# Predicting Divorce with the DS Test: a quantitative approach

CYO - Data Science: Capstone

Luz E Rodriguez - October 2020

```r
if(!require(magrittr)) install.packages("magrittr",
                                        repos = "http://cran.us.r-project.org")
if(!require(utils)) install.packages("utils",
                                        repos = "http://cran.us.r-project.org")
if(!require(ggplot2)) install.packages("ggplot2",
                                       repos = "http://cran.us.r-project.org")
if(!require(dplyr)) install.packages("dplyr", repos = "http://cran.us.r-project.org")
if(!require(kableExtra)) install.packages("kableExtra",
                                          repos = "http://cran.us.r-project.org")
if(!require(knitr)) install.packages("knitr",
                                     repos = "http://cran.us.r-project.org")
if(!require(tibble)) install.packages("tibble",
                                      repos = "http://cran.us.r-project.org")
if(!require(corrplot)) install.packages("corrplot",
                                        repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret",
                                     repos = "http://cran.us.r-project.org")
if(!require(mlbench)) install.packages("mlbench",
                                       repos = "http://cran.us.r-project.org")
if(!require(gridExtra)) install.packages("gridExtra",
                                         repos = "http://cran.us.r-project.org")
if(!require(ggridges)) install.packages("ggridges",
                                        repos = "http://cran.us.r-project.org")
if(!require(factoextra)) install.packages("factoextra",
                                          repos = "http://cran.us.r-project.org")
if(!require(rpart)) install.packages("rpart",
                                     repos = "http://cran.us.r-project.org")
if(!require(rpart.plot)) install.packages("rpart.plot",
                                          repos = "http://cran.us.r-project.org")
if(!require(randomForest)) install.packages("randomForest",
                                            repos = "http://cran.us.r-project.org")
if(!require(Rtools)) install.packages("Rtools",
                                      repos = "https://cran.rstudio.com/bin/windows/Rtools/")
```

## 1. Project Overview

I found this interesting article about Divorce Prediction https://dergipark.org.tr/en/download/article-file/748448. This is based on a Gottman's couples therapy research of more than 20 years and proposes that couples can find alert signs of divorce. If diagnosed on time, marriages can work on tackling their problems to have a happy marriage and avoid divorce. Gottman has designed therapies to help marriages to improve their quality of life and relationships.

The article analyzes a sample survey in Turkey and predicts divorce with 98% accuracy by using correlation based feature selection and artificial neural networks.

It is very exciting to be able to quantify emotions and behaviors and apply analytics in social sciences. It is also more difficult and biased because there are a lot of assumptions, like people are completely honest and that every person interpret the questions in the same way. That is why I want to explore this dataset and apply other prediction methods to see if I get the same prediction rate. I will predict a categorical variable, in this case whether a person will remain married(0) or will get divorced(1) based on their current behavior explained by the answers to the survey.

# 2. Analysis

## Data Set Description

The dataset is taken from UCI Machine Learning Repository - Center for Machine Learning And Intelligent Systems https://archive.ics.uci.edu/ml/machine-learning-databases/00497/. This corresponds to a dataset from a questionnaire that ask participants several questions related to their behavior with their partners based on the Divorce Predictors Scale (DPS) from Gottman couples therapy.

The initial research https://dergipark.org.tr/en/download/article-file/748448 was developed by Dr. Mustafa Kemal Yöntem, Dr. Kemal ADEM, Prof. Dr. Tahsin İlhan, and Lecturer Serhat Kılıçarslan, inspired by Gottman's DPS couples therapy.

The survey had a total of 170 Turkish participants, 84 males and 86 females between 20 to 63 years old, from which 84 are divorced(49%) and 86 are married(51%) who answered 54 questions. All responses were collected on a 5-point scale: (0=Never, 1=Seldom, 2=Averagely, 3=Frequently, 4=Always).

The last variable **Class** indicates marital status as 0 for married and 1 for divorced.

*Downloading the dataset*

```
current_dir <- getwd()
divorce <- utils::read.table(file.path(current_dir,"divorce.csv"),
                             sep = ";", header = TRUE)
```

```
print("Split between married/divorced in dateset")
```

```
## [1] "Split between married/divorced in dateset"
```

```
(table(divorce$Class))
```

```
##
##  0  1
## 86 84
```

```
print("Summary scale responses")
```

```
## [1] "Summary scale responses"
```

```
with(divorce, summary(Atr1))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   2.000   1.776   3.000   4.000
```

```r
print("Dataset glimpse")
```

```
## [1] "Dataset glimpse"
```

```r
tibble::glimpse(divorce)
```

```
## Rows: 170
## Columns: 55
## $ Atr1  <int> 2, 4, 2, 3, 2, 0, 3, 2, 2, 1, 4, 4, 3, 3, 3, 4, 4, 4, 3, 4, 4...
## $ Atr2  <int> 2, 4, 2, 2, 2, 0, 3, 1, 2, 1, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 3...
## $ Atr3  <int> 4, 4, 2, 3, 1, 1, 3, 2, 1, 1, 4, 4, 3, 3, 3, 3, 3, 4, 4, 4, 3...
## $ Atr4  <int> 1, 4, 2, 2, 1, 0, 2, 2, 0, 1, 3, 3, 4, 4, 4, 2, 2, 3, 4, 3, 3...
## $ Atr5  <int> 0, 4, 1, 3, 1, 0, 1, 2, 0, 1, 4, 4, 3, 3, 3, 4, 4, 4, 3, 4, 4...
## $ Atr6  <int> 0, 0, 3, 3, 1, 2, 3, 1, 4, 2, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1...
## $ Atr7  <int> 0, 0, 2, 3, 0, 0, 4, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0...
## $ Atr8  <int> 0, 4, 1, 3, 0, 0, 3, 3, 3, 2, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 3...
## $ Atr9  <int> 0, 4, 1, 3, 0, 0, 2, 3, 3, 2, 4, 4, 3, 3, 3, 3, 3, 4, 4, 4, 3...
## $ Atr10 <int> 0, 4, 2, 3, 0, 1, 2, 2, 3, 2, 3, 3, 4, 4, 4, 2, 2, 3, 4, 3, 3...
## $ Atr11 <int> 1, 4, 3, 4, 0, 0, 2, 4, 3, 3, 4, 4, 3, 3, 3, 4, 4, 4, 3, 4, 4...
## $ Atr12 <int> 0, 3, 4, 3, 1, 2, 2, 3, 3, 0, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 3...
## $ Atr13 <int> 1, 4, 2, 3, 0, 1, 2, 2, 3, 0, 4, 4, 3, 3, 3, 4, 4, 4, 3, 4, 4...
## $ Atr14 <int> 1, 0, 3, 4, 1, 0, 3, 3, 3, 2, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 3...
## $ Atr15 <int> 0, 4, 3, 3, 1, 2, 2, 4, 3, 1, 4, 4, 3, 3, 3, 3, 3, 4, 4, 4, 3...
## $ Atr16 <int> 1, 4, 3, 3, 1, 0, 3, 3, 3, 0, 3, 3, 4, 4, 4, 2, 2, 3, 4, 3, 3...
## $ Atr17 <int> 0, 4, 3, 3, 1, 2, 3, 2, 3, 1, 4, 4, 3, 3, 3, 4, 4, 4, 3, 4, 4...
## $ Atr18 <int> 0, 4, 3, 3, 1, 1, 3, 3, 3, 2, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 3...
## $ Atr19 <int> 0, 3, 3, 3, 2, 0, 3, 2, 3, 1, 4, 4, 3, 3, 3, 4, 4, 4, 3, 4, 4...
## $ Atr20 <int> 1, 2, 2, 4, 1, 1, 2, 1, 3, 0, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 3...
## $ Atr21 <int> 0, 1, 1, 1, 1, 0, 3, 2, 2, 0, 4, 4, 3, 3, 3, 3, 3, 4, 4, 4, 3...
## $ Atr22 <int> 0, 1, 0, 1, 0, 0, 3, 1, 2, 0, 3, 3, 4, 4, 4, 2, 2, 3, 4, 3, 3...
## $ Atr23 <int> 0, 0, 1, 1, 0, 0, 3, 1, 2, 0, 4, 4, 3, 3, 3, 4, 4, 4, 3, 4, 4...
## $ Atr24 <int> 0, 2, 2, 1, 0, 0, 3, 2, 3, 1, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 3...
## $ Atr25 <int> 0, 2, 2, 2, 0, 2, 2, 3, 2, 1, 4, 4, 3, 3, 3, 4, 4, 4, 3, 4, 4...
## $ Atr26 <int> 0, 1, 2, 1, 2, 2, 3, 3, 3, 1, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 3...
## $ Atr27 <int> 0, 2, 2, 1, 1, 0, 3, 2, 2, 1, 4, 4, 3, 3, 3, 3, 3, 4, 4, 4, 3...
## $ Atr28 <int> 0, 0, 2, 1, 2, 0, 2, 2, 3, 1, 3, 3, 4, 4, 4, 2, 2, 3, 4, 3, 3...
## $ Atr29 <int> 0, 1, 3, 1, 1, 0, 2, 2, 2, 1, 4, 4, 3, 3, 3, 4, 4, 4, 3, 4, 4...
## $ Atr30 <int> 1, 1, 2, 3, 1, 0, 2, 3, 3, 1, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 3...
## $ Atr31 <int> 1, 0, 3, 2, 1, 4, 1, 1, 1, 1, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4...
## $ Atr32 <int> 2, 4, 3, 3, 1, 1, 2, 1, 1, 1, 4, 4, 4, 4, 4, 4, 4, 3, 4, 3, 4...
## $ Atr33 <int> 1, 2, 1, 2, 1, 1, 2, 0, 1, 0, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4...
## $ Atr34 <int> 2, 3, 1, 2, 1, 1, 1, 2, 1, 1, 4, 4, 4, 4, 4, 4, 4, 3, 4, 3, 4...
## $ Atr35 <int> 0, 0, 1, 1, 0, 1, 1, 2, 1, 0, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4...
## $ Atr36 <int> 1, 2, 1, 1, 0, 1, 2, 1, 1, 0, 4, 4, 4, 4, 4, 4, 4, 3, 4, 3, 4...
## $ Atr37 <int> 2, 3, 2, 3, 0, 1, 3, 4, 1, 1, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4...
## $ Atr38 <int> 1, 4, 1, 3, 0, 2, 2, 4, 2, 1, 4, 4, 4, 4, 4, 4, 4, 3, 4, 3, 4...
## $ Atr39 <int> 3, 2, 3, 4, 2, 0, 2, 4, 2, 2, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4...
## $ Atr40 <int> 3, 4, 3, 4, 1, 2, 3, 4, 2, 2, 4, 4, 4, 4, 4, 4, 4, 3, 4, 3, 4...
## $ Atr41 <int> 2, 2, 3, 2, 0, 2, 3, 4, 2, 1, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4...
## $ Atr42 <int> 1, 2, 3, 2, 2, 1, 3, 4, 2, 2, 4, 4, 4, 4, 4, 4, 4, 3, 4, 3, 4...
```

```
## $ Atr43 <int> 1, 3, 2, 3, 3, 2, 3, 3, 2, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4...
## $ Atr44 <int> 2, 4, 3, 2, 0, 3, 4, 2, 2, 2, 4, 4, 4, 4, 3, 4, 4, 3, 4, 3, 4...
## $ Atr45 <int> 3, 2, 2, 3, 2, 0, 3, 0, 2, 2, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4...
## $ Atr46 <int> 2, 2, 3, 2, 2, 2, 3, 0, 1, 2, 4, 4, 4, 4, 4, 4, 4, 3, 4, 3, 4...
## $ Atr47 <int> 1, 2, 2, 2, 1, 2, 2, 1, 1, 0, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4...
## $ Atr48 <int> 3, 3, 3, 3, 2, 1, 3, 2, 1, 2, 4, 4, 4, 4, 4, 4, 4, 3, 4, 3, 4...
## $ Atr49 <int> 3, 4, 1, 3, 3, 2, 2, 2, 1, 2, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4...
## $ Atr50 <int> 3, 4, 1, 3, 2, 1, 3, 2, 1, 2, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4...
## $ Atr51 <int> 2, 4, 1, 3, 2, 1, 3, 1, 1, 2, 4, 4, 4, 4, 4, 4, 4, 3, 4, 3, 4...
## $ Atr52 <int> 3, 4, 2, 2, 2, 1, 2, 1, 1, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4...
## $ Atr53 <int> 2, 2, 2, 2, 1, 2, 2, 1, 1, 3, 4, 4, 4, 4, 4, 4, 4, 3, 4, 3, 4...
## $ Atr54 <int> 1, 2, 2, 2, 0, 0, 2, 0, 1, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4...
## $ Class <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
```

The survey design has some bias. From the initial article, the researchers explained that Turkey is among the 12 countries with increasing divorce rate (as of 2019 when the study was published). Divorced participants answered based on their marriage experience, while the married participants included were the ones who considered themselves as having a "happy marriage", without any thought of divorce. This means, married people who are having problems and have signs of divorce were excluded.

The dataset does not include any demographic variables and only have the 54 questions' responses and marital status(output variable).

*Attribute Information*

1. When one of us apologizes when our discussions go bad, the issue does not extend.
2. I know we can ignore our differences, even if things get hard sometimes.
3. When we need to, we can take our discussions from the beginning and correct it.
4. When I argue with my spouse, it will eventually work for me to contact him.
5. The time I spent with my spouse is special for us.
6. We don't have time at home as partners.
7. We are like two strangers who share the same environment at home rather than family.
8. I enjoy our holidays with my spouse.
9. I enjoy traveling with my spouse.
10. My spouse and most of our goals are common.
11. I think that some day, my spouse and I will bee in harmony with each other.
12. My spouse and I have similar values regarding personal freedom.
13. My spouse and I have similar entertainment.
14. Most of our goals in regards to people (children, friends, etc.) are the same.
15. My dreams of living are similar and harmonious with those of my spouse.
16. I'm compatible with my spouse about what love should be.
17. I share the same views with my spouse about being happy.
18. My spouse and I have similar ideas about how marriage should be.
19. My spouse and I have similar ideas about how roles should be in marriage.
20. My spouse and I have similar values regarding trust.
21. I know exactly what my spouse likes.
22. I know how my spouse wants to be taken care of when she's sick.
23. I know my spouse's favorite food.
24. I can tell you what kind of stress my spouse is having in life.
25. I have knowledge of my spouse's inner world.
26. I know my spouse's basic concerns.
27. I know what my spouse's current sources of stress are.
28. I know my spouse's hopes and wishes.
29. I know my spouse very well.
30. I know my spouse's friends and their social relationships.

31. I feel aggressive when I argue with my spouse.
32. When discussing with my spouse, I usually use expressions such as X, Y, Z.
33. I can use negative statements about my spouse's personality during our discussions.
34. I can use offensive expressions during our discussions.
35. I can insult our discussions.
36. I can be humiliating when we argue.
37. My argument with my spouse is not calm.
38. I hate my spouse's way of bringing it up.
39. Fights often occur suddenly.
40. We're just starting a fight before I know what's going on.
41. When I talk to my spouse about something, my calm suddenly breaks.
42. When I argue with my spouse, it only snaps in and I don't say a word.
43. I'm mostly willing to calm the environment a little bit.
44. Sometimes I think it's good for me to leave home for a while.
45. I'd rather stay silent than argue with my spouse.
46. Even if I'm right in the argument, I'm willing not to upset the other side.
47. When I argue with my spouse, I remain silent because I am afraid of not being able to control my anger.
48. I feel right in our discussions.
49. I have nothing to do with what I've been accused of.
50. I'm not actually the one who's guilty of what I'm accused of.
51. I'm not the one who's wrong about problems at home.
52. I wouldn't hesitate to tell her about my spouse's inadequacy.
53. I remind my spouse of her inadequacies during our discussion.
54. I'm not afraid to tell her about my spouse's incompetence.

## Data wrangling

The decision variable *Class* is a binary variable, having married as 0 and divorced 1. I will adjust this vaiable as factor. Other than that, the dataste is very clean.

```r
divorce <- divorce %>% dplyr::mutate(Class = factor(Class))
```

## Datasets for model creation

This dataset has only 170 rows. Being a small dataset, I will split the data in a 80-20 proportion for training and testing. There will not be a validation dataset.

From the training dataset 50% of participants are divorced. From the testing dataset 48% are divorced. This is a very balanced split.

```r
set.seed(1, sample.kind="Rounding")

test_index <- createDataPartition(y = divorce$Class, times = 1, p = 0.2, list = FALSE)

divorce_training <-divorce[-test_index, ]

divorce_testing <- divorce[test_index, ]

print("Training set proportions:")
```

```
## [1] "Training set proportions:"
```

```r
prop.table(table(divorce_training$Class))
```

```
##
##         0         1
## 0.5037037 0.4962963
```

```r
print("Testing set proportions:")
```

```
## [1] "Testing set proportions:"
```

```r
prop.table(table(divorce_testing$Class))
```

```
##
##         0         1
## 0.5142857 0.4857143
```

## Data Exploration

- Dataset overview

As explained before, there are 54 predictors, which are answers to a survey. The response scale is (0=Never, 1=Seldom, 2=Averagely, 3=Frequently, 4=Always).

From the data summary we can see that more than 20 answers have a mean greater than 2.

```r
summary <- summary(divorce)
summary
```

```
##       Atr1            Atr2            Atr3            Atr4      
##  Min.   :0.000   Min.   :0.000   Min.   :0.000   Min.   :0.000  
##  1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.000  
##  Median :2.000   Median :2.000   Median :2.000   Median :1.000  
##  Mean   :1.776   Mean   :1.653   Mean   :1.765   Mean   :1.482  
##  3rd Qu.:3.000   3rd Qu.:3.000   3rd Qu.:3.000   3rd Qu.:3.000  
##  Max.   :4.000   Max.   :4.000   Max.   :4.000   Max.   :4.000  
##       Atr5            Atr6             Atr7             Atr8      
##  Min.   :0.000   Min.   :0.0000   Min.   :0.0000   Min.   :0.000  
##  1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.000  
##  Median :1.000   Median :0.0000   Median :0.0000   Median :1.000  
##  Mean   :1.541   Mean   :0.7471   Mean   :0.4941   Mean   :1.453  
##  3rd Qu.:3.000   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:3.000  
##  Max.   :4.000   Max.   :4.0000   Max.   :4.0000   Max.   :4.000  
##       Atr9           Atr10           Atr11           Atr12      
##  Min.   :0.000   Min.   :0.000   Min.   :0.000   Min.   :0.000  
##  1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.000  
##  Median :1.000   Median :2.000   Median :1.000   Median :1.500  
##  Mean   :1.459   Mean   :1.576   Mean   :1.688   Mean   :1.653  
##  3rd Qu.:3.000   3rd Qu.:3.000   3rd Qu.:3.000   3rd Qu.:3.000  
##  Max.   :4.000   Max.   :4.000   Max.   :4.000   Max.   :4.000  
##      Atr13           Atr14           Atr15           Atr16
```

```
## Min.   :0.000   Min.   :0.000   Min.   :0.000   Min.   :0.000  
## 1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.000  
## Median :2.000   Median :1.000   Median :1.000   Median :1.000  
## Mean   :1.835   Mean   :1.571   Mean   :1.571   Mean   :1.476  
## 3rd Qu.:3.000   3rd Qu.:3.000   3rd Qu.:3.000   3rd Qu.:3.000  
## Max.   :4.000   Max.   :4.000   Max.   :4.000   Max.   :4.000  
##     Atr17           Atr18           Atr19           Atr20      
## Min.   :0.000   Min.   :0.000   Min.   :0.000   Min.   :0.000  
## 1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.000  
## Median :1.000   Median :1.000   Median :1.000   Median :1.000  
## Mean   :1.653   Mean   :1.518   Mean   :1.641   Mean   :1.459  
## 3rd Qu.:3.000   3rd Qu.:3.000   3rd Qu.:3.000   3rd Qu.:3.000  
## Max.   :4.000   Max.   :4.000   Max.   :4.000   Max.   :4.000  
##     Atr21           Atr22           Atr23           Atr24      
## Min.   :0.000   Min.   :0.000   Min.   :0.000   Min.   :0.000  
## 1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.000  
## Median :1.000   Median :0.000   Median :0.000   Median :1.000  
## Mean   :1.388   Mean   :1.247   Mean   :1.412   Mean   :1.512  
## 3rd Qu.:3.000   3rd Qu.:3.000   3rd Qu.:3.000   3rd Qu.:3.000  
## Max.   :4.000   Max.   :4.000   Max.   :4.000   Max.   :4.000  
##     Atr25           Atr26           Atr27         Atr28           Atr29      
## Min.   :0.000   Min.   :0.000   Min.   :0.0   Min.   :0.000   Min.   :0.000  
## 1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.0   1st Qu.:0.000   1st Qu.:0.000  
## Median :1.000   Median :1.000   Median :1.0   Median :0.500   Median :1.000  
## Mean   :1.629   Mean   :1.488   Mean   :1.4   Mean   :1.306   Mean   :1.494  
## 3rd Qu.:3.000   3rd Qu.:3.000   3rd Qu.:3.0   3rd Qu.:3.000   3rd Qu.:3.000  
## Max.   :4.000   Max.   :4.000   Max.   :4.0   Max.   :4.000   Max.   :4.000  
##     Atr30           Atr31           Atr32           Atr33           Atr34    
## Min.   :0.000   Min.   :0.000   Min.   :0.000   Min.   :0.000   Min.   :0.0  
## 1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.0  
## Median :1.000   Median :2.000   Median :2.000   Median :1.000   Median :1.0  
## Mean   :1.494   Mean   :2.124   Mean   :2.059   Mean   :1.806   Mean   :1.9  
## 3rd Qu.:3.000   3rd Qu.:4.000   3rd Qu.:4.000   3rd Qu.:4.000   3rd Qu.:4.0  
## Max.   :4.000   Max.   :4.000   Max.   :4.000   Max.   :4.000   Max.   :4.0  
##     Atr35           Atr36           Atr37           Atr38      
## Min.   :0.000   Min.   :0.000   Min.   :0.000   Min.   :0.000  
## 1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.000  
## Median :0.500   Median :0.000   Median :2.000   Median :1.000  
## Mean   :1.671   Mean   :1.606   Mean   :2.088   Mean   :1.859  
## 3rd Qu.:4.000   3rd Qu.:4.000   3rd Qu.:4.000   3rd Qu.:4.000  
## Max.   :4.000   Max.   :4.000   Max.   :4.000   Max.   :4.000  
##     Atr39           Atr40           Atr41           Atr42      
## Min.   :0.000   Min.   :0.000   Min.   :0.000   Min.   :0.000  
## 1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.000  
## Median :2.000   Median :1.500   Median :2.000   Median :2.000  
## Mean   :2.088   Mean   :1.871   Mean   :1.994   Mean   :2.159  
## 3rd Qu.:4.000   3rd Qu.:4.000   3rd Qu.:4.000   3rd Qu.:4.000  
## Max.   :4.000   Max.   :4.000   Max.   :4.000   Max.   :4.000  
##     Atr43           Atr44           Atr45           Atr46      
## Min.   :0.000   Min.   :0.000   Min.   :0.000   Min.   :0.000  
## 1st Qu.:2.000   1st Qu.:0.000   1st Qu.:1.000   1st Qu.:2.000  
## Median :3.000   Median :2.000   Median :3.000   Median :3.000  
## Mean   :2.706   Mean   :1.941   Mean   :2.459   Mean   :2.553  
## 3rd Qu.:4.000   3rd Qu.:4.000   3rd Qu.:4.000   3rd Qu.:4.000  
```

```
##  Max.   :4.000   Max.   :4.000   Max.   :4.000   Max.   :4.000  
##      Atr47           Atr48           Atr49           Atr50      
##  Min.   :0.000   Min.   :0.000   Min.   :0.000   Min.   :0.000  
##  1st Qu.:1.000   1st Qu.:2.000   1st Qu.:1.000   1st Qu.:1.000  
##  Median :2.000   Median :3.000   Median :3.000   Median :2.000  
##  Mean   :2.271   Mean   :2.741   Mean   :2.382   Mean   :2.429  
##  3rd Qu.:4.000   3rd Qu.:4.000   3rd Qu.:4.000   3rd Qu.:4.000  
##  Max.   :4.000   Max.   :4.000   Max.   :4.000   Max.   :4.000  
##      Atr51           Atr52           Atr53           Atr54       Class 
##  Min.   :0.000   Min.   :0.000   Min.   :0.000   Min.   :0.000   0:86  
##  1st Qu.:2.000   1st Qu.:1.000   1st Qu.:1.000   1st Qu.:0.000   1:84  
##  Median :3.000   Median :3.000   Median :2.000   Median :2.000         
##  Mean   :2.476   Mean   :2.518   Mean   :2.241   Mean   :2.012         
##  3rd Qu.:4.000   3rd Qu.:4.000   3rd Qu.:4.000   3rd Qu.:4.000         
##  Max.   :4.000   Max.   :4.000   Max.   :4.000   Max.   :4.000         
```

- Missing values

The dataset does not have any missing values that need to be removed or handle.

```r
table(is.na(divorce))
```

```
## 
## FALSE 
##  9350
```

- Histograms of predictor variables

A visual inspection tells us that the attributes that could predict divorce based on the counts and more filled as divorced are: atr1, atr3,atr13, atr25, atr32, atr39.

```r
## store predictor variables
attributes <- setdiff(colnames(divorce), "Class")

## initialize an empty list to store plots
histograms = list()

## initialize plot index
i = 1

## create plot and fill plot list

for (attr in attributes) {
  hist = ggplot2::ggplot(divorce_training) +
    ggplot2::geom_histogram(aes_string(x=attr, fill="Class"),
                   position="stack", alpha=0.6) +
    ggplot2::theme_classic() +
    ggplot2::theme(legend.position="bottom")

  histograms[[i]] = hist
  i = i + 1
}
```

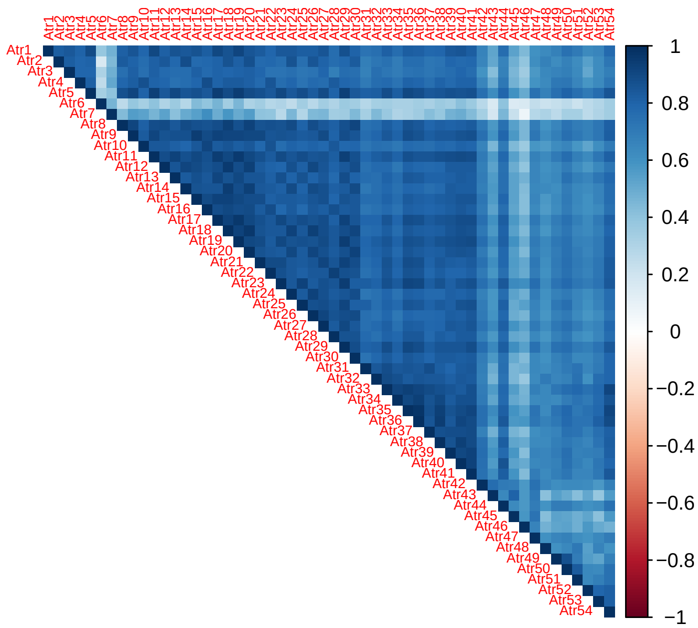
```
## display plots. I will plot this graphs at the end after the references for clarity
#histograms
#do.call("grid.arrange", c(histograms, ncol=3))
```

- Evaluating variables correlation

```
main_vars <- divorce_training[, 1:54]

#calculating correlation among variables in divorce
corr_divorce <- cor(main_vars)

corrplot::corrplot(corr_divorce, type="upper", method="shade",tl.cex = 0.6)
```



The correlation chart shows there are many variables highly correlated, as shown with the darkets blue shade (values close to 1). We need to find the relevant variables to run a predictive model, thus, we want to remove highly correlated variables to avoid overfitting.

- Eliminating highly correlated variables

To efficiently reduce the amount of predictors, we will be using Principal Component Analisys (PCA)

The first 9 elements account for 90% of the variability. The 25 first elements account for 98% of the variability.Only the first component explains 77% of the variability.

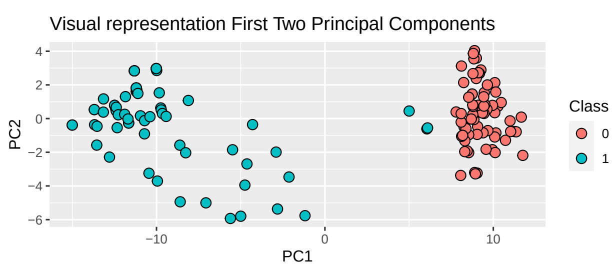
```
x <- divorce_training[ ,attributes] #Selecting attributes from dataset

divorce_training_pca <- prcomp(x) #Calculating PC from attributes
summary(divorce_training_pca)
```

```
## Importance of components:
##                           PC1     PC2     PC3     PC4     PC5     PC6     PC7
```

```
## Standard deviation     9.9169 2.10905 1.90120 1.40838 1.32143 1.26750 1.20208
## Proportion of Variance 0.7724 0.03493 0.02839 0.01558 0.01371 0.01262 0.01135
## Cumulative Proportion  0.7724 0.80732 0.83571 0.85129 0.86501 0.87762 0.88897
##                            PC8     PC9    PC10    PC11    PC12    PC13    PC14
## Standard deviation     1.12240 1.09812 1.01164 0.97886 0.93897 0.85697 0.83375
## Proportion of Variance 0.00989 0.00947 0.00804 0.00753 0.00692 0.00577 0.00546
## Cumulative Proportion  0.89887 0.90834 0.91637 0.92390 0.93082 0.93659 0.94205
##                           PC15    PC16    PC17    PC18    PC19    PC20    PC21
## Standard deviation     0.81812 0.74315 0.72554 0.69914 0.68521 0.61560 0.57906
## Proportion of Variance 0.00526 0.00434 0.00413 0.00384 0.00369 0.00298 0.00263
## Cumulative Proportion  0.94731 0.95165 0.95578 0.95962 0.96331 0.96628 0.96892
##                           PC22    PC23    PC24    PC25    PC26    PC27    PC28
## Standard deviation     0.57025 0.54391 0.53732 0.51584 0.50180 0.47519 0.44590
## Proportion of Variance 0.00255 0.00232 0.00227 0.00209 0.00198 0.00177 0.00156
## Cumulative Proportion  0.97147 0.97379 0.97606 0.97815 0.98013 0.98190 0.98346
##                           PC29    PC30    PC31    PC32    PC33    PC34    PC35
## Standard deviation     0.43946 0.43497 0.41335 0.38767 0.36893 0.36494 0.35122
## Proportion of Variance 0.00152 0.00149 0.00134 0.00118 0.00107 0.00105 0.00097
## Cumulative Proportion  0.98498 0.98647 0.98781 0.98899 0.99006 0.99110 0.99207
##                           PC36    PC37   PC38    PC39    PC40    PC41    PC42
## Standard deviation     0.34239 0.32768 0.3189 0.29698 0.27924 0.26448 0.25400
## Proportion of Variance 0.00092 0.00084 0.0008 0.00069 0.00061 0.00055 0.00051
## Cumulative Proportion  0.99299 0.99384 0.9946 0.99533 0.99594 0.99649 0.99700
##                           PC43   PC44    PC45    PC46    PC47    PC48    PC49
## Standard deviation     0.24348 0.2247 0.21123 0.20330 0.19254 0.18120 0.16258
## Proportion of Variance 0.00047 0.0004 0.00035 0.00032 0.00029 0.00026 0.00021
## Cumulative Proportion  0.99746 0.9979 0.99821 0.99853 0.99882 0.99908 0.99929
##                           PC50    PC51    PC52    PC53    PC54
## Standard deviation     0.15695 0.14094 0.13272 0.12741 0.10986
## Proportion of Variance 0.00019 0.00016 0.00014 0.00013 0.00009
## Cumulative Proportion  0.99948 0.99964 0.99978 0.99991 1.00000
```

```r
data.frame(divorce_training_pca$x[,1:2], Class=divorce_training$Class) %>%
  ggplot2::ggplot(aes(PC1,PC2, fill = Class))+
  ggplot2::geom_point(cex=3, pch=21) +
  ggplot2::coord_fixed(ratio = 1)+
  ggplot2::ggtitle("Visual representation First Two Principal Components")
```



```r
data.frame(divorce_training_pca$x[,2:3], Class=divorce_training$Class) %>%
  ggplot2::ggplot(aes(PC2,PC3, fill = Class))+
  ggplot2::geom_point(cex=3, pch=21) +
  ggplot2::coord_fixed(ratio = 1)+
  ggplot2::ggtitle("Visual representation PC 2 and 3")
```


Now we will create a new training and testing dataset with principal components to use in the modeling session.

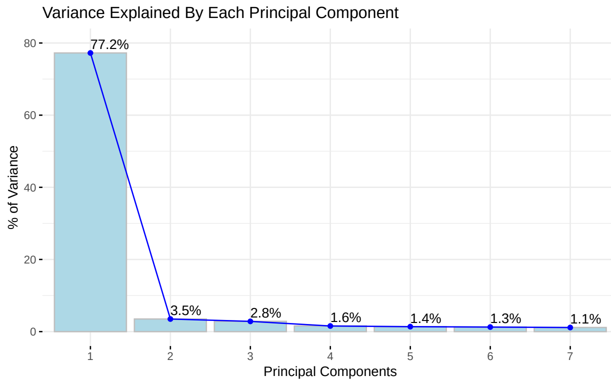
```r
divorce_training_pc = as.data.frame(divorce_training_pca$x)
divorce_training_pc$Class = divorce_training$Class
head(divorce_training_pc)
```

```
##          PC1        PC2        PC3        PC4         PC5        PC6       PC7
## 1  5.001927  0.4482030 -1.9183911 -0.4050889  2.07916074 -2.4164406 -0.494351
## 3 -2.131388 -3.4694589  1.6256846  2.3380452 -0.78482827 -1.1632855  1.539369
## 4 -4.752328 -3.9502921  0.9283251  0.8547924  0.52184214 -3.3619236  1.761967
## 5  6.056497 -0.6087961  1.7336813 -0.6574911  0.78846621 -0.2286895 -0.505369
## 6  6.095386 -0.5550061 -1.3081149  2.7564768  0.77774284  1.6858438  1.009736
## 7 -4.629296 -2.6933806  2.7921849  0.4245280  0.03805505 -0.8893236  1.111097
##          PC8        PC9      PC10       PC11       PC12       PC13        PC14
## 1 -0.3828507  0.3213686 0.9630951  3.1690760 -1.7297686  0.7022726 -0.09165216
## 3  1.3076764  2.0083988 0.8576773 -0.8056156  0.4054828 -0.1688673 -0.13026595
## 4  0.1093031 -0.4881266 0.9061661  0.2117650  0.1587790  0.1896811  1.14357272
## 5 -0.6234749 -0.3774705 0.4807929  0.9321649  0.5315559 -1.7625150 -0.36416526
## 6  1.1923196  1.6190475 1.2334267 -2.2687860 -0.1356265  0.9835357  1.26326869
## 7  1.9948610 -0.2144704 1.2097839 -0.4789832 -1.3445787 -1.4860855 -1.38843781
##         PC15       PC16       PC17       PC18        PC19       PC20       PC21
## 1  0.8780461 -0.1607495 -1.2302091  0.7919400  0.36003064 -1.1979412 -1.2224653
## 3  1.7475863 -0.1860201  1.5511447  0.4575502 -0.09111096 -0.9224207 -0.4927741
## 4 -0.9600942  1.3454632  0.8642120  2.0205829  0.29516376 -1.0793850  0.3175153
## 5  0.8694334 -0.8452825  1.1500063  0.9531644 -0.43131328 -0.0697104  0.2276545
## 6 -0.5504636 -1.5397232  0.9459050 -1.4566373  0.11364548 -1.3958371 -0.6464744
## 7  0.2923353  0.3649304 -0.9555119  0.1066875 -1.01624678  1.1742153  0.2529383
##           PC22        PC23        PC24         PC25       PC26       PC27
## 1 -0.127571579 -0.07161427  0.25600716  0.368479171 0.77664993  0.8931457
## 3  0.003912372 -0.47243435 -0.18952314 -1.001357301 0.20558906 -0.4275691
## 4 -0.052130053  0.65243890 -0.53103061  0.009104412 0.16427742  0.8605826
## 5 -0.398490059 -0.36803277 -1.15774333 -0.012233225 1.66455407 -1.5377373
```

```
## 6  0.252193479  0.67587544  0.09738649  1.040229824 0.21877661 -1.2836662
## 7  0.299361558  1.32286848  1.80090766 -0.705892002 0.01290306  0.7781087
##         PC28        PC29        PC30        PC31         PC32        PC33
## 1  0.2831462  0.18575733  1.00722010  0.27743906  0.561596537 -0.13752957
## 3 -0.5251967  0.80721182 -0.03782090 -1.73513270  0.687582514 -0.17270602
## 4  1.4692071 -1.23530007 -0.26844745  0.18048194  0.003543012  0.09525928
## 5  0.2890830  0.08404252  1.36860394  0.12308114 -0.640749245  0.11019362
## 6  0.5390956  0.68130435  0.07000135 -0.07103052 -0.303898503  0.37072636
## 7 -1.1739270 -0.15805067  0.81946269  0.06812937  0.762958374  0.84774121
##        PC34         PC35       PC36       PC37        PC38       PC39
## 1 0.1583347 -0.005603301 -0.6540078 -0.3213327 -0.04936921 -0.9146279
## 3 0.1022437  0.305162933 -0.9323578  0.5605979 -0.20728209  0.1951162
## 4 0.3866160  0.496901705  0.5589019  0.3444398  0.22239626 -0.1210227
## 5 0.1986656 -0.175479502 -0.4004802 -0.4990597 -0.33526177  0.2112228
## 6 0.7961522 -0.608458418  0.8994890 -0.7179210  0.02497511 -0.2468381
## 7 0.3746570 -0.265292997  0.6837553 -0.3537728  0.36073001  0.3866860
##          PC40       PC41        PC42         PC43       PC44         PC45
## 1 -0.02966737  0.1178952  0.20361309 -0.217933068  0.2823155 -0.313384785
## 3  0.39829938 -0.5904353 -0.03062511 -0.256609267 -0.2817708  0.198534294
## 4  0.49401989  0.6681529  0.19735677  0.032685234 -0.1299638 -0.007100035
## 5 -0.37936225  0.4833759 -0.36830480 -0.030735760  0.1918664  0.701884264
## 6 -0.27982142 -0.1132089  0.09504793 -0.100302811  0.3339586 -0.655476789
## 7 -0.40690683 -0.1807528 -0.35601455 -0.002339044  0.2880148  0.334591304
##          PC46        PC47        PC48         PC49        PC50        PC51
## 1  0.13014615 -0.07598285  0.05177478 -0.029211762 -0.02074489  0.11460665
## 3  0.60036620  0.17002531 -0.12623810 -0.131477388 -0.13925237  0.09832337
## 4 -0.08448601  0.42703425 -0.05156991 -0.258556486  0.25761210  0.08405680
## 5 -0.23484856  0.11673140 -0.08753785  0.094750062  0.04304332 -0.31857092
## 6 -0.15935518  0.50241346 -0.10830897  0.003326409 -0.04236347  0.03420550
## 7 -0.04713392 -0.07623524  0.04462132 -0.338139890  0.02031832 -0.18400468
##          PC52        PC53         PC54 Class
## 1 -0.45447250  0.01791921  0.006505286     1
## 3 -0.16046745 -0.18603486 -0.045096214     1
## 4 -0.11839101 -0.04804378 -0.056844668     1
## 5  0.21332102 -0.14694180 -0.173686280     1
## 6 -0.06626001  0.02315277  0.113020238     1
## 7 -0.30166746  0.11309212 -0.032099722     1
```

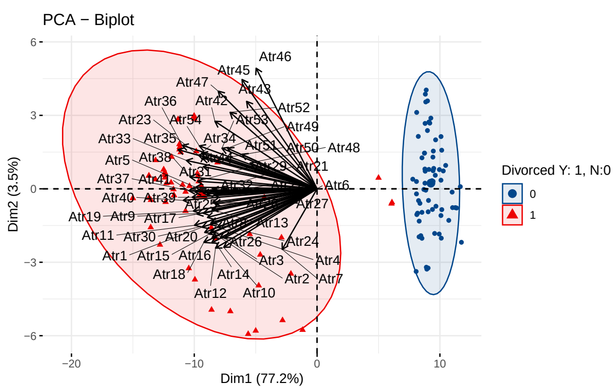
```r
divorce_testing_pc = predict(divorce_training_pca, newdata = divorce_testing)
divorce_testing_pc = as.data.frame(divorce_training_pc)
divorce_testing_pc$class = divorce_testing$class
```

```r
factoextra::fviz_eig(divorce_training_pca, addlabels=TRUE,
                     ylim=c(0,80), geom = c("bar", "line"),
                     barfill="lightblue", barcolor="grey", linecolor="blue", ncp=7) +
 labs(title = "Variance Explained By Each Principal Component",
         x = "Principal Components", y = "% of Variance")
```




The following chart also shows how the first component explains 72% of variability.

```
factoextra::fviz_pca_biplot(divorce_training_pca,
                            col.ind = as.factor(divorce_training$Class), col="black",
                palette = "lancet", geom = c("point"), repel=TRUE,
                legend.title="Divorced Y: 1, N:0", addEllipses = TRUE)
```




## Models

In this section I will explore different methods to predict divorce like logistic regression, kNN, decision trees and random forest and then compare which models provides the best accuracy. I will also use the models in the original data and with the principal components for comparison.

### Baseline prediction

Randomly guessing whether a person will get divorced based on the answers to the questionare, we get that 51% of them would get divorce. We assume that each person has an equal chance of getting divorced. Randomly guessing gives a divorce estimated greater than the value in the data set which is 43%. The accuracy of randomly guessing is just 63%.

```
set.seed(3, sample.kind = "Rounding")
# guess with equal probability of divorce
guess_divorce <- sample(c(0,1), nrow(divorce_testing), replace = TRUE)
#how many people would be divorced
print("Proportion of people whow would get divorced")
```

```
## [1] "Proportion of people whow would get divorced"
```

```r
mean(guess_divorce)
```

```
## [1] 0.5142857
```

```r
#Calculating accuracy comparing guessing with testing data set
print("Accuracy")
```

```
## [1] "Accuracy"
```

```r
mean(guess_divorce == divorce_testing$Class)
```

```
## [1] 0.6285714
```

```r
### Accuracy table
summary_accuracy <- tibble(Method = "Baseline Prediction", Accuracy = 0.6285)
kable(summary_accuracy, booktabs = T) %>%
  kableExtra::kable_styling(position = "center", latex_options = "striped")
```

| Method | Accuracy |
|---|---|
| Baseline Prediction | 0.6285 |

### Logistic regression

Logistic regression is a regression method where we calculate the probabilities of an output variable to belong to a certain class. Since the logistic regression model provides probabilites between 0-1, to predict the outcome I determine them as 1 or 0 like : if probability > 0.5 the outcome is 1 (divorced, that is our target variable), otherwise it is 0.

- glm model with all data

```r
set.seed(3, sample.kind = "Rounding")
logistic_model <- glm(Class ~.,family=binomial(link='logit'),data=divorce_training)

#summary(logistic_model)
```

```r
glm_preds_divorce <- predict(logistic_model, type="response")
print("Divorce rate in logistic regression")
```

```
## [1] "Divorce rate in logistic regression"
```

```r
mean((ifelse(glm_preds_divorce > 0.5, 1, 0)) == divorce_testing$Class)
```

```
## [1] 0.5037037
```

- Accuracy in testing set Logistic regression

```r
## create prediction probabilities (on test dataset)
glm_test_pred_probs = predict(logistic_model, type="response", newdata=divorce_testing)

## create predictions (on test dataset)
glm_test_preds = as.factor(ifelse(glm_test_pred_probs > 0.5, 1,0))

## evaluate performance (on test dataset)
confusionMatrix(glm_test_preds, divorce_testing$Class)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 18  2
##          1  0 15
##
##                Accuracy : 0.9429
##                  95% CI : (0.8084, 0.993)
##     No Information Rate : 0.5143
##     P-Value [Acc > NIR] : 4.406e-08
##
##                   Kappa : 0.8852
##
##  Mcnemar's Test P-Value : 0.4795
##
##             Sensitivity : 1.0000
##             Specificity : 0.8824
##          Pos Pred Value : 0.9000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.5143
##          Detection Rate : 0.5143
##    Detection Prevalence : 0.5714
##       Balanced Accuracy : 0.9412
##
##        'Positive' Class : 0
##
```

```r
### Accuracy table
# Update table based on testing data
summary_accuracy <- bind_rows(summary_accuracy,
                    tibble(Method = "Logistic Regression",
                           Accuracy = 0.9429))
# Show table
knitr::kable(summary_accuracy)%>%
  kableExtra::kable_styling(position = "center", latex_options = "striped")
```

| Method | Accuracy |
|---|---|
| Baseline Prediction | 0.6285 |
| Logistic Regression | 0.9429 |

- glm model with Principal Components data

```r
set.seed(3, sample.kind = "Rounding")
logistic_model_pc <- glm(Class ~.,family=binomial(link='logit'),data=divorce_training_pc)
```

- Accuracy in testing set Principal Components Logistic regression

```r
## create prediction probabilities (on test dataset)
glm_test_pred_probs_pc = predict(logistic_model_pc, type="response",
                                 newdata=divorce_testing_pc)

## create predictions (on test dataset)
glm_test_preds_pc = as.factor(ifelse(glm_test_pred_probs_pc > 0.5, 1,0))

## evaluate performance (on test dataset)
confusionMatrix(glm_test_preds_pc, divorce_testing_pc$Class)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 68  0
##          1  0 67
##
##                Accuracy : 1
##                  95% CI : (0.973, 1)
##     No Information Rate : 0.5037
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.5037
##          Detection Rate : 0.5037
##    Detection Prevalence : 0.5037
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : 0
##
```

```r
### Accuracy table
# Update the accuracy table base on testing data
summary_accuracy <- bind_rows(summary_accuracy,
                    tibble(Method = "Logistic Regression PC",
                           Accuracy = 1))
# Show the RMSE improvement
knitr::kable(summary_accuracy)%>%
  kableExtra::kable_styling(position = "center", latex_options = "striped")
```

| Method | Accuracy |
|---|---|
| Baseline Prediction | 0.6285 |
| Logistic Regression | 0.9429 |
| Logistic Regression PC | 1.0000 |

### kNN Model

- Model with all the data

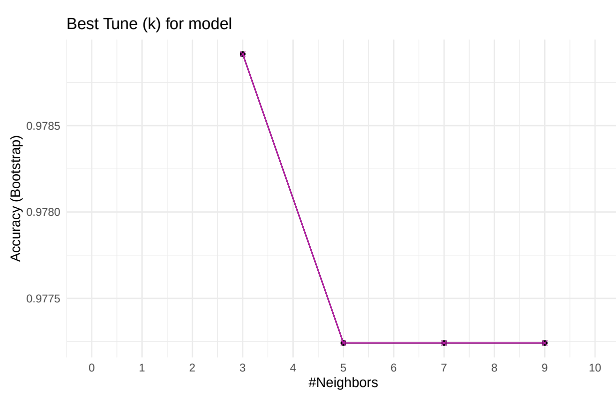
The kNN model (k-nearest neighbors) is a classification algorithm that consists of finding areas called neighbors. It will group the data points in areas measuring the distance between the points. The final value is the value more common among the neighbor.

First, we finding best tune. For this model the best parameter is 3. It means that with 3 neighbors the model can classify the information accurately. Additional neighbors do not improve the accuracy, but actually decreases it.

```
set.seed(3, sample.kind = "Rounding")    # simulate R 3.5
train_knn_divorce <- train(Class ~ .,
                     method = "knn",
                     data = divorce_training,
                     tuneGrid = data.frame(k = seq(3, 30, 2)))
train_knn_divorce$bestTune
```

```
##   k
## 1 3
```

```
ggplot2::ggplot(train_knn_divorce) +
  ggplot2::geom_line(colour="#ba1ea8") +
  ggplot2::geom_point(colour="#ba1ea8", shape=4)+
  ggplot2::scale_x_continuous(limits = c(0,10), breaks=seq(0,12,1)) +
  ggplot2::ggtitle("Best Tune (k) for model")+
  theme_minimal()
```




```
knn_preds_divorce <- predict(train_knn_divorce, divorce_testing)
mean(knn_preds_divorce == divorce_testing$Class)
```

```
## [1] 0.9714286
```

```r
### Accuracy table
# Update table
summary_accuracy <- bind_rows(summary_accuracy,
                    tibble(Method = "kNN original data",
                           Accuracy = 0.9714))
# Show table
knitr::kable(summary_accuracy)%>%
  kableExtra::kable_styling(position = "center", latex_options = "striped")
```

| Method | Accuracy |
|---|---|
| Baseline Prediction | 0.6285 |
| Logistic Regression | 0.9429 |
| Logistic Regression PC | 1.0000 |
| kNN original data | 0.9714 |

- kNN Crossvalidation

Now I want to try crossvalidation to see if we can get a better model. Crossvalidation here is a nice tool since the dataset is small.

```r
set.seed(8, sample.kind = "Rounding")    # simulate R 3.5
train_knn_cv_divorce <- train(Class ~ .,
                         method = "knn",
                         data = divorce_training,
                         tuneGrid = data.frame(k = seq(3, 30, 2)),
                         trControl = trainControl(method = "cv", number = 10, p = 0.9))
train_knn_cv_divorce$bestTune
```

```
##     k
## 14 29
```

```r
knn_cv_preds_divorce <- predict(train_knn_cv_divorce, divorce_testing)
mean(knn_cv_preds_divorce == divorce_testing$Class)
```

```
## [1] 0.9714286
```

```r
### Accuracy table
# Update the table
summary_accuracy <- bind_rows(summary_accuracy,
                    tibble(Method = "kNN original data Cross Validation",
                           Accuracy = 0.9714))
# Show table
knitr::kable(summary_accuracy)%>%
  kableExtra::kable_styling(position = "center", latex_options = "striped")
```

| Method | Accuracy |
|---|---|
| Baseline Prediction | 0.6285 |
| Logistic Regression | 0.9429 |
| Logistic Regression PC | 1.0000 |
| kNN original data | 0.9714 |
| kNN original data Cross Validation | 0.9714 |

Crossvalidation in this case does not improve the accuracy.

- kNN with only PCA

Now I want to test if I get better results using the data with principal components. For this model, k is the same as the one obtained with the full training data.

```
set.seed(3, sample.kind = "Rounding")     # simulate R 3.5
train_knn_divorce_pc <- train(Class ~ .,
                     method = "knn",
                     data = divorce_training_pc,
                     tuneGrid = data.frame(k = seq(3, 30, 2)))
train_knn_divorce_pc$bestTune
```

```
##   k
## 1 3
```

```
knn_cv_preds_divorce_pc <- predict(train_knn_divorce_pc, divorce_testing_pc)
mean(knn_cv_preds_divorce_pc == divorce_testing_pc$Class)
```

```
## [1] 0.9777778
```

```
### Accuracy table
# Update the error table
summary_accuracy <- bind_rows(summary_accuracy,
                    tibble(Method = "kNN PC",
                           Accuracy = 0.9778))
# Show the RMSE improvement
knitr::kable(summary_accuracy)%>%
  kableExtra::kable_styling(position = "center", latex_options = "striped")
```

| Method | Accuracy |
|---|---|
| Baseline Prediction | 0.6285 |
| Logistic Regression | 0.9429 |
| Logistic Regression PC | 1.0000 |
| kNN original data | 0.9714 |
| kNN original data Cross Validation | 0.9714 |
| kNN PC | 0.9778 |

### Classification Tree

- All the data

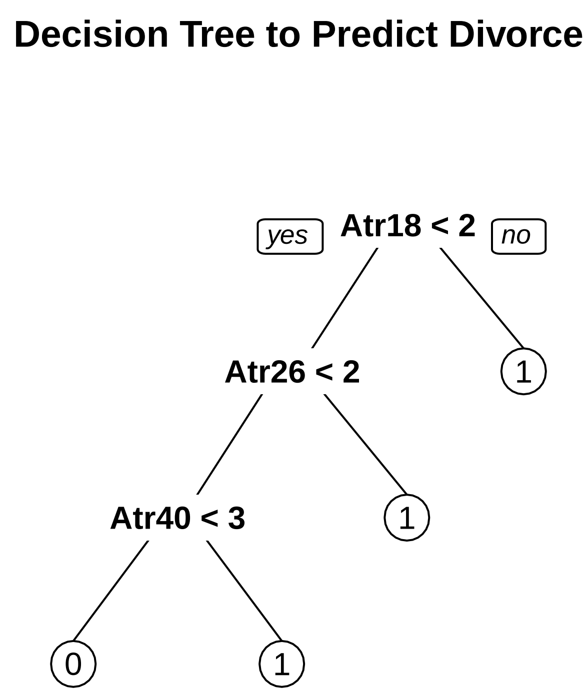
A classification tree starts by moving into branches, which are observations about the the output variables to predict it. This is an instintive model that starts splitting the data and evaluates if it contributes to the decision variable. The process is repeated in a recursive manner evaluating all the variables until the new branches does not add value to the outcome variable.

```r
train_rpart_divorce = rpart::rpart(Class ~ .,
                     data = divorce_training,
                     method = 'class',
                     control = rpart.control(minsplit=2),
                     model = TRUE)

## create prediction probabilities (on train dataset)
rpart_train_pred_probs = predict(train_rpart_divorce)

## create predictions (on train dataset)
rpart_train_preds = as.factor(ifelse(rpart_train_pred_probs[ , 2] > 0.5, 1, 0))
```

```r
rpart.plot::prp(train_rpart_divorce, main="Decision Tree to Predict Divorce")
```




```r
ct_preds_divorce <- predict(train_rpart_divorce, divorce_testing)
mean(ct_preds_divorce == divorce_testing$Class)
```

```
## [1] 0.5
```

```r
### Accuracy table
# Update the error table
summary_accuracy <- bind_rows(summary_accuracy,
                    tibble(Method = "Classification tree",
                           Accuracy = 0.5))
```

```r
# Show the Accuracy table
knitr::kable(summary_accuracy)%>%
  kableExtra::kable_styling(position = "center", latex_options = "striped")
```

| Method | Accuracy |
|---|---|
| Baseline Prediction | 0.6285 |
| Logistic Regression | 0.9429 |
| Logistic Regression PC | 1.0000 |
| kNN original data | 0.9714 |
| kNN original data Cross Validation | 0.9714 |
| kNN PC | 0.9778 |
| Classification tree | 0.5000 |

```r
varImp(train_rpart_divorce)
```

```
##          Overall
## Atr11 59.9407407
## Atr17 61.6425414
## Atr18 61.7498174
## Atr19 61.6425414
## Atr20 61.6872196
## Atr26  3.7754644
## Atr28  1.8607646
## Atr3   0.9710145
## Atr34  0.3043478
## Atr36  2.4140108
## Atr39  0.9710145
## Atr40  3.8317791
## Atr1   0.0000000
## Atr2   0.0000000
## Atr4   0.0000000
## Atr5   0.0000000
## Atr6   0.0000000
## Atr7   0.0000000
## Atr8   0.0000000
## Atr9   0.0000000
## Atr10  0.0000000
## Atr12  0.0000000
## Atr13  0.0000000
## Atr14  0.0000000
## Atr15  0.0000000
## Atr16  0.0000000
## Atr21  0.0000000
## Atr22  0.0000000
## Atr23  0.0000000
## Atr24  0.0000000
## Atr25  0.0000000
## Atr27  0.0000000
## Atr29  0.0000000
## Atr30  0.0000000
## Atr31  0.0000000
## Atr32  0.0000000
```

```
## Atr33  0.0000000
## Atr35  0.0000000
## Atr37  0.0000000
## Atr38  0.0000000
## Atr41  0.0000000
## Atr42  0.0000000
## Atr43  0.0000000
## Atr44  0.0000000
## Atr45  0.0000000
## Atr46  0.0000000
## Atr47  0.0000000
## Atr48  0.0000000
## Atr49  0.0000000
## Atr50  0.0000000
## Atr51  0.0000000
## Atr52  0.0000000
## Atr53  0.0000000
## Atr54  0.0000000
```

From the classification tree we found that the main attributes to predict divorce are Atrr 20,19,18,17,11,26,28,3,34,36,39 and 40. All this questions are related to knowing your spouse and her/his needs.When people have low scores on those questions that highly predicts divorce.

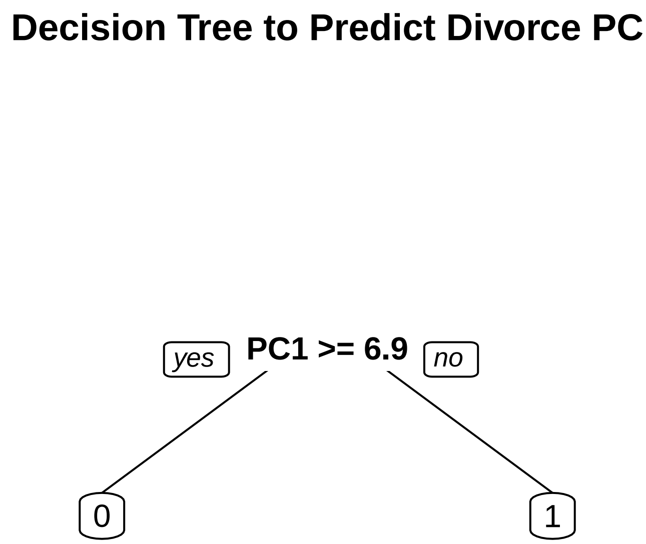
- Classification tree principal components

```
train_rpart_divorce_pc = rpart::rpart(Class ~ .,
                     data = divorce_training_pc,
                     method = 'class',
                     control = rpart.control(minsplit=2),
                     model = TRUE)

## create prediction probabilities (on train dataset)
rpart_train_pred_probs_pc = predict(train_rpart_divorce_pc)

## create predictions (on train dataset)
rpart_train_preds_pc = as.factor(ifelse(rpart_train_pred_probs_pc[ , 2] > 0.5, 1, 0))
```

```
rpart.plot::prp(train_rpart_divorce_pc , main="Decision Tree to Predict Divorce PC")
```




Since the first principal component explains 73% of the variability, it is not surprised that PC1 is picked on the decision tree. However the accuracy is not improved.

Accuracy

```
ct_preds_divorce_pc <- predict(train_rpart_divorce_pc, divorce_testing_pc)
mean(ct_preds_divorce_pc == divorce_testing_pc$Class)
```

```
## [1] 0.5
```

```
### Accuracy table
# Update the error table
summary_accuracy <- bind_rows(summary_accuracy,
                    tibble(Method = "Classification tree PC",
                           Accuracy = 0.5))
# Show the Accuracy table
knitr::kable(summary_accuracy)%>%
  kableExtra::kable_styling(position = "center", latex_options = "striped")
```

| Method | Accuracy |
|---|---|
| Baseline Prediction | 0.6285 |
| Logistic Regression | 0.9429 |
| Logistic Regression PC | 1.0000 |
| kNN original data | 0.9714 |
| kNN original data Cross Validation | 0.9714 |
| kNN PC | 0.9778 |
| Classification tree | 0.5000 |
| Classification tree PC | 0.5000 |

### Random Forest

- All data

```
set.seed(14, sample.kind = "Rounding")     # simulate R 3.5
train_rf_divorce <- train(Class ~ .,
                    data = divorce_training,
                    method = "rf",
                    ntree = 100,
                    tuneGrid = data.frame(mtry = seq(1:7)))
train_rf_divorce$bestTune
```

```
##   mtry
## 1    1
```

```
rf_preds_divorce <- predict(train_rf_divorce, divorce_testing)
mean(rf_preds_divorce == divorce_testing$Class)
```

```
## [1] 0.9714286
```

```
varImp(train_rf_divorce)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 54)
##
##         Overall
## Atr23  100.00
## Atr40   99.80
## Atr36   80.88
## Atr9    74.12
## Atr11   62.61
## Atr17   60.54
## Atr38   57.16
## Atr44   56.42
## Atr41   54.81
## Atr21   54.48
## Atr50   50.18
## Atr27   48.95
## Atr39   48.30
```

```
## Atr26   43.84
## Atr20   43.04
## Atr54   38.87
## Atr33   38.44
## Atr3    36.36
## Atr10   32.97
## Atr31   31.81
```

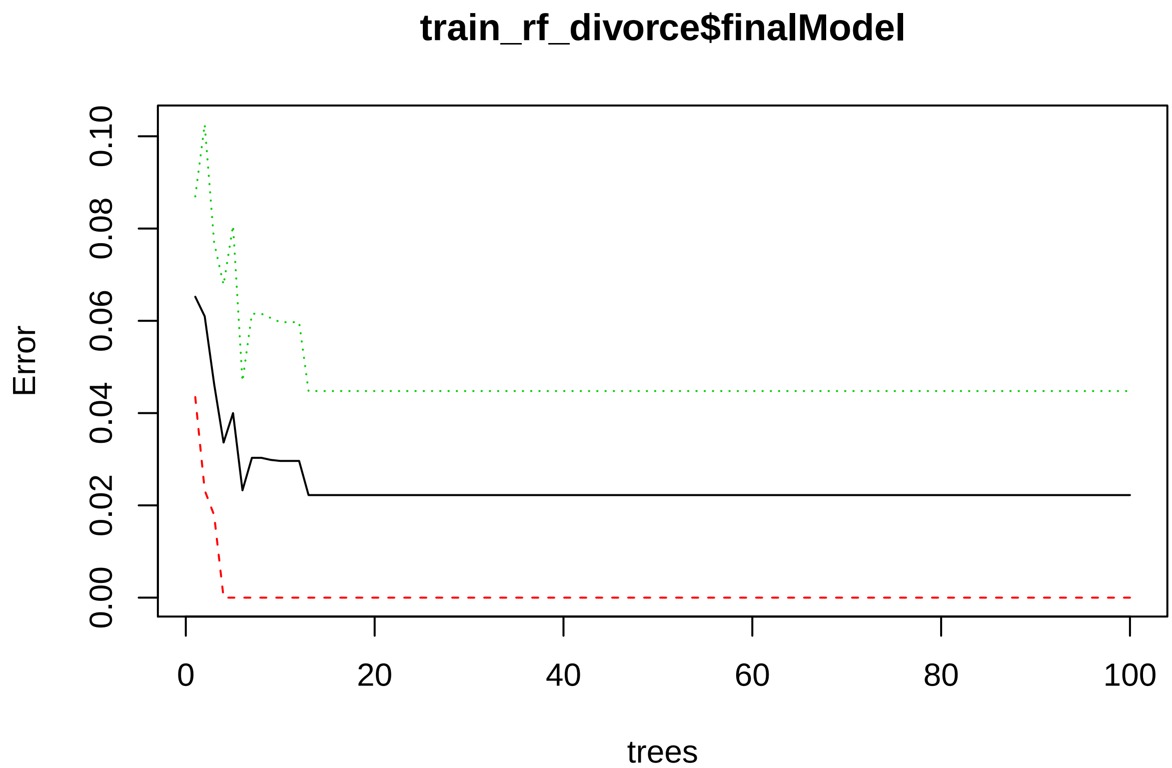
Random forest shows the top 3 variables predicting divorce are Atr 23,40,36.

23. I know my spouse's favorite food.
24. I can be humiliating when we argue.
25. We're just starting a fight before I know what's going on.

```r
train_rf_divorce$finalModel # inspect final model
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 100, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 2.22%
## Confusion matrix:
##    0  1 class.error
## 0 68  0  0.00000000
## 1  3 64  0.04477612
```

```r
# make plot of decision tree
plot(train_rf_divorce$finalModel, margin = 0.1)
```




The error graphs shows that even though the algorithm ran 100 trees, the error stays constant from tree number 15.

```
train_rf <- randomForest(Class ~ ., data=divorce_training)

confusionMatrix(predict(train_rf, divorce_testing),
                divorce_testing$Class)$overall["Accuracy"]
```

```
##  Accuracy
## 0.9714286
```

```
### Accuracy table
# Update the error table
summary_accuracy <- bind_rows(summary_accuracy,
                    tibble(Method = "Random Forest",
                           Accuracy = 0.9714))
# Show table
knitr::kable(summary_accuracy)%>%
  kableExtra::kable_styling(position = "center", latex_options = "striped")
```

| Method | Accuracy |
|---|---|
| Baseline Prediction | 0.6285 |
| Logistic Regression | 0.9429 |
| Logistic Regression PC | 1.0000 |
| kNN original data | 0.9714 |
| kNN original data Cross Validation | 0.9714 |
| kNN PC | 0.9778 |
| Classification tree | 0.5000 |
| Classification tree PC | 0.5000 |
| Random Forest | 0.9714 |

- Random Forest with Principal Components

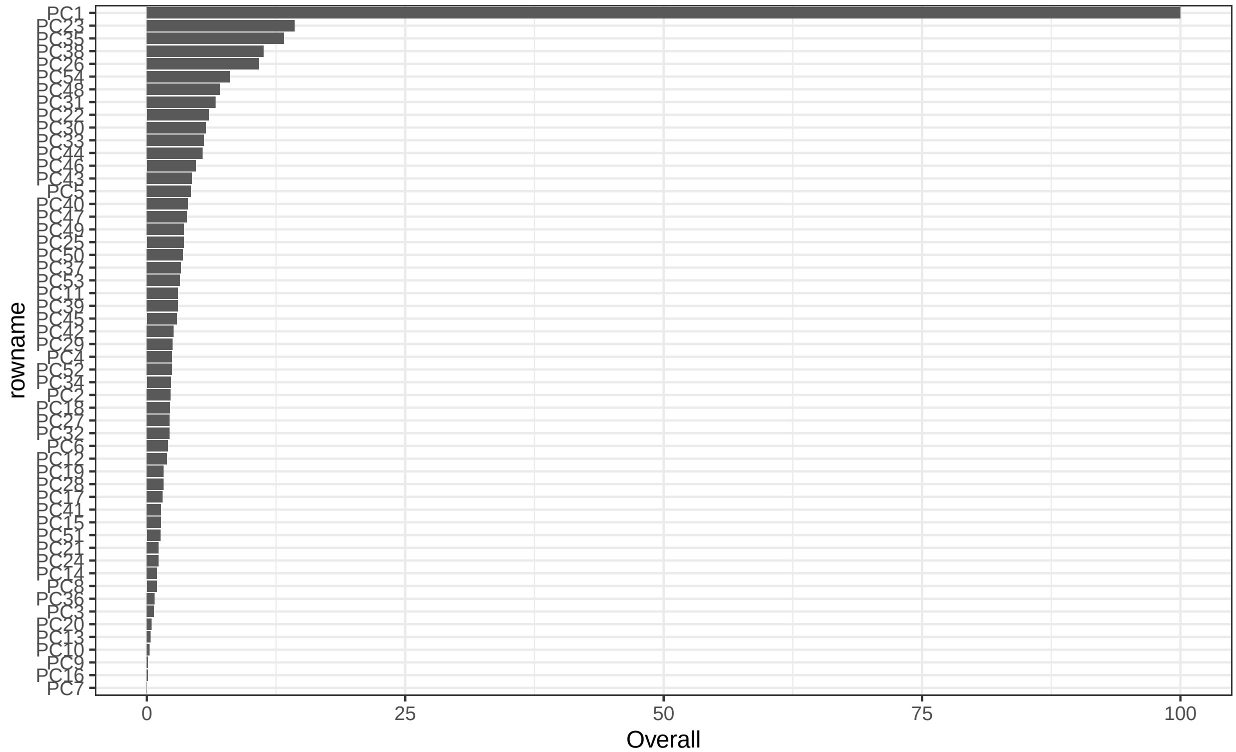
```
set.seed(14, sample.kind = "Rounding")     # simulate R 3.5
train_rf_divorce_pc <- train(Class ~ .,
                    data = divorce_training_pc,
                    method = "rf",
                    ntree = 100,
                    tuneGrid = data.frame(mtry = seq(1:7)))
train_rf_divorce$bestTune
```

```
##   mtry
## 1    1
```

```
rf_preds_divorce_pc <- predict(train_rf_divorce_pc, divorce_testing_pc)
mean(rf_preds_divorce_pc == divorce_testing_pc$Class)
```

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

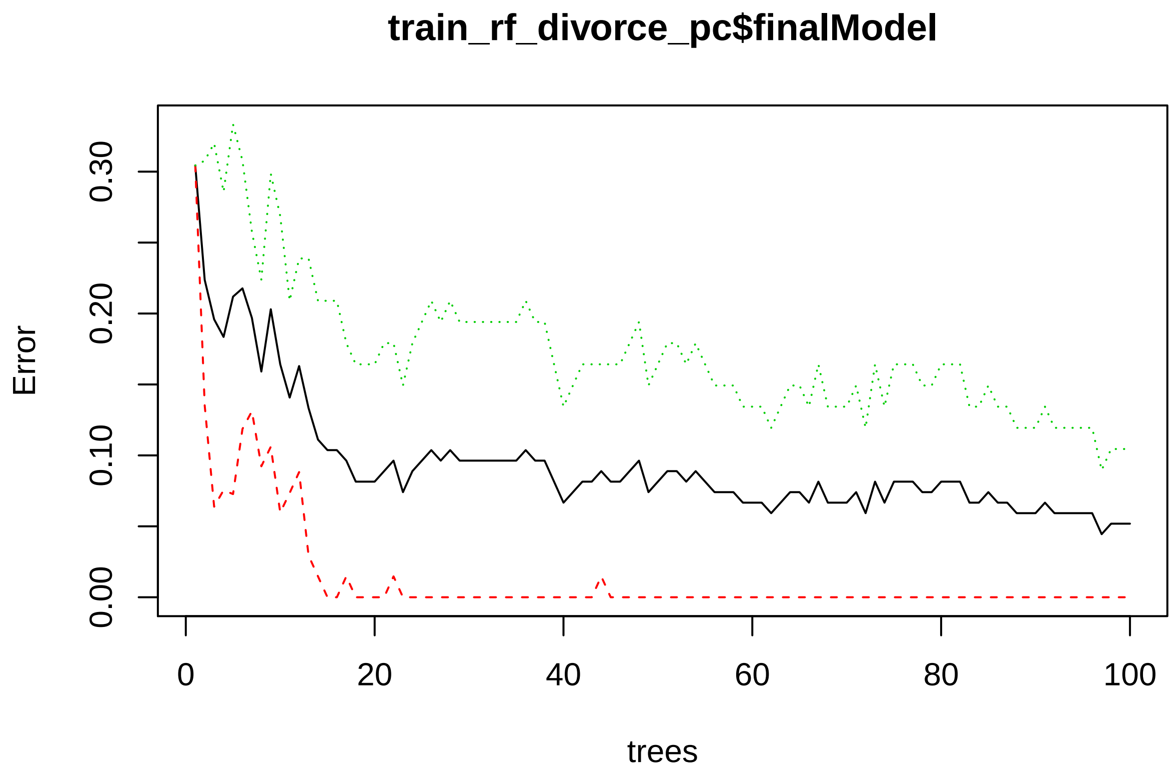
```
varImp(train_rf_divorce_pc)$importance %>%
  as.data.frame() %>%
  rownames_to_column() %>%
  arrange(Overall) %>%
  mutate(rowname = forcats::fct_inorder(rowname )) %>%
  ggplot()+
    geom_col(aes(x = rowname, y = Overall))+
    coord_flip()+
    theme_bw()
```




```r
train_rf_divorce_pc$finalModel # inspect final model
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 100, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 5.19%
## Confusion matrix:
##    0  1 class.error
## 0 68  0   0.0000000
## 1  7 60   0.1044776
```

```r
# make plot of decision tree
plot(train_rf_divorce_pc$finalModel, margin = 0.1)
```




```
#text(train_rf_divorce$finalModel)
```

The error graphs shows that even though the algorithm ran 100 trees, the error stays constant after tree number 20.

```
train_rf_pc <- randomForest(Class ~ ., data=divorce_training_pc)

confusionMatrix(predict(train_rf_pc, divorce_testing_pc),
                divorce_testing_pc$Class)$overall["Accuracy"]
```

```
## Accuracy
##        1
```

```
### Accuracy table
# Update the error table
summary_accuracy <- bind_rows(summary_accuracy,
                    tibble(Method = "Random Forest PC",
                           Accuracy = 1))
# Show the RMSE improvement
knitr::kable(summary_accuracy)%>%
  kableExtra::kable_styling(position = "center", latex_options = "striped")
```

| Method | Accuracy |
|---|---|
| Baseline Prediction | 0.6285 |
| Logistic Regression | 0.9429 |
| Logistic Regression PC | 1.0000 |
| kNN original data | 0.9714 |
| kNN original data Cross Validation | 0.9714 |
| kNN PC | 0.9778 |
| Classification tree | 0.5000 |
| Classification tree PC | 0.5000 |
| Random Forest | 0.9714 |
| Random Forest PC | 1.0000 |

## Results

After running logistic regression, kNN method, classification tree and Random Forest on the divorce dataset to predict divorce in couples based on their answers to the survey, I found that the best models are Logistic Regression and Random Forest when using Principal Components in both with Accuracy of 1. The worse model was classification tree. It seems to be because once it takes one branch, adding other variables are not very powerful when predicting in the test data. Random Forest and kNN with the original data have the same accuracy.

Even though this is a very subjective topic,it is interesting that from random forest the main variable to predict divorce is not knowing something basic as your spouse's favorite food. The other 2 variables are related to unhealthy behaviors on the relationship.

*Main variables that predict divorce based on Random Forest with all data*

23. I know my spouse's favorite food.
24. I can be humiliating when we argue.
25. We're just starting a fight before I know what's going on.

```
knitr::kable(summary_accuracy)%>%
  kableExtra::kable_styling(position = "center", latex_options = "striped")
```

| Method | Accuracy |
|---|---|
| Baseline Prediction | 0.6285 |
| Logistic Regression | 0.9429 |
| Logistic Regression PC | 1.0000 |
| kNN original data | 0.9714 |
| kNN original data Cross Validation | 0.9714 |
| kNN PC | 0.9778 |
| Classification tree | 0.5000 |
| Classification tree PC | 0.5000 |
| Random Forest | 0.9714 |
| Random Forest PC | 1.0000 |

## Conclusion

This was a great exercise to compare different classification methods. I wanted to explore this type of models since this is usually applied for situations that are more difficult to represent in a model. In this case, evaluating whether a person would get divorce is very subjective and it is a field out of my expertise.

Just starting, the data exploration requires creativity, We can't have regular histograms or counts like what you do with continuous variables. An additional issue, is that we have more than 50 predictors. Trying to understand all of them was not easy. It was helpful to use the *for* loop.

On the modeling side, it was useful to run the models to see how much the accuracy changes if you use the original data or principal components. Regarding Principal Components, since in this dataset the first component explained more than 70% of the variability it was not surprising that this was the best predictor.

## References

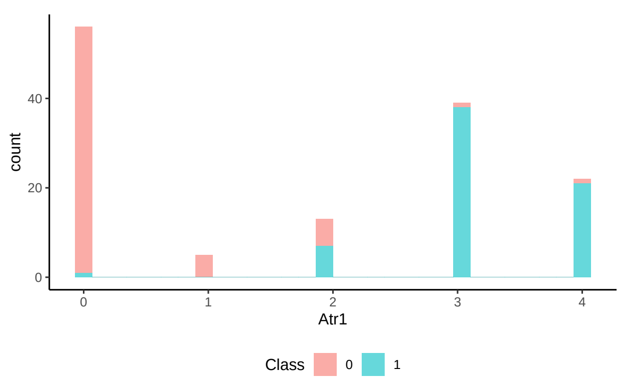

1. Divorce Analysis in R by Howard, https://www.rpubs.com/howard_song/538809
2. Data Science Book from Rafael Irizarry, https://rafalab.github.io/dsbook/
3. Predicting whether a couple is going to get divorced or not using artificial neural networks by Ibrahim M. Nasser: http://dstore.alazhar.edu.ps/xmlui/bitstream/handle/123456789/545/IJEAIS191007.pdf?sequence=1&isAllowed=y
4. Initial research by Mustafa Kemal, Kemal Adem, Tahsin Ilhan and Serhat Kilicarslan: Divorce prediction using correlation based feature selection and artificial neural networks https://dergipark.org.tr/en/download/article-file/748448


## Others

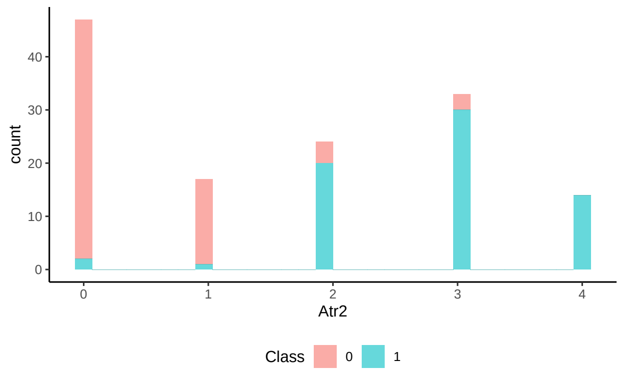
Dataset variables histograms

```
histograms
```

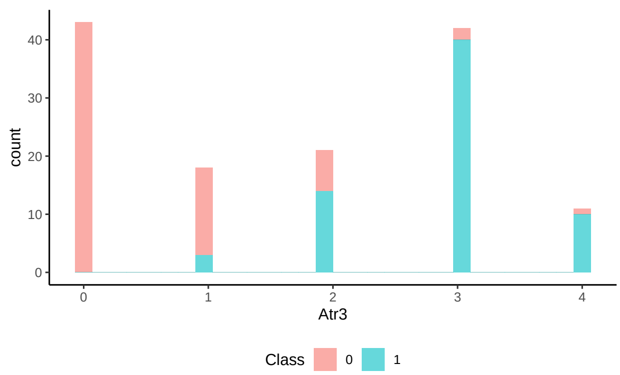
```
## [[1]]
```



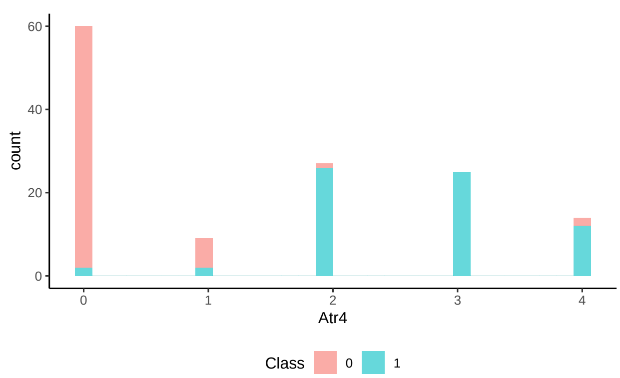

```
##
## [[2]]
```



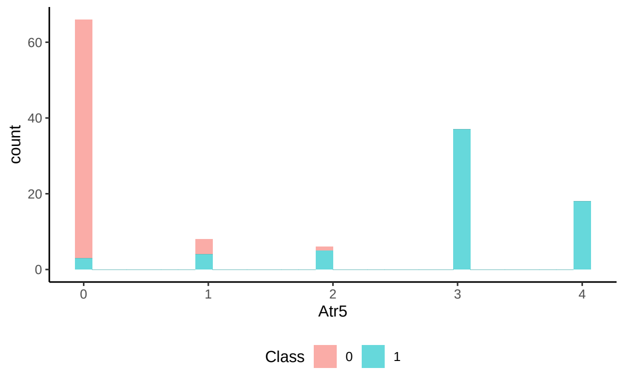

```
##
## [[3]]
```



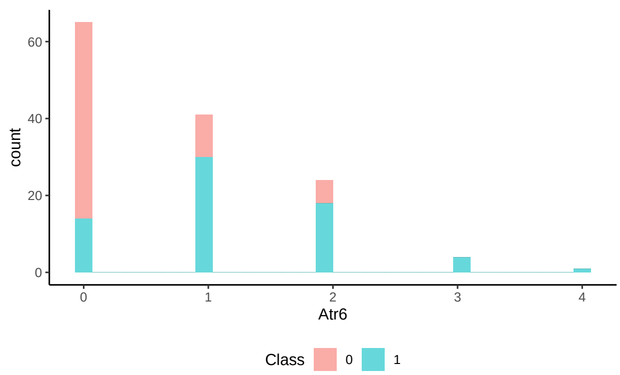

```
##
## [[4]]
```



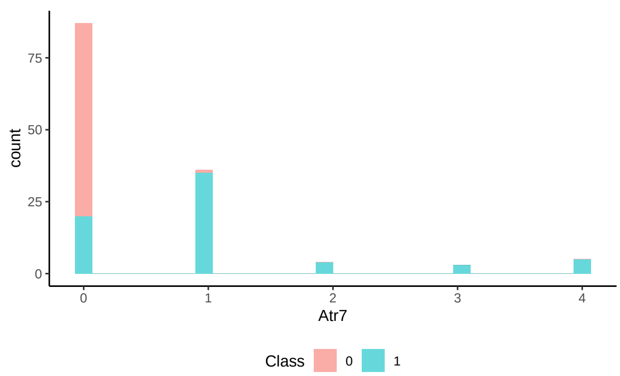

```
##
## [[5]]
```


count
60
40
20
0
0
1
2
3
4
Atr5
Class
0
1

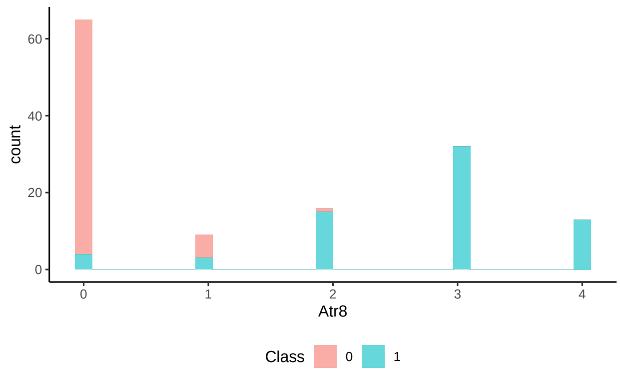
```
##
## [[6]]
```


count
60
40
20
0
0
1
2
3
4
Atr6
Class
0
1

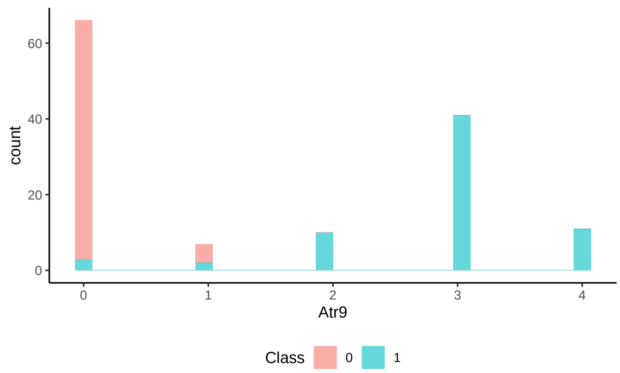
```
##
## [[7]]
```


count
75
50
25
0
0
1
2
3
4
Atr7
Class
0
1

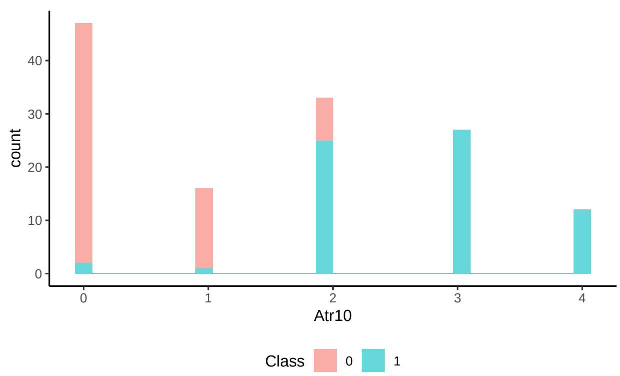
```
##
## [[8]]
```


60
40
20
0
count
0
1
2
3
4
Atr8
Class
0
1

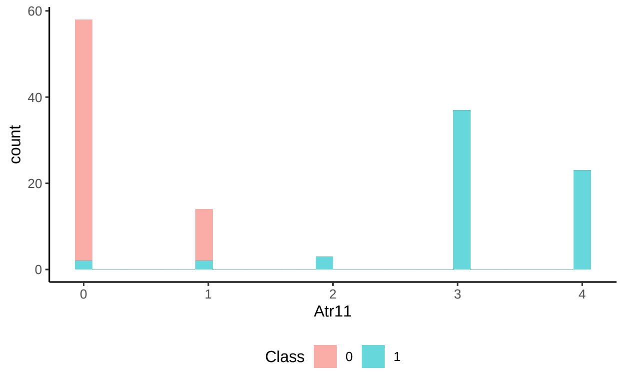
```
##
## [[9]]
```


60
40
20
0
count
0
1
2
3
4
Atr9
Class
0
1

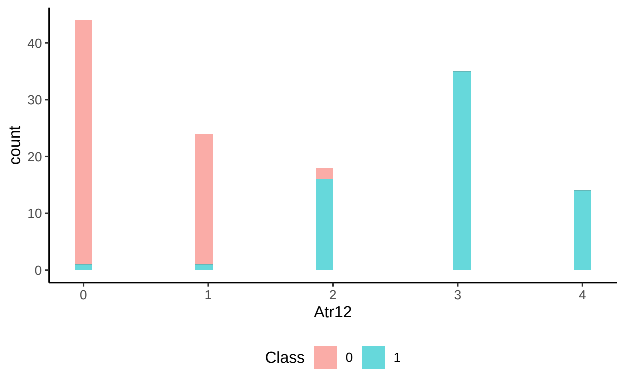
```
##
## [[10]]
```


40
30
20
10
0
count
0
1
2
3
4
Atr10
Class
0
1

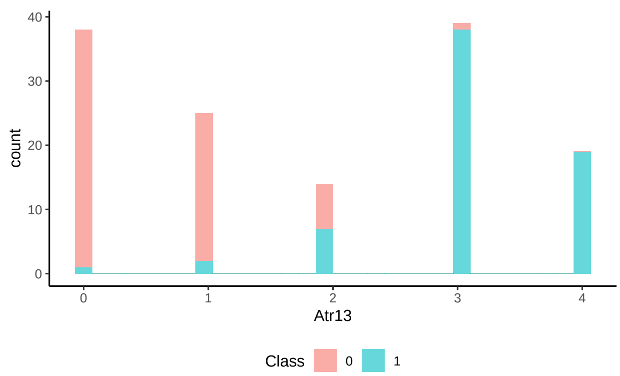
```
##
## [[11]]
```


60
40
20
0
count
0
1
2
3
4
Atr11
Class
0
1

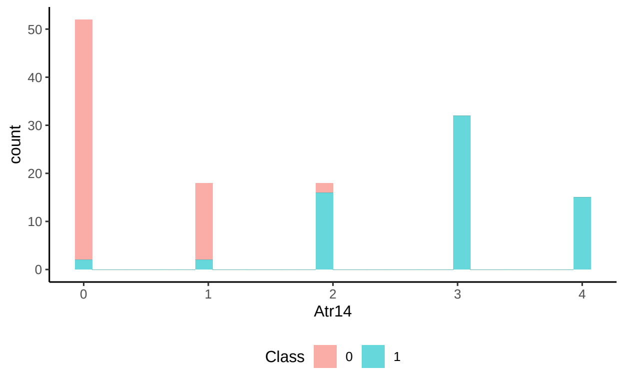
```
##
## [[12]]
```


40
30
20
10
0
count
0
1
2
3
4
Atr12
Class
0
1

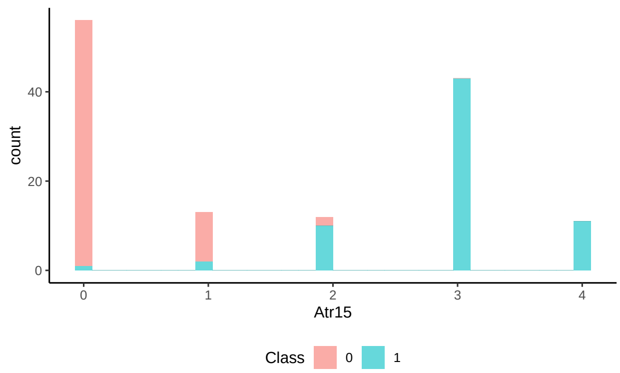
```
##
## [[13]]
```


40
30
20
10
0
count
0
1
2
3
4
Atr13
Class
0
1

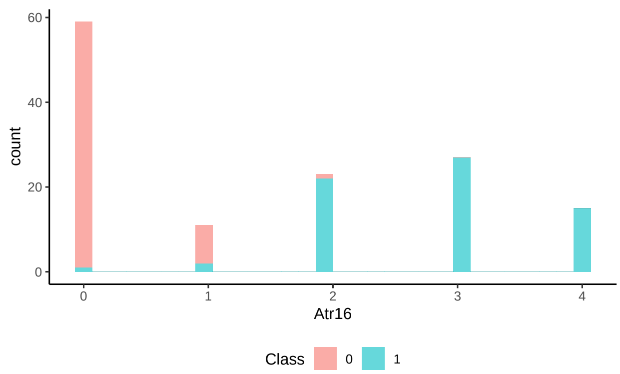
```
##
## [[14]]
```


count
Atr14
Class 0 1

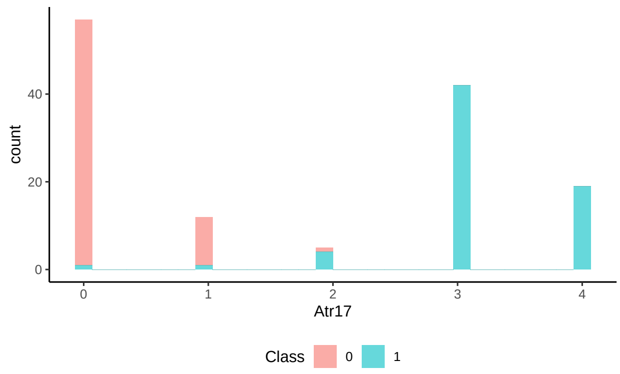
```
##
## [[15]]
```


count
Atr15
Class 0 1

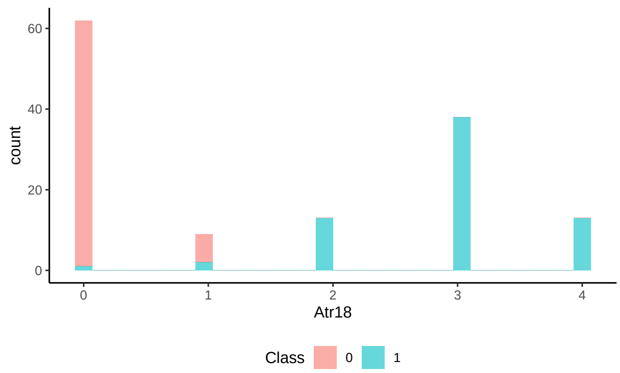
```
##
## [[16]]
```


count
Atr16
Class 0 1

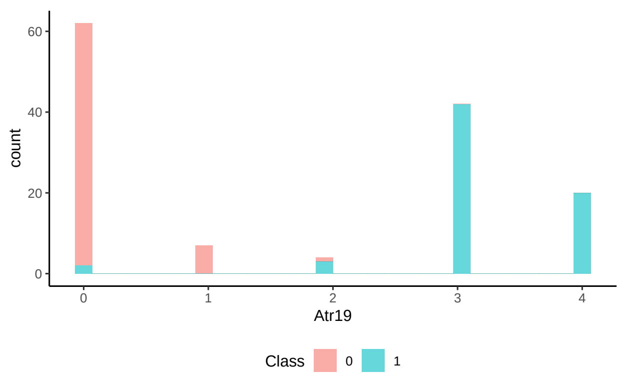
```
##
## [[17]]
```


count
40
20
0
0
1
2
3
4
Atr17
Class
0
1

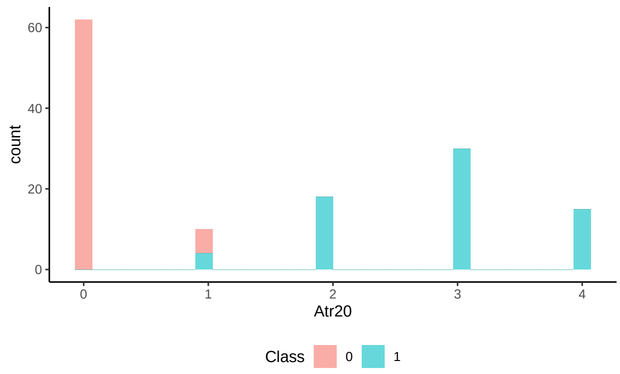
```
##
## [[18]]
```


count
60
40
20
0
0
1
2
3
4
Atr18
Class
0
1

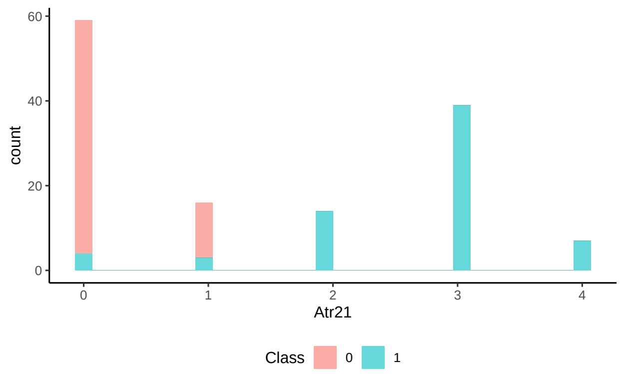
```
##
## [[19]]
```


count
60
40
20
0
0
1
2
3
4
Atr19
Class
0
1

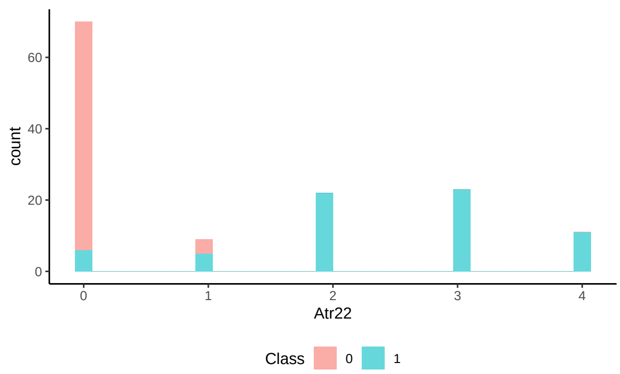
```
##
## [[20]]
```


60
40
20
0
count
0
1
2
3
4
Atr20
Class
0
1

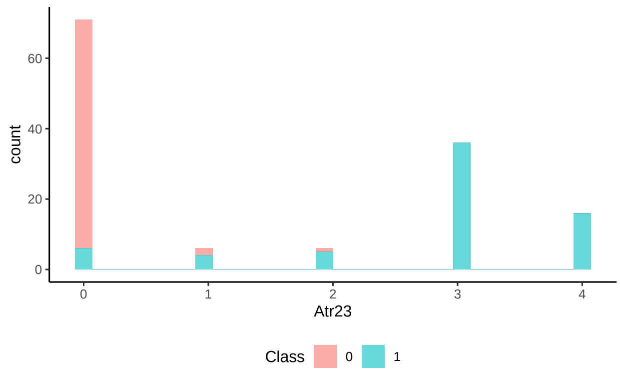
```
##
## [[21]]
```


60
40
20
0
count
0
1
2
3
4
Atr21
Class
0
1

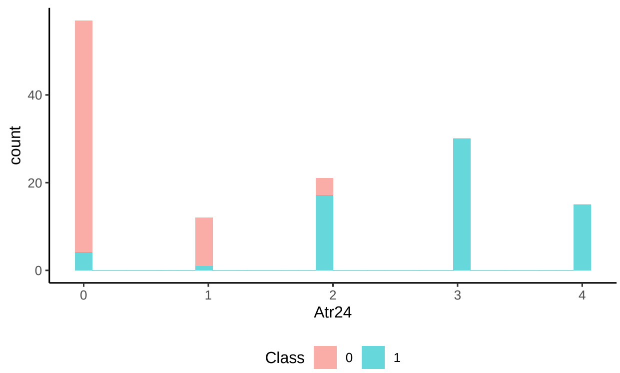
```
##
## [[22]]
```


60
40
20
0
count
0
1
2
3
4
Atr22
Class
0
1

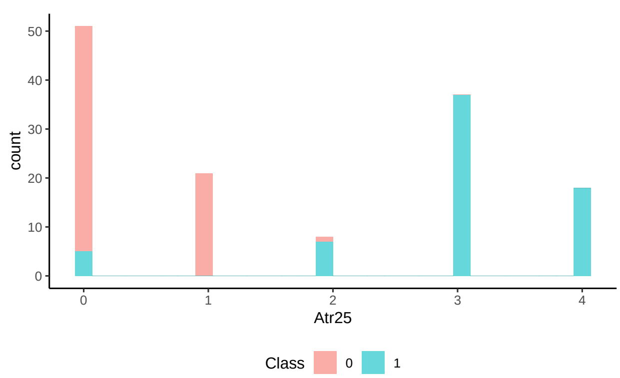
```
##
## [[23]]
```


count
60
40
20
0
0
1
2
3
4
Atr23
Class
0
1

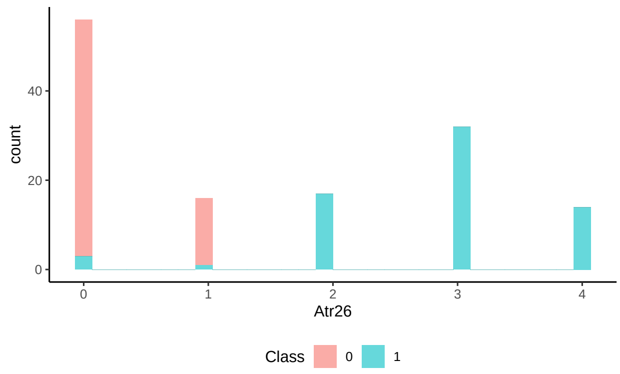
```
##
## [[24]]
```


count
40
20
0
0
1
2
3
4
Atr24
Class
0
1

```
##
## [[25]]
```

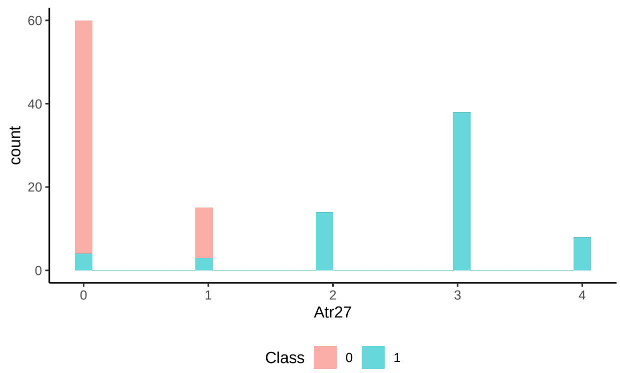

count
50
40
30
20
10
0
0
1
2
3
4
Atr25
Class
0
1

```
##
## [[26]]
```



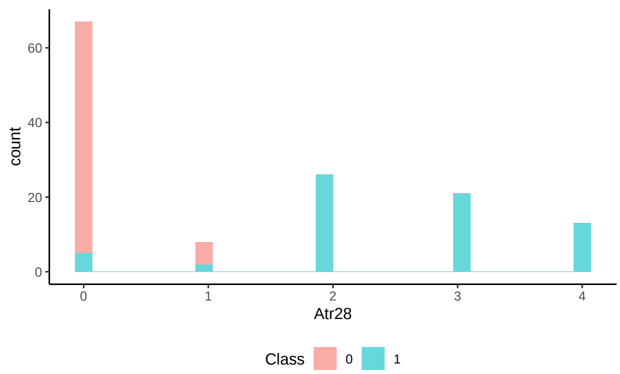
40
20
0
count
0
1
2
3
4
Atr26
Class 0 1


```
##
## [[27]]
```



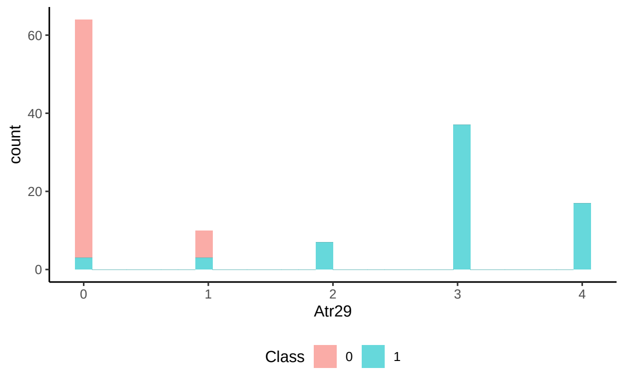
60
40
20
0
count
0
1
2
3
4
Atr27
Class 0 1


```
##
## [[28]]
```

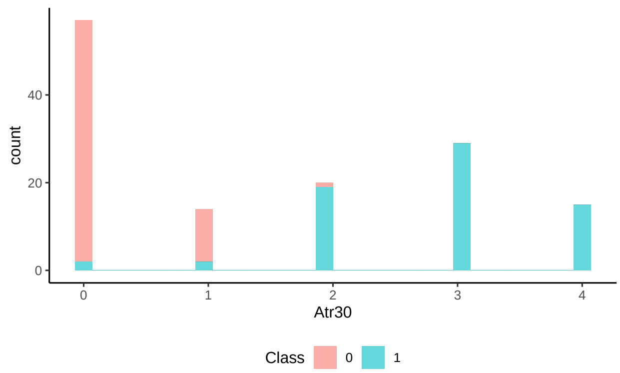


60
40
20
0
count
0
1
2
3
4
Atr28
Class 0 1

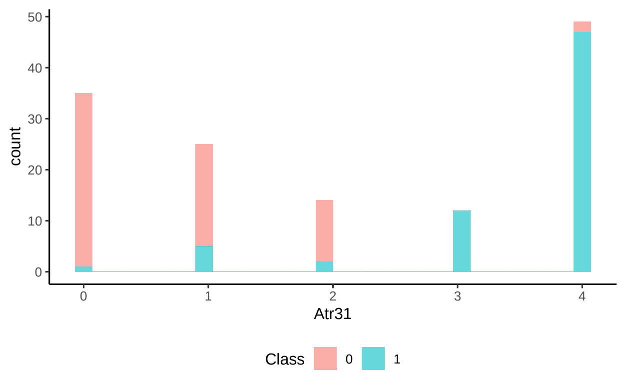

```
##
## [[29]]
```


60
40
20
0
count
0
1
2
3
4
Atr29
Class
0
1

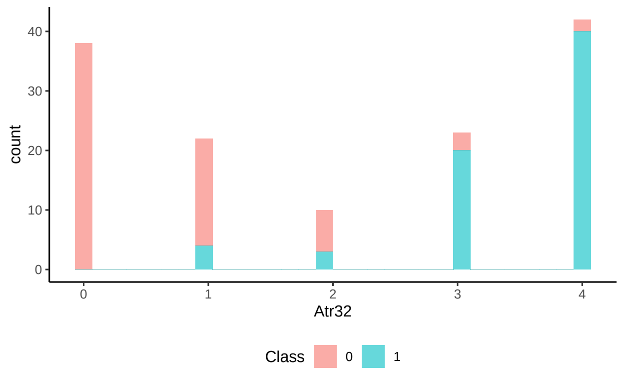
```
##
## [[30]]
```


40
20
0
count
0
1
2
3
4
Atr30
Class
0
1

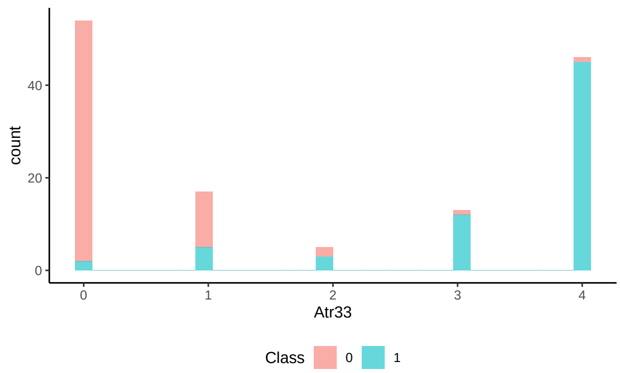
```
##
## [[31]]
```


50
40
30
20
10
0
count
0
1
2
3
4
Atr31
Class
0
1

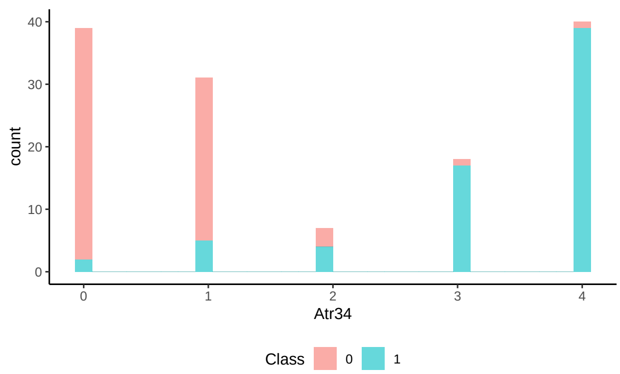
```
##
## [[32]]
```


40
30
20
10
0
count
0
1
2
3
4
Atr32
Class
0
1

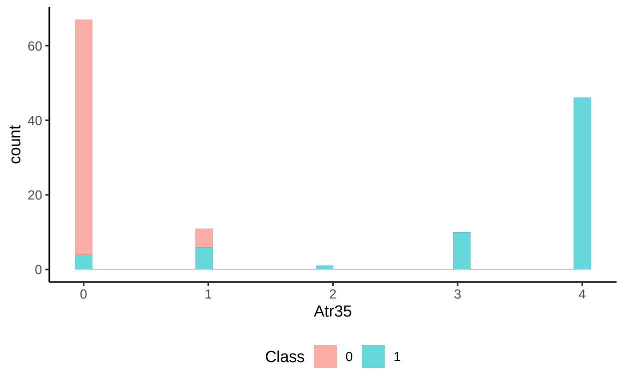
```
##
## [[33]]
```


40
20
0
count
0
1
2
3
4
Atr33
Class
0
1

```
##
## [[34]]
```

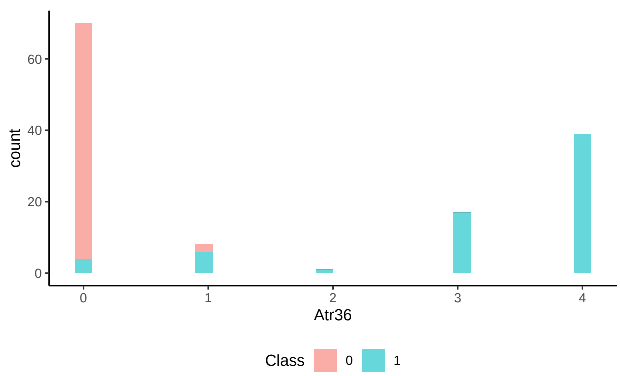

40
30
20
10
0
count
0
1
2
3
4
Atr34
Class
0
1

```
##
## [[35]]
```



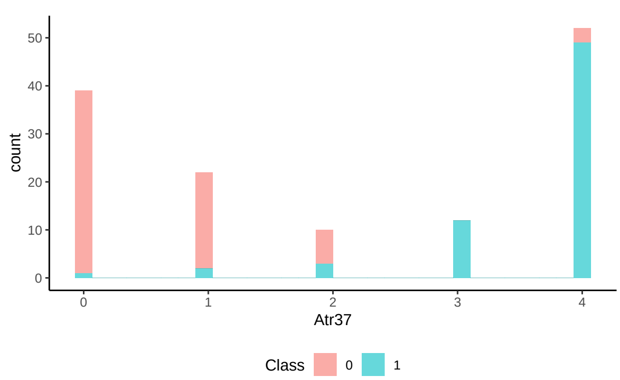
60
40
20
0
count
0
1
2
3
4
Atr35
Class
0
1


```
##
## [[36]]
```



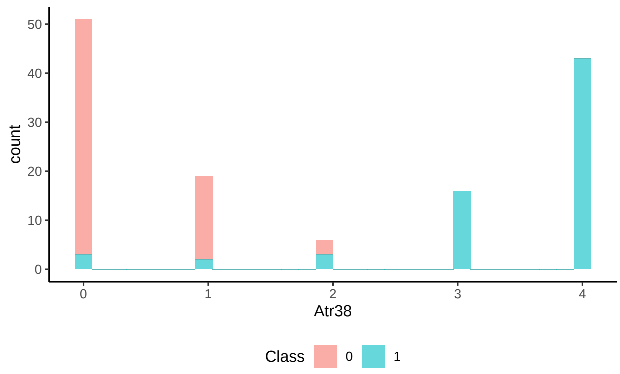
60
40
20
0
count
0
1
2
3
4
Atr36
Class
0
1


```
##
## [[37]]
```



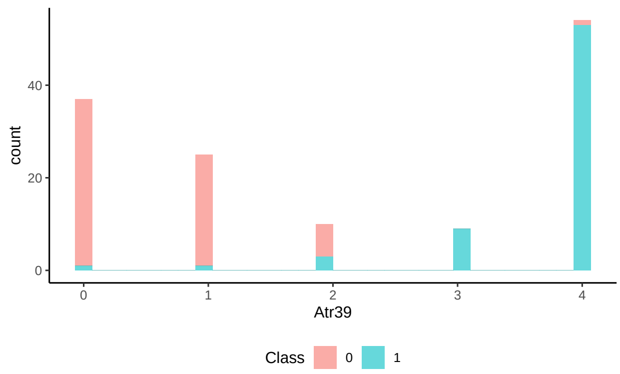
50
40
30
20
10
0
count
0
1
2
3
4
Atr37
Class
0
1


```
##
## [[38]]
```



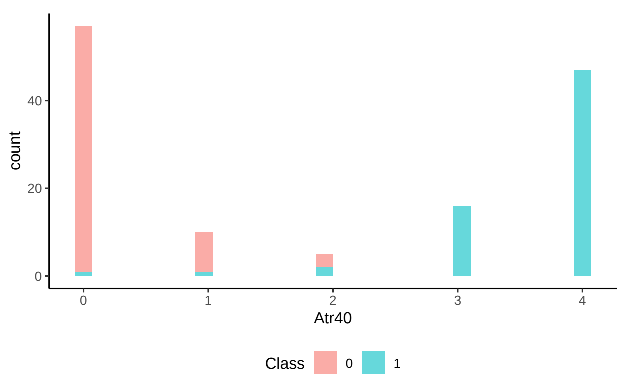
count
Atr38
Class
0
1


```
##
## [[39]]
```



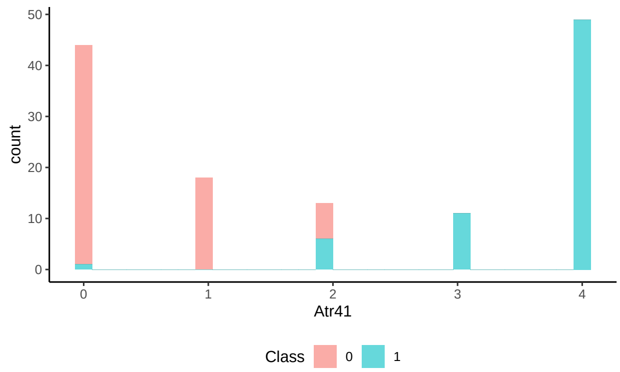
count
Atr39
Class
0
1


```
##
## [[40]]
```

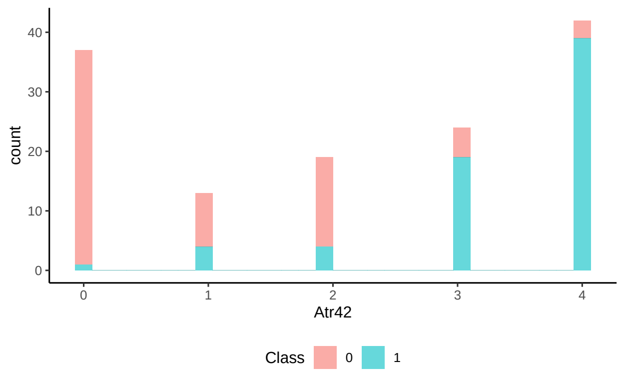


count
Atr40
Class
0
1

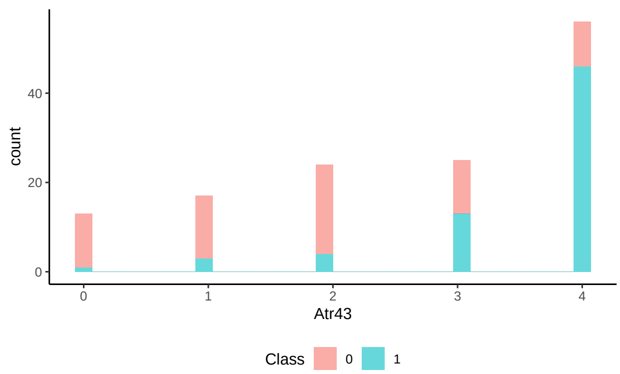

```
##
## [[41]]
```


50
40
30
20
10
0
count
0
1
2
3
4
Atr41
Class
0
1

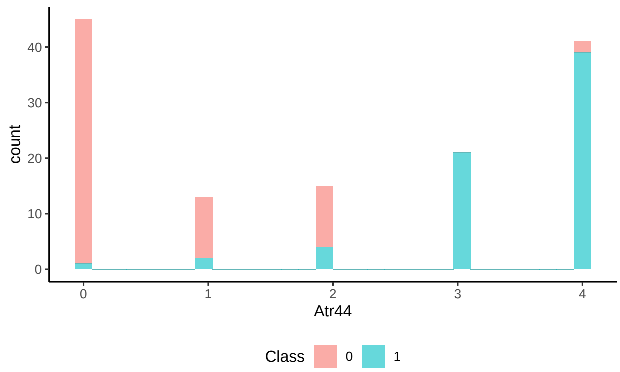
##
## [[42]]


40
30
20
10
0
count
0
1
2
3
4
Atr42
Class
0
1

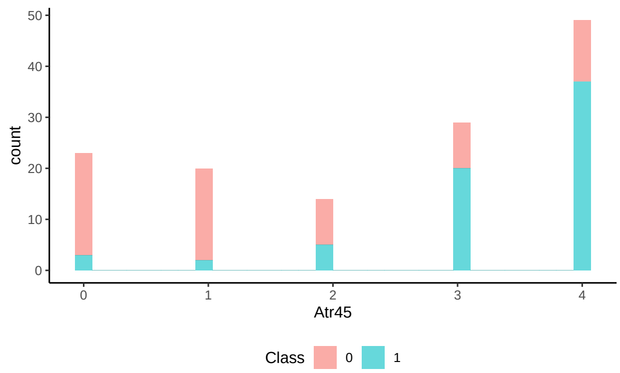
##
## [[43]]


40
20
0
count
0
1
2
3
4
Atr43
Class
0
1

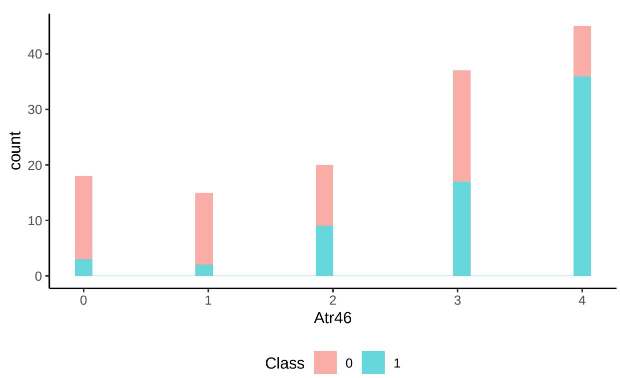
##
## [[44]]


count
Atr44
Class 0 1

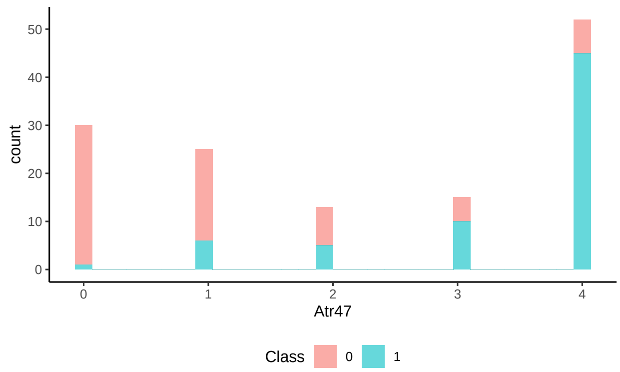
```
##
## [[45]]
```


count
Atr45
Class 0 1

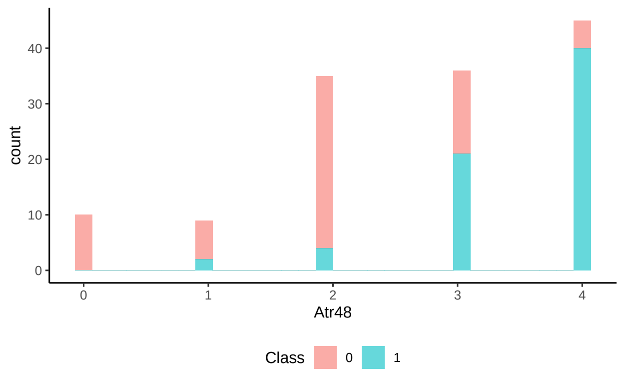
```
##
## [[46]]
```


count
Atr46
Class 0 1

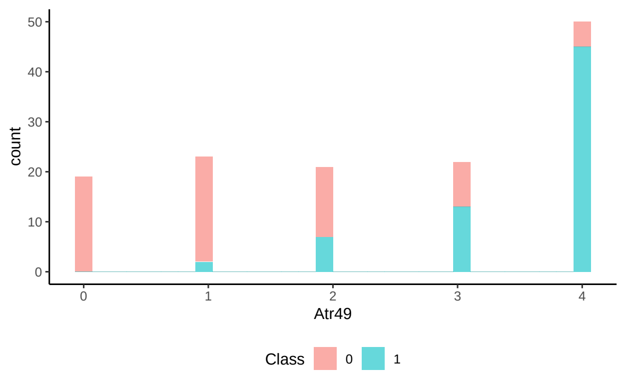
```
##
## [[47]]
```


count
Atr47
Class
0
1

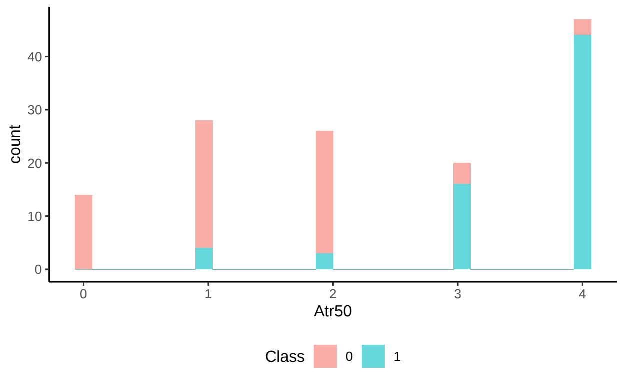
```
##
## [[48]]
```


count
Atr48
Class
0
1

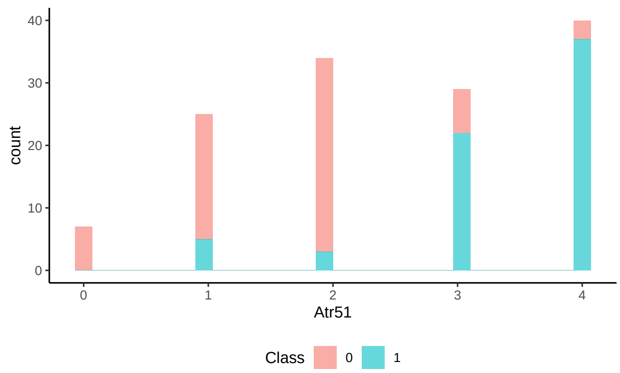
```
##
## [[49]]
```


count
Atr49
Class
0
1

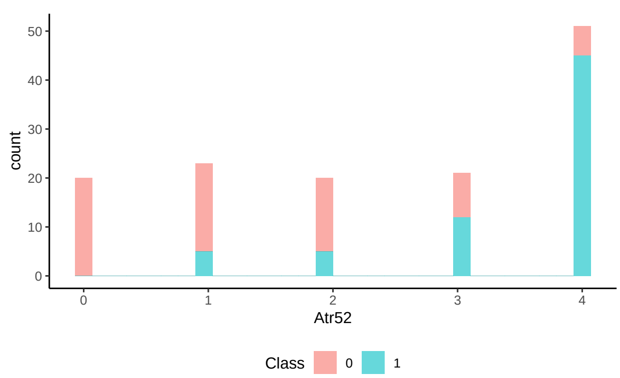
```
##
## [[50]]
```


40
30
20
10
0
count
0
1
2
3
4
Atr50
Class
0
1

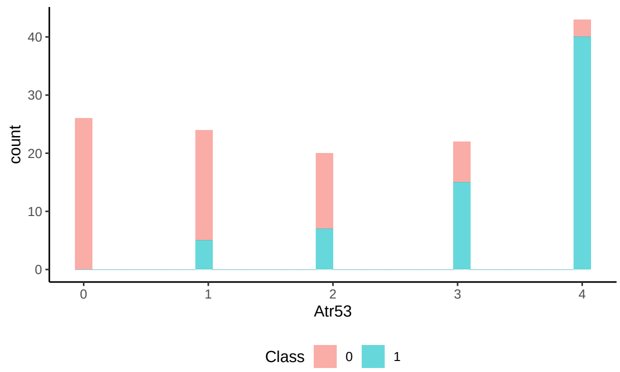
```
##
## [[51]]
```


40
30
20
10
0
count
0
1
2
3
4
Atr51
Class
0
1

```
##
## [[52]]
```

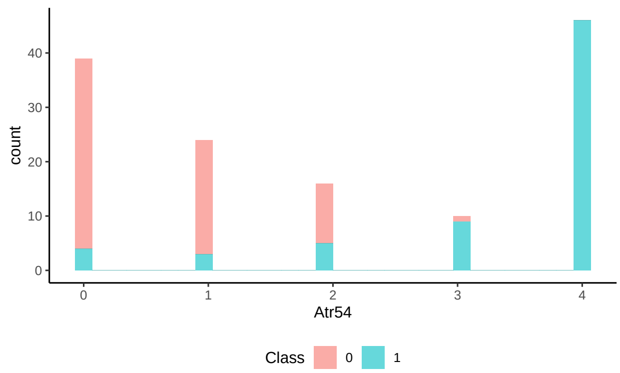

50
40
30
20
10
0
count
0
1
2
3
4
Atr52
Class
0
1

```
##
## [[53]]
```



40
30
20
10
0
count
0
1
2
3
4
Atr53
Class
0
1


```
##
## [[54]]
```



40
30
20
10
0
count
0
1
2
3
4
Atr54
Class
0
1