homework2

2025-04-26

```r
#Imported all the necessary libraries
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ dplyr     1.1.4     ✔ readr     2.1.5
## ✔ forcats   1.0.0     ✔ stringr   1.5.1
## ✔ ggplot2   3.5.2     ✔ tibble    3.2.1
## ✔ lubridate 1.9.4     ✔ tidyr     1.3.1
## ✔ purrr     1.0.4
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to be
come errors
```

```r
library(e1071)
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(ggplot2)
library(dplyr)
```

```r
# I've set the working directory and loaded 'nhis_2022.csv' into data
data <- read.csv("C:/Users/alekh/Downloads/nhis_2022.csv")

# Exploring the data
head(data)
```

```
##   YEAR SERIAL STRATA PSU        NHISHID REGION PERNUM          NHISPID     HHX
## 1 2022      1    143  16 0002022H000001      4      1 0002022H00000110 H000001
## 2 2022      2    106  53 0002022H000003      3      1 0002022H00000310 H000003
## 3 2022      2    106  53 0002022H000003      3      2 0002022H00000320 H000003
## 4 2022      3    134  13 0002022H000006      2      1 0002022H00000610 H000006
## 5 2022      4    106  53 0002022H000007      3      1 0002022H00000710 H000007
## 6 2022      4    106  53 0002022H000007      3      2 0002022H00000720 H000007
##   SAMPWEIGHT ASTATFLG CSTATFLG AGE SEX MARSTCUR EDUC HOURSWRK POVERTY HEIGHT
## 1       8018        1        0  61   1        1  201       45      34     69
## 2      10117        1        0  43   1        1  301       45      37     70
## 3       7933        0        1  12   2        0    0        0      37     60
## 4       2681        1        0  68   1        5  505        0      31     75
## 5      10233        1        0  73   1        1  201        0      32     71
## 6       7712        0        1  16   2        0    0        0      32     65
##   WEIGHT BMICALC HINOTCOVE CANCEREV CHEARTDIEV DIABETICEV HEARTATTEV STROKEV
## 1    260    38.4         1        1          1          1          1       1
## 2    190    27.3         1        1          1          1          1       1
## 3     96    18.7         1        0          0          1          0       0
## 4    200    25.0         1        1          1          1          1       1
## 5    172    24.0         1        1          1          1          1       1
## 6    106    17.6         1        0          0          1          0       0
##   ALCANYNO ALCDAYSYR CIGDAYMO MOD10DMIN VIG10DMIN FRUTNO VEGENO JUICEMNO
## 1        2       104       96         0         0      5     15        0
## 2        1        52       96        20         0      1      1        1
## 3      996       996       96         0         0    996    996      996
## 4        7       364       96        60         0      3      1        0
## 5        0         0       96       690         0      2      4        0
## 6      996       996       96         0         0    996    996      996
##   SALADSNO BEANNO SALSAMNO TOMSAUCEMNO SODAPNO FRIESPNO SPORDRMNO FRTDRINKMNO
## 1       10      5        5           2       0      110         3           0
## 2        1      1        1           1       0        1         0           0
## 3      996    996      996         996     996      996       996         996
## 4        1      1        2           1       1        1         0           2
## 5        4      2        0           3      30        5         1           0
## 6      996    996      996         996     996      996       996         996
##   COFETEAMNO POTATONO PIZZANO HRSLEEP CVDSHT
## 1          0        3       2       8      1
## 2          1        1       1       6      2
## 3        996      996     996       0      2
## 4          0        1       1       6      2
## 5         30        6       2       8      2
## 6        996      996     996       0      1
```

```
summary(data)
```

```
##       YEAR          SERIAL          STRATA           PSU        
##  Min.   :2022   Min.   :    1   Min.   :100.0   Min.   :  1.00  
##  1st Qu.:2022   1st Qu.: 7184   1st Qu.:112.0   1st Qu.:  8.00  
##  Median :2022   Median :14403   Median :126.0   Median : 23.00  
##  Mean   :2022   Mean   :14419   Mean   :125.8   Mean   : 30.94  
##  3rd Qu.:2022   3rd Qu.:21648   3rd Qu.:140.0   3rd Qu.: 48.00  
##  Max.   :2022   Max.   :28854   Max.   :151.0   Max.   :153.00  
##    NHISHID              REGION          PERNUM         NHISPID         
##  Length:35115       Min.   :1.000   Min.   :1.000   Length:35115      
##  Class :character   1st Qu.:2.000   1st Qu.:1.000   Class :character  
##  Mode  :character   Median :3.000   Median :1.000   Mode  :character  
##                     Mean   :2.712   Mean   :1.178                     
##                     3rd Qu.:4.000   3rd Qu.:1.000                     
##                     Max.   :4.000   Max.   :2.000                     
##      HHX              SAMPWEIGHT       ASTATFLG         CSTATFLG     
##  Length:35115       Min.   :  740   Min.   :0.0000   Min.   :0.0000  
##  Class :character   1st Qu.: 5095   1st Qu.:1.0000   1st Qu.:0.0000  
##  Mode  :character   Median : 7947   Median :1.0000   Median :0.0000  
##                     Mean   : 9343   Mean   :0.7874   Mean   :0.2126  
##                     3rd Qu.:11777   3rd Qu.:1.0000   3rd Qu.:0.0000  
##                     Max.   :43112   Max.   :1.0000   Max.   :1.0000  
##       AGE             SEX           MARSTCUR          EDUC      
##  Min.   :  0.0   Min.   :1.000   Min.   :0.000   Min.   :  0.0  
##  1st Qu.: 23.0   1st Qu.:1.000   1st Qu.:1.000   1st Qu.:103.0  
##  Median : 45.0   Median :2.000   Median :1.000   Median :301.0  
##  Mean   : 45.3   Mean   :1.532   Mean   :3.349   Mean   :248.4  
##  3rd Qu.: 65.0   3rd Qu.:2.000   3rd Qu.:6.000   3rd Qu.:400.0  
##  Max.   :999.0   Max.   :9.000   Max.   :9.000   Max.   :999.0  
##     HOURSWRK        POVERTY          HEIGHT          WEIGHT     
##  Min.   : 0.00   Min.   :11.00   Min.   : 0.00   Min.   :  0.0  
##  1st Qu.: 0.00   1st Qu.:24.00   1st Qu.:62.00   1st Qu.:130.0  
##  Median : 0.00   Median :33.00   Median :66.00   Median :165.0  
##  Mean   :17.64   Mean   :30.28   Mean   :60.72   Mean   :215.2  
##  3rd Qu.:40.00   3rd Qu.:37.00   3rd Qu.:70.00   3rd Qu.:203.0  
##  Max.   :99.00   Max.   :37.00   Max.   :99.00   Max.   :999.0  
##     BMICALC        HINOTCOVE        CANCEREV       CHEARTDIEV    
##  Min.   : 11.5   Min.   :1.000   Min.   :0.000   Min.   :0.0000  
##  1st Qu.: 24.0   1st Qu.:1.000   1st Qu.:1.000   1st Qu.:1.0000  
##  Median : 28.3   Median :1.000   Median :1.000   Median :1.0000  
##  Mean   :215.8   Mean   :1.093   Mean   :0.892   Mean   :0.8533  
##  3rd Qu.: 36.8   3rd Qu.:1.000   3rd Qu.:1.000   3rd Qu.:1.0000  
##  Max.   :996.0   Max.   :9.000   Max.   :9.000   Max.   :9.0000  
##    DIABETICEV      HEARTATTEV        STROKEV         ALCANYNO       ALCDAYSYR  
##  Min.   :1.000   Min.   :0.0000   Min.   :0.000   Min.   :  0.0   Min.   :  0  
##  1st Qu.:1.000   1st Qu.:1.0000   1st Qu.:1.000   1st Qu.:  1.0   1st Qu.:  6  
##  Median :1.000   Median :1.0000   Median :1.000   Median :  3.0   Median :104  
##  Mean   :1.092   Mean   :0.8247   Mean   :0.824   Mean   :330.5   Mean   :372  
##  3rd Qu.:1.000   3rd Qu.:1.0000   3rd Qu.:1.000   3rd Qu.:996.0   3rd Qu.:996  
##  Max.   :9.000   Max.   :9.0000   Max.   :9.000   Max.   :999.0   Max.   :999  
##     CIGDAYMO       MOD10DMIN       VIG10DMIN          FRUTNO     
##  Min.   : 0.00   Min.   :  0.0   Min.   :  0.00   Min.   :  0.0  
##  1st Qu.:96.00   1st Qu.:  0.0   1st Qu.:  0.00   1st Qu.:  1.0  
##  Median :96.00   Median : 20.0   Median :  0.00   Median :  3.0  
##  Mean   :94.36   Mean   : 34.6   Mean   : 16.59   Mean   :245.5  
##  3rd Qu.:96.00   3rd Qu.: 45.0   3rd Qu.: 15.00   3rd Qu.: 30.0  
```

```
##  Max.   :99.00   Max.   :999.0   Max.   :999.00   Max.   :999.0  
##      VEGENO         JUICEMNO        SALADSNO         BEANNO     
##  Min.   :  0.0   Min.   :  0.0   Min.   :  0.0   Min.   :  0.0  
##  1st Qu.:  1.0   1st Qu.:  0.0   1st Qu.:  1.0   1st Qu.:  1.0  
##  Median :  3.0   Median :  1.0   Median :  3.0   Median :  2.0  
##  Mean   :247.2   Mean   :244.1   Mean   :245.1   Mean   :245.6  
##  3rd Qu.: 30.0   3rd Qu.: 30.0   3rd Qu.: 30.0   3rd Qu.: 20.0  
##  Max.   :999.0   Max.   :999.0   Max.   :999.0   Max.   :999.0  
##     SALSAMNO       TOMSAUCEMNO        SODAPNO          FRIESPNO    
##  Min.   :  0.0   Min.   :  0.0   Min.   :  0.0   Min.   :  0.0  
##  1st Qu.:  0.0   1st Qu.:  1.0   1st Qu.:  0.0   1st Qu.:  1.0  
##  Median :  2.0   Median :  2.0   Median :  1.0   Median :  2.0  
##  Mean   :245.3   Mean   :246.5   Mean   :243.1   Mean   :244.6  
##  3rd Qu.: 20.0   3rd Qu.: 15.0   3rd Qu.: 30.0   3rd Qu.: 20.0  
##  Max.   :999.0   Max.   :999.0   Max.   :999.0   Max.   :999.0  
##    SPORDRMNO       FRTDRINKMNO      COFETEAMNO        POTATONO    
##  Min.   :  0.0   Min.   :  0.0   Min.   :  0.0   Min.   :  0.0  
##  1st Qu.:  0.0   1st Qu.:  0.0   1st Qu.:  0.0   1st Qu.:  1.0  
##  Median :  0.0   Median :  0.0   Median :  1.0   Median :  2.0  
##  Mean   :242.3   Mean   :242.5   Mean   :243.2   Mean   :245.7  
##  3rd Qu.: 15.0   3rd Qu.: 10.0   3rd Qu.: 30.0   3rd Qu.: 20.0  
##  Max.   :999.0   Max.   :999.0   Max.   :999.0   Max.   :999.0  
##     PIZZANO          HRSLEEP           CVDSHT     
##  Min.   :  0.0   Min.   : 0.000   Min.   :0.000  
##  1st Qu.:  1.0   1st Qu.: 5.000   1st Qu.:1.000  
##  Median :  2.0   Median : 7.000   Median :2.000  
##  Mean   :244.8   Mean   : 8.135   Mean   :1.791  
##  3rd Qu.: 10.0   3rd Qu.: 8.000   3rd Qu.:2.000  
##  Max.   :999.0   Max.   :99.000   Max.   :9.000
```

```
str(data)
```

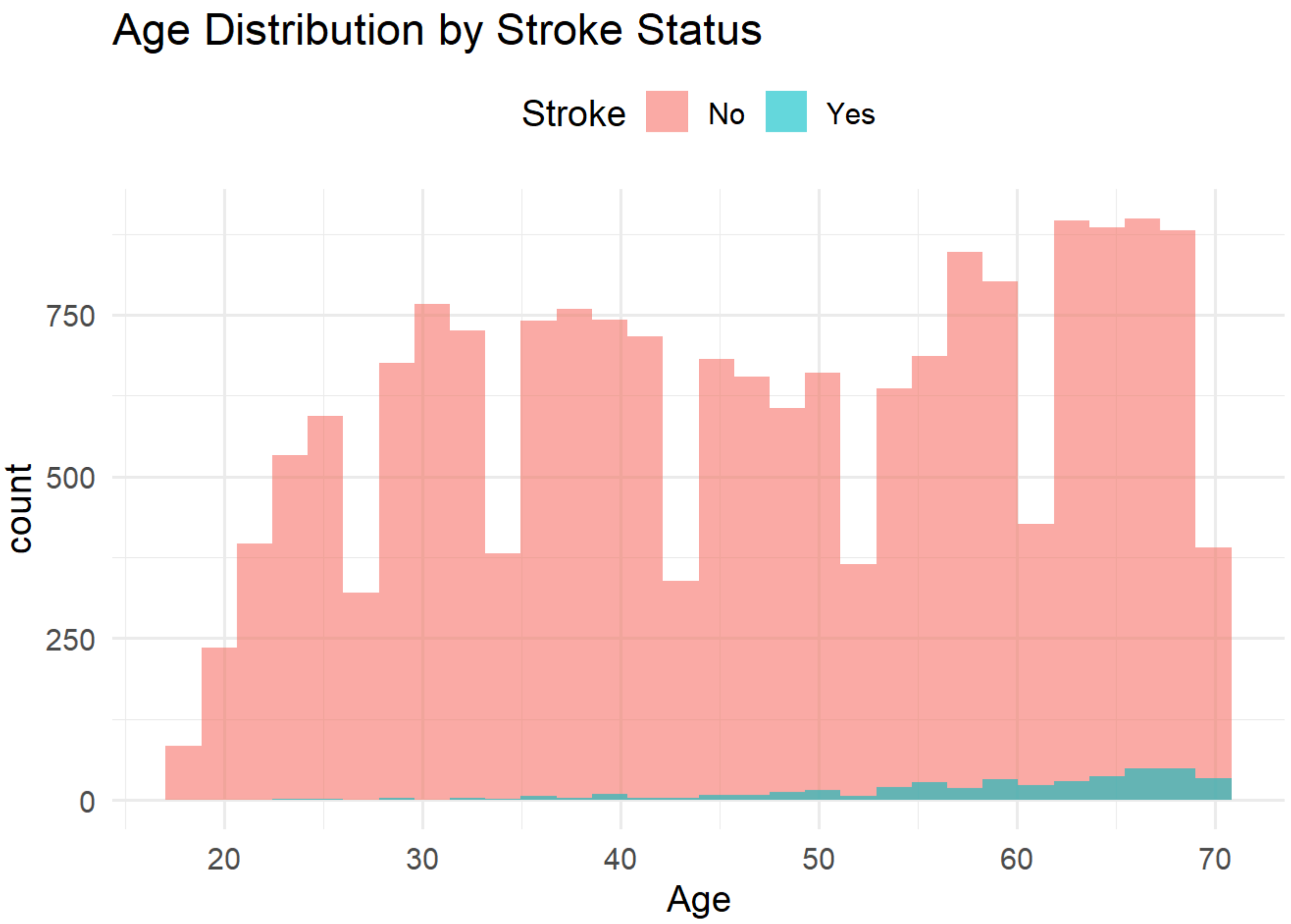
```
## 'data.frame':	35115 obs. of  48 variables:
##  $ YEAR       : int  2022 2022 2022 2022 2022 2022 2022 2022 2022 2022 ...
##  $ SERIAL     : int  1 2 2 3 4 4 5 6 7 8 ...
##  $ STRATA     : int  143 106 106 134 106 106 127 111 143 105 ...
##  $ PSU        : int  16 53 53 13 53 53 26 11 14 61 ...
##  $ NHISHID    : chr  "0002022H000001" "0002022H000003" "0002022H000003" "0002022H000006"
...
##  $ REGION     : int  4 3 3 2 3 3 2 4 4 1 ...
##  $ PERNUM     : int  1 1 2 1 1 2 1 1 1 1 ...
##  $ NHISPID    : chr  "0002022H00000110" "0002022H00000310" "0002022H00000320" "0002022H000
00610" ...
##  $ HHX        : chr  "H000001" "H000003" "H000003" "H000006" ...
##  $ SAMPWEIGHT : num  8018 10117 7933 2681 10233 ...
##  $ ASTATFLG   : int  1 1 0 1 1 0 1 1 1 1 ...
##  $ CSTATFLG   : int  0 0 1 0 0 1 0 0 0 0 ...
##  $ AGE        : int  61 43 12 68 73 16 73 21 59 67 ...
##  $ SEX        : int  1 1 2 1 1 2 1 1 1 2 ...
##  $ MARSTCUR   : int  1 1 0 5 1 0 1 8 7 7 ...
##  $ EDUC       : int  201 301 0 505 201 0 201 303 201 400 ...
##  $ HOURSWRK   : int  45 45 0 0 0 0 0 0 0 6 ...
##  $ POVERTY    : int  34 37 37 31 32 32 36 23 33 37 ...
##  $ HEIGHT     : int  69 70 60 75 71 65 71 68 68 63 ...
##  $ WEIGHT     : int  260 190 96 200 172 106 190 200 175 169 ...
##  $ BMICALC    : num  38.4 27.3 18.7 25 24 17.6 26.5 30.4 26.6 29.9 ...
##  $ HINOTCOVE  : int  1 1 1 1 1 1 1 9 2 1 ...
##  $ CANCEREV   : int  1 1 0 1 1 0 1 1 2 2 ...
##  $ CHEARTDIEV : int  1 1 0 1 1 0 2 1 1 2 ...
##  $ DIABETICEV : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ HEARTATTEV : int  1 1 0 1 1 0 1 1 1 2 ...
##  $ STROKEV    : int  1 1 0 1 1 0 1 1 1 1 ...
##  $ ALCANYNO   : int  2 1 996 7 0 996 2 996 4 997 ...
##  $ ALCDAYSYR  : int  104 52 996 364 0 996 2 996 4 997 ...
##  $ CIGDAYMO   : int  96 96 96 96 96 96 96 96 96 30 ...
##  $ MOD10DMIN  : int  0 20 0 60 690 0 60 45 15 120 ...
##  $ VIG10DMIN  : int  0 0 0 0 0 0 0 45 0 0 ...
##  $ FRUTNO     : int  5 1 996 3 2 996 1 0 1 1 ...
##  $ VEGENO     : int  15 1 996 1 4 996 2 2 0 1 ...
##  $ JUICEMNO   : int  0 1 996 0 0 996 10 3 0 0 ...
##  $ SALADSNO   : int  10 1 996 1 4 996 5 2 1 3 ...
##  $ BEANNO     : int  5 1 996 1 2 996 0 2 0 3 ...
##  $ SALSAMNO   : int  5 1 996 2 0 996 0 1 2 1 ...
##  $ TOMSAUCEMNO: int  2 1 996 1 3 996 4 0 1 3 ...
##  $ SODAPNO    : int  0 0 996 1 30 996 5 2 0 0 ...
##  $ FRIESPNO   : int  110 1 996 1 5 996 3 0 4 0 ...
##  $ SPORDRMNO  : int  3 0 996 0 1 996 3 0 1 5 ...
##  $ FRTDRINKMNO: int  0 0 996 2 0 996 3 0 0 0 ...
##  $ COFETEAMNO : int  0 1 996 0 30 996 0 0 0 1 ...
##  $ POTATONO   : int  3 1 996 1 6 996 1 0 3 2 ...
##  $ PIZZANO    : int  2 1 996 1 2 996 1 1 1 3 ...
##  $ HRSLEEP    : int  8 6 0 6 8 0 6 9 9 8 ...
##  $ CVDSHT     : int  1 2 2 2 2 1 2 2 1 2 ...
```

```r
# Subsetting the data, taking only adults, i.e., between the ages 18 and 70.
# Converting numeric variables to categoric factors.
data <- data %>%
  filter(AGE >= 18, AGE <= 70, STROKEV %in% c(1,2)) %>%
  mutate(
    Sex = factor(SEX, levels = c(1,2), labels = c("Male","Female")),
    Stroke = factor(STROKEV, levels = c(1,2), labels = c("No","Yes"))
  )
```

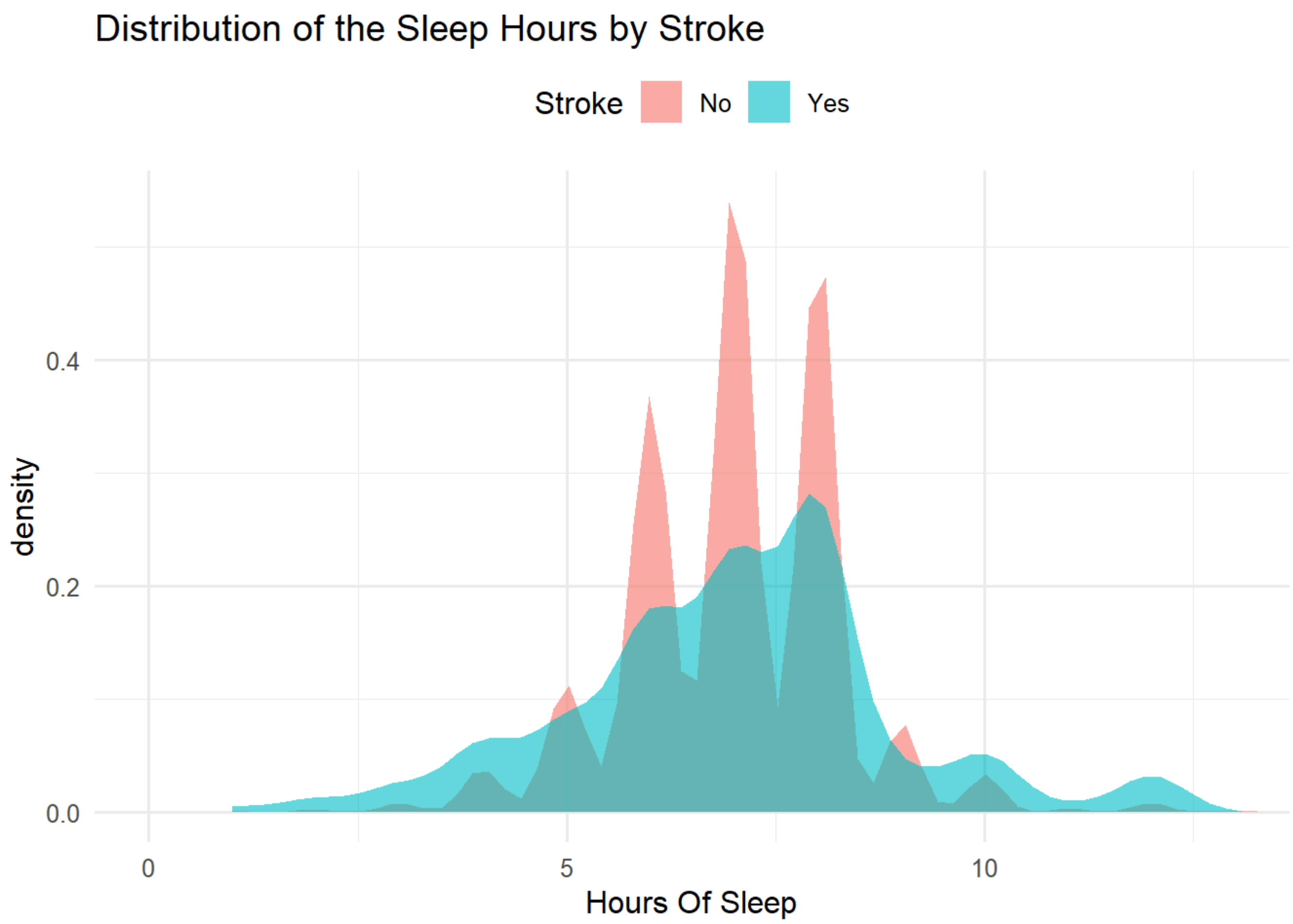
```r
# Renaming variables to clear column names.
names(data)[names(data)=="AGE"] <- "Age"
names(data)[names(data)=="HRSLEEP"] <- "Hours Of Sleep"
names(data)[names(data)=="HOURSWRK"] <- "Hours Worked"
names(data)[names(data)=="ALCDAYSYR"] <- "Alcohol Consumption Days Per Year"
names(data)[names(data)=="CIGDAYMO"] <- "Cigarettes Consumed Per Month"
names(data)[names(data)=="MOD10DMIN"] <- "Duration Of Moderate Activity(in mins)"
names(data)[names(data)=="VIG10DMIN"] <- "Duration Of Vigorous Activity(in mins)"
```

```r
# Predicting the stroke status (Yes/No) in adults (18–70)
# using SVMs on predictors:
# Age, Sex, Hours Of Sleep, Hours Worked,Alcohol Consumption Days
# Cigarettes/Month, Moderate & Vigorous Activity (mins).



# Cleaning the invalid codes and then replace those with NA,
#lastly, drop those null values.
codes <- c(996, 997, 998, 999)
variables <- c("Age", "Hours Of Sleep", "Hours Worked",
               "Alcohol Consumption Days Per Year",
               "Cigarettes Consumed Per Month",
               "Duration Of Moderate Activity(in mins)",
               "Duration Of Vigorous Activity(in mins)")
# Keeping both Moderate and Vigorous activity
#since they are not highly correlated.
data <- data %>%
  mutate(across(all_of(variables), ~ ifelse(. %in% codes, NA, .))) %>%
  na.omit()
```

```r
#Exploratory Data Analysis
# Histogram of Age distribution by Age
ggplot(data, aes(x = Age, fill = Stroke)) +
  geom_histogram(position = "identity", alpha = 0.6, bins = 30) +
  labs(
    title = "Age Distribution by Stroke Status",
    fill  = "Stroke"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    legend.position   = "top"
  )
```




```
# A density plot of sleeping hours by Stroke
ggplot(data, aes(x = `Hours Of Sleep`, fill = Stroke)) +
  geom_density(alpha = 0.6, color = NA) +
  coord_cartesian(xlim = c(0, 13)) +
  labs(
    title = "Distribution of the Sleep Hours by Stroke",
    fill  = "Stroke"
  ) +
  theme_minimal(base_size = 12) +
  theme(
    legend.position  = "top"
  )
```

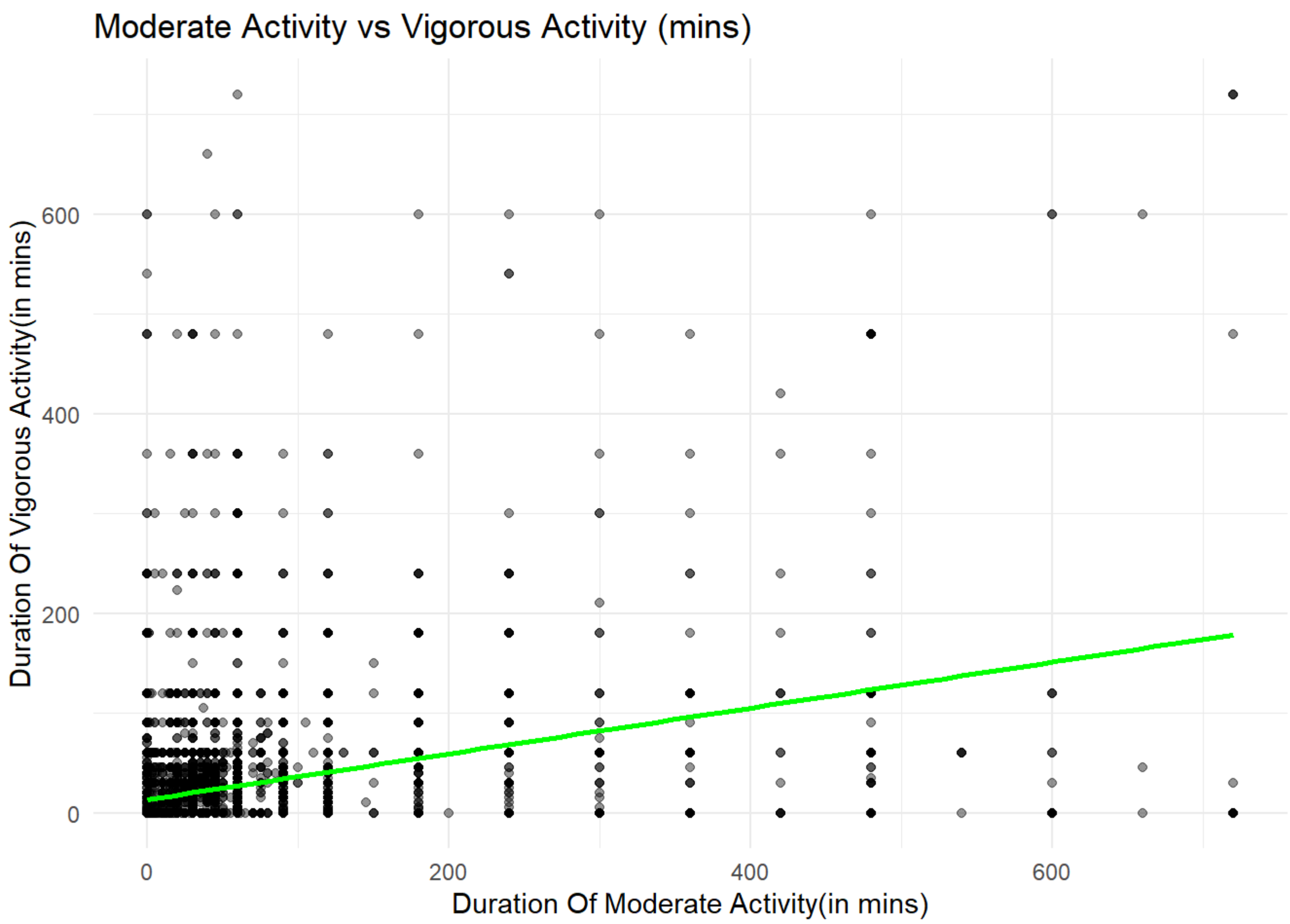



```
# Computing correlation between moderate & vigorous activity
correlation <- cor(
  data$`Duration Of Moderate Activity(in mins)`,
  data$`Duration Of Vigorous Activity(in mins)`
)
print(paste("Correlation (r) =", round(correlation, 5)))
```

```
## [1] "Correlation (r) = 0.31152"
```

```
# Plotting the above
ggplot(data, aes(
    x = `Duration Of Moderate Activity(in mins)`,
    y = `Duration Of Vigorous Activity(in mins)`
  )) +
  geom_point(alpha = 0.4) +
  geom_smooth(method = "lm", se = FALSE,
              color = "green") +
  labs(
    title = "Moderate Activity vs Vigorous Activity (mins)",
  ) +
  theme_minimal()
```

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




```
# Scaling the variables variables
data[variables] <- scale(data[variables])

# Splitting the data into train and test sets.
set.seed(42)
train_data <- createDataPartition(data$Stroke, p = 0.7, list = FALSE)
train_set <- data[train_data, ]
test_set <- data[-train_data, ]

# Increased 'Yes' class weight to 50
#to force the model to better recognize
#minority class during training.
# Chosen this setting of balance, to address the imbalance
# while avoiding extremes. This ratio improved
# stroke detection without excessive false alarms.
weights <- c("No" = 1, "Yes" = 50)
```

```
#PART 1: Linear SVM
set.seed(42)
tune_one <- tune(svm,
                 Stroke ~ `Age` + Sex + `Hours Of Sleep` + `Hours Worked`
                 + `Alcohol Consumption Days Per Year`
                 + `Cigarettes Consumed Per Month`
                 + `Duration Of Moderate Activity(in mins)`
                 + `Duration Of Vigorous Activity(in mins)`,
                 data          = train_set, kernel        = "linear",
                 ranges        = list(cost = c(0.01, 0.1, 1)),
                 class.weights = weights)
```

Usingclassweightstohandleimbalancewithoutdroppingdata

```
# Choosing the best linear svm from tuning
svm_one <- tune_one$best.model
svm_one
```

```
##
## Call:
## best.tune(METHOD = svm, train.x = Stroke ~ Age + Sex + `Hours Of Sleep` +
##     `Hours Worked` + `Alcohol Consumption Days Per Year` + `Cigarettes Consumed Per Month`
+
##     `Duration Of Moderate Activity(in mins)` + `Duration Of Vigorous Activity(in mins)`,
##     data = train_set, ranges = list(cost = c(0.01, 0.1, 1)), kernel = "linear",
##     class.weights = weights)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  8739
```

```
# Prediction and evaluation metrics
prediction_one <- predict(svm_one, test_set)
confusion_mat_one <- confusionMatrix(prediction_one, test_set$Stroke)
print(confusion_mat_one)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  3846   35
##        Yes 1650   89
##
##                Accuracy : 0.7002
##                  95% CI : (0.688, 0.7121)
##     No Information Rate : 0.9779
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0567
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.69978
##             Specificity : 0.71774
##          Pos Pred Value : 0.99098
##          Neg Pred Value : 0.05118
##              Prevalence : 0.97794
##          Detection Rate : 0.68434
##    Detection Prevalence : 0.69057
##       Balanced Accuracy : 0.70876
##
##        'Positive' Class : No
##
```

```
precision_one <- posPredValue(prediction_one, test_set$Stroke, positive = "Yes")
recall_one <- sensitivity(prediction_one,  test_set$Stroke, positive = "Yes")
f1_one <- 2 * (precision_one * recall_one) / (precision_one + recall_one)
accuracy_one <- confusion_mat_one$overall["Accuracy"]
precision_one
```

```
## [1] 0.05117884
```

```
recall_one
```

```
## [1] 0.7177419
```

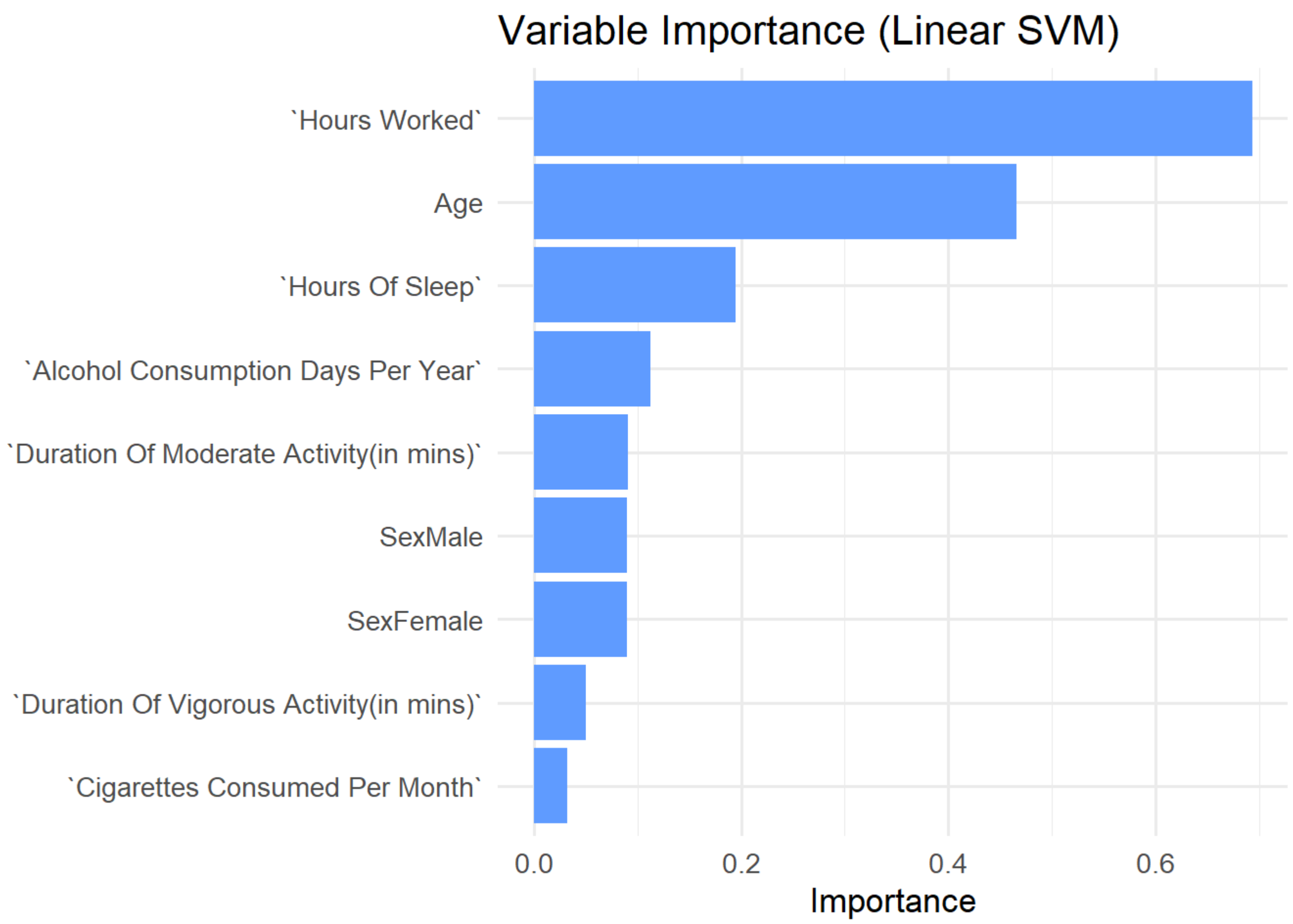
```
f1_one
```

```
## [1] 0.09554482
```

```
accuracy_one
```

```
##  Accuracy
## 0.7001779
```

```
# Extracting variable importance from the fitted linear SVM model
weight_vector <- as.numeric(t(svm_one$coefs) %*% svm_one$SV)
names(weight_vector) <- colnames(svm_one$SV)

#'importance_df' consists ofabsolute weights and sort them in decreasing order
importance_df <- data.frame(
  Variable = names(weight_vector),
  Importance = abs(weight_vector)
)
importance_df <- importance_df[order(importance_df$Importance, decreasing = TRUE), ]

#Plotting the variable importance
ggplot(importance_df, aes(
  x = reorder(Variable, Importance),
  y = Importance
)) +
  geom_col(fill = "#619CFF") +
  coord_flip() +
  labs(
    title = "Variable Importance (Linear SVM)",
    x     = NULL,
    y     = "Importance"
  ) +
  theme_minimal(base_size = 13)
```




```
ggsave("variable_imp.png", width = 5, height = 5, dpi = 350, bg = "white")
```

Evaluated model with confusion matrix (Accuracy is 70%)

```r
#PART 2: Radial SVM
tune_two <- tune(svm,
                 Stroke ~ `Age` + Sex + `Hours Of Sleep` + `Hours Worked`
                 + `Alcohol Consumption Days Per Year`
                 + `Cigarettes Consumed Per Month`
                 + `Duration Of Moderate Activity(in mins)`
                 + `Duration Of Vigorous Activity(in mins)`,
                 data          = train_set,
                 kernel        = "radial",
                 ranges        = list(cost = c(0.1, 1), gamma = c(0.01, 0.1)),
                 class.weights = weights)
```

```r
# Choosing the best radial svm from tuning
svm_two <- tune_two$best.model
svm_two
```

```
##
## Call:
## best.tune(METHOD = svm, train.x = Stroke ~ Age + Sex + `Hours Of Sleep` +
##     `Hours Worked` + `Alcohol Consumption Days Per Year` + `Cigarettes Consumed Per Month`
+
##     `Duration Of Moderate Activity(in mins)` + `Duration Of Vigorous Activity(in mins)`,
##     data = train_set, ranges = list(cost = c(0.1, 1), gamma = c(0.01,
##         0.1)), kernel = "radial", class.weights = weights)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  7352
```

```r
# Prediction and evaluation metrics
prediction_two <- predict(svm_two,    test_set)
confusion_mat_two <- confusionMatrix(prediction_two, test_set$Stroke)
print(confusion_mat_two)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  3896   36
##        Yes 1600   88
##
##                Accuracy : 0.7089
##                  95% CI : (0.6968, 0.7208)
##     No Information Rate : 0.9779
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0584
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.70888
##             Specificity : 0.70968
##          Pos Pred Value : 0.99084
##          Neg Pred Value : 0.05213
##              Prevalence : 0.97794
##          Detection Rate : 0.69324
##    Detection Prevalence : 0.69964
##       Balanced Accuracy : 0.70928
##
##        'Positive' Class : No
##
```

```
precision_two <- posPredValue(prediction_two, test_set$Stroke, positive = "Yes")
recall_two<- sensitivity(prediction_two,  test_set$Stroke, positive = "Yes")
f1_two<- 2 * (precision_two * recall_two) / (precision_two + recall_two)
accuracy_two<- confusion_mat_two$overall["Accuracy"]
precision_two
```

```
## [1] 0.0521327
```

```
recall_two
```

```
## [1] 0.7096774
```

```
f1_two
```

```
## [1] 0.09713024
```

```
accuracy_two
```

```
##  Accuracy
## 0.7088968
```

Radial SVM achieved is equal to 70% accuracy on test set.

```
#PART 3: Polynomial SVM
tune_three <- tune(svm,
                   Stroke ~ `Age` + Sex + `Hours Of Sleep` + `Hours Worked`
                   + `Alcohol Consumption Days Per Year`
                   + `Cigarettes Consumed Per Month`
                   + `Duration Of Moderate Activity(in mins)`
                   + `Duration Of Vigorous Activity(in mins)`,
                   data          = train_set, kernel        = "polynomial",
                   ranges        = list(cost   = c(0.1,1), degree = c(3,4),
                                        coef0  = c(0.5,1)),
                   class.weights = weights)
```

```
# Choosing the best polynomial svm from tuning
svm_three <- tune_three$best.model
svm_three
```

```
##
## Call:
## best.tune(METHOD = svm, train.x = Stroke ~ Age + Sex + `Hours Of Sleep` +
##     `Hours Worked` + `Alcohol Consumption Days Per Year` + `Cigarettes Consumed Per Month`
+
##     `Duration Of Moderate Activity(in mins)` + `Duration Of Vigorous Activity(in mins)`,
##     data = train_set, ranges = list(cost = c(0.1, 1), degree = c(3,
##         4), coef0 = c(0.5, 1)), kernel = "polynomial", class.weights = weights)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  1
##      degree:  4
##      coef.0:  0.5
##
## Number of Support Vectors:  6850
```

```
# Prediction and evaluation metrics
prediction_three<- predict(svm_three,    test_set)
confusion_mat_three  <- confusionMatrix(prediction_three, test_set$Stroke)
print(confusion_mat_three)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  3996   36
##        Yes 1500   88
##
##                Accuracy : 0.7267
##                  95% CI : (0.7148, 0.7383)
##     No Information Rate : 0.9779
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0645
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.72707
##             Specificity : 0.70968
##          Pos Pred Value : 0.99107
##          Neg Pred Value : 0.05542
##              Prevalence : 0.97794
##          Detection Rate : 0.71103
##    Detection Prevalence : 0.71744
##       Balanced Accuracy : 0.71838
##
##        'Positive' Class : No
##
```

```
precision_three<- posPredValue(prediction_three, test_set$Stroke, positive = "Yes")
recall_three<- sensitivity(prediction_three,  test_set$Stroke, positive = "Yes")
f1_three <- 2 * (precision_three * recall_three) / (precision_three + recall_three)
accuracy_three  <- confusion_mat_three$overall["Accuracy"]
precision_three
```

```
## [1] 0.05541562
```

```
recall_three
```

```
## [1] 0.7096774
```

```
f1_three
```

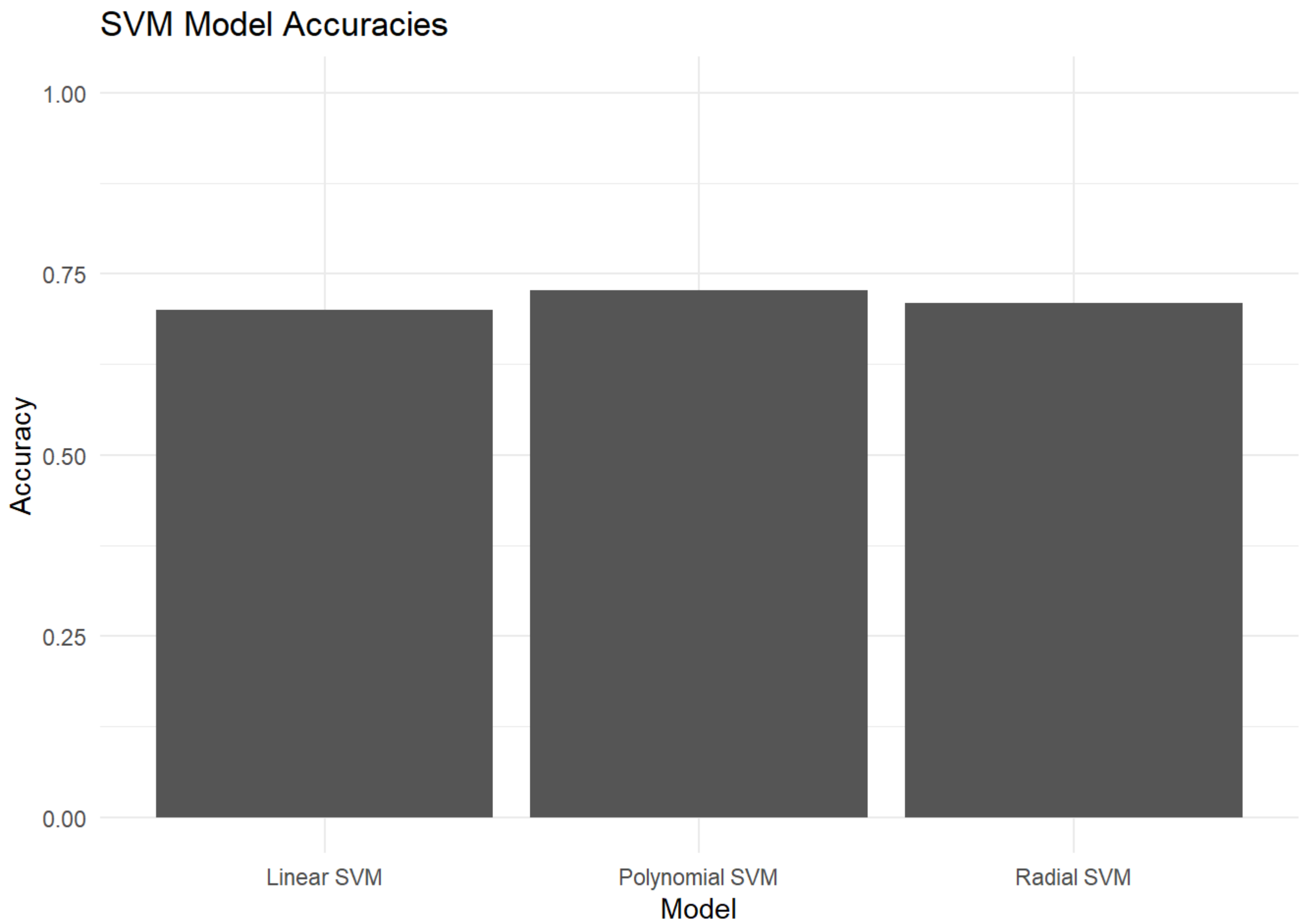
```
## [1] 0.1028037
```

```
accuracy_three
```

```
##  Accuracy
## 0.7266904
```

Polynomial SVM achieved is close to 73% accuracy on test set.

```
# Comparing the accuarcy results by plotting
model_results <- data.frame(
  Model = c("Linear SVM", "Radial SVM", "Polynomial SVM"),
  Accuracy = c(accuracy_one, accuracy_two, accuracy_three)
)

ggplot(model_results, aes(x = Model, y = Accuracy)) +
  geom_col() +
  ylim(0, 1) +
  labs(title = "SVM Model Accuracies", x = "Model", y = "Accuracy") +
  theme_minimal()
```




```
# creating the model evaluation table
evaluation_table <- data.frame(
  Model     = c("Linear SVM", "Radial SVM", "Polynomial SVM"),
  Accuracy  = c(accuracy_one, accuracy_two, accuracy_three),
  Precision = c(precision_one, precision_two, precision_three),
  Recall    = c(recall_one, recall_two, recall_three),
  F1_Score  = c(f1_one, f1_two, f1_three)
)
# rounding off the numeric values to present
evaluation_table <- evaluation_table %>%
  mutate(across(where(is.numeric), ~ round(., 3)))

print(evaluation_table)
```

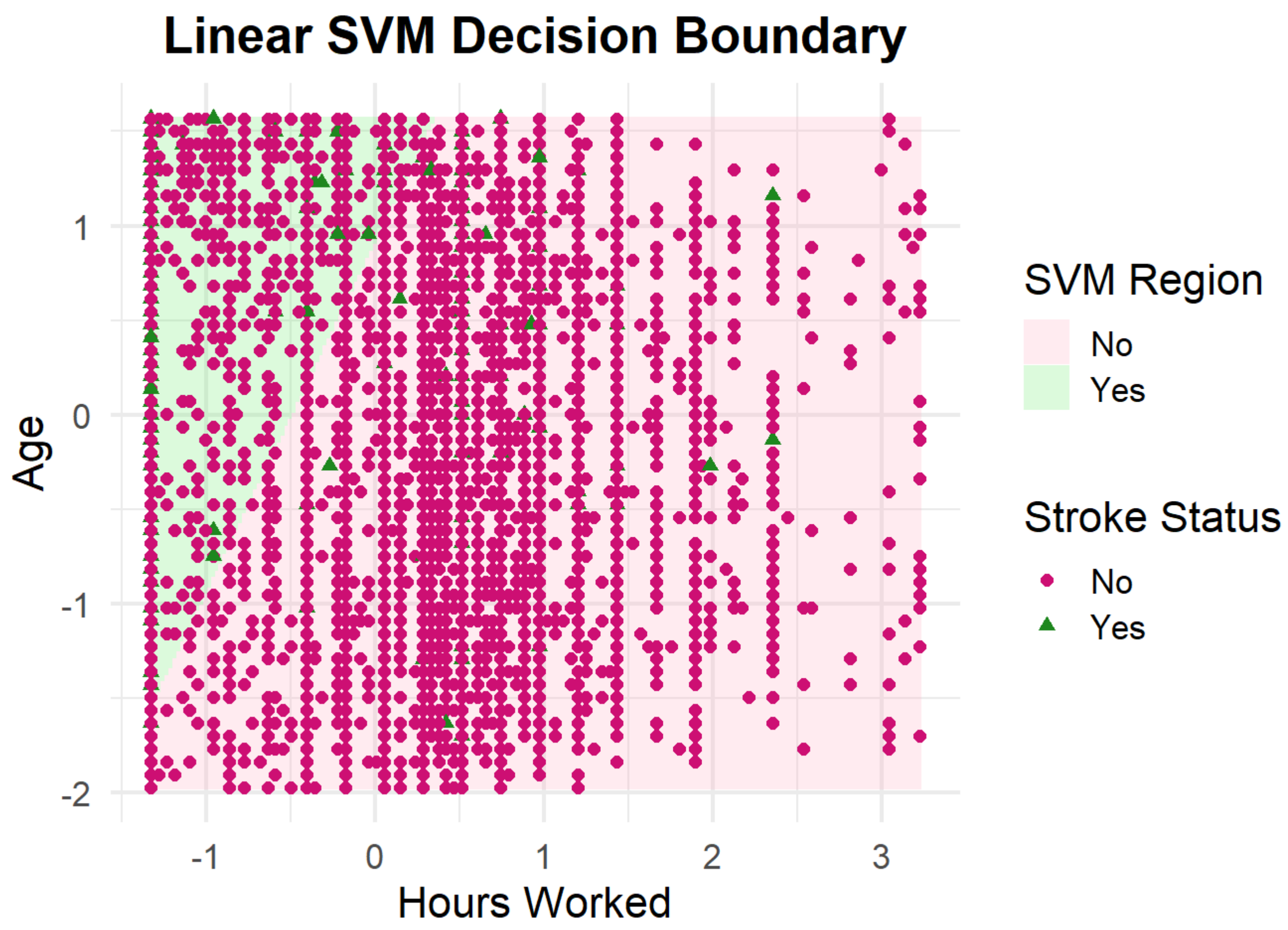
```
##            Model Accuracy Precision Recall F1_Score
## 1     Linear SVM    0.700     0.051  0.718    0.096
## 2     Radial SVM    0.709     0.052  0.710    0.097
## 3 Polynomial SVM    0.727     0.055  0.710    0.103
```

From the above plot of SVM accuracies comparison, polynomial model has performed the highest, about 73 % , radial performs next best , and then linear svm performs good, does purely linear seperation

```
# Subset training data with the two strong predictors
train_subset <- train_set %>%
  select(`Hours Worked`, Age, Stroke)
```

```
# Fitting the Linear SVM with the two predictors and one respone variables.
svm_four <- svm(Stroke ~ `Hours Worked` + Age,
                         data = train_subset,
                         kernel = "linear",
                         cost = 1,
                         scale = TRUE,
                         class.weights = weights)
```

```
# Creating a grid and then plotting
x_seq <- seq(min(train_subset$`Hours Worked`), max(train_subset$`Hours Worked`), length.out =
200)
y_seq <- seq(min(train_subset$Age), max(train_subset$Age), length.out = 200)
grid <- expand.grid(`Hours Worked` = x_seq, Age = y_seq)
grid$Prediction <- predict(svm_four, grid)

ggplot() +
  geom_tile(data = grid, aes(x = `Hours Worked`, y = Age, fill = Prediction), alpha = 0.3) +
  geom_point(data = train_subset, aes(x = `Hours Worked`, y = Age, shape = Stroke, color = St
roke), size = 2.0) +
  scale_fill_manual(values = c("No" = "#FFC0CB", "Yes" = "#90EE90")) +
  scale_color_manual(values = c("No" = "deeppink3", "Yes" = "forestgreen")) +
  labs(title = "Linear SVM Decision Boundary",
       x = "Hours Worked",
       y = "Age",
       fill = "SVM Region",
       color = "Stroke Status",
       shape = "Stroke Status") +
  theme_minimal(base_size = 16) +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold"),
    legend.position = "right"
  )
```




```
ggsave("linear_svm2.png", width = 4, height = 4, dpi = 350, bg = "white")

#From the above plots, polynomial model has performed the highest,
#about 73 % , radial performs next best ,
#and then linear svm performs good, does purely linear separation.


#RESULTS: The linear SVM, with best cost 1 , has 8739 support vectors,
#suggesting high complexity, and it has an accuracy of 70%.
#It is poor at detecting strokes, precision is 5.1%;

#The radial SVM, with best cost 1 and gamma 0.1 ,
#has 7352 support vectors,so its efiicient,
#and it has an accuracy of 70.89%.
#It is slightly better than linear svm,
#but still poor at detecting strokes, precision is 5.21%;

#The polynomial SVM is the best among all,
#with best cost 1, degree 4, and coef0 0.5 , has 6850 support vectors,
#and it has an accuracy of 72.66%.
#It is still suffers from low precision like the rest of the two models,
#for "Yes" , but has better F1-score.

#The minority class is "yes" stroke and
#all the three models perform poorly in this one.
```