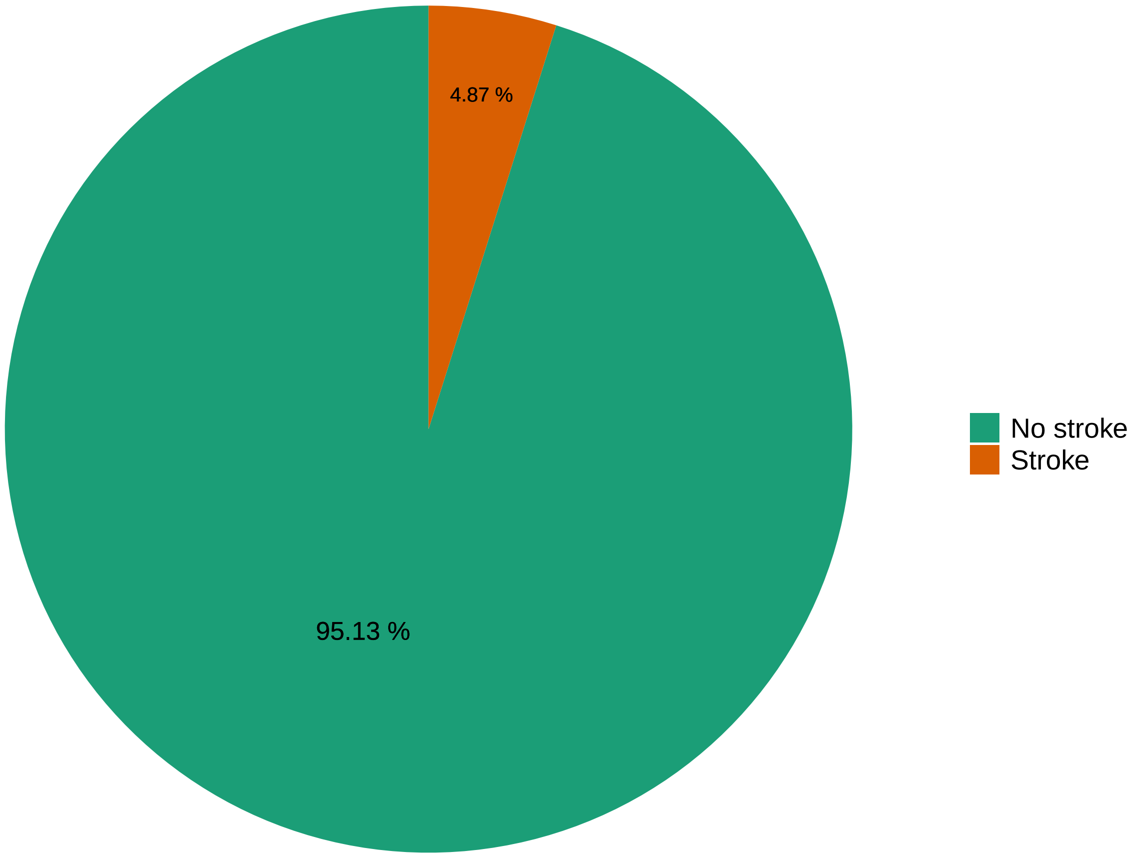
```
stroke_df = read.csv("./data/healthcare-dataset-stroke-data.csv")
# head(stroke_df)

head(stroke_df)
```

```
##      id gender age hypertension heart_disease ever_married     work_type
## 1  9046   Male  67            0             1          Yes       Private
## 2 51676 Female  61            0             0          Yes Self-employed
## 3 31112   Male  80            0             1          Yes       Private
## 4 60182 Female  49            0             0          Yes       Private
## 5  1665 Female  79            1             0          Yes Self-employed
## 6 56669   Male  81            0             0          Yes       Private
##   Residence_type avg_glucose_level  bmi  smoking_status stroke
## 1          Urban            228.69 36.6 formerly smoked      1
## 2          Rural            202.21  N/A    never smoked      1
## 3          Rural            105.92 32.5    never smoked      1
## 4          Urban            171.23 34.4          smokes      1
## 5          Rural            174.12   24    never smoked      1
## 6          Urban            186.21   29 formerly smoked      1
```

EDA and Visualization

Distribution of stroke:

```
dataplot10 = stroke_df %>% dplyr::count(stroke)
dataplot1 = dataplot10 %>% mutate(ntotal=sum(dataplot10$n), perc= n/ntotal)
plot1= ggplot(dataplot1, aes(x="", y=perc*100, fill=as.factor(stroke), group=as.factor(stroke)))+theme_
  geom_bar(width = 1, stat = "identity") + theme_void() +
  labs(x=" ",y=" ", fill=" ") +
  scale_fill_brewer(palette = "Dark2",labels = c("No stroke", "Stroke"))+
  geom_text( y=55, label="95.13 %", size=5)+geom_text(aes(label="4.87 %"),y=2.5, x=1.3, size=4)+
  coord_polar("y", start=0) + theme(legend.text=element_text(size=15))

plot1
```



We could see that only 4.87% of the 5110 individuals in the dataset suffered a stroke.

```r
#genders

dataplot2=stroke_df %>% dplyr::count(stroke, gender) %>% spread(stroke, n)
names(dataplot2)=c("gender", "neg", "pos")

dataplot2 = dataplot2 %>% mutate(perc_gender=pos/(pos+neg))

plot2 = ggplot(dataplot2 %>% filter(gender!="Other"), aes(x=gender,
                        y=perc_gender*100, fill=as.factor(gender),
                        group=as.factor(gender))) + theme_bw()+
  geom_bar(stat = "identity")+
  labs(title="Gender",x="",y="Probability of stroke (%)") + scale_fill_brewer(palette = "Dark2") +
  theme(legend.position = "none")+  theme(text = element_text(size=13.07,colour="black"))+
  theme(axis.text.x = element_text(colour="black",size=13.07))+
  theme(axis.text.y = element_text(colour="black",size=13.07))
```

Smoking status:

```r
dataplot3_1=stroke_df %>% dplyr::count(stroke, smoking_status) %>% spread(stroke, n)
names(dataplot3_1)=c("smoking_status", "neg", "pos")

dataplot3_1 = dataplot3_1 %>% mutate(perc_smoke=pos/(pos+neg))

plot3 = ggplot(dataplot3_1, aes(x=smoking_status,
                        y=perc_smoke*100, fill=as.factor(smoking_status),
                        group=as.factor(smoking_status))) + theme_bw()+
  geom_bar(stat = "identity")+
  labs(title="Smoking status",x=" ",y="Probability of stroke (%)") + scale_fill_brewer(palette = "Dark2
  scale_x_discrete(labels=c("formerly smoked" = "Formerly smoked", "never smoked" = "Never smoked", "smo
  theme(legend.position = "none")+  theme(text = element_text(size=13.07,colour="black"))+
  theme(axis.text.x = element_text(colour="black",size=13.07))+
  theme(axis.text.y = element_text(colour="black",size=13.07))
```

People who identified as former smokers have the highest probability of having a stroke (~8%), followed by smokers and then people who never smoked.

```
# hypertension

dataplot3_1a=stroke_df %>% dplyr::count(stroke, hypertension) %>% spread(stroke, n)
names(dataplot3_1a)=c("hypertension", "neg", "pos")

dataplot3_1a = dataplot3_1a %>% mutate(perc_hyp=pos/(pos+neg))

plot4 =ggplot(dataplot3_1a, aes(x=as.factor(hypertension) ,
                        y=perc_hyp*100, fill=as.factor(hypertension ),
                        group=as.factor(hypertension ))) + theme_bw()+
  geom_bar(stat = "identity")+
  labs(title="Hypertension", x=" ",y="Probability of stroke (%)", fill=" ") +
  scale_fill_brewer(palette = "Dark2") +
  scale_x_discrete(breaks=c("0","1"), labels=c("0" = "No hypertension", "1" = "Hypertension")) + theme(l
  theme(text = element_text(size=13.07,colour="black"))+
  theme(axis.text.x = element_text(colour="black",size=13.07))+
  theme(axis.text.y = element_text(colour="black",size=13.07))
```

```
#heart disease
dataplot3_1b=stroke_df %>% dplyr::count(stroke, heart_disease) %>% spread(stroke, n)
names(dataplot3_1b)=c("heart_disease", "neg", "pos")

dataplot3_1b = dataplot3_1b %>% mutate(perc_hd=pos/(pos+neg))

plot5 = ggplot(dataplot3_1b, aes(x=as.factor(heart_disease) ,
                         y=perc_hd*100, fill=as.factor(heart_disease ),
                         group=as.factor(heart_disease ))) + theme_bw()+
  geom_bar(stat = "identity")+
  labs(title="Heart disease",x="", y="Probability of stroke (%)", fill="Heart disease") +
  scale_fill_brewer(palette = "Dark2") +
  scale_x_discrete(breaks=c("0","1"), labels=c("0" = "No HD", "1" = "HD")) + theme(legend.position = "no
  theme(text = element_text(size=13.07,colour="black"))+
  theme(axis.text.x = element_text(colour="black",size=13.07))+
  theme(axis.text.y = element_text(colour="black",size=13.07))
```

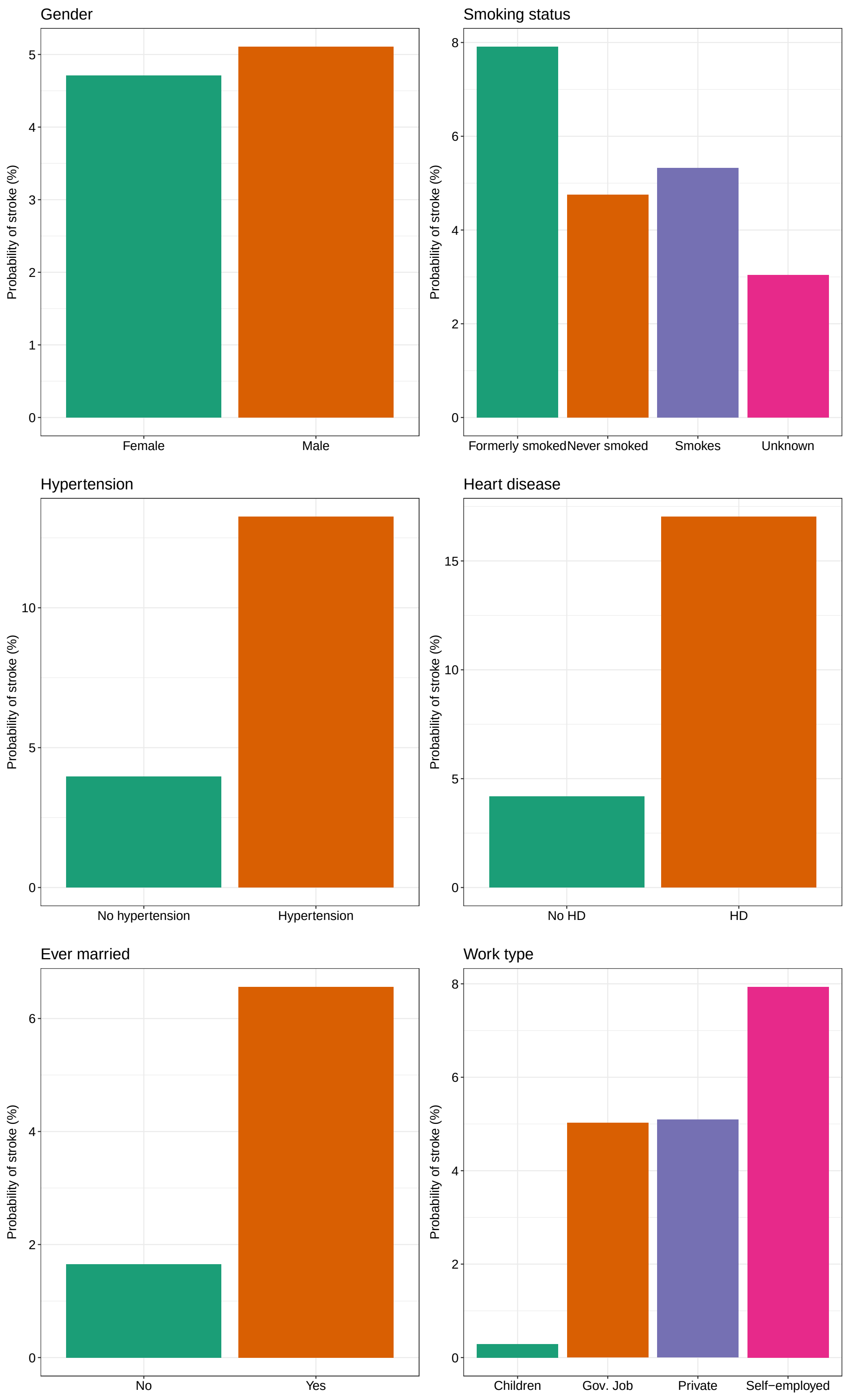
```
#ever_married
dataplot3_1c=stroke_df %>% dplyr::count(stroke, ever_married) %>% spread(stroke, n)
names(dataplot3_1c)=c("ever_married", "neg", "pos")

dataplot3_1c = dataplot3_1c %>% mutate(perc_em=pos/(pos+neg))

plot6 = ggplot(dataplot3_1c, aes(x=ever_married ,
                         y=perc_em*100, fill=as.factor(ever_married ),
                         group=as.factor(ever_married ))) + theme_bw()+
  geom_bar(stat = "identity")+
  labs(title="Ever married",x="", y="Probability of stroke (%)", fill=" ") +
  scale_fill_brewer(palette = "Dark2") +
  theme(legend.position = "none")+
  theme(text = element_text(size=13.07,colour="black"))+
  theme(axis.text.x = element_text(colour="black",size=13.07))+
  theme(axis.text.y = element_text(colour="black",size=13.07))
```

```
# work type

dataplot3_1d= stroke_df %>% dplyr::count(stroke, work_type) %>% spread(stroke, n)
names(dataplot3_1d)=c("work_type", "neg", "pos")

dataplot3_1d = dataplot3_1d %>% mutate(perc_wt=pos/(pos+neg))

plot7=ggplot(dataplot3_1d %>% filter(work_type!="Never_worked"), aes(x=work_type ,
                         y=perc_wt*100, fill=as.factor(work_type ),
                         group=as.factor(work_type ))) + theme_bw()+
  geom_bar(stat = "identity")+
  labs(title="Work type",x=" ",y="Probability of stroke (%)", fill=" ") +
  scale_fill_brewer(palette = "Dark2") +
  scale_x_discrete(labels=c("children" = "Children", "Govt_job" = "Gov. Job")) +
  theme(legend.position = "none")+
  theme(text = element_text(size=13.07,colour="black"))+
  theme(axis.text.x = element_text(colour="black",size=13.07))+
  theme(axis.text.y = element_text(colour="black",size=13.07))
```

```
#residence type

dataplot3_1e=stroke_df %>% dplyr::count(stroke, Residence_type) %>% spread(stroke, n)
names(dataplot3_1e)=c("Residence_type", "neg", "pos")

dataplot3_1e = dataplot3_1e %>% mutate(perc_rt=pos/(pos+neg))

plot8 = ggplot(dataplot3_1e, aes(x=Residence_type ,
                         y=perc_rt*100, fill=as.factor(Residence_type ),
                         group=as.factor(Residence_type ))) + theme_bw()+
  geom_bar(stat = "identity")+
  labs(title="Residence type",x=" ", y="Probability of stroke (%)", fill=" ") +
  scale_fill_brewer(palette = "Dark2") +
  theme(legend.position = "none")+
  theme(text = element_text(size=13.07,colour="black"))+
  theme(axis.text.x = element_text(colour="black",size=13.07))+
  theme(axis.text.y = element_text(colour="black",size=13.07))
```

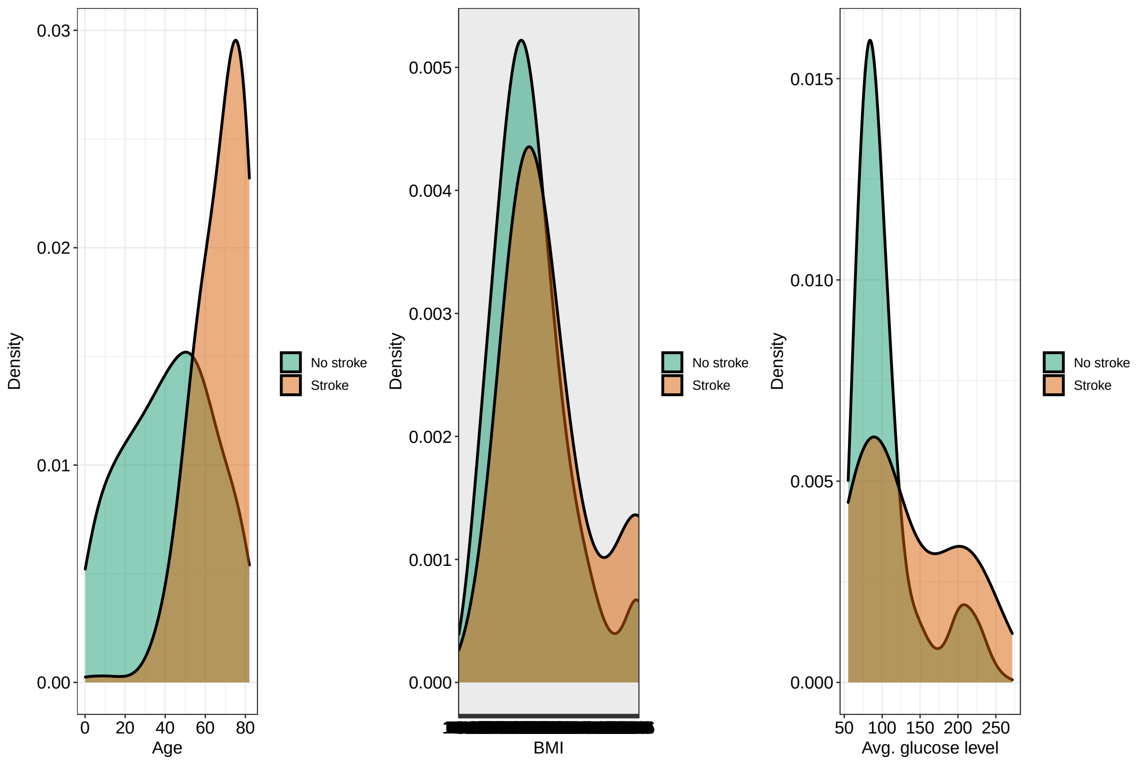
Categorical variables:

```
#figures
allplotslist_1 <- align_plots(plot2, plot3, plot4, plot5, plot6, plot7, plot8, align = "hv")


grid_1=grid.arrange(allplotslist_1[[1]],allplotslist_1[[2]],
                  allplotslist_1[[3]],allplotslist_1[[4]],
                  allplotslist_1[[5]], allplotslist_1[[6]],nrow = 3)
```


Gender
Probability of stroke (%)
0
1
2
3
4
5
Female
Male
Smoking status
Probability of stroke (%)
0
2
4
6
8
Formerly smoked
Never smoked
Smokes
Unknown
Hypertension
Probability of stroke (%)
0
5
10
No hypertension
Hypertension
Heart disease
Probability of stroke (%)
0
5
10
15
No HD
HD
Ever married
Probability of stroke (%)
0
2
4
6
No
Yes
Work type
Probability of stroke (%)
0
2
4
6
8
Children
Gov. Job
Private
Self−employed

Continuous variable:

```
#age
plot9 =
  stroke_df %>%
  ggplot() +
  geom_density(aes(x=age  , group=as.factor(stroke),fill=as.factor(stroke)),
               size=1,alpha=0.5, adjust=2)  +
  theme_bw()+
  ylab("Density")+ labs(fill=' ',x="Age") +
  scale_fill_brewer(palette = "Dark2",labels = c("No stroke", "Stroke"))+
  theme(text = element_text(size=13.07,colour="black"))+
  theme(axis.text.x = element_text(colour="black",size=13.07))+
  theme(axis.text.y = element_text(colour="black",size=13.07))



# bmi
plot10 =
  stroke_df %>%
  ggplot() +
  geom_density(aes(x=bmi, group=as.factor(stroke),fill=as.factor(stroke)),
               size=1,alpha=0.5, adjust=2)  +
  theme_bw()+
  ylab("Density")+ labs(fill=' ',x="BMI") +
  scale_fill_brewer(palette = "Dark2",labels = c("No stroke", "Stroke"))+
  theme(text = element_text(size=13.07,colour="black"))+
  theme(axis.text.x = element_text(colour="black",size=13.07))+
  theme(axis.text.y = element_text(colour="black",size=13.07))



# avg_glucose_level
plot11 =
  stroke_df %>%
  ggplot() +
  geom_density(aes(x=avg_glucose_level  , group=as.factor(stroke),fill=as.factor(stroke)),
               size=1,alpha=0.5, adjust=2)  +
  theme_bw()+
  ylab("Density")+ labs(fill=' ',x="Avg. glucose level") +
  scale_fill_brewer(palette = "Dark2",labels = c("No stroke", "Stroke"))+
  theme(text = element_text(size=13.07,colour="black"))+
  theme(axis.text.x = element_text(colour="black",size=13.07))+
  theme(axis.text.y = element_text(colour="black",size=13.07))

#combine plots

allplotslist_2 <- align_plots(plot9, plot10, plot11, align = "hv")

grid_3=grid.arrange(allplotslist_2[[1]],allplotslist_2[[2]],
                  allplotslist_2[[3]],ncol = 3)
```



Comment: From these plots we can see that:

Formerly smokers are more prone to suffer a stroke than smokers. This could be due to the fact that former smokers quit after acquiring health conditions that raised their risk of having a stroke.

Self-employed are under higher risk of suffering a stroke than private and government jobs. Maybe due to higher stress and lack of insurance that are results of being self-employed?

Urban residents, males and people with hypertension or heart disease are prone to suffer a stroke. In addition, people who have been married are also more likely to suffer a stroke than the single people.

Age seems to be an important factor, with higher age comes higher chance of having a stroke. There are far more people who developed a stroke that have high glucose level than people with low glucose level.

Change categorical variables to binary for model training

```
stroke_df$stroke = as.factor(stroke_df$stroke)
stroke_df$gender = factor(stroke_df$gender) %>% as.numeric()
stroke_df$ever_married = factor(stroke_df$ever_married) %>% as.numeric()
stroke_df$work_type = factor(stroke_df$work_type) %>% as.numeric()
stroke_df$Residence_type = factor(stroke_df$Residence_type) %>% as.numeric()
stroke_df$smoking_status = factor(stroke_df$smoking_status) %>% as.numeric()
stroke_df$heart_disease = factor(stroke_df$heart_disease) %>% as.numeric()
stroke_df$hypertension = as.numeric(factor(stroke_df$hypertension))
stroke_df$work_type = as.factor(stroke_df$work_type) %>% as.numeric()
stroke_df$bmi = as.numeric(stroke_df$bmi)
```

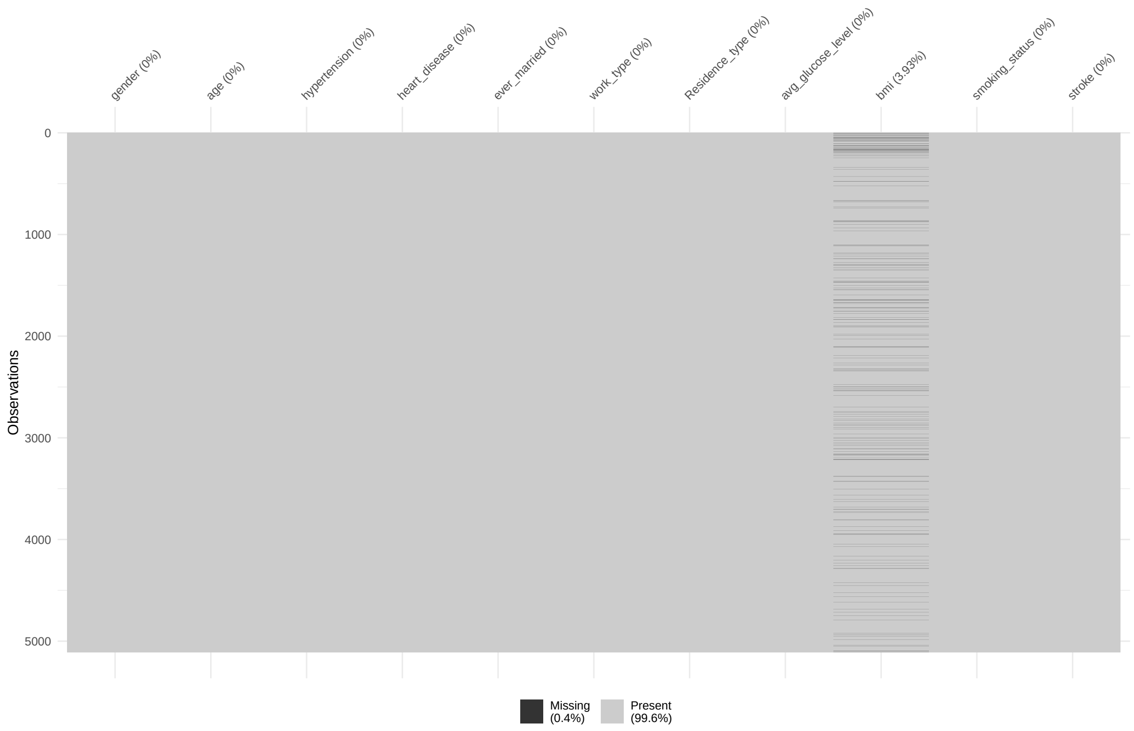
```
## Warning: NAs introduced by coercion
```

```
stroke_df = stroke_df[, -1] %>%
    mutate(stroke = recode(stroke,
                           `0` = "No",
```

```r
                           `1` = "Yes"),
           stroke = factor(stroke)) %>%
    filter(gender < 3)


summary(stroke_df)
```

```
##      gender           age         hypertension   heart_disease
##  Min.   :1.000   Min.   : 0.08   Min.   :1.000   Min.   :1.000
##  1st Qu.:1.000   1st Qu.:25.00   1st Qu.:1.000   1st Qu.:1.000
##  Median :1.000   Median :45.00   Median :1.000   Median :1.000
##  Mean   :1.414   Mean   :43.23   Mean   :1.097   Mean   :1.054
##  3rd Qu.:2.000   3rd Qu.:61.00   3rd Qu.:1.000   3rd Qu.:1.000
##  Max.   :2.000   Max.   :82.00   Max.   :2.000   Max.   :2.000
##
##   ever_married     work_type     Residence_type  avg_glucose_level
##  Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   : 55.12
##  1st Qu.:1.000   1st Qu.:2.000   1st Qu.:1.000   1st Qu.: 77.24
##  Median :2.000   Median :4.000   Median :2.000   Median : 91.88
##  Mean   :1.656   Mean   :3.495   Mean   :1.508   Mean   :106.14
##  3rd Qu.:2.000   3rd Qu.:4.000   3rd Qu.:2.000   3rd Qu.:114.09
##  Max.   :2.000   Max.   :5.000   Max.   :2.000   Max.   :271.74
##
##       bmi        smoking_status  stroke
##  Min.   :10.30   Min.   :1.000   No :4860
##  1st Qu.:23.50   1st Qu.:2.000   Yes: 249
##  Median :28.10   Median :2.000
##  Mean   :28.89   Mean   :2.586
##  3rd Qu.:33.10   3rd Qu.:4.000
##  Max.   :97.60   Max.   :4.000
##  NA's   :201
```

```r
vis_miss(stroke_df)
```



```r
head(stroke_df)
```

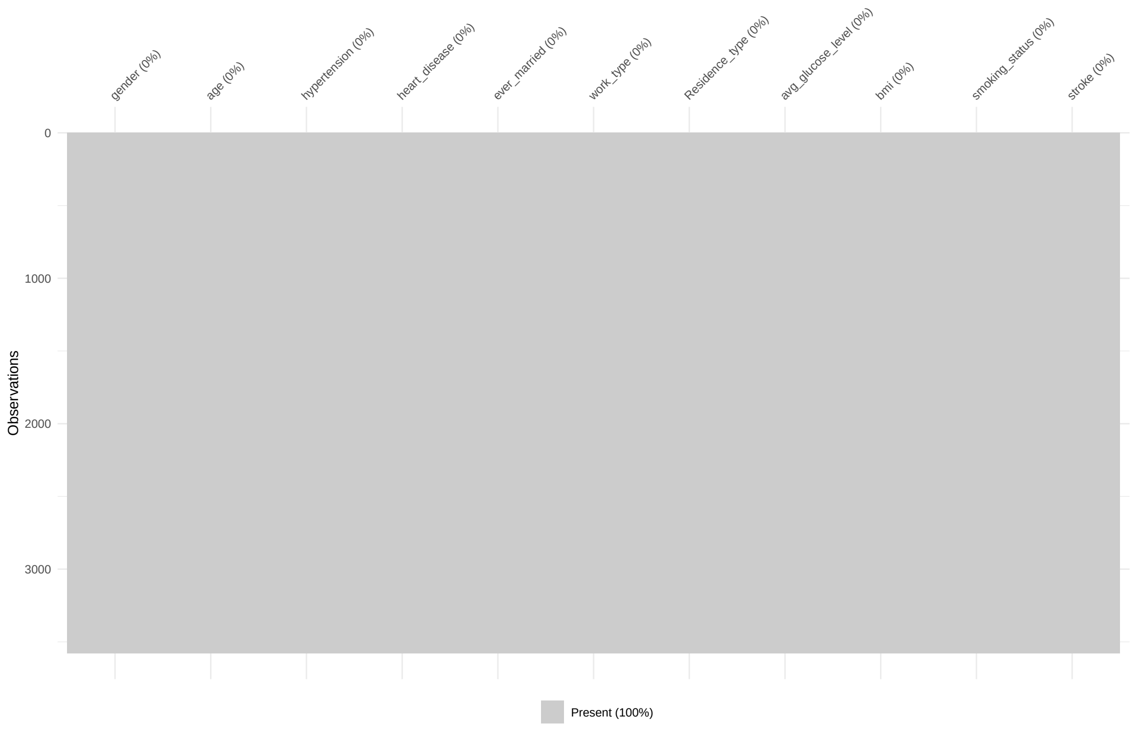
```
##   gender age hypertension heart_disease ever_married work_type Residence_type
## 1      2  67            1             2            2         4              2
## 2      1  61            1             1            2         5              1
## 3      2  80            1             2            2         4              1
## 4      1  49            1             1            2         4              2
## 5      1  79            2             1            2         5              1
## 6      2  81            1             1            2         4              2
##   avg_glucose_level  bmi smoking_status stroke
## 1            228.69 36.6              1    Yes
## 2            202.21   NA              2    Yes
## 3            105.92 32.5              2    Yes
## 4            171.23 34.4              3    Yes
## 5            174.12 24.0              2    Yes
## 6            186.21 29.0              1    Yes
```

Partition the dataset

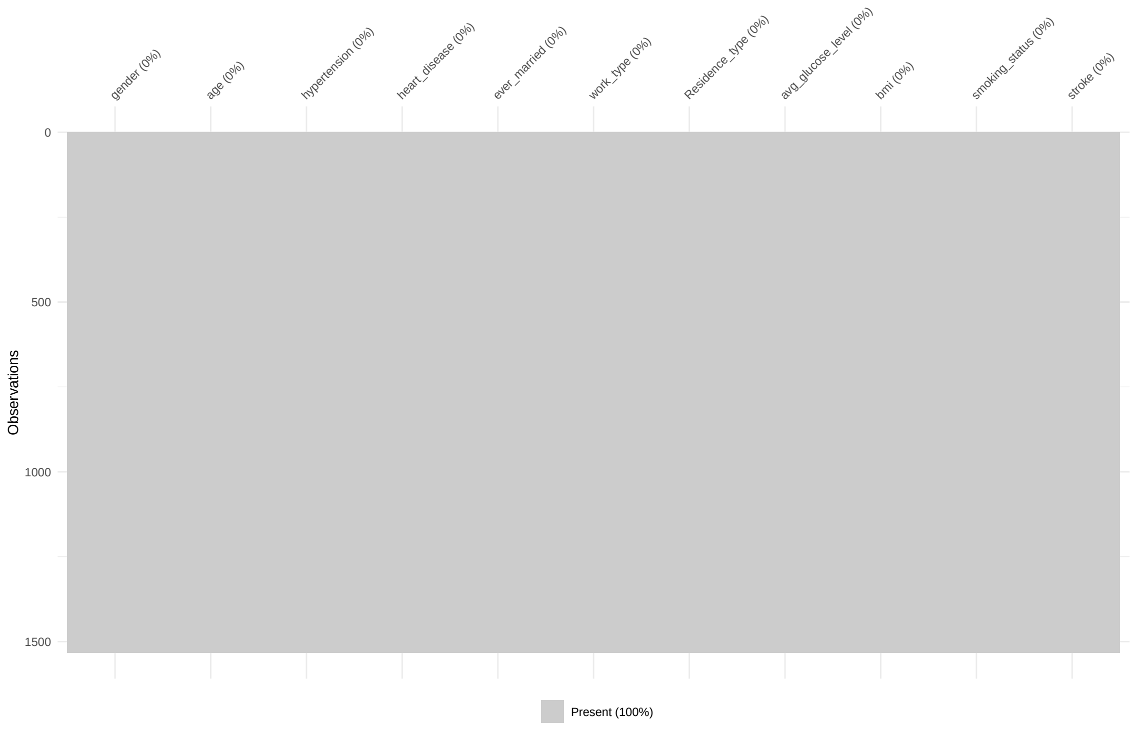
```r
set.seed(123)
trRow = createDataPartition(y = stroke_df$stroke, p = 0.7, list = F)
train.data = stroke_df[trRow, ]
test.data = stroke_df[-trRow, ]
```

Imputation with preProcess()

```r
knnImp = preProcess(train.data, method = "knnImpute", k = 3)
train.data.imp = predict(knnImp, train.data)
vis_miss(train.data.imp)
```



```
test.data.imp = predict(knnImp,test.data)
vis_miss(test.data.imp)
```



Try following models to see which algorithm fits the best because our outcome is binary and it would better to proceed with which classification performs the best. We will have accuracy and ROC/AUC as our evaluation metrics.

In most cases I used grid search with repeated cross-validation (10 folds repeated 3 times) to tune the parameters.

```r
ctrl <- trainControl(
  method = "repeatedcv",
  number = 10,repeats=3,
  summaryFunction = twoClassSummary,
  classProbs = TRUE)
```

Models

Try following models to see which algorithm fits the best because our outcome is binary and it would better to proceed with which classification performs the best. We will have accuracy and ROC/AUC as our evaluation metrics.

In most cases I used grid search with repeated cross-validation (10 folds repeated 3 times) to tune the parameters.

Logistic regression

GLM

```r
# Using caret

set.seed(1)
model.glm <- train(x = train.data.imp[, c(1:10)],
                   y = train.data.imp$stroke,
                   method = "glm",
                   metric = "ROC",
                   trControl = ctrl)

summary(model.glm)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.1082  -0.3228  -0.1715  -0.0864   3.6880
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -3.97326    0.16468 -24.128   <2e-16 ***
## gender             0.02252    0.08224   0.274   0.7843
## age                1.64490    0.14451  11.382   <2e-16 ***
## hypertension       0.08525    0.05926   1.439   0.1502
## heart_disease      0.06894    0.05172   1.333   0.1825
## ever_married      -0.12673    0.12379  -1.024   0.3060
## work_type         -0.10400    0.10999  -0.946   0.3444
## Residence_type     0.01134    0.08236   0.138   0.8905
## avg_glucose_level  0.14815    0.06574   2.253   0.0242 *
## bmi                0.01233    0.10444   0.118   0.9060
## smoking_status     0.00644    0.08590   0.075   0.9402
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1397.4  on 3576  degrees of freedom
## Residual deviance: 1117.6  on 3566  degrees of freedom
## AIC: 1139.6
##
## Number of Fisher Scoring iterations: 7
```

```
glm.pred = predict(model.glm, newdata = test.data.imp, type = "prob")

glm.prob = ifelse(glm.pred$Yes > 0.5,  "Yes", "No")

confusionMatrix(data = as.factor(glm.prob),
                reference = test.data.imp$stroke,
                positive = "Yes")
```

```
## Warning in confusionMatrix.default(data = as.factor(glm.prob), reference =
## test.data.imp$stroke, : Levels are not in the same order for reference and data.
## Refactoring data to match.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  1458   74
##        Yes    0    0
##
##                Accuracy : 0.9517
##                  95% CI : (0.9397, 0.9619)
##     No Information Rate : 0.9517
##     P-Value [Acc > NIR] : 0.5309
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.0000
##             Specificity : 1.0000
##          Pos Pred Value :    NaN
##          Neg Pred Value : 0.9517
##              Prevalence : 0.0483
##          Detection Rate : 0.0000
##    Detection Prevalence : 0.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : Yes
##
```

Penalized logistic regression

```
tune_grid = expand.grid(
  alpha=0:1,
  lambda = seq(0.0001, 1, length = 20)
)
```

```r
model.logistic <- train(
  x = train.data.imp[, c(1:10)],
  y = train.data.imp$stroke,
  method = "glmnet",
  metric = "ROC",
  trControl = ctrl,
  tuneGrid = tune_grid)

log.pred = predict(model.logistic, newdata = test.data.imp, type = "prob")

log.prob = ifelse(log.pred$Yes > 0.5,  "Yes", "No")

confusionMatrix(data = as.factor(log.prob),
                reference = test.data.imp$stroke,
                positive = "Yes")
```

```
## Warning in confusionMatrix.default(data = as.factor(log.prob), reference =
## test.data.imp$stroke, : Levels are not in the same order for reference and data.
## Refactoring data to match.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  1458   74
##        Yes    0    0
##
##                Accuracy : 0.9517
##                  95% CI : (0.9397, 0.9619)
##     No Information Rate : 0.9517
##     P-Value [Acc > NIR] : 0.5309
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.0000
##             Specificity : 1.0000
##          Pos Pred Value :    NaN
##          Neg Pred Value : 0.9517
##              Prevalence : 0.0483
##          Detection Rate : 0.0000
##    Detection Prevalence : 0.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : Yes
##
```

KNN

```
## 161-nearest neighbor model
## Training set outcome distribution:
##
```

```
##   No  Yes
## 3402  175
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  1458   74
##        Yes    0    0
##
##                Accuracy : 0.9517
##                  95% CI : (0.9397, 0.9619)
##     No Information Rate : 0.9517
##     P-Value [Acc > NIR] : 0.5309
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.0000
##             Specificity : 1.0000
##          Pos Pred Value :    NaN
##          Neg Pred Value : 0.9517
##              Prevalence : 0.0483
##          Detection Rate : 0.0000
##    Detection Prevalence : 0.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : Yes
##
```

GAM

```
set.seed(1)

model.gam <- train(x = train.data.imp[,c(1:10)],
                   y = train.data.imp$stroke,
                   method = "gam",
                   metric = "ROC",
                   trControl = ctrl)

summary(model.gam)
```

```
##
## Family: binomial
## Link function: logit
##
## Formula:
## .outcome ~ gender + hypertension + heart_disease + ever_married +
##     Residence_type + smoking_status + work_type + s(age) + s(bmi) +
##     s(avg_glucose_level)
##
## Parametric coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -4.0884994  0.2099125 -19.477   <2e-16 ***
## gender          0.0153919  0.0819196   0.188    0.851
## hypertension    0.0870821  0.0585888   1.486    0.137
## heart_disease   0.0751334  0.0513023   1.465    0.143
## ever_married   -0.1596949  0.1288478  -1.239    0.215
## Residence_type  0.0148814  0.0821419   0.181    0.856
## smoking_status -0.0008626  0.0858376  -0.010    0.992
## work_type      -0.0775830  0.1104279  -0.703    0.482
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                           edf Ref.df  Chi.sq p-value
## s(age)                3.704359      9 120.058  <2e-16 ***
## s(bmi)                0.001716      9   0.000  0.9891
## s(avg_glucose_level) 0.808853      9   4.189  0.0228 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.0905   Deviance explained = 20.6%
## UBRE = -0.68293  Scale est. = 1         n = 3577
```

```
model.gam$finalModel
```

```
##
## Family: binomial
## Link function: logit
##
## Formula:
## .outcome ~ gender + hypertension + heart_disease + ever_married +
##     Residence_type + smoking_status + work_type + s(age) + s(bmi) +
##     s(avg_glucose_level)
##
## Estimated degrees of freedom:
## 3.7044 0.0017 0.8089  total = 12.51
##
## UBRE score: -0.6829292
```

```
gam.pred = predict(model.gam, newdata = test.data.imp, type = "prob")

gam.prob = ifelse(gam.pred$Yes > 0.5,  "Yes", "No")

confusionMatrix(data = as.factor(gam.prob),
                reference = test.data.imp$stroke,
                positive = "Yes")
```

```
## Warning in confusionMatrix.default(data = as.factor(gam.prob), reference =
## test.data.imp$stroke, : Levels are not in the same order for reference and data.
## Refactoring data to match.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  1458   74
##        Yes    0    0
```

```
##
##                Accuracy : 0.9517
##                  95% CI : (0.9397, 0.9619)
##     No Information Rate : 0.9517
##     P-Value [Acc > NIR] : 0.5309
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.0000
##             Specificity : 1.0000
##          Pos Pred Value :    NaN
##          Neg Pred Value : 0.9517
##              Prevalence : 0.0483
##          Detection Rate : 0.0000
##    Detection Prevalence : 0.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : Yes
##
```

Linear Discriminant Analysis (LDA)

```
set.seed(1)
model.lda <- train(x = train.data.imp[,c(1:10)],
                   y = train.data.imp$stroke,
                   method = "lda",
                   metric = "ROC",
                   trControl = ctrl)

lda.pred = predict(model.lda, newdata = test.data.imp, type = "prob")

lda.prob = ifelse(lda.pred$Yes > 0.5,  "Yes", "No")


roc.lda <- roc(test.data.imp$stroke, lda.pred[,2])
```

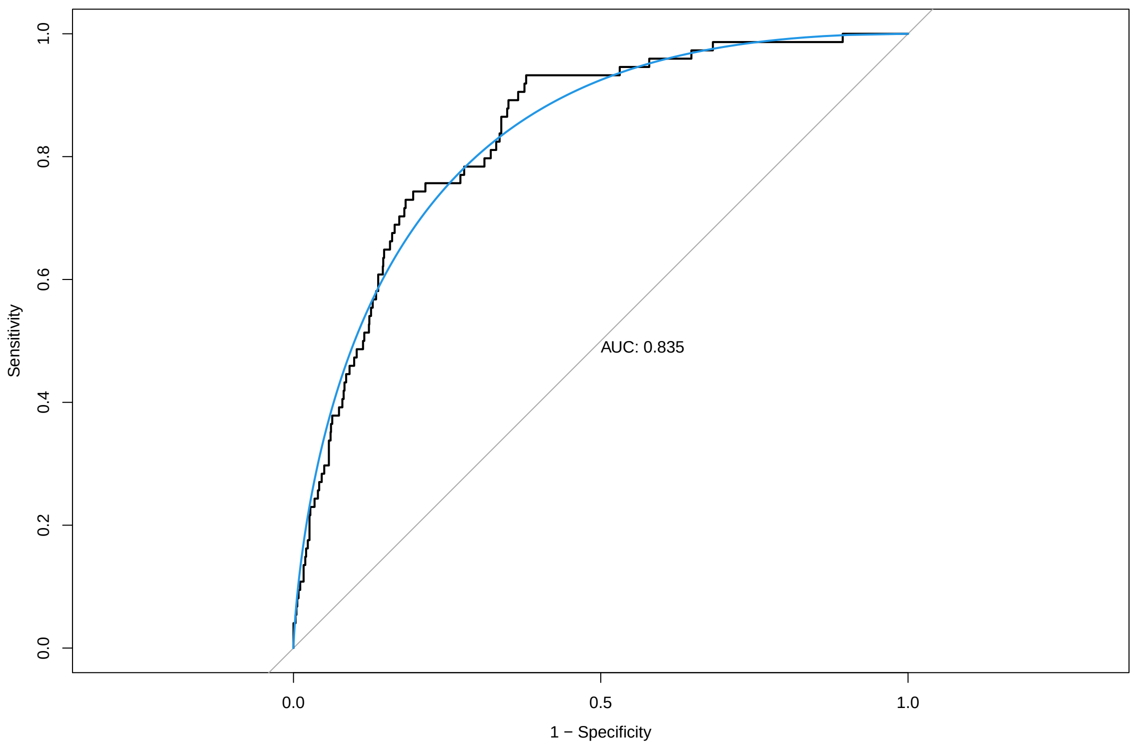
```
## Setting levels: control = No, case = Yes
```

```
## Setting direction: controls < cases
```

```
auc.lda = roc.lda$auc[1]
auc.lda
```

```
## [1] 0.8354002
```

```
plot(roc.lda, legacy.axes = TRUE, print.auc = TRUE)
plot(smooth(roc.lda), col = 4, add = TRUE)
```



```r
confusionMatrix(data = as.factor(lda.prob),
                reference = test.data.imp$stroke,
                positive = "Yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  1452   70
##        Yes    6    4
##
##                Accuracy : 0.9504
##                  95% CI : (0.9383, 0.9607)
##     No Information Rate : 0.9517
##     P-Value [Acc > NIR] : 0.6233
##
##                   Kappa : 0.0847
##
##  Mcnemar's Test P-Value : 4.953e-13
##
##             Sensitivity : 0.054054
##             Specificity : 0.995885
##          Pos Pred Value : 0.400000
##          Neg Pred Value : 0.954008
##              Prevalence : 0.048303
##          Detection Rate : 0.002611
##    Detection Prevalence : 0.006527
##       Balanced Accuracy : 0.524969
##
##        'Positive' Class : Yes
```

```
##
```

Classification trees

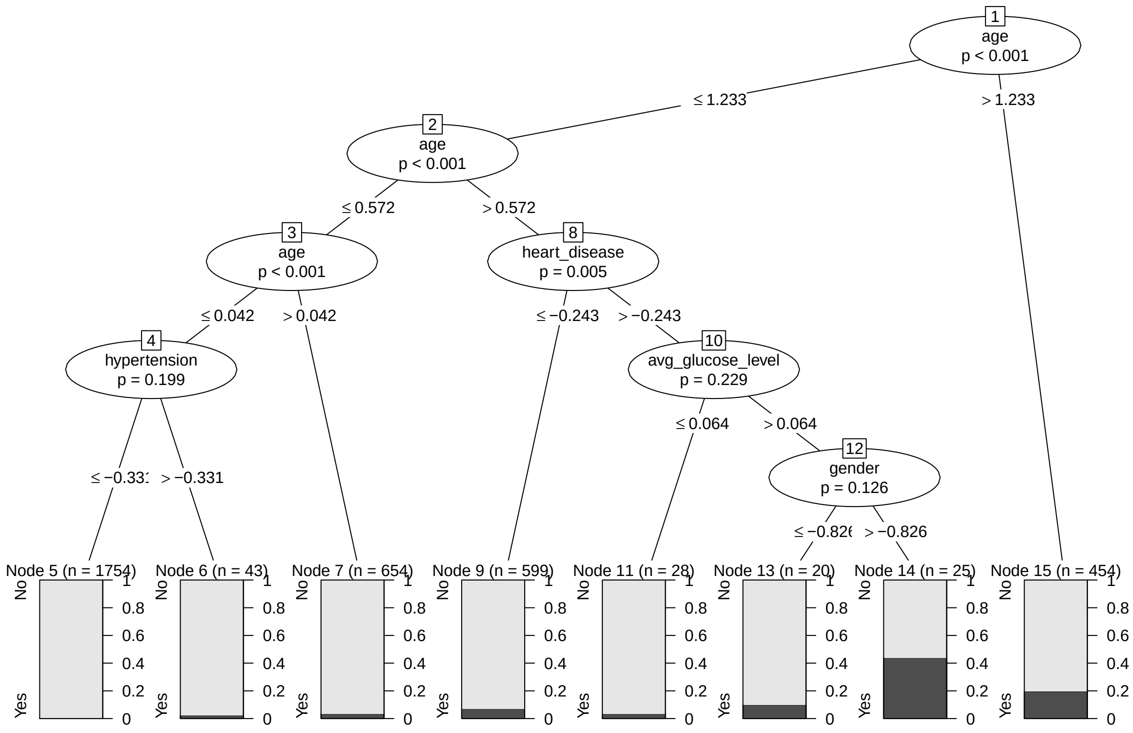
Logistic regression assumes that the data is linearly separable in space but decision trees do not. Decision trees also handle skewed data better.

Conditional Inference tree

```
set.seed(1)

#ctree.fit <- train(stroke ~ . ,  stroke_df,  subset = trRow,method = "ctree",metric = "ROC",trControl =

#plot(ctree.fit$finalModel)

#ctree.pred <- predict(ctree.fit, newdata = stroke_df[-trRow,],type = "prob")[,1]

#roc.ctree <- roc(stroke_df$stroke[-trRow], ctree.pred)

#roc.ctree$auc[1]

#plot(roc.ctree, legacy.axes = TRUE, print.auc = TRUE)



model.ctree <- train(x = train.data.imp[,c(1:10)],
                    y = train.data.imp$stroke,
                    method = "ctree",
                    tuneGrid = data.frame(mincriterion = 1-exp(seq(-2, -1, length = 50))),
                    metric = "ROC",
                    trControl = ctrl)

model.ctree$finalModel #conditional inferece tree with 8 terminal nodes
```

```
##
##   Conditional inference tree with 8 terminal nodes
##
## Response:  .outcome
## Inputs:  gender, age, hypertension, heart_disease, ever_married, work_type, Residence_type, avg_gluc
## Number of observations:  3577
##
## 1) age <= 1.233368; criterion = 1, statistic = 220.125
##   2) age <= 0.5715063; criterion = 1, statistic = 84.341
##     3) age <= 0.04201719; criterion = 1, statistic = 24.818
##       4) hypertension <= -0.3313682; criterion = 0.801, statistic = 5.249
##         5)*  weights = 1754
##       4) hypertension > -0.3313682
##         6)*  weights = 43
##     3) age > 0.04201719
##       7)*  weights = 654
##   2) age > 0.5715063
##     8) heart_disease <= -0.2426811; criterion = 0.995, statistic = 12.052
##       9)*  weights = 599
##     8) heart_disease > -0.2426811
##       10) avg_glucose_level <= 0.0637711; criterion = 0.771, statistic = 4.974
##         11)*  weights = 28
##       10) avg_glucose_level > 0.0637711
##         12) gender <= -0.8260215; criterion = 0.874, statistic = 6.113
##           13)*  weights = 20
##         12) gender > -0.8260215
##           14)*  weights = 25
## 1) age > 1.233368
##   15)*  weights = 454
```

```
plot(model.ctree$finalModel)
```

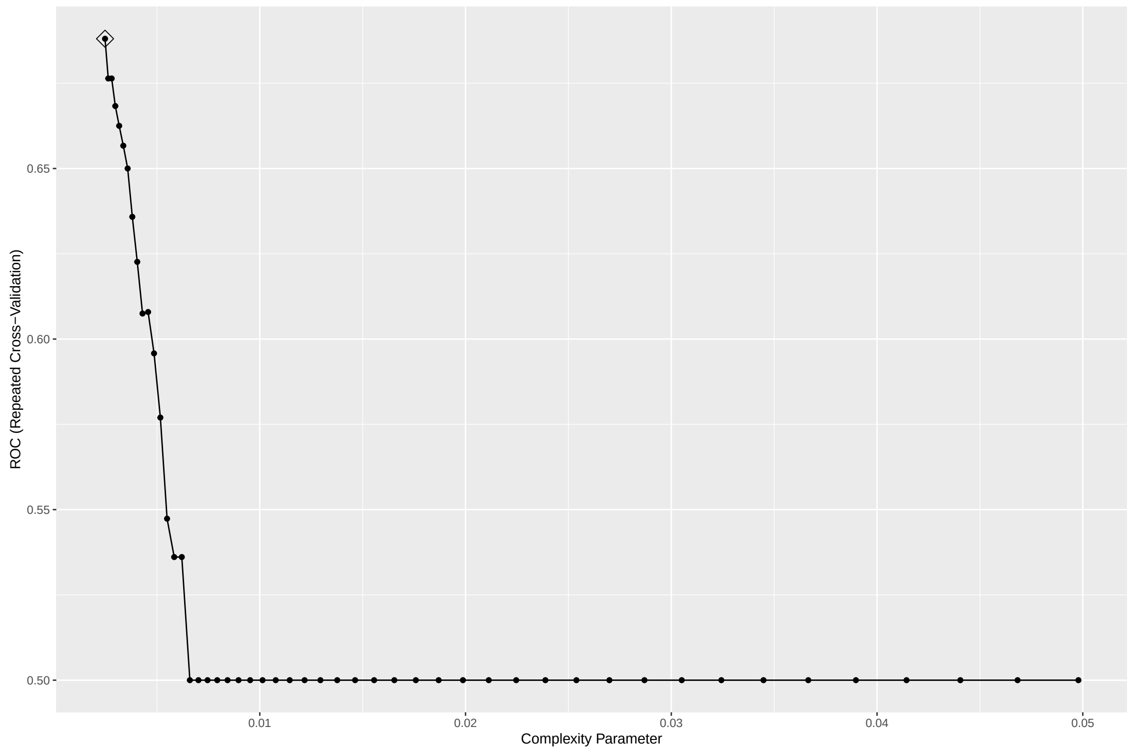


CART

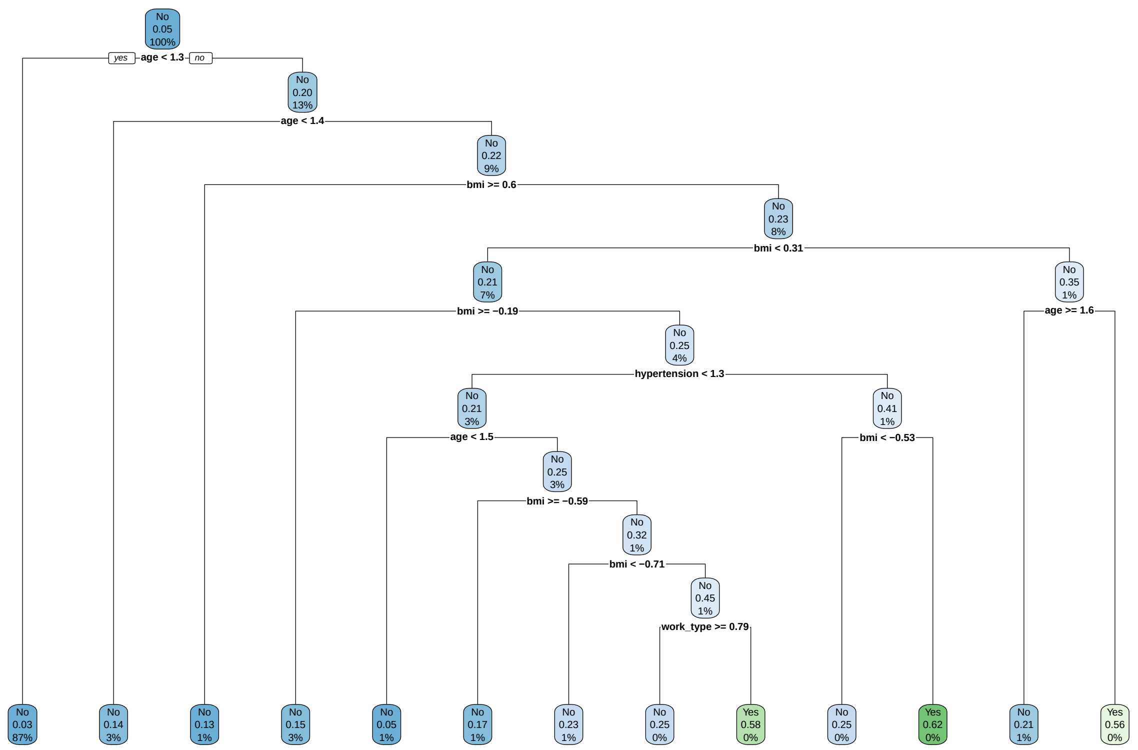
```
set.seed(1)
rpart.fit <- train(x = train.data.imp[,c(1:10)],
                   y = train.data.imp$stroke,
                   method = "rpart",
                   tuneGrid = data.frame(cp = exp(seq(-6,-3, len = 50))),
                   trControl = ctrl,
                   metric = "ROC")

rpart.fit
```

```
## CART
##
## 3577 samples
##   10 predictor
##    2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 3220, 3220, 3219, 3218, 3220, 3219, ...
## Resampling results across tuning parameters:
##
##   cp           ROC        Sens       Spec
##   0.002478752  0.6880472  0.9890271  0.040196078
##   0.002635255  0.6763975  0.9900072  0.034422658
##   0.002801638  0.6763975  0.9900072  0.034422658
##   0.002978527  0.6683119  0.9904974  0.030718954
##   0.003166583  0.6625134  0.9913797  0.028758170
##   0.003366514  0.6566881  0.9917716  0.026906318
##   0.003579067  0.6500128  0.9919677  0.024945534
##   0.003805041  0.6358360  0.9920657  0.023093682
##   0.004045282  0.6226361  0.9928500  0.023093682
##   0.004300691  0.6074861  0.9950055  0.011546841
##   0.004572226  0.6079300  0.9957898  0.009694989
##   0.004860905  0.5958067  0.9962785  0.009694989
##   0.005167811  0.5769863  0.9967679  0.007734205
##   0.005494094  0.5473425  0.9985311  0.003812636
##   0.005840977  0.5360977  0.9993143  0.003812636
##   0.006209762  0.5360977  0.9993143  0.003812636
##   0.006601832  0.5000000  1.0000000  0.000000000
##   0.007018655  0.5000000  1.0000000  0.000000000
##   0.007461796  0.5000000  1.0000000  0.000000000
##   0.007932915  0.5000000  1.0000000  0.000000000
##   0.008433780  0.5000000  1.0000000  0.000000000
##   0.008966268  0.5000000  1.0000000  0.000000000
##   0.009532376  0.5000000  1.0000000  0.000000000
##   0.010134227  0.5000000  1.0000000  0.000000000
##   0.010774078  0.5000000  1.0000000  0.000000000
##   0.011454327  0.5000000  1.0000000  0.000000000
##   0.012177525  0.5000000  1.0000000  0.000000000
##   0.012946384  0.5000000  1.0000000  0.000000000
##   0.013763787  0.5000000  1.0000000  0.000000000
##   0.014632799  0.5000000  1.0000000  0.000000000
##   0.015556678  0.5000000  1.0000000  0.000000000
##   0.016538888  0.5000000  1.0000000  0.000000000
##   0.017583113  0.5000000  1.0000000  0.000000000
##   0.018693268  0.5000000  1.0000000  0.000000000
##   0.019873515  0.5000000  1.0000000  0.000000000
##   0.021128280  0.5000000  1.0000000  0.000000000
##   0.022462268  0.5000000  1.0000000  0.000000000
##   0.023880480  0.5000000  1.0000000  0.000000000
##   0.025388235  0.5000000  1.0000000  0.000000000
##   0.026991186  0.5000000  1.0000000  0.000000000
##   0.028695344  0.5000000  1.0000000  0.000000000
##   0.030507097  0.5000000  1.0000000  0.000000000
##   0.032433241  0.5000000  1.0000000  0.000000000
##   0.034480996  0.5000000  1.0000000  0.000000000
##   0.036658042  0.5000000  1.0000000  0.000000000
##   0.038972541  0.5000000  1.0000000  0.000000000
##   0.041433172  0.5000000  1.0000000  0.000000000
##   0.044049161  0.5000000  1.0000000  0.000000000
##   0.046830317  0.5000000  1.0000000  0.000000000
##   0.049787068  0.5000000  1.0000000  0.000000000
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.002478752.
```

```
ggplot(rpart.fit, highlight = TRUE)
```



```
rpart.plot(rpart.fit$finalModel)
```



```r
summary(resamples(list(rpart.fit, model.ctree)))
```

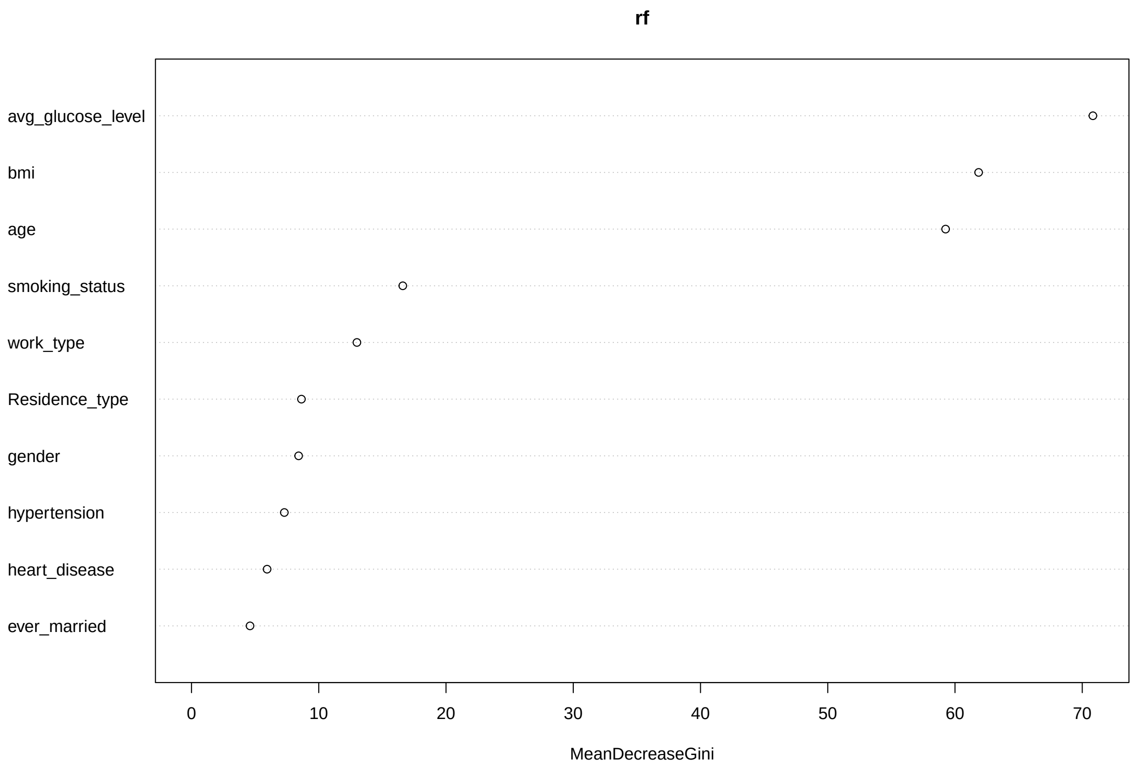
```
##
## Call:
## summary.resamples(object = resamples(list(rpart.fit, model.ctree)))
##
## Models: Model1, Model2
## Number of resamples: 30
##
## ROC
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## Model1 0.5000000 0.6589317 0.6870994 0.6880472 0.7407475 0.8249957    0
## Model2 0.7445502 0.7982399 0.8273150 0.8222961 0.8487048 0.9058824    0
##
## Sens
##             Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## Model1 0.9735294 0.9852941 0.9882353 0.9890271 0.9941176    1    0
## Model2 0.9911765 1.0000000 1.0000000 0.9994126 1.0000000    1    0
##
## Spec
##        Min. 1st Qu.     Median        Mean    3rd Qu.       Max. NA's
## Model1    0       0 0.02777778 0.040196078 0.05882353 0.11764706    0
## Model2    0       0 0.00000000 0.001851852 0.00000000 0.05555556    0
```

```r
#ctree is a better fit
```

Bagging and Random Forests

```r
set.seed(1)
```

```r
bagging <- randomForest(stroke ~ . ,
                        stroke_df[trRow,],
                        mtry = 8,
                        na.action = na.exclude)

set.seed(1)
rf <- randomForest(stroke ~ . ,
                   stroke_df[trRow,],
                   mtry = 3,
                   na.action = na.exclude)



varImpPlot(rf)
```



Random forests using caret

```r
rf.grid <- expand.grid(mtry = 1:8,
                       splitrule = "gini",
                       min.node.size = seq(from = 2, to = 10, by = 2))
set.seed(1)

rf.fit <- train(x = train.data.imp[,c(1:10)],
                  y = train.data.imp$stroke,
                method = "ranger",
                tuneGrid = rf.grid,
                metric = "ROC",
                trControl = ctrl)
```

```r
#rf.pred = predict(rf.fit, newdata = test.data, type = "prob")

#rf.prob = ifelse(rf.pred$Yes > 0.5,  "Yes", "No")



#rf.prob = recode_factor(rf.prob, `0` = "No", `1` = "Yes")

#confusionMatrix(data = as.factor(rf.pred),reference = stroke_df$stroke[-trRow], positive = "Yes")
```

SVM

Since we are considering the two-class classification problem in our context of question, SVM may serve as a good solution.

- What predictor variables did you include?

From the previous model, I decide to pick up features with variable importance greater than 10 : avg_glucose_level, age, bmi, smoking status and working type.

First of all, we will still process the model with all variables included in the model and plot the variable importance to see which variables should we keep in the model.

- What technique did you use? What assumptions, if any, are being made by using this technique?

The only assumptions of support vector machines are independent and identically distributed data. SVM is quite tolerant of input data, especially the soft-margin version.

- If there were tuning parameters, how did you pick their values?

C also known as cost parameter is the tuning parameter in the SVM algorithm. I will used the exponentiated sequence of number with cross-validation to see if there's any best parameters for this model. Plotting out the tuning will also help us have a better vizualization on deciding hyperparameters.

Linear SVM

Radial SVM

```r
trainSmoted <- SMOTE(stroke ~ ., train.data)



trainSmoted$stroke = factor(trainSmoted$stroke, levels = c("No", "Yes"))
train.data$stroke = factor(train.data$stroke, levels = c("No", "Yes"))
test.data$stroke = factor(test.data$stroke, levels = c("No", "Yes"))

set.seed(123)

svm.linear = tune.svm(stroke ~ .,
                      data = trainSmoted,
                      kernel = "linear",
                      cost = exp(seq(-5,2,len=50)),
                      scale = TRUE)



summary(svm.linear)
```

```r
plot(svm.linear)

svm.linear$best.parameters

best.linear = svm.linear$best.model
summary(best.linear)


pred_train_lsvm = predict(best.linear, newdata = train.data)

pred_train_lsvm = recode_factor(pred_train_lsvm, `0` = "No", `1` = "Yes")

confusionMatrix(data = pred_train_lsvm, reference = train.data$stroke , positive = "Yes")

pred_test_lsvm = predict(best.linear, newdata = test.data)
confusionMatrix(data = pred_test_lsvm, reference = test.data$stroke, positive = "Yes")

pred_test_lsvm_numeric = as.numeric(pred_test_lsvm) -1
roc.lsvm = roc(test.data$stroke, pred_test_lsvm_numeric)

auc.lsvm = roc.lsvm$auc[1]
auc.lsvm
plot(roc.lsvm, legacy.axes = TRUE, print.auc = TRUE)
# plot(smooth(roc.lsvm), col = 4, add = TRUE)
```

Radial kernel (RBF)

```r
set.seed(123)

svm.rbf = tune.svm(stroke ~ .,
                      data = trainSmoted,
                      kernel = "radial",
                      cost = exp(seq(-4,1,len=10)),
                      gamma = exp(seq(-5,3,len = 10)))



summary(svm.rbf)
plot(svm.rbf)

svm.rbf$best.parameters

best.rbf = svm.rbf$best.model
summary(best.rbf)

pred_train_rsvm = predict(best.rbf, newdata = train.data)
confusionMatrix(data = pred_train_rsvm, reference = train.data$stroke, positive = "Yes")

pred_test_rsvm = predict(best.rbf, newdata = test.data)
confusionMatrix(data = pred_test_rsvm, reference = test.data$stroke, positive = "Yes")

pred_test_rsvm_numeric = as.numeric(pred_test_rsvm) -1
roc.rsvm = roc(test.data$stroke, pred_test_rsvm_numeric)
```

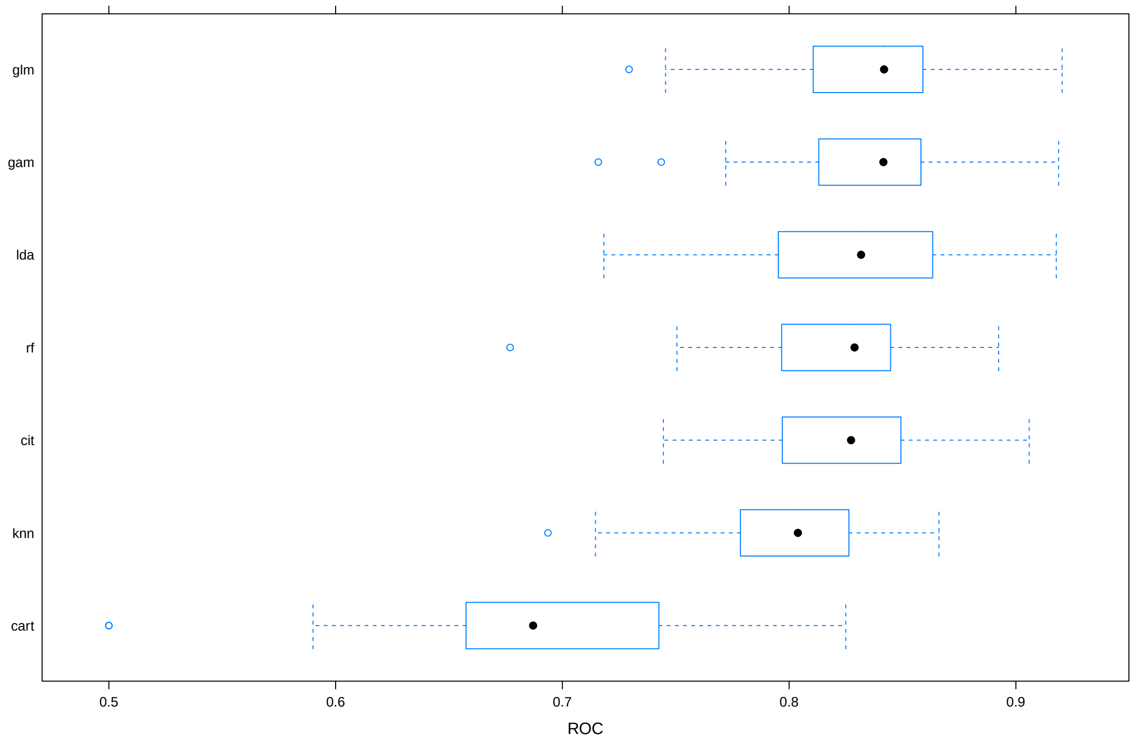
```r
auc.rsvm = roc.rsvm$auc[1]
auc.rsvm
plot(roc.rsvm, legacy.axes = TRUE, print.auc = TRUE)
# plot(smooth(roc.rsvm), col = 4, add = TRUE)
```

Compare models

```r
# based on cv

res <- resamples(list(glm = model.glm,
                      gam = model.gam,
                      knn = model.knn,
                      lda = model.lda,
                      cart = rpart.fit,
                      cit = model.ctree,
                      rf = rf.fit))
summary(res)
```

```
##
## Call:
## summary.resamples(object = res)
##
## Models: glm, gam, knn, lda, cart, cit, rf
## Number of resamples: 30
##
## ROC
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## glm  0.7294118 0.8135885 0.8418678 0.8357881 0.8586866 0.9204152    0
## gam  0.7158497 0.8147347 0.8415756 0.8355627 0.8578143 0.9188581    0
## knn  0.6936275 0.7790405 0.8038735 0.8003922 0.8260102 0.8660948    0
## lda  0.7183007 0.7977772 0.8317360 0.8298767 0.8612577 0.9178201    0
## cart 0.5000000 0.6589317 0.6870994 0.6880472 0.7407475 0.8249957    0
## cit  0.7445502 0.7982399 0.8273150 0.8222961 0.8487048 0.9058824    0
## rf   0.6769608 0.7972078 0.8288591 0.8187658 0.8441112 0.8923875    0
##
## Sens
##           Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
## glm  0.9970588 1.0000000 1.0000000 0.9999020 1.0000000    1    0
## gam  0.9970588 1.0000000 1.0000000 0.9999020 1.0000000    1    0
## knn  1.0000000 1.0000000 1.0000000 1.0000000 1.0000000    1    0
## lda  0.9824047 0.9911765 0.9941176 0.9928497 0.9970588    1    0
## cart 0.9735294 0.9852941 0.9882353 0.9890271 0.9941176    1    0
## cit  0.9911765 1.0000000 1.0000000 0.9994126 1.0000000    1    0
## rf   1.0000000 1.0000000 1.0000000 1.0000000 1.0000000    1    0
##
## Spec
##      Min. 1st Qu.     Median        Mean    3rd Qu.       Max. NA's
## glm     0       0 0.00000000 0.000000000 0.00000000 0.00000000    0
## gam     0       0 0.00000000 0.000000000 0.00000000 0.00000000    0
## knn     0       0 0.00000000 0.000000000 0.00000000 0.00000000    0
## lda     0       0 0.00000000 0.034422658 0.05882353 0.16666667    0
## cart    0       0 0.02777778 0.040196078 0.05882353 0.11764706    0
## cit     0       0 0.00000000 0.001851852 0.00000000 0.05555556    0
## rf      0       0 0.00000000 0.000000000 0.00000000 0.00000000    0
```

```
bwplot(res, metric = "ROC")
```



```
# GLM and GAM perform better compared to KNN. KNN usually requires a larger dataset to perform as good
```

```
#1st column is probability of negative, 2nd is positive
glm.pred <- predict(model.glm, newdata = test.data.imp, type = "prob")[,2]
gam.pred <- predict(model.gam, newdata = test.data.imp, type = "prob")[,2]
lda.pred <- predict(model.lda, newdata = test.data.imp, type = "prob")[,2]
knn.pred <- predict(model.knn, newdata = test.data.imp, type = "prob")[,2]
rf.pred <- predict(rf.fit, newdata = test.data.imp, type = "prob")[,2]
cit.pred <- predict(model.ctree, newdata = test.data.imp, type = "prob")[,2]
cart.pred <- predict(rpart.fit, newdata = test.data.imp, type = "prob")[,2]

roc.glm <- roc(test.data.imp$stroke, glm.pred)
```

```
## Setting levels: control = No, case = Yes

## Setting direction: controls < cases
```

```
roc.gam <- roc(test.data.imp$stroke, gam.pred)
```

```
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
```

```
roc.lda <- roc(test.data.imp$stroke, lda.pred)
```

```
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
```

```
roc.knn <- roc(test.data.imp$stroke, knn.pred)
```

```
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
```

```
roc.rf <- roc(test.data.imp$stroke, rf.pred)
```

```
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
```

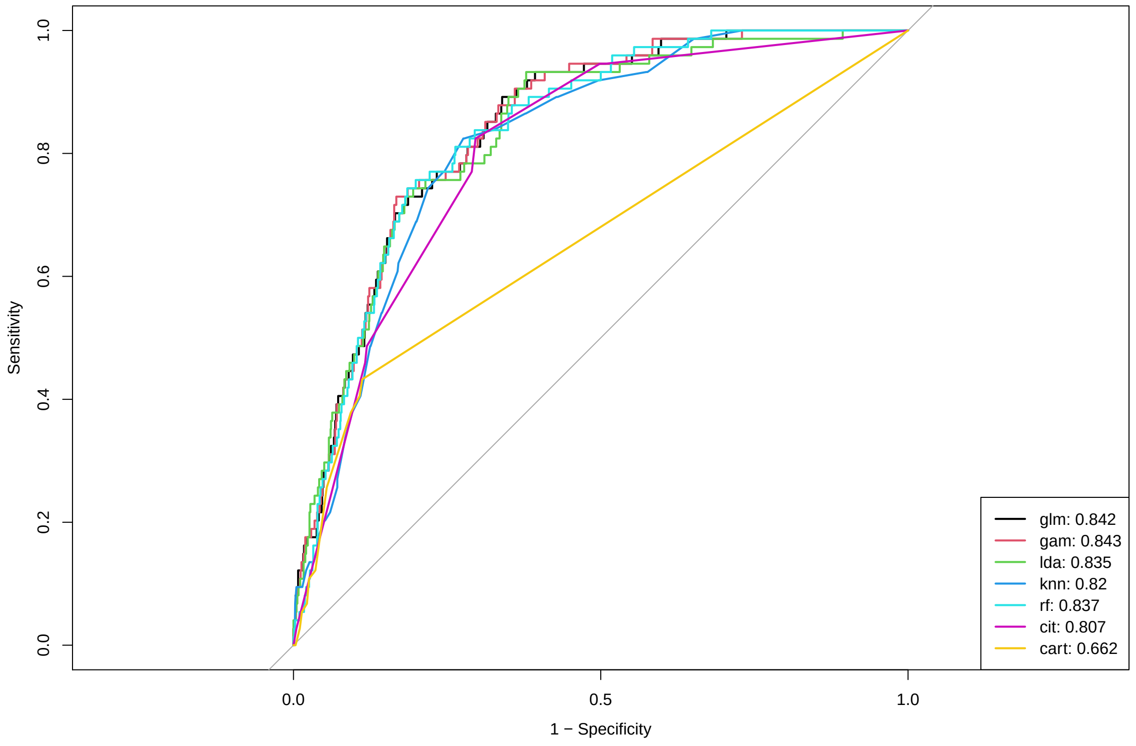
```
roc.cit <- roc(test.data.imp$stroke, cit.pred)
```

```
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
```

```
roc.cart <- roc(test.data.imp$stroke, cart.pred)
```

```
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
```

```
auc <- c(roc.glm$auc[1], roc.gam$auc[1],
         roc.lda$auc[1], roc.knn$auc[1],
         roc.rf$auc[1], roc.cit$auc[1],
         roc.cart$auc[1])

plot(roc.glm, legacy.axes = TRUE)
plot(roc.gam, col = 2, add = TRUE)
plot(roc.lda, col = 3, add = TRUE)
plot(roc.knn, col = 4, add = TRUE)
plot(roc.rf, col = 5, add = TRUE)
plot(roc.cit, col = 6, add = TRUE)
plot(roc.cart, col = 7, add = TRUE)

modelNames <- c("glm", "gam","lda", "knn","rf", "cit", "cart")
legend("bottomright", legend = paste0(modelNames, ": ", round(auc,3)),
       col = 1:7, lwd = 2)
```



Conclusion

We could fix this by ovesampling, however,for the purpose of our analysis, I opted to evaluate normal sampling data to avoid biased prediction results.The linear discriminant model would be more stable than the logistic regresion model if the distribution of the predictors is approximately normal, which is not the case in this example. LDA is also more popular when we have more than two response classes.

Variable Importance

```r
varImp(model.glm)
```

```
## glm variable importance
##
##                    Overall
## age               100.0000
## avg_glucose_level  19.2663
## hypertension       12.0604
## heart_disease      11.1260
## ever_married        8.3908
## work_type           7.6988
## gender              1.7581
## Residence_type      0.5543
## bmi                 0.3810
## smoking_status      0.0000
```