Stroke Prediction

Andrew Nguyen

2023-08-25

Libraries

```
library(tidyverse) # data wrangling
library(caret) # ML , training and testing
library(rpart) # ML
library(rpart.plot) # plotting ML models
library(vip) # identifying feature importance
library(smotefamily) # for handling imbalnces in dataset
#ggplot2 settings I like:
T<-theme_bw()+theme(,text=element_text(size=18),
                    axis.text=element_text(size=18),
                    panel.grid.major=element_blank(),
                    panel.grid.minor.x = element_blank(),
                    panel.grid = element_blank(),
                    legend.key = element_blank(),
                    axis.title.y=element_text(margin=margin(t=0,r=15,b=0,l=0)),
                    axis.title.x=element_text(margin=margin(t=15,r=,b=0,l=0)))
#+ theme(legend.position="none")
```

Loading the data

I downloaded the Stroke Prediction Database from kaggle. There are 11 clinical features for predicting stroke events.

Let's load it in and see what it looks like! We also need to process the data and remove ID because it is only an unique identifier, remove unknown smokers, and remove bmi NA values.

```
dat<-read.csv("data/healthcare-dataset-stroke-data.csv")

#get a sense of the data with glimpse
glimpse(dat)
```

```
## Rows: 5,110
## Columns: 12
## $ id                <int> 9046, 51676, 31112, 60182, 1665, 56669, 53882, 10434~
## $ gender            <chr> "Male", "Female", "Male", "Female", "Female", "Male"~
## $ age               <dbl> 67, 61, 80, 49, 79, 81, 74, 69, 59, 78, 81, 61, 54, ~
## $ hypertension      <int> 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1~
```

```
## $ heart_disease     <int> 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0~
## $ ever_married      <chr> "Yes", "Yes", "Yes", "Yes", "Yes", "Yes", "Yes", "No~
## $ work_type         <chr> "Private", "Self-employed", "Private", "Private", "S~
## $ Residence_type    <chr> "Urban", "Rural", "Rural", "Urban", "Rural", "Urban"~
## $ avg_glucose_level <dbl> 228.69, 202.21, 105.92, 171.23, 174.12, 186.21, 70.0~
## $ bmi               <chr> "36.6", "N/A", "32.5", "34.4", "24", "29", "27.4", "~
## $ smoking_status    <chr> "formerly smoked", "never smoked", "never smoked", "~
## $ stroke            <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
```

```
#need to make bmi a numeric
dat$bmi<-as.numeric(dat$bmi)
```

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

```
#need to remove ID and and samples with no BMI measurements
#filter out unknown smokers too
dat<-dat%>%
  dplyr::select(!id)%>%
  dplyr::filter(bmi!="N/A")%>%
  dplyr::filter(smoking_status!="Unknown")
#let's se how many people get strokes in this sample
dat%>%
  group_by(stroke)%>%
  count()%>%
  knitr::kable()
```

stroke	n
0	3246
1	180

```
dat$stroke<-factor(dat$stroke)
glimpse(dat)
```

```
## Rows: 3,426
## Columns: 11
## $ gender            <chr> "Male", "Male", "Female", "Female", "Male", "Male", ~
## $ age               <dbl> 67, 80, 49, 79, 81, 74, 69, 81, 61, 54, 79, 50, 64, ~
## $ hypertension      <int> 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0~
## $ heart_disease     <int> 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0~
## $ ever_married      <chr> "Yes", "Yes", "Yes", "Yes", "Yes", "Yes", "No", "Yes~
## $ work_type         <chr> "Private", "Private", "Private", "Self-employed", "P~
## $ Residence_type    <chr> "Urban", "Rural", "Urban", "Rural", "Urban", "Rural"~
## $ avg_glucose_level <dbl> 228.69, 105.92, 171.23, 174.12, 186.21, 70.09, 94.39~
## $ bmi               <dbl> 36.6, 32.5, 34.4, 24.0, 29.0, 27.4, 22.8, 29.7, 36.8~
## $ smoking_status    <chr> "formerly smoked", "never smoked", "smokes", "never ~
## $ stroke            <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
```

For the dataset:

I'm not seeing many stroke events here.

For the variables:

What I see missing here is diet, exercise, and sleep, which would be nice to have. But these variables can definitely impact the ones listed such as heart health.

I would like to try a decision tree model because I want to see if it produces intuitive cut-offs in these features. The interpretation could be cleaner because the data don't have to be standardized and the continue variables retain their original units.

We can compare the performance of a decision tree with a logistic regression. To avoid overfitting our models, let's split the dataset into training and test.

Splitting dataset into training and testing

- Training = 70% of data
- Testing = 30% of data

```
intrain <- createDataPartition(y = dat$stroke, p= 0.7, list = FALSE)
training <- dat[intrain,]
testing <- dat[-intrain,]
```

Fit a logistic regression

- fit three 10-fold cross validated logistic regression models

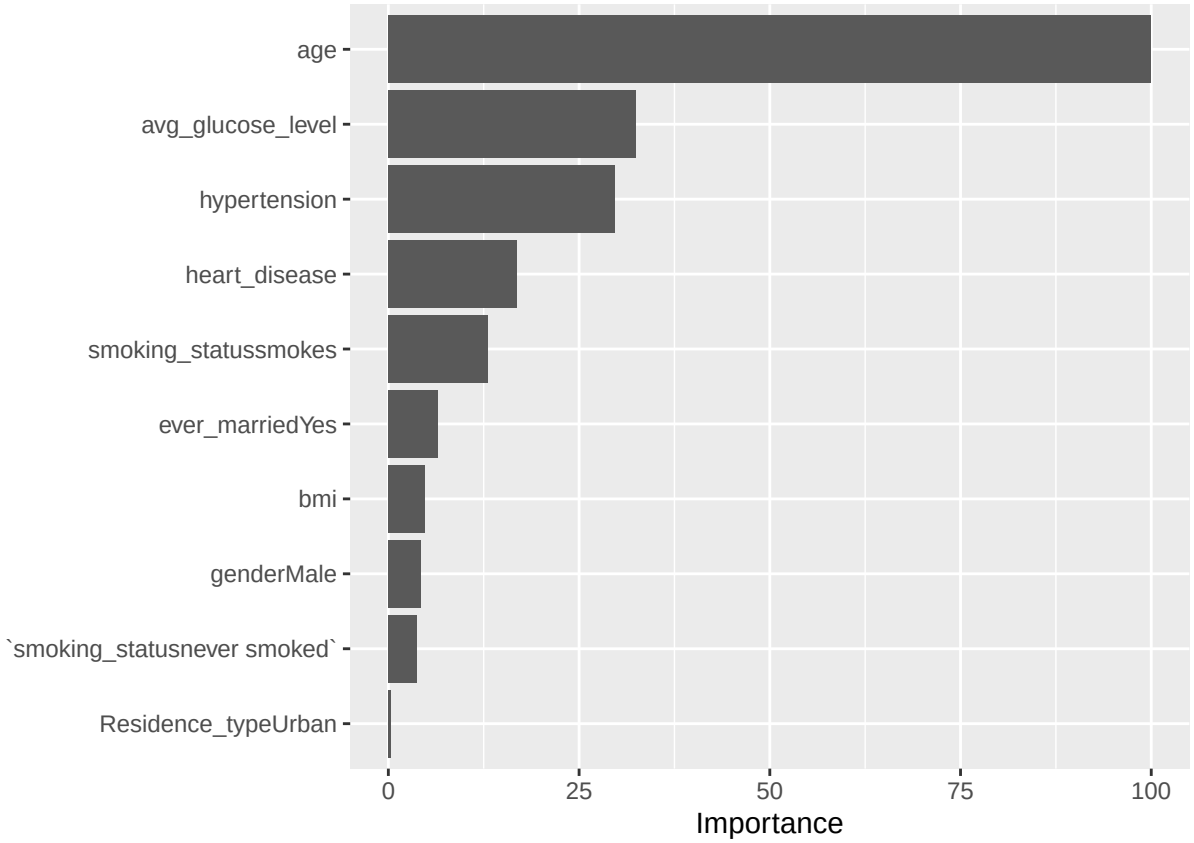
```
#for reproducibility
set.seed(123)
logistic.reg <- train(
  stroke ~ .,
  data = dat,
  method = "glm",
  family = "binomial",
  trControl = trainControl(method = "cv", number = 10)
)
# predict class on test dataset
pred_class <- predict(logistic.reg,testing)

# create confusion matrix
confusionMatrix(table(pred_class, testing$stroke))
```

```
## Confusion Matrix and Statistics
##
##
## pred_class   0   1
##          0 973  54
##          1   0   0
##
##                Accuracy : 0.9474
##                  95% CI : (0.9319, 0.9603)
##     No Information Rate : 0.9474
##     P-Value [Acc > NIR] : 0.5361
##
##                   Kappa : 0
##
```

```
##  Mcnemar's Test P-Value : 5.498e-13
##
##             Sensitivity : 1.0000
##             Specificity : 0.0000
##          Pos Pred Value : 0.9474
##          Neg Pred Value :    NaN
##              Prevalence : 0.9474
##          Detection Rate : 0.9474
##    Detection Prevalence : 1.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : 0
##
```

```
vip(logistic.reg) ### figuring out variable importance
```



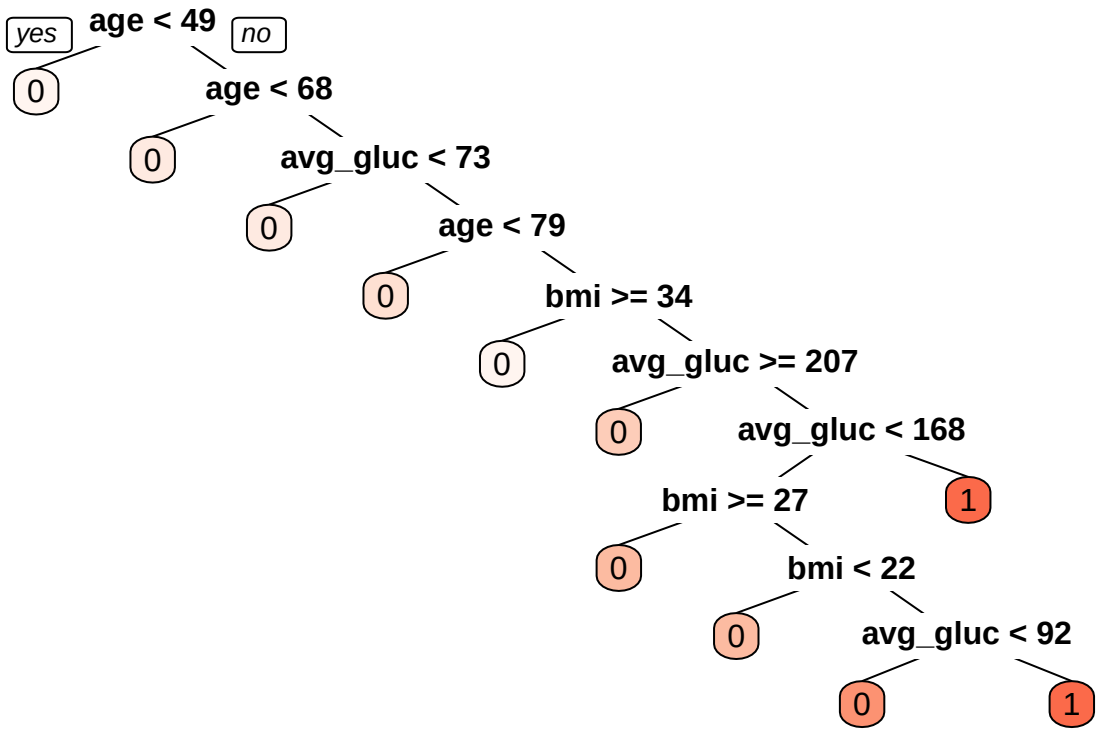
Performance is terrible, lots of false positives.

Fit a decision tree

```
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)

set.seed(3333)
#dtree_fit<-rpart(stroke~.,data=training,method="class")
```

```
dtree_fit <- train(stroke ~., data = training, method = "rpart",
                   parms = list(split = "information"),
                   trControl=trctrl,
                   tuneLength = 10)
prp(dtree_fit$finalModel, box.palette = "Reds")
```



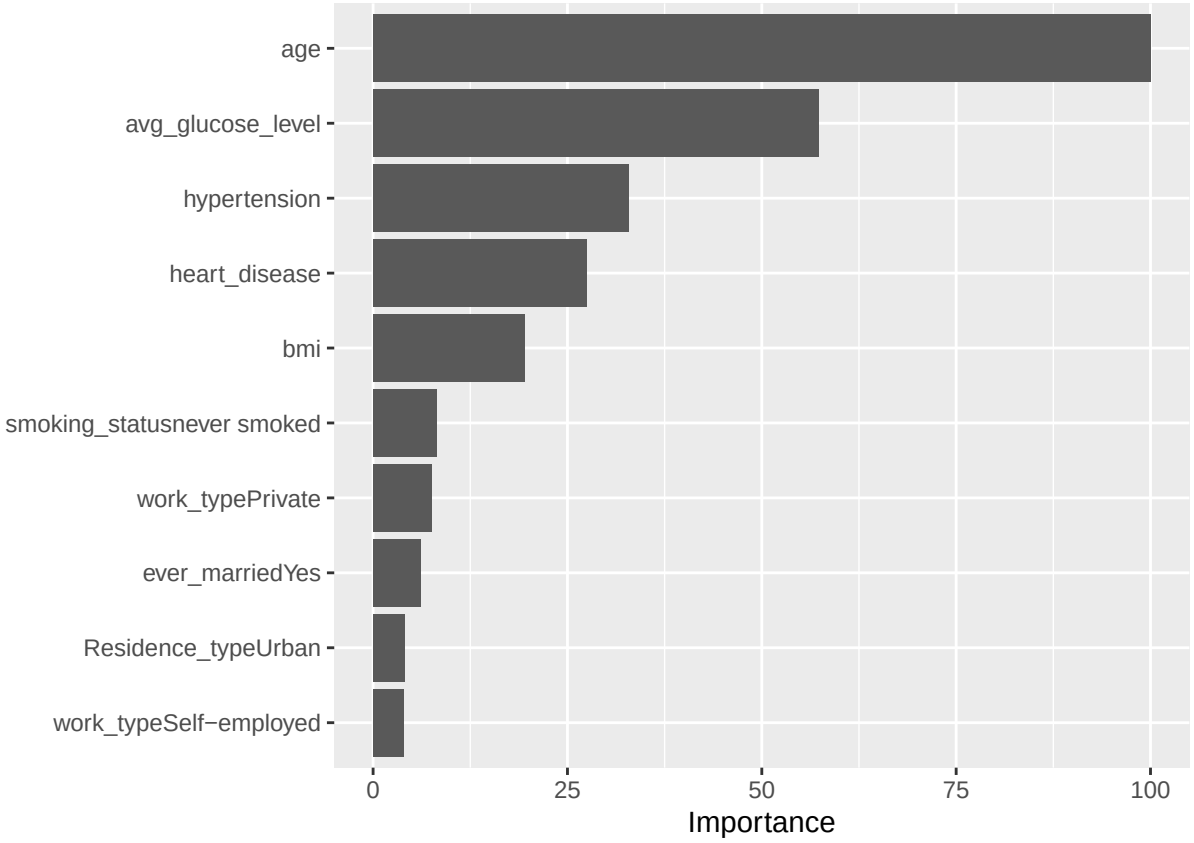

```
pred_class2 <- predict(dtree_fit,newdata=testing)

confusionMatrix(table(pred_class2, testing$stroke))
```

```
## Confusion Matrix and Statistics
##
##
## pred_class2   0   1
##           0 963  52
##           1  10   2
##
##                Accuracy : 0.9396
##                  95% CI : (0.9233, 0.9534)
##     No Information Rate : 0.9474
##     P-Value [Acc > NIR] : 0.8812
##
##                   Kappa : 0.0423
##
##  Mcnemar's Test P-Value : 1.919e-07
```

```
##
##             Sensitivity : 0.98972
##             Specificity : 0.03704
##          Pos Pred Value : 0.94877
##          Neg Pred Value : 0.16667
##              Prevalence : 0.94742
##          Detection Rate : 0.93768
##    Detection Prevalence : 0.98832
##       Balanced Accuracy : 0.51338
##
##        'Positive' Class : 0
##
```

```
vip(dtree_fit)
```



These performances are terrible. I'm finding that others are finding the same issue I'm having in that the features don't have a great association with the event, stroke. And there is a suggestion to use SMOTE as a way to handle imbalances within the dataset.

SessionInfo

```
sessionInfo()
```

```
## R version 4.3.1 (2023-06-16 ucrt)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19045)
##
## Matrix products: default
##
##
## locale:
## [1] LC_COLLATE=English_United States.utf8
## [2] LC_CTYPE=English_United States.utf8
## [3] LC_MONETARY=English_United States.utf8
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.utf8
##
## time zone: America/New_York
## tzcode source: internal
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] smotefamily_1.3.1 vip_0.4.1         rpart.plot_3.1.1  rpart_4.1.19
##  [5] caret_6.0-94      lattice_0.21-8    lubridate_1.9.2   forcats_1.0.0
##  [9] stringr_1.5.0     dplyr_1.1.2       purrr_1.0.2       readr_2.1.4
## [13] tidyr_1.3.0       tibble_3.2.1      ggplot2_3.4.3     tidyverse_2.0.0
##
## loaded via a namespace (and not attached):
##  [1] gtable_0.3.4         xfun_0.40            recipes_1.0.7
##  [4] tzdb_0.4.0           vctrs_0.6.3          tools_4.3.1
##  [7] generics_0.1.3       stats4_4.3.1         parallel_4.3.1
## [10] proxy_0.4-27         fansi_1.0.4          highr_0.10
## [13] ModelMetrics_1.2.2.2 pkgconfig_2.0.3      Matrix_1.6-1
## [16] data.table_1.14.8    lifecycle_1.0.3      farver_2.1.1
## [19] compiler_4.3.1       munsell_0.5.0        codetools_0.2-19
## [22] htmltools_0.5.6      class_7.3-22         yaml_2.3.7
## [25] prodlim_2023.03.31   pillar_1.9.0         MASS_7.3-60
## [28] gower_1.0.1          iterators_1.0.14     foreach_1.5.2
## [31] parallelly_1.36.0    nlme_3.1-163         lava_1.7.2.1
## [34] tidyselect_1.2.0     digest_0.6.33        stringi_1.7.12
## [37] future_1.33.0        reshape2_1.4.4       listenv_0.9.0
## [40] labeling_0.4.2       splines_4.3.1        fastmap_1.1.1
## [43] grid_4.3.1           colorspace_2.1-0     cli_3.6.1
## [46] magrittr_2.0.3       survival_3.5-7       utf8_1.2.3
## [49] e1071_1.7-13         future.apply_1.11.0  withr_2.5.0
## [52] scales_1.2.1         timechange_0.2.0     rmarkdown_2.24
## [55] globals_0.16.2       nnet_7.3-19          timeDate_4022.108
## [58] hms_1.1.3            evaluate_0.21        knitr_1.43
## [61] hardhat_1.3.0        rlang_1.1.1          Rcpp_1.0.11
## [64] glue_1.6.2           pROC_1.18.4          ipred_0.9-14
## [67] rstudioapi_0.15.0    R6_2.5.1             plyr_1.8.8
```