Applied Machine Learning and Predictive Modelling I: Modelling Stroke Data

Authors: Larissa Eisele, Fabian Lüthard, Yves Maillard

Module: Applied Machine Learning and Predictive Modelling I

Submitted on 10th of June, 2022

SUPERVISOR: MATTEO TANADINI AND DANIEL MEISTER

Lucerne University of Applied Sciences and Arts

Table of Contents

1 Introduction

Use case: We are a smart watch manufacturer working on a new feature for stroke prevention. We are going to analyze survey data that we plan to ask our users, complementing it with HR (Heart Rate) and CGM (Continuous Glucose Monitoring) data that our product already measures. We hope that our feature can prevent serious health issues and motivate our users to adopt healthier lifestyles.

2 Importing Data

```r
stroke_data <- read_csv('./data/healthcare-dataset-stroke-data.csv')
```

```
## Rows: 5110 Columns: 12
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (6): gender, ever_married, work_type, Residence_type, bmi, smoking_status
## dbl (6): id, age, hypertension, heart_disease, avg_glucose_level, stroke
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
stroke_data$bmi <- as.numeric(stroke_data$bmi)
```

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

```r
stroke_data$bmi[is.na(stroke_data$bmi)] <- mean(stroke_data$bmi, na.rm = TRUE)
stroke_data$stroke <- as.factor(stroke_data$stroke)

stroke_data
```

```
## # A tibble: 5,110 x 12
##       id gender   age hypertension heart_disease ever_married work_type
##    <dbl> <chr>  <dbl>        <dbl>         <dbl> <chr>        <chr>
##  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
##  7 53882 Male      74            1             1 Yes          Private
##  8 10434 Female    69            0             0 No           Private
##  9 27419 Female    59            0             0 Yes          Private
## 10 60491 Female    78            0             0 Yes          Private
## # ... with 5,100 more rows, and 5 more variables: Residence_type <chr>,
## #   avg_glucose_level <dbl>, bmi <dbl>, smoking_status <chr>, stroke <fct>
```

2.1 Summaries

```r
#testing the effect of non smokers and smokers
count_by_smoke_status <- stroke_data %>%
  select(smoking_status, stroke) %>%
  group_by(smoking_status, stroke) %>%
  summarise( N = n())
```

```
## `summarise()` has grouped output by 'smoking_status'. You can override using the `.groups` argument.
```

```
#testing the effect of work type
 count_by_work_type <- stroke_data %>%
   select(work_type, stroke) %>%
   group_by(work_type, stroke) %>%
   summarise( N = n())
```

```
## `summarise()` has grouped output by 'work_type'. You can override using the `.groups` argument.
```

```
 # testing the effects of residence type
count_by_Residence_type <- stroke_data %>%
   select(Residence_type, stroke) %>%
   group_by(Residence_type, stroke) %>%
   summarise( N = n())
```

```
## `summarise()` has grouped output by 'Residence_type'. You can override using the `.groups` argument.
```

```
 # testing the effects of gender
count_by_gender <- stroke_data %>%
   select(gender, stroke) %>%
   group_by(gender, stroke) %>%
   summarise( N = n())
```

```
## `summarise()` has grouped output by 'gender'. You can override using the `.groups` argument.
```

```
 # testing the effects of hypertension
count_by_hypertension <- stroke_data %>%
   select(hypertension, stroke) %>%
   group_by(hypertension, stroke) %>%
   summarise( N = n())
```

```
## `summarise()` has grouped output by 'hypertension'. You can override using the `.groups` argument.
```

```
# testing the effects of heart disease
count_by_heart_disease <- stroke_data %>%
   select(heart_disease, stroke) %>%
   group_by(heart_disease, stroke) %>%
   summarise( N = n())
```

```
## `summarise()` has grouped output by 'heart_disease'. You can override using the `.groups` argument.
```

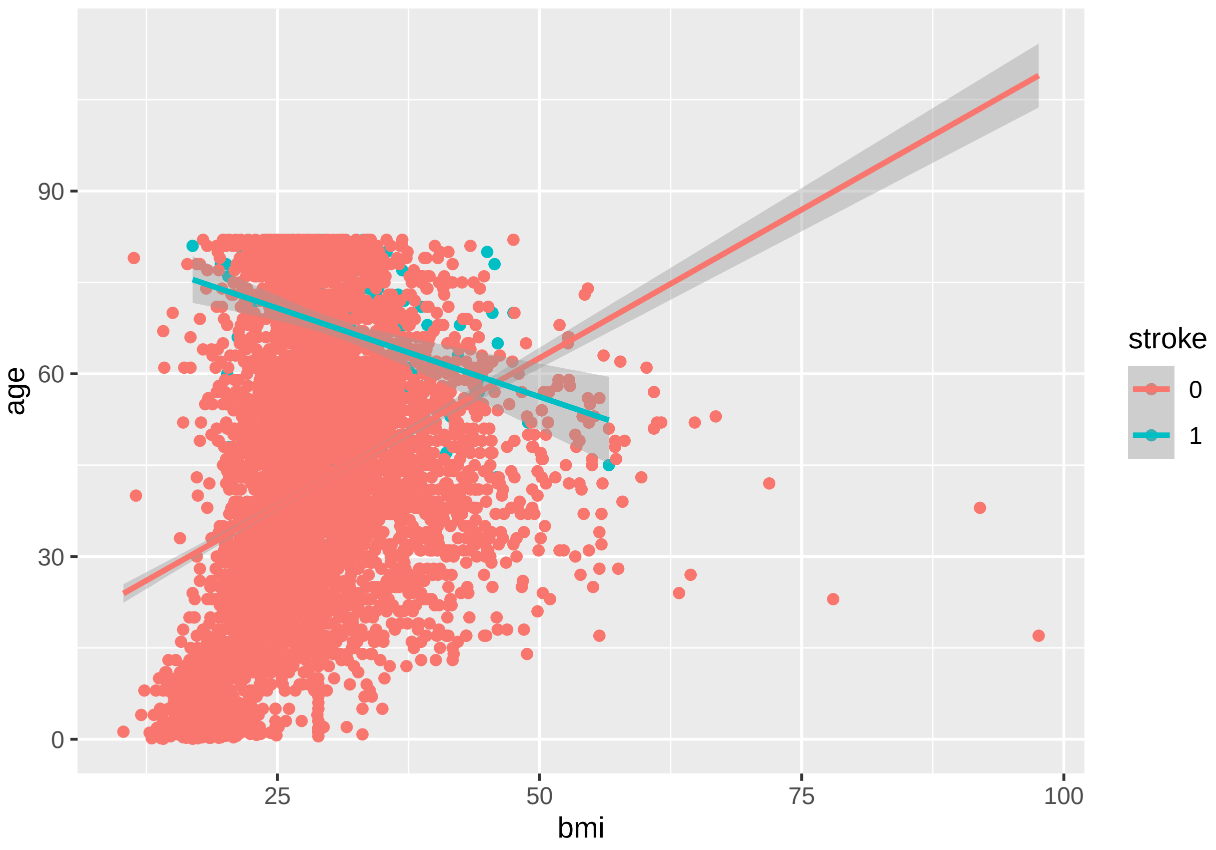
```
# testing the effects of marriage status
count_by_marriage <- stroke_data %>%
   select(ever_married, stroke) %>%
   group_by(ever_married, stroke) %>%
   summarise( N = n())
```

```
## `summarise()` has grouped output by 'ever_married'. You can override using the `.groups` argument.
```

2.2 Scatterplots

```
stroke_data %>%
  ggplot(mapping = aes(x = bmi, y = age, color = stroke)) +
  geom_point() +
  geom_smooth(method = 'lm')
```

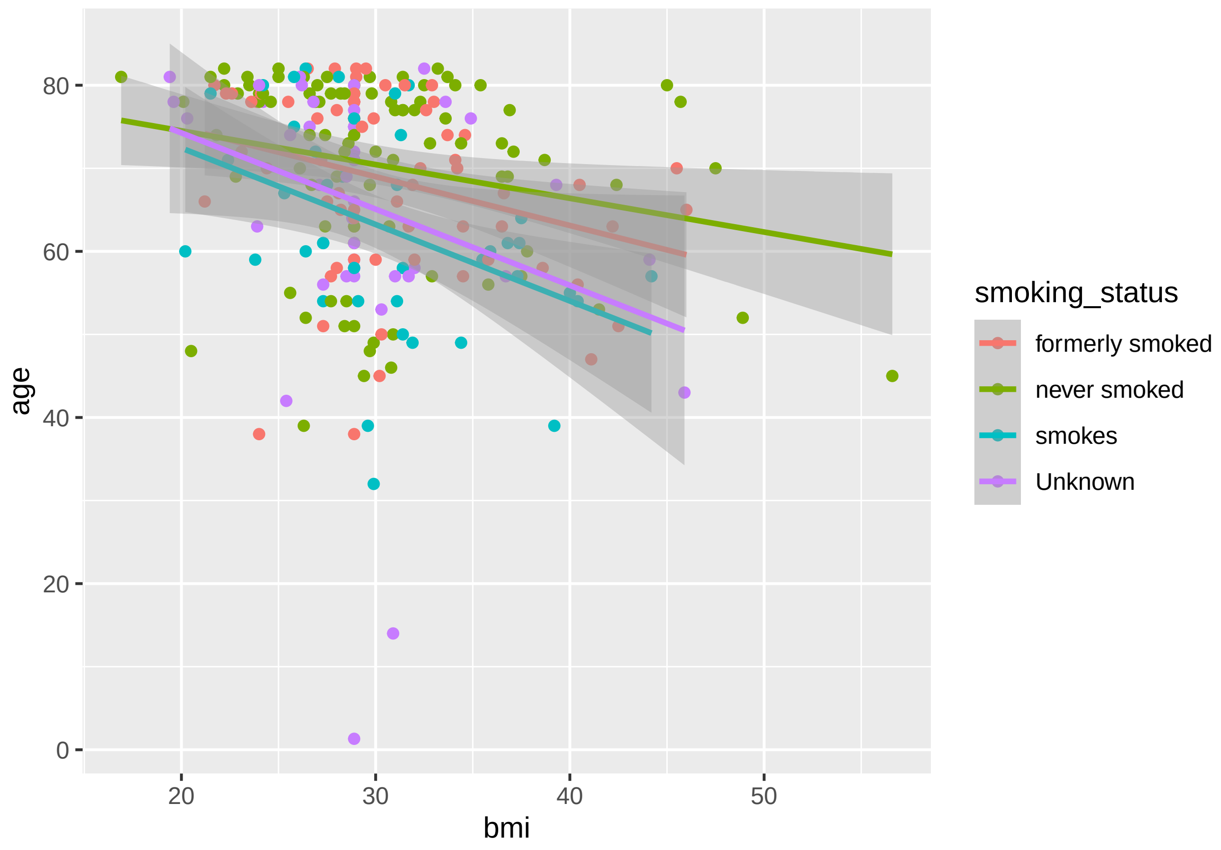
```
## `geom_smooth()` using formula 'y ~ x'
```



```r
stroke_data %>%
  filter(stroke == 1) %>%
  ggplot(mapping = aes(x = bmi, y = age, color = smoking_status)) +
  geom_point(method = 'lm') +
  geom_smooth(method = 'lm')
```

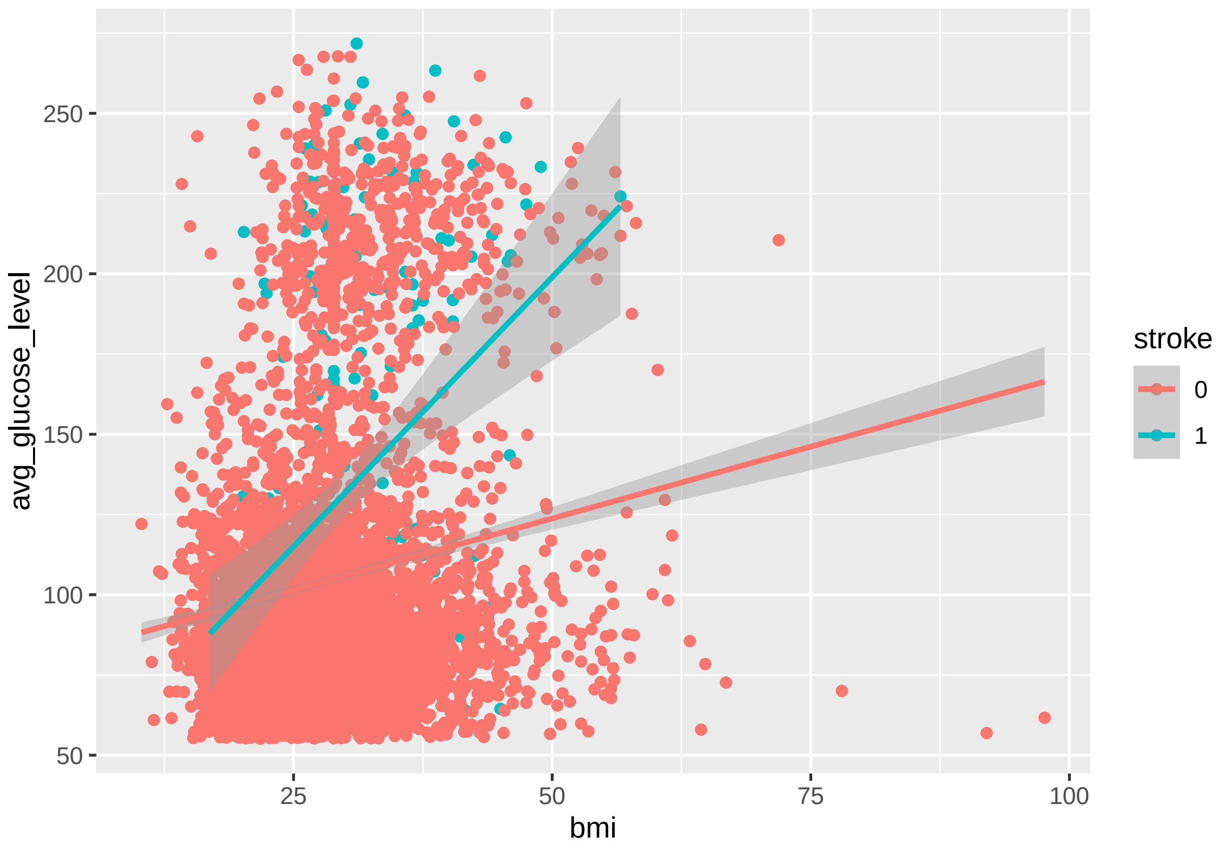
```
## Warning: Ignoring unknown parameters: method
```

```
## `geom_smooth()` using formula 'y ~ x'
```



```r
stroke_data %>%
  ggplot(mapping = aes(x = bmi, y = avg_glucose_level, color = stroke)) +
  geom_point() +
  geom_smooth(method = 'lm')
```

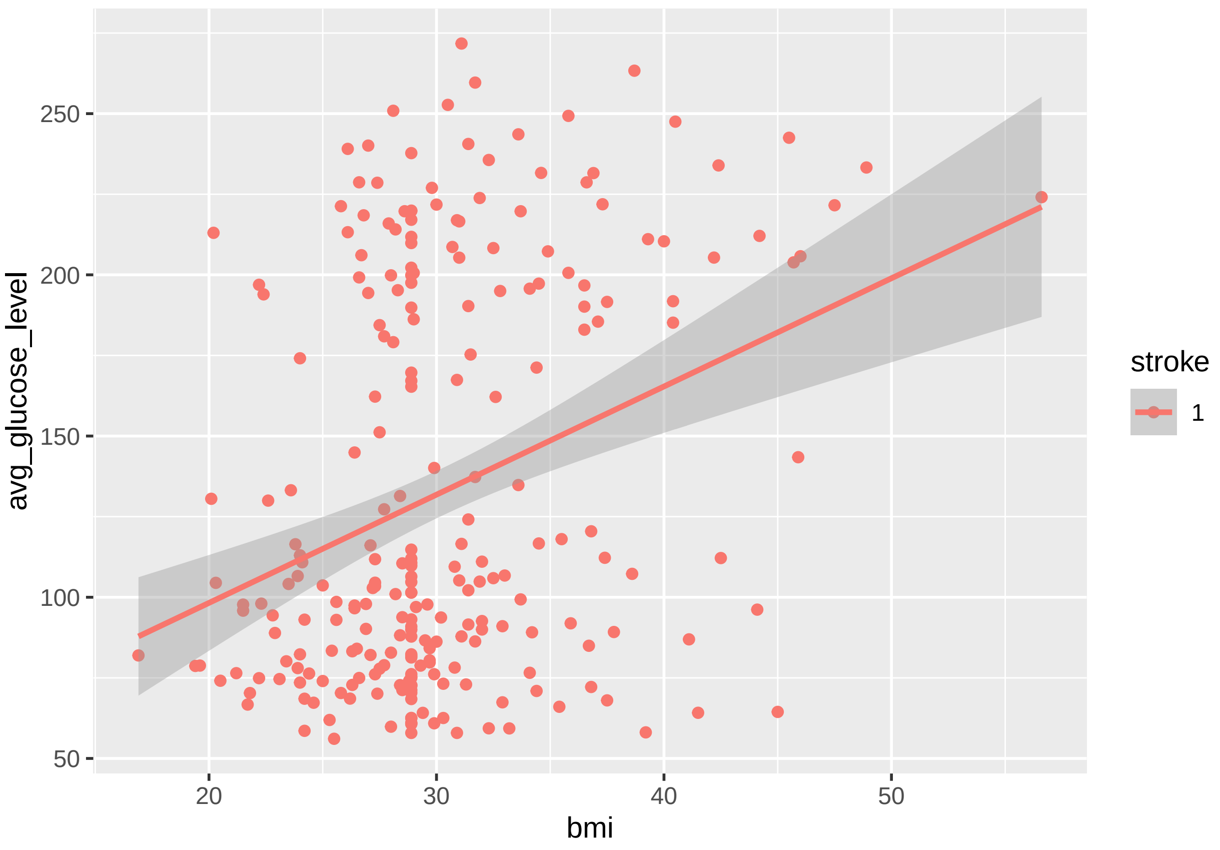
```
## `geom_smooth()` using formula 'y ~ x'
```



```
stroke_data %>%
  filter(stroke == 1) %>%
  ggplot(mapping = aes(x = bmi, y = avg_glucose_level, color = stroke)) +
  geom_point(method = 'lm') +
  geom_smooth(method = 'lm')
```

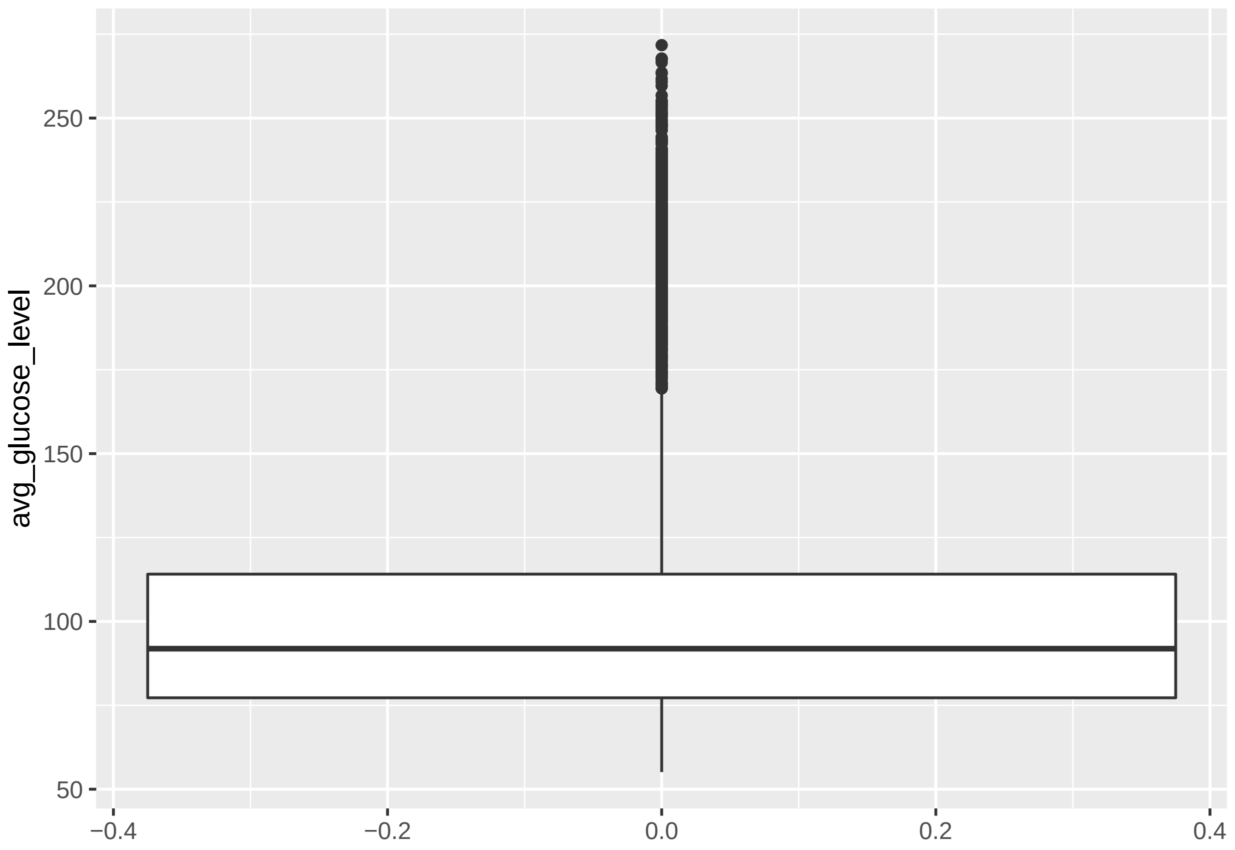
```
## Warning: Ignoring unknown parameters: method
```

```
## `geom_smooth()` using formula 'y ~ x'
```

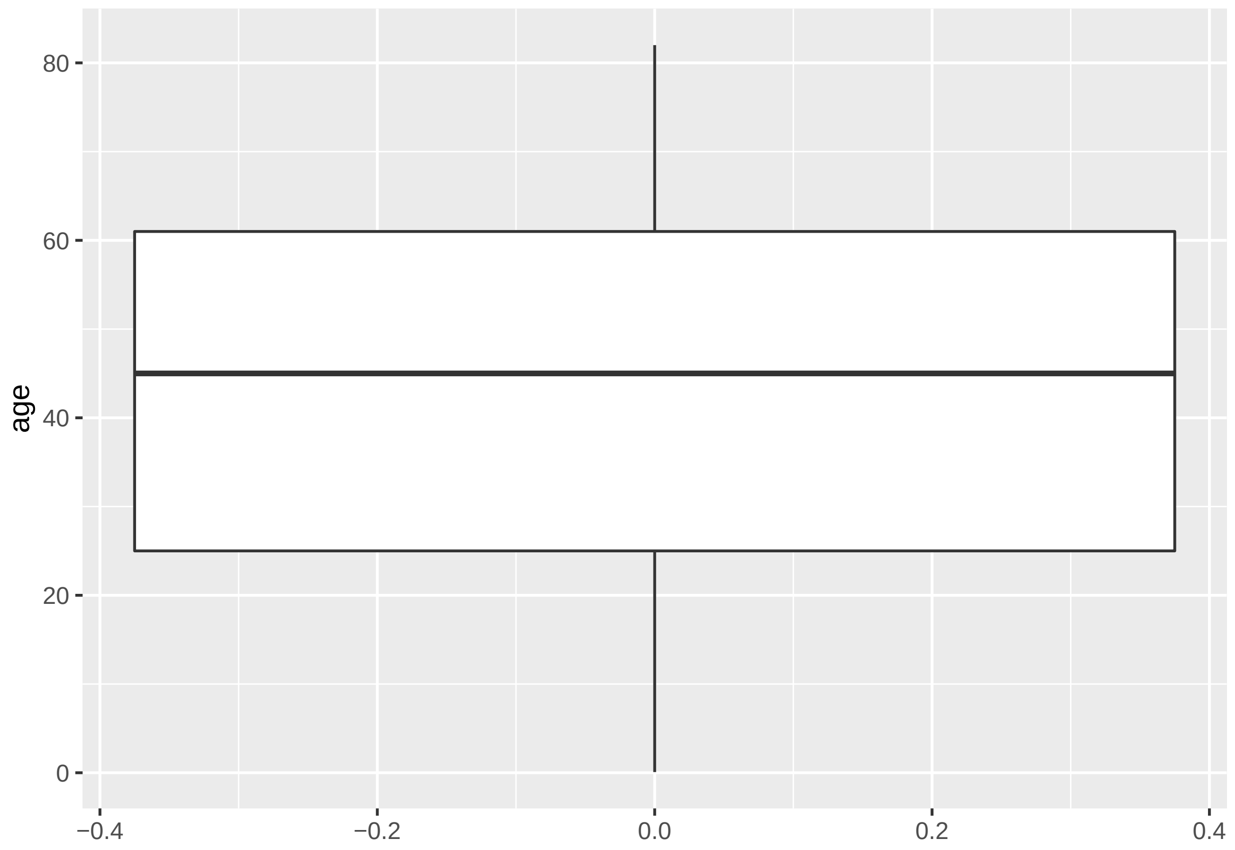


2.3 Boxplots

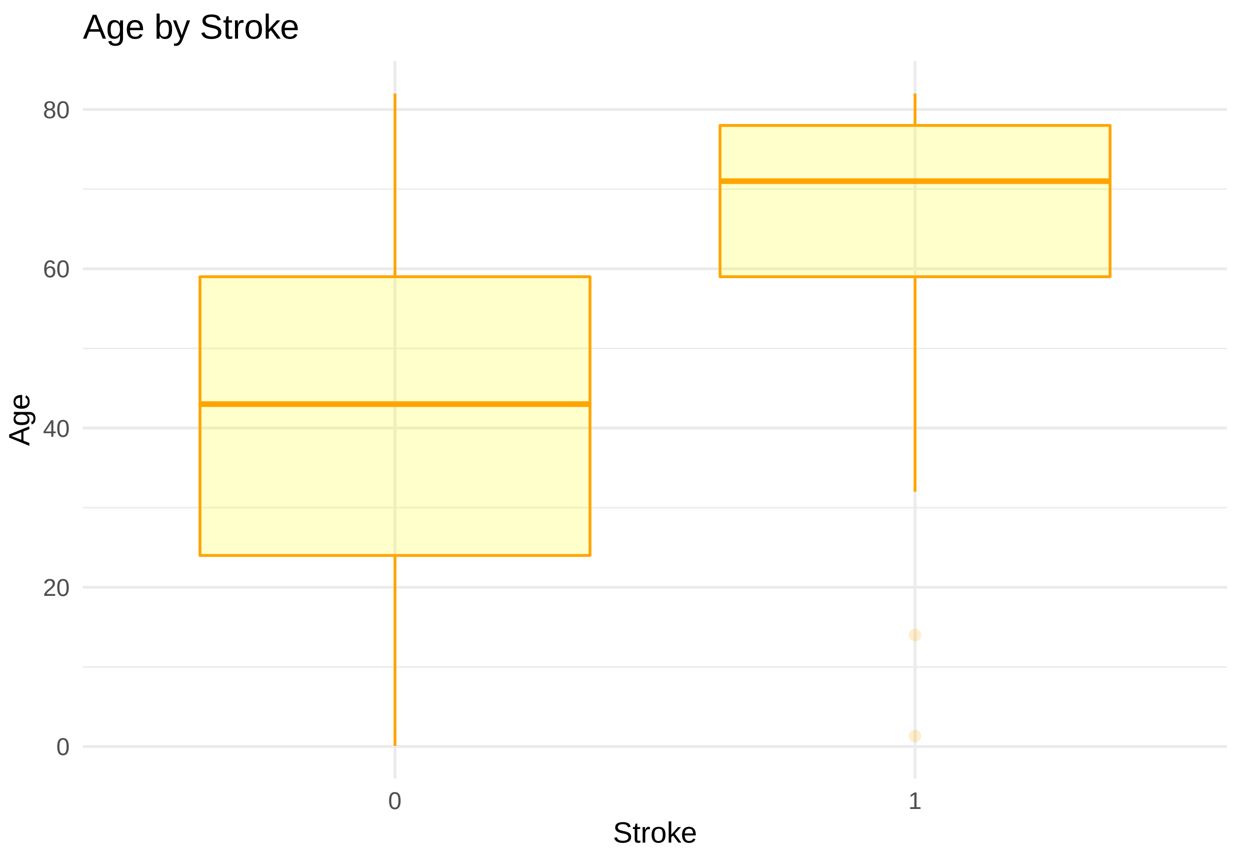
```r
#Boxplot on avg. glucose_level
ggplot(stroke_data, aes(y = avg_glucose_level)) +
  geom_boxplot()
```



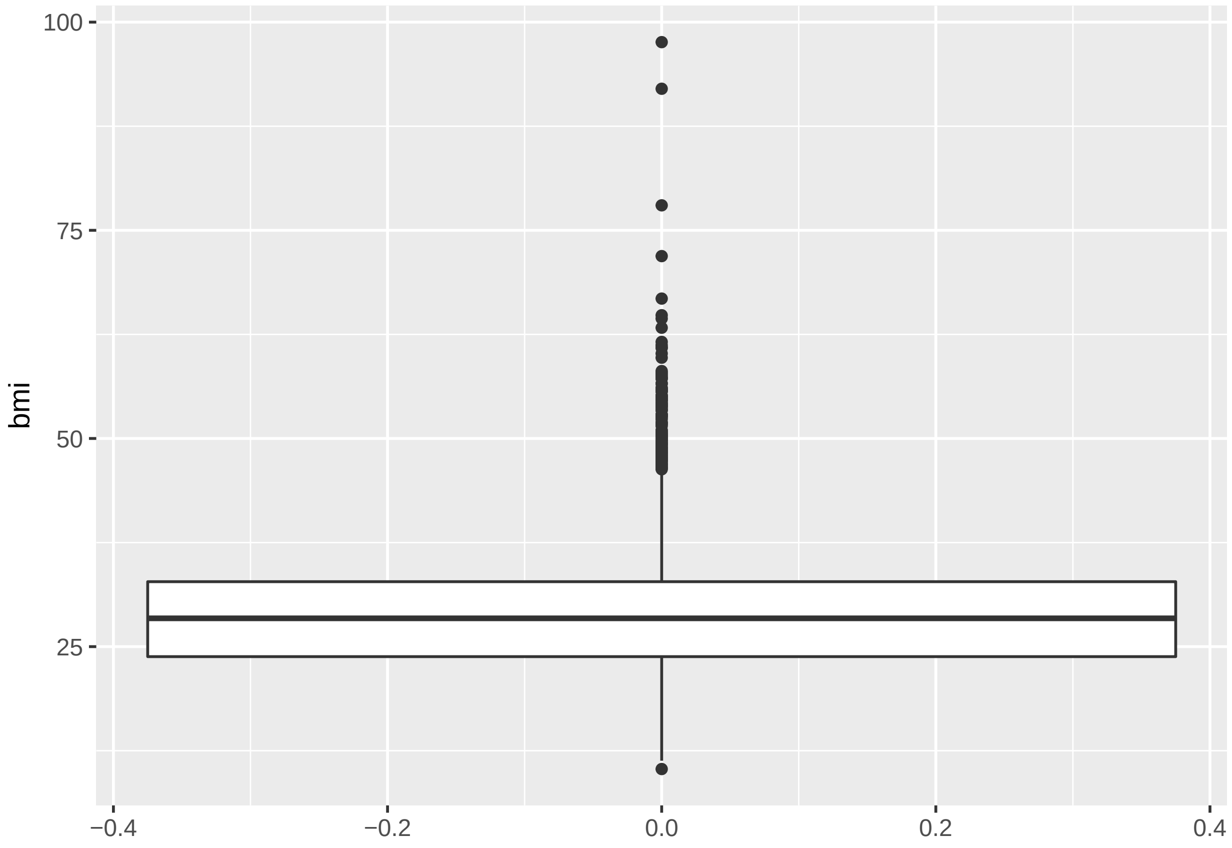
```r
#Boxplot on Age
ggplot(stroke_data, aes(y = age)) +
  geom_boxplot()
```



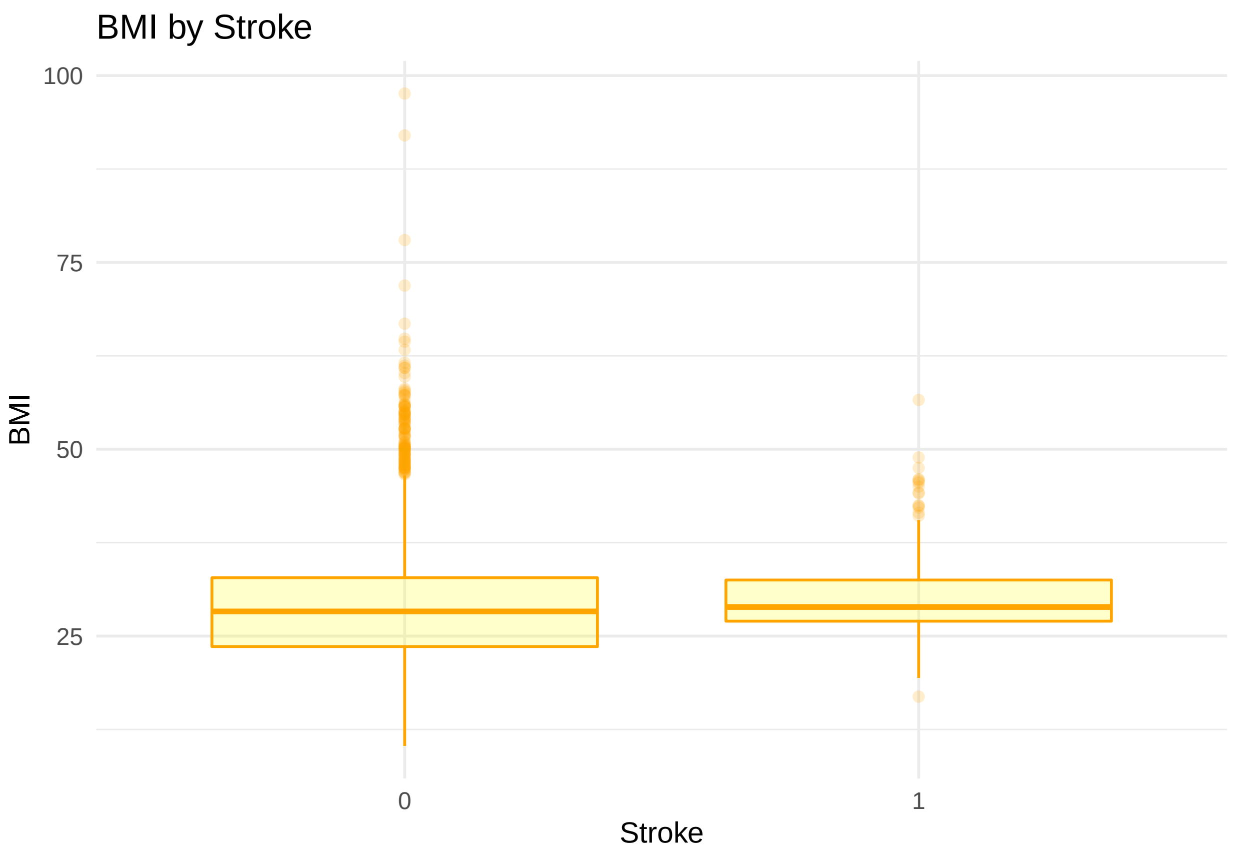
```
stroke_by_age <- stroke_data %>%
# dplyr::filter(stroke == 1) %>%
 ggplot(aes(x = stroke,
            y = age)) +
  geom_boxplot(color="orange", fill="yellow", alpha=0.2) +
  ggtitle("Age by Stroke") +
  xlab("Stroke") + ylab("Age") +
  theme_minimal() + theme(axis.text.x = element_text(angle = 0))

stroke_by_age
```



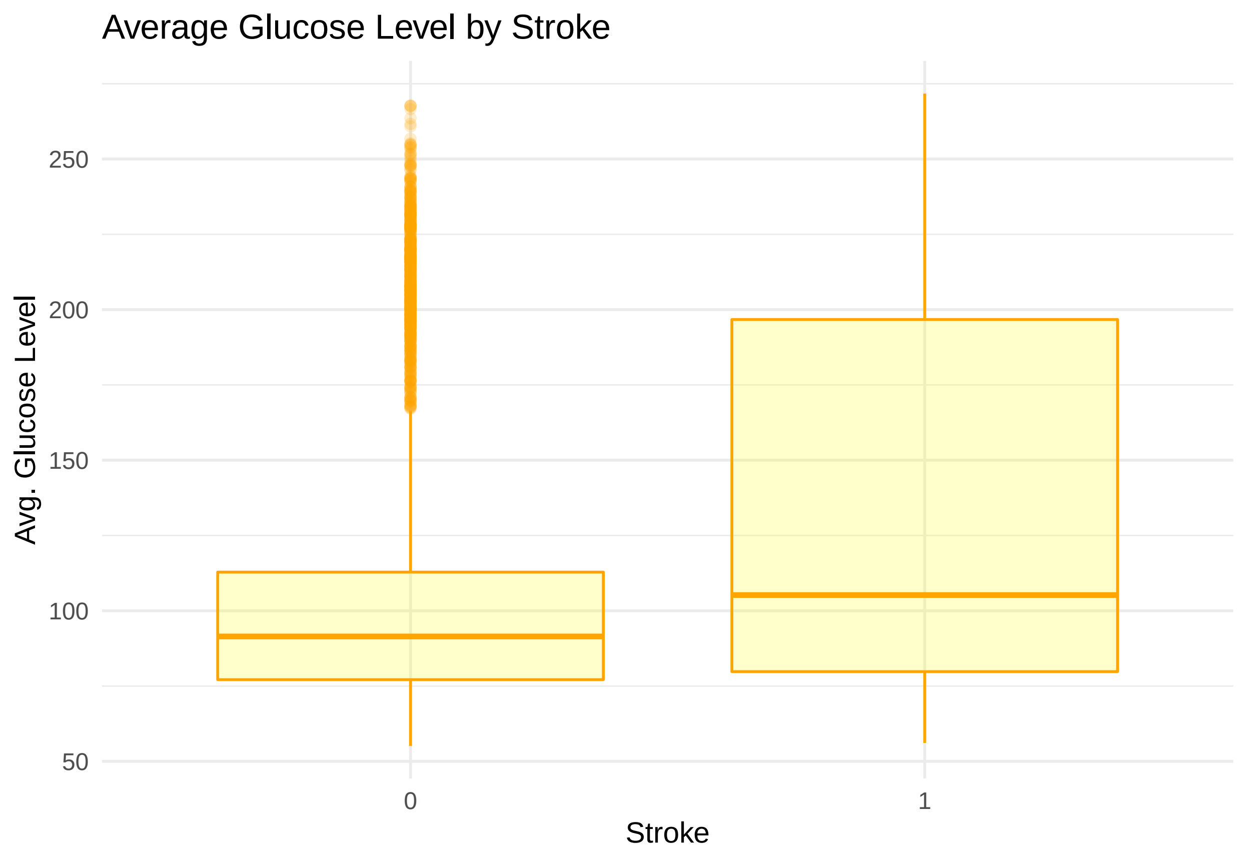
```
#Boxplot on BMI
ggplot(stroke_data, aes(y = bmi)) +
geom_boxplot()
```



```
stroke_by_bmi <- stroke_data %>%
# dplyr::filter(stroke == 1) %>%
 ggplot(aes(x = stroke,
            y = bmi)) +
  geom_boxplot(color="orange", fill="yellow", alpha=0.2) +
  ggtitle("BMI by Stroke") +
  xlab("Stroke") + ylab("BMI") +
  theme_minimal() + theme(axis.text.x = element_text(angle = 0))
stroke_by_bmi
```



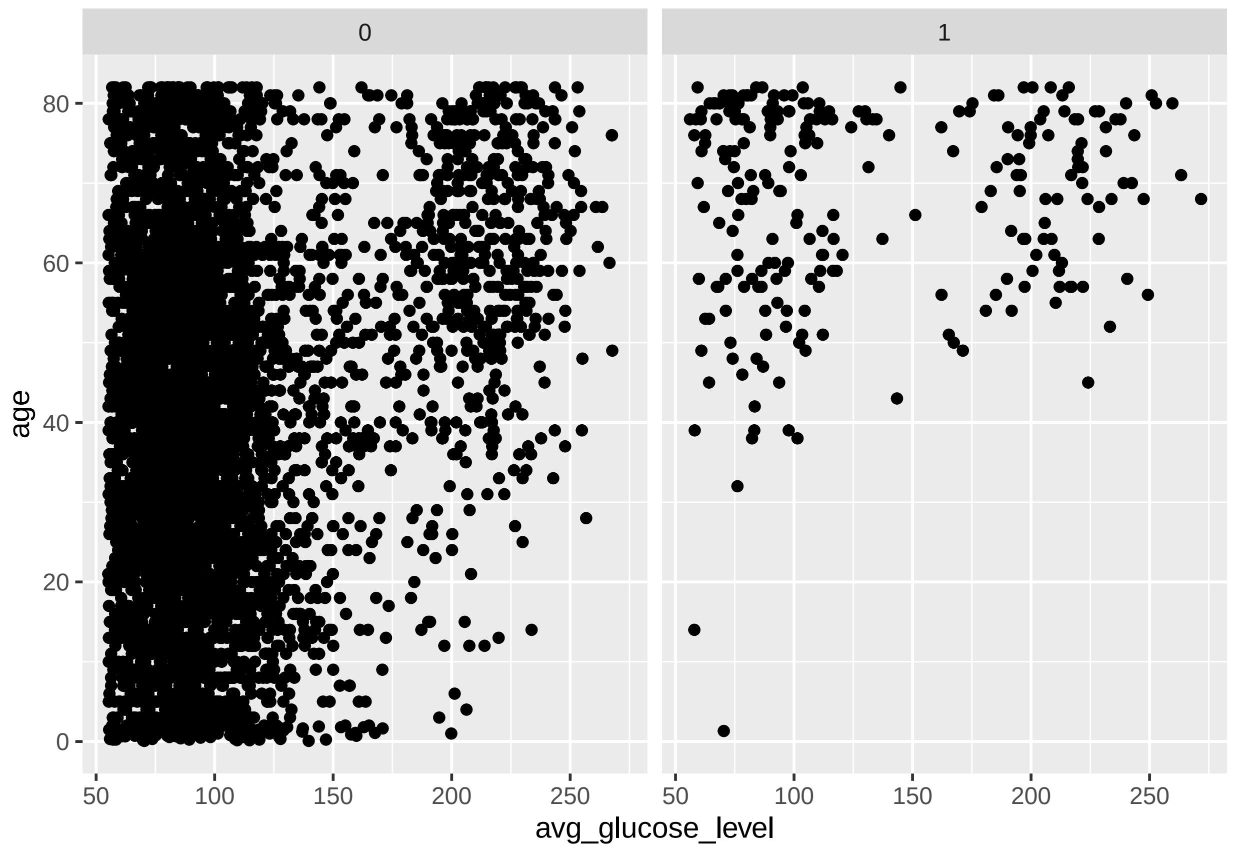
```r
#boxplot by avg_glucose_level and stroke
stroke_by_avg_glucose_level <- stroke_data %>%
# dplyr::filter(stroke == 1) %>%
 ggplot(aes(x = stroke,
            y = avg_glucose_level)) +
  geom_boxplot(color="orange", fill="yellow", alpha=0.2) +
  ggtitle("Average Glucose Level by Stroke") +
  xlab("Stroke") + ylab("Avg. Glucose Level") +
  theme_minimal() + theme(axis.text.x = element_text(angle = 0))

stroke_by_avg_glucose_level
```



3 Methodology

4 Linear Model

```
# qplot -> age/glucose level per stroke
library(ggplot2)
 qplot(y = age, x = avg_glucose_level,
        data = stroke_data,
        facets = ~ stroke)
```



```r
# fitting models for simple regression model
lm.stroke <- lm(age ~ avg_glucose_level, data = stroke_data)
summary(lm.stroke)
```

```
##
## Call:
## lm(formula = age ~ avg_glucose_level, data = stroke_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -53.368 -16.775   1.115  16.626  44.648
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       30.602224   0.783098   39.08   <2e-16 ***
## avg_glucose_level  0.118932   0.006786   17.53   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.96 on 5108 degrees of freedom
## Multiple R-squared:  0.05673,    Adjusted R-squared:  0.05654
## F-statistic: 307.2 on 1 and 5108 DF,  p-value: < 2.2e-16
```

“‘

5 Generalised Linear Model with family set to Poisson

6 Generalised Linear Model with family set to Binomial

7 Generalised Additive Model

8 Neural Network

9 Support Vector Machine

10 solve an optimisation problem