Logistic regression

Anastasia Chanbour

15/10/2021

Load the required packages

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.5     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

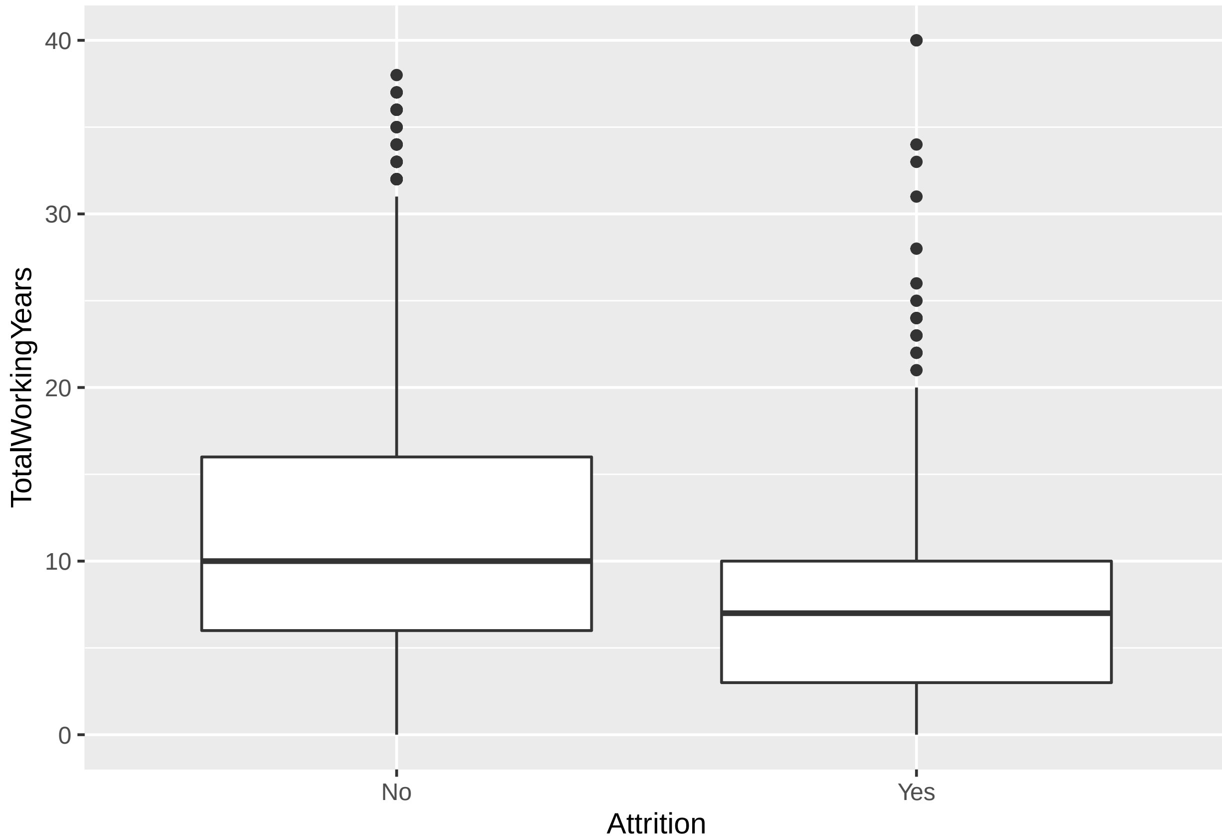
```
library(broom)
library(modeldata)
data(attrition)  #package with data
```

Aim: Classification with logistic regression

Part 1: Categorical outcome data

Comparing the distribution of a numeric predictor variable between the two outcome classes

```
ggplot(attrition, aes(x=Attrition, y = TotalWorkingYears))+
  geom_boxplot()
```



Testing for difference in means

```
t.test(TotalWorkingYears~Attrition, data = attrition)
```

```
##
##  Welch Two Sample t-test
##
## data:  TotalWorkingYears by Attrition
## t = 7.0192, df = 350.88, p-value = 1.16e-11
## alternative hypothesis: true difference in means between group No and group Yes is not equal to 0
## 95 percent confidence interval:
##  2.604401 4.632019
## sample estimates:
##  mean in group No mean in group Yes
##         11.862936          8.244726
```

Checking for class balance:

```
attrition %>% count(Attrition)
```

```
##   Attrition    n
## 1        No 1233
## 2       Yes  237
```

```
table(attrition$Attrition)
```

```
##
##   No  Yes
## 1233  237
```

Creating a balanced dataset with the same number of observations in both classes

Reason: Classifiers (such as Logistic Regression) tend to ignore small classes while concentrating on classifying the large ones accurately

```
attr_No <- attrition %>%
  filter(Attrition == "No") %>%
  sample_n(size = 237)

attr_Yes <- attrition %>%
  filter(Attrition == "Yes")

attr <- rbind(attr_No, attr_Yes)

# or

attr <- attrition %>%
  group_by(Attrition) %>%
  slice_sample(n = 237)


# transform outcome to numeric 0-1
nattr <- attr %>%
  mutate(Y = as.numeric(Attrition) - 1) %>%
  select(Y, TotalWorkingYears)
```

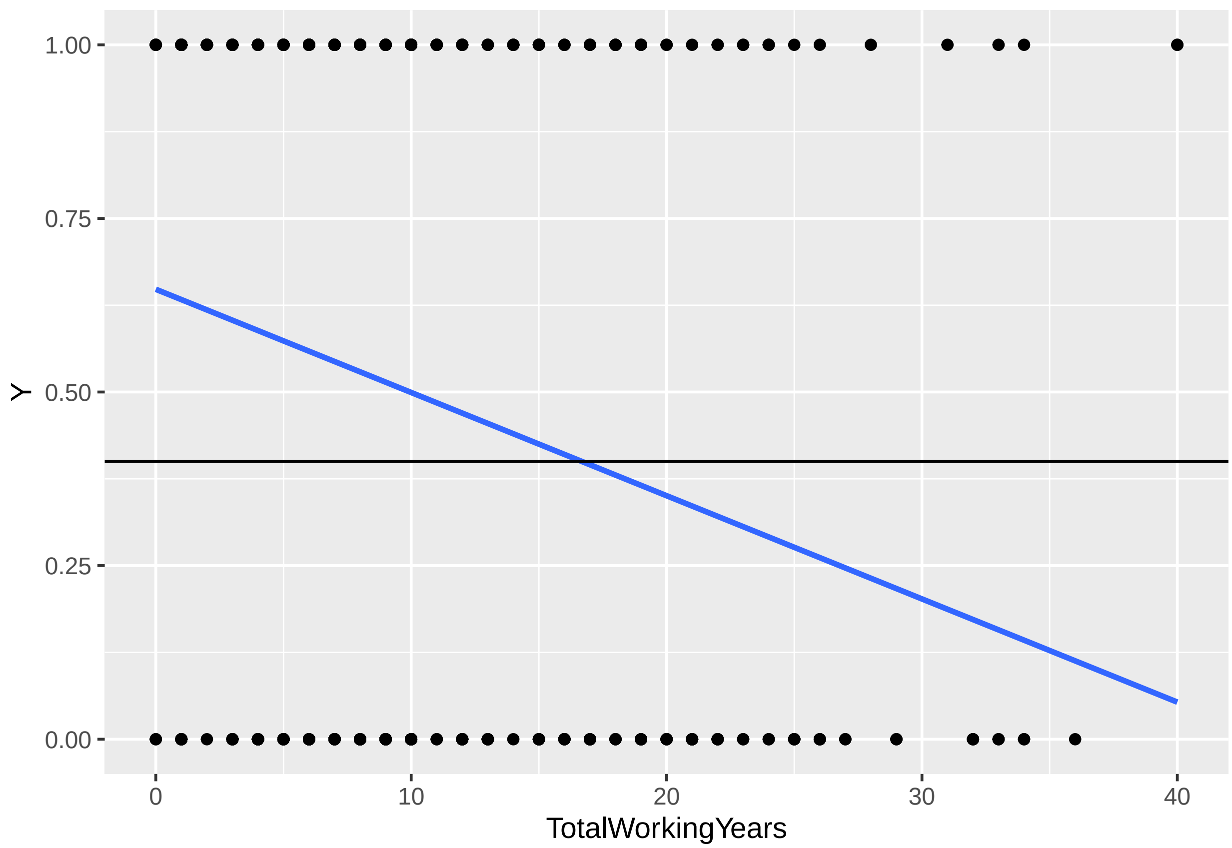
```
## Adding missing grouping variables: `Attrition`
```

Classification: linear regression with the linear probability model (LPM)?

Plotting linear regression line, change the threshold

```
ggplot(nattr, aes(x = TotalWorkingYears, y = Y)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  geom_hline(yintercept = .4) # or e.g. mean(nattr$Y)
```

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



Problems:

- Can predict outside 0-1 range
- Not directly interpretable as probabilities

Thresholding ideas

Choose a threshold/cutoff value for predictor X, say c, and then classify

- $\hat{Y} = 1$ if $X \geq c$
- $\hat{Y} = 0$ otherwise

Or if the association is negative, change the sign

As we vary c, we trade-off between kinds of errors: false positives and false negatives

In the simple case with thresholding one predictor, the classification/decision rules are all equivalent whether we use linear regression or logistic regression (as long as the fitted relationship is monotone)

For **multiple** regression–when we have more predictors–we can then transform a numeric prediction from the model $\hat{Y}$ to a classification by using a threshold rule on the scale of the predictions (instead of on the scale of one predictor as before)

- $\hat{Y} = 1$ if $x^T\hat{\beta} \geq c$
- $\hat{Y} = 0$ otherwise

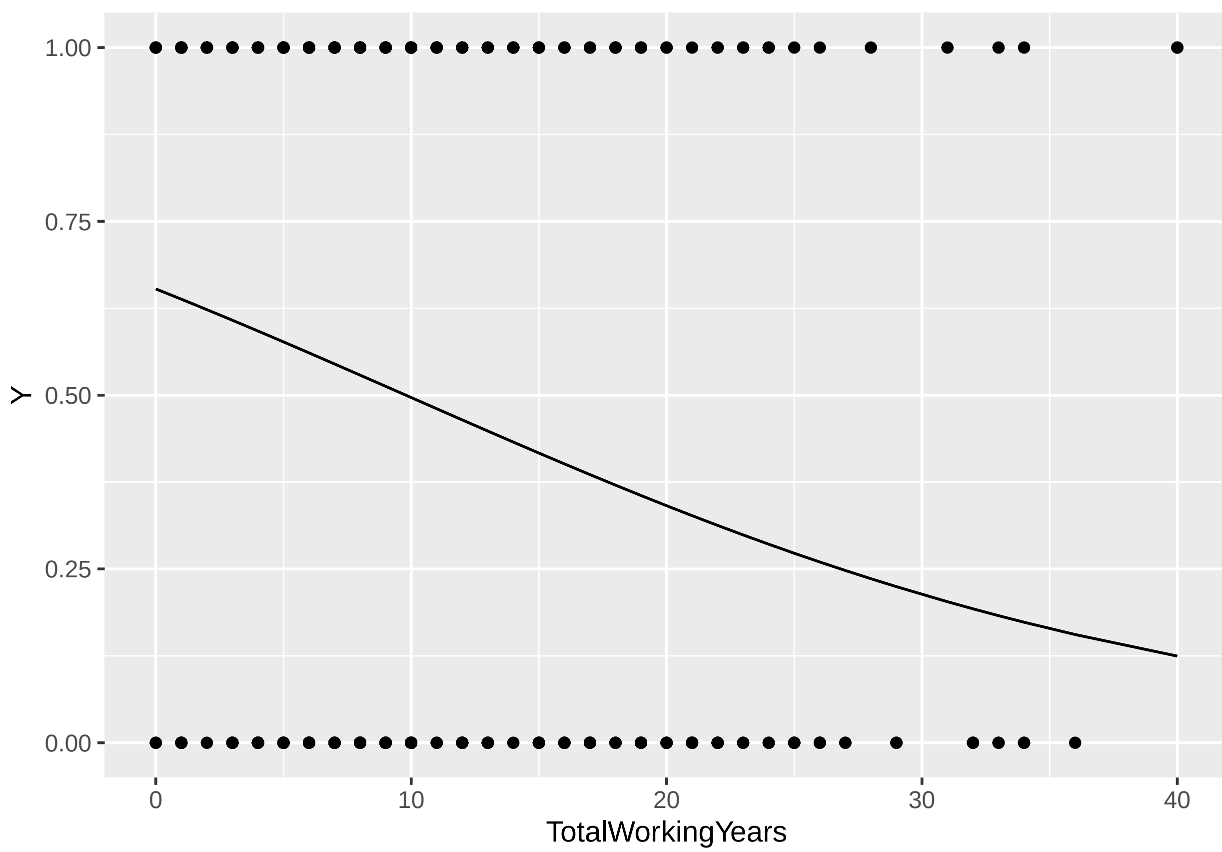
Logistic regression

```r
model_glm <- glm(Y~TotalWorkingYears,data = nattr, family = binomial)
model_glm
```

```
##
## Call:  glm(formula = Y ~ TotalWorkingYears, family = binomial, data = nattr)
##
## Coefficients:
##       (Intercept)  TotalWorkingYears
##           0.63147           -0.06451
##
## Degrees of Freedom: 473 Total (i.e. Null);  472 Residual
## Null Deviance:       657.1
## Residual Deviance: 632   AIC: 636
```

Compare the fit of the glm to LPM

```r
augment(model_glm, type.predict = "response") %>%
  ggplot(aes(TotalWorkingYears, Y)) +
  geom_point() +
  geom_line(aes(y = .fitted))
```



Modeling assumption

$$\text{logit}[P(Y = 1|X)] = \beta_0 + \beta_1 X$$

some function (logit) of the mean of Y is equal to a linear function in X

$$P(Y = 1|X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}$$