IBM HR Analytics Employee Attrition & Performance

Predict attrition of your valuable employees

Francesco Pudda

6/4/2020

Abstract

The meaning of attrition in a work environment refers to loss of employees through a number of circumstances in a given period of time. The cause of attrition may be either voluntary, like employees moving or retiring, taking another job, or considering themselves to be ill-suited to their position, or involuntary, such as a company's manoeuvre to reduce its cost. However, this does not come at no cost. It was shown, in fact, that the average cost to fill a vacant position due to preventable attrition is $4,129 (Mariotti, Robinson, and Esen 2017) that is a significant amount especially for a small company. Understanding thus, the main reasons causing attrition, may bring very valuable insights for companies willing to restrict such phenomenon.

Inspection and pre-processing

Such project was carried out by using the IBM dataset published at Kaggle (Subhash 2017). It is mostly important to state that this dataset is fictional and it was created by IBM. It is nonetheless important though, because by studying such dataset it is possible to gain insights about how to treat real data and clues on how to carry out an efficient exploratory and predictive analysis. Accordingly, every plot, every discussion and every suggestion based on results will be said as if data were real and applicable to real world problem, which could be actually the case, because even if fictional I suppose that it was based on some real estimates and not completely random (even though I do not have source to prove it).

These are the libraries used for this work:

```
library(tidyverse)
library(knitr)
library(kableExtra)
library(ggcorrplot)
library(caret)

# External function needed later
source("IBM-HR-DrawCM.R")
```

First thing to do is to load and visually inspect the dataset to understand how it is made up, how we can treat it, and if there are any missing values.

```
dataset <- read_csv("IBM-HR-Dataset.csv")
cat("Samples: ", nrow(dataset), "\nFeatures: ", ncol(dataset), "\n")
```

```
## Samples:  1470
## Features:  35
```

Age	EducationField	JobLevel	Over18	TotalWorkingYears
Attrition	EmployeeCount	JobRole	OverTime	TrainingTimesLastYear
BusinessTravel	EmployeeNumber	JobSatisfaction	PercentSalaryHike	WorkLifeBalance
DailyRate	EnvironmentSatisfaction	MaritalStatus	PerformanceRating	YearsAtCompany
Department	Gender	MonthlyIncome	RelationshipSatisfaction	YearsInCurrentRole
DistanceFromHome	HourlyRate	MonthlyRate	StandardHours	YearsSinceLastPromotion
Education	JobInvolvement	NumCompaniesWorked	StockOptionLevel	YearsWithCurrManager

```
table(sapply(dataset, class))
```

```
##
## character   numeric
##         9        26
```

```
cat("Any NA?", any(is.na(dataset)), "\n")
```

```
## Any NA? FALSE
```

We can see that the dataset is made up by 1470 sample classified by 35 features which are mostly numeric and few of them factors. Interestingly there are no NA values in the dataset so we won't need to check for those in later analysis.

Many features of the dataset are self-explaining from their names but some need further discussion. In particular, there is some feature stored as numeric but should be more properly defined as factor, since each value is a key for a specific value, so I am going to convert them.

Key	Education	EnvironmentSatisfaction	JobInvolvement	JobSatisfaction	PerformanceRating	RelationshipSatisfaction	WorkLifeBalance
1	Below college	Low	Low	Low	Low	Low	Bad
2	College	Medium	Medium	Medium	Good	Medium	Good
3	Bachelor	High	High	High	Excellent	High	Better
4	Master	Very high	Very high	Very high	Outstanding	Very high	Best
5	Doctorate						

Before starting any sort of analysis I am going to optimise memory usage of the dataset. It is not so important in this particular case since it is a pretty small one, but it is a good habit anyway. Firstly, all *numeric* features will be converted to *integer* because they takes half the memory, and then *character* features will be converted to *factor* for the same reason. *Numeric* columns that could be treated as factors will be left as are. Finally some feature does not provide information since the value is always the same and one is just the ID of the employee, and thus they will be removed. This is how the dataset looks like after this operations:

```
##                     Age                  Attrition             BusinessTravel
##               "integer"                   "factor"                   "factor"
##                DailyRate                  Department           DistanceFromHome
##               "integer"                   "factor"                   "integer"
##                Education             EducationField    EnvironmentSatisfaction
##                 "factor"                   "factor"                   "factor"
##                   Gender                 HourlyRate             JobInvolvement
##                 "factor"                   "integer"                  "factor"
##                 JobLevel                    JobRole            JobSatisfaction
##                "integer"                   "factor"                   "factor"
##           MaritalStatus               MonthlyIncome                MonthlyRate
##                 "factor"                   "integer"                  "integer"
##       NumCompaniesWorked                  OverTime          PercentSalaryHike
##                "integer"                   "factor"                   "integer"
##        PerformanceRating    RelationshipSatisfaction           StockOptionLevel
##                 "factor"                   "factor"                   "integer"
##        TotalWorkingYears      TrainingTimesLastYear            WorkLifeBalance
##                "integer"                   "integer"                   "factor"
```

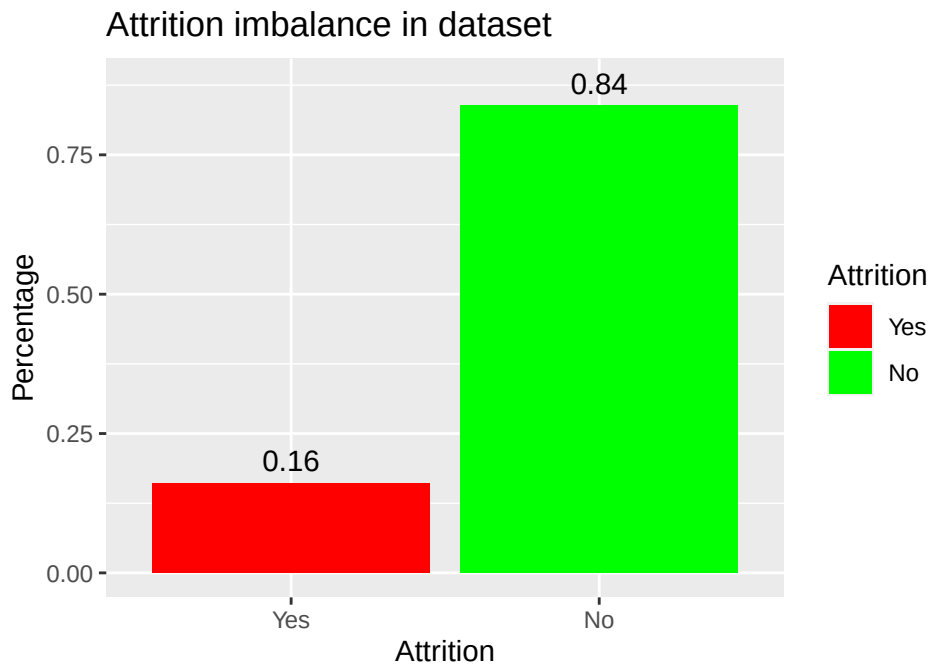
```
##           YearsAtCompany       YearsInCurrentRole  YearsSinceLastPromotion
##                "integer"                "integer"                "integer"
##     YearsWithCurrManager
##                "integer"
```

Exploratory data analysis

Before starting any sort of training it is necessary to visually inspect data to find clues for the training as well as insights about important features correlation. Given the size of the datset, it would be extremely long and also boring for the reader to inspect `Attrition` by every other variable or every possible column combinations. On the other hand, it is much more suitable instead to get less but more significative relationships, and discuss them.

First let's study if dataset is balanced with regard to `Attrition`.

```
# Attrition percentage in general.
dataset %>%
  group_by(Attrition) %>%
  summarise(Average = n() / nrow(dataset)) %>%
  ggplot(aes(Attrition, Average, fill = Attrition)) +
  geom_col() +
  ggtitle("Attrition imbalance in dataset") +
  ylab("Percentage") +
  geom_text(aes(Attrition, Average + 0.04, label = round(Average, 2))) +
  scale_fill_manual(values=c("#FF0000", "#00FF00"))
```

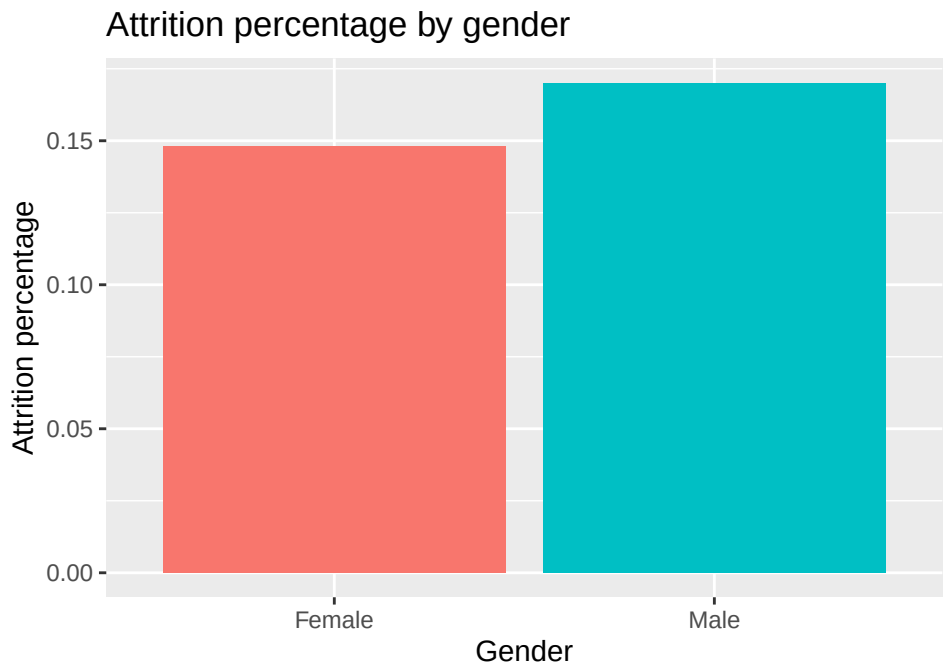


Dataset is definetely imbalanced and this will have effects on the final prediction phase because metrics such accuracy will not be suitable given that it will probably be close di 80%-90%.

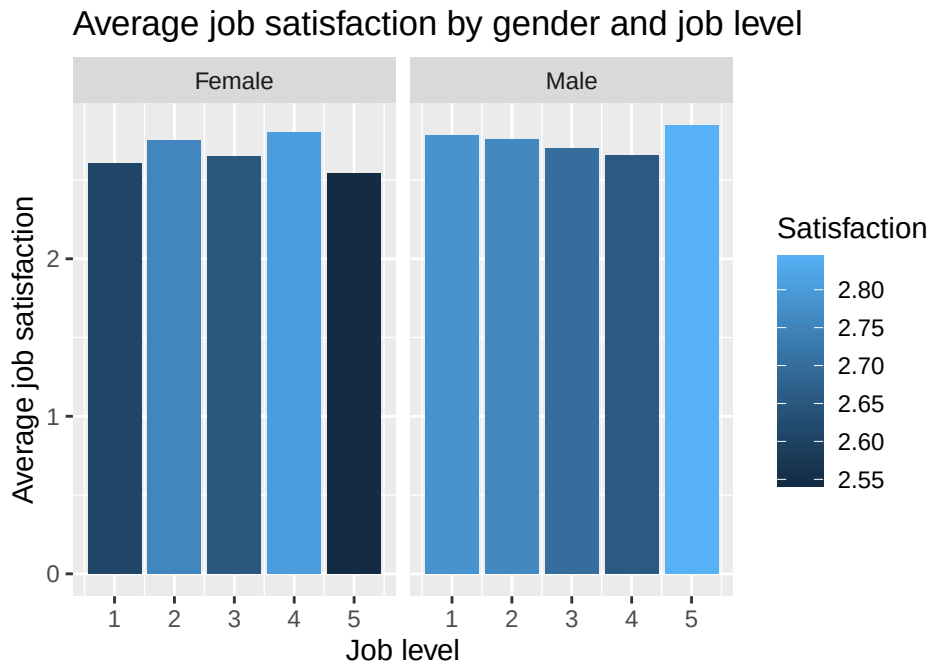
Gender factors

First factor to take into account is gender. It is probably the first anyone can think about when considering work inequality, and therefore the first to be studied here.

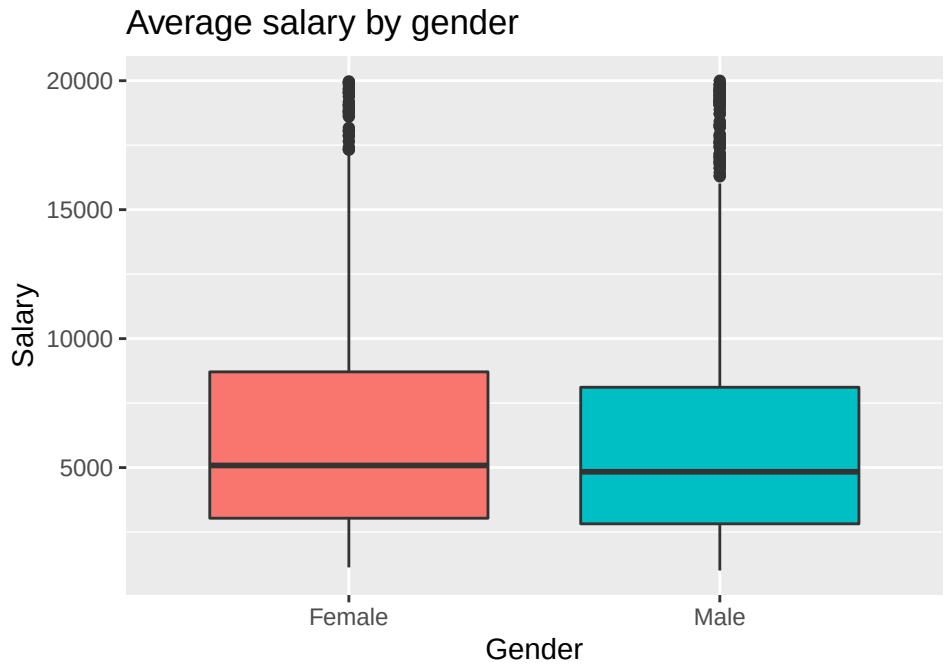
```r
# Attrition percentage by gender.
dataset %>%
  group_by(Gender) %>%
  summarise(Attrition_perc = sum(Attrition == "Yes")/n()) %>%
  ggplot(aes(Gender, Attrition_perc, fill = Gender)) +
  geom_col() +
  ggtitle("Attrition percentage by gender") +
  ylab("Attrition percentage") +
  theme(legend.position = "none")
```




```r
# Job satisfaction and position by gender
dataset %>%
  select(JobLevel, JobSatisfaction, Gender) %>%
  mutate(JobSatisfaction = as.integer(JobSatisfaction)) %>%
  group_by(JobLevel, Gender) %>%
  summarise(Satisfaction = mean(JobSatisfaction)) %>%
  ggplot(aes(x = JobLevel, y = Satisfaction, group = JobLevel,
             fill = Satisfaction)) +
  geom_col() +
  facet_wrap(~Gender) +
  ggtitle("Average job satisfaction by gender and job level") +
  ylab("Average job satisfaction") +
  xlab("Job level")
```




```
# Differences in salary by gender.
dataset %>%
  group_by(Gender) %>%
  ggplot(aes(Gender, MonthlyIncome, fill = Gender)) +
  geom_boxplot() +
  ggtitle("Average salary by gender") +
  ylab("Salary") +
  theme(legend.position = "none")
```



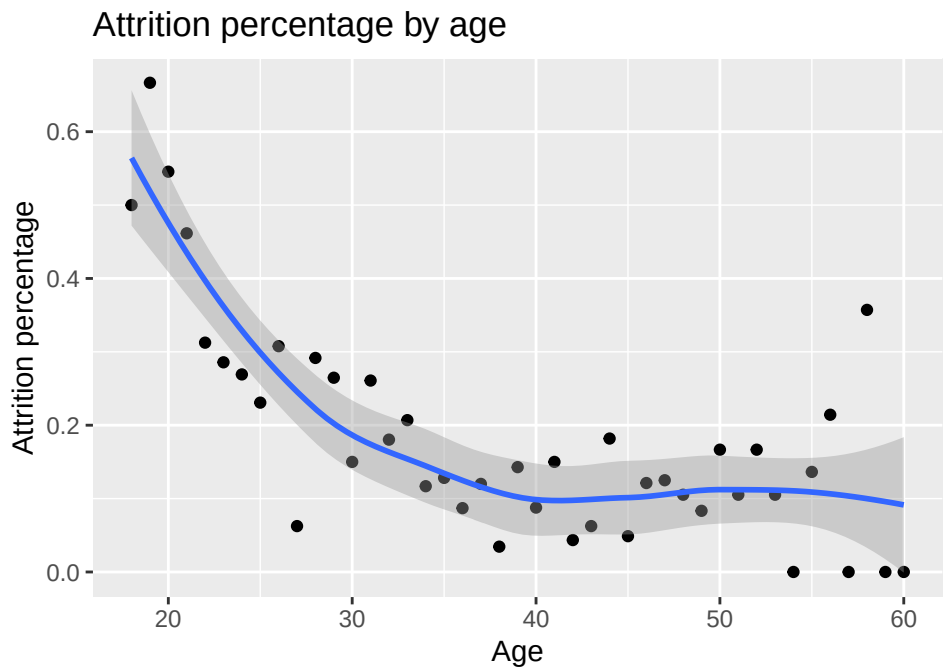
First thing to note is the difference in attrition between the two genders. Males displays, in fact, averagely

more attrition then females even though, as shown in the second plot, their mean job satisfaction is a bit higher, especially for to job level (tier 5). Despite such differences differences in salary are basically zero, just slightly favoured towards females despite the average thought of males earnings more.

Age and generation factors

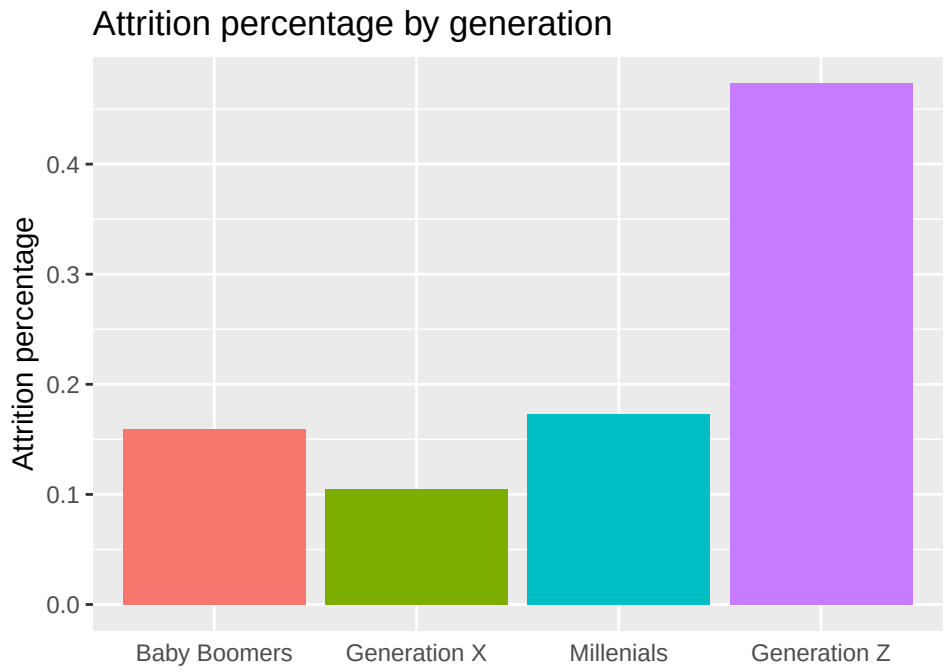
The second very important feature is age. It is, in fact, reasonable to think that different generation may have different mindsets and different ways of approaching the work life, as well as more experience and knowledge to deal with problems as they get older.

```
# Attrition percentage by age.
dataset %>%
  group_by(Age) %>%
  summarise(Attrition_perc = mean(Attrition == "Yes")) %>%
  ggplot(aes(Age, Attrition_perc)) +
  geom_point() +
  geom_smooth() +
  ggtitle("Attrition percentage by age") +
  ylab("Attrition percentage")
```

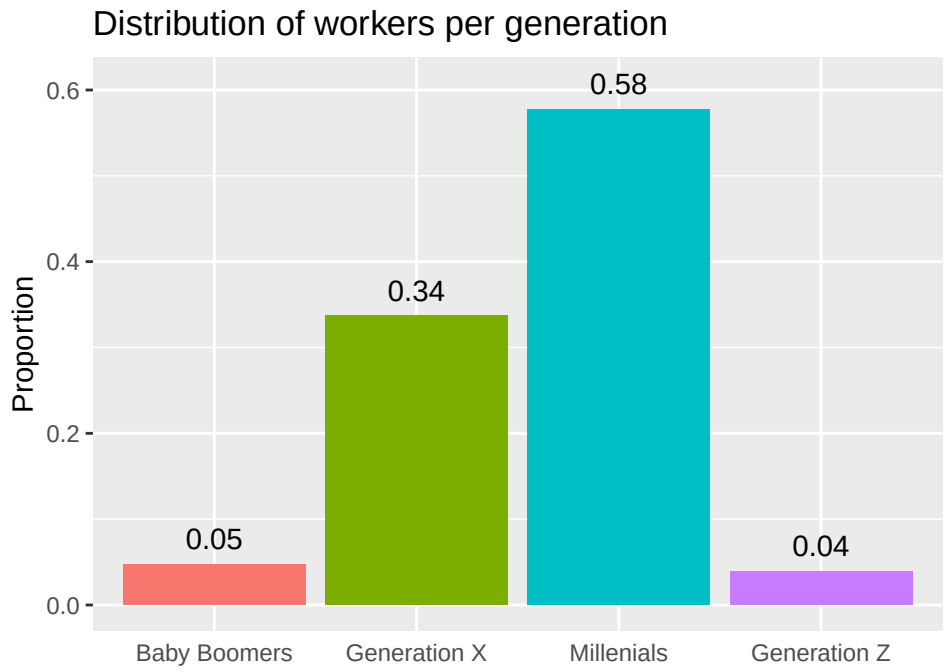



```
# Attrition by generation
dataset %>%
  mutate(Generation = factor(sapply(Age, getGeneration),
                             labels = c("Baby Boomers", "Generation X",
                                        "Millenials", "Generation Z"))) %>%
  group_by(Generation) %>%
  summarise(Attrition_perc = mean(Attrition == "Yes")) %>%
  ggplot(aes(Generation, Attrition_perc, fill = Generation)) +
  geom_col() +
  ggtitle("Attrition percentage by generation") +
  ylab("Attrition percentage") +
  theme(axis.title.x = element_blank(),
        axis.ticks.x = element_blank(),
```

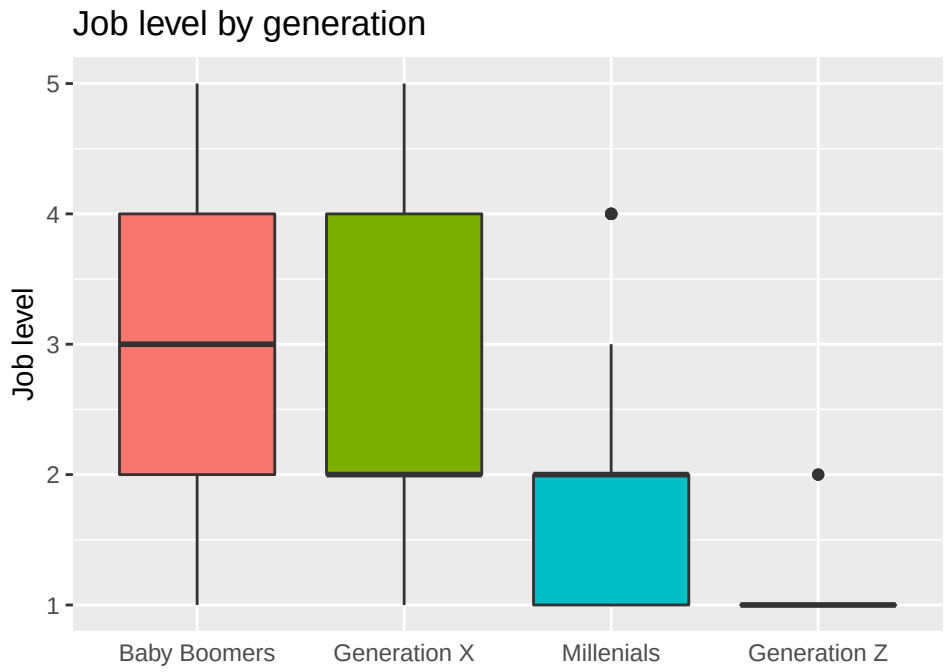
```
        legend.position = "none")
```




```
# Distribution of generations
dataset %>%
  mutate(Generation = factor(sapply(Age, getGeneration),
                             labels = c("Baby Boomers", "Generation X",
                                        "Millenials", "Generation Z"))) %>%
  group_by(Generation) %>%
  summarise(Proportion = n()/nrow(dataset)) %>%
  ggplot(aes(x = Generation, y = Proportion, fill = Generation)) +
  geom_col(position = "dodge") +
  ggtitle("Distribution of workers per generation") +
  ylab("Proportion") +
  theme(axis.title.x = element_blank(),
        axis.ticks.x = element_blank(),
        legend.position = "none") +
  geom_text(aes(x = Generation, y = Proportion+0.03, label = round(Proportion,2)))
```




```
# Job level by generation
dataset %>%
  mutate(Generation = factor(sapply(Age, getGeneration),
                             labels = c("Baby Boomers", "Generation X",
                                        "Millenials", "Generation Z"))) %>%
  ggplot(aes(Generation, JobLevel, fill = Generation)) +
  geom_boxplot() +
  ggtitle("Job level by generation") +
  ylab("Job level") +
  theme(axis.title.x = element_blank(),
        axis.ticks.x = element_blank(),
        legend.position = "none")
```

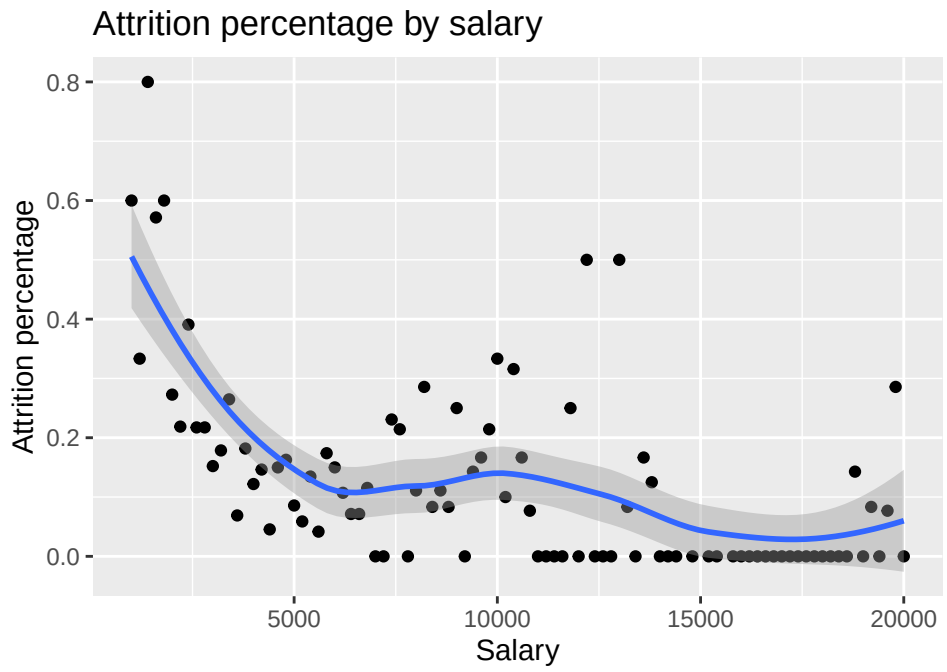


As supposed, there is a clear increase in attrition as the age decrease, stalling at around 40 years old. Similar pattern can be seen if we group age by generation. Generations are defined as per Wikipedia ("Generation," n.d.). Coherently there is also a huge majority of millenials ($age \geq 23 \wedge age \leq 38$), followed by Generation X ($age \geq 39 \wedge age \leq 54$), and finally Boomers and Generation Z who take the smallest share of the population. Finally, as shown in the graph, we can see a decreasing average job level as the age falls, with Boomers and Generation X being approximately between level 2 and 4, Millenials between 1 and 2, and Generation at level 1.

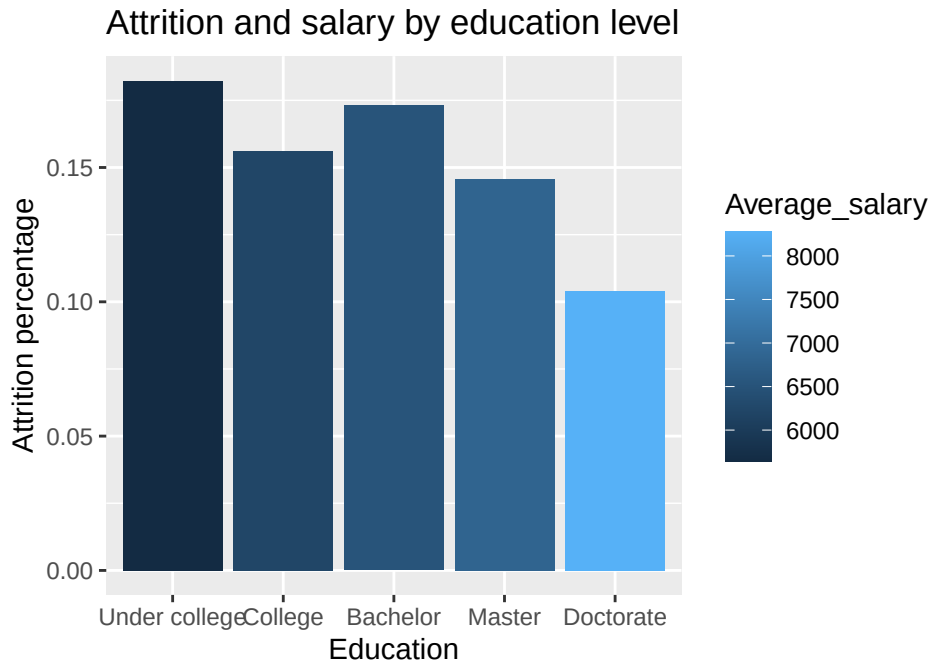
Education and income factors

Let's move on to study how salary and educational level may influence attrition. One can suppose, in fact, that salary should play the most important role and that it is averagely dependent on education level.

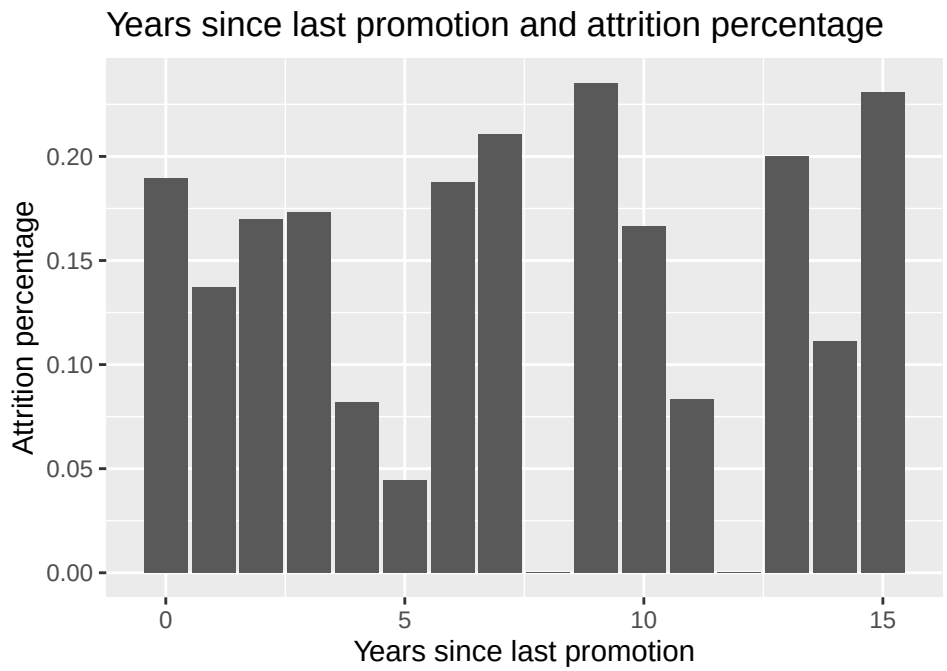
```
# Attrition percentage by monthly income.
# Income is rounded to the closest 200$ to have more
# data points to average.
dataset %>%
  mutate(MonthlyIncome = round(MonthlyIncome/200)*200) %>%
  group_by(MonthlyIncome) %>%
  summarise(Attrition_perc = mean(Attrition == "Yes")) %>%
  ggplot(aes(MonthlyIncome, Attrition_perc)) +
  geom_point() +
  geom_smooth() +
  ggtitle("Attrition percentage by salary") +
  ylab("Attrition percentage") +
  xlab("Salary")
```



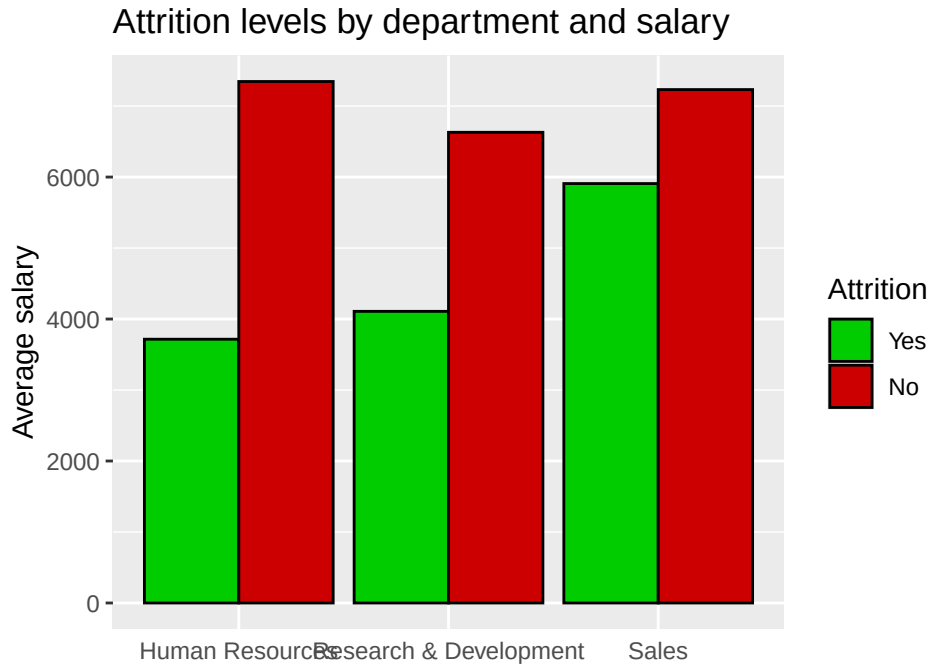

```
# Attrition and salary by education level
dataset %>%
  group_by(Education) %>%
  summarise(Attrition_perc = mean(Attrition == "Yes"),
            Average_salary = mean(MonthlyIncome)) %>%
  ggplot(aes(Education, Attrition_perc, fill = Average_salary)) +
  geom_col(position = "dodge") +
  ggtitle("Attrition and salary by education level") +
  ylab("Attrition percentage")
```



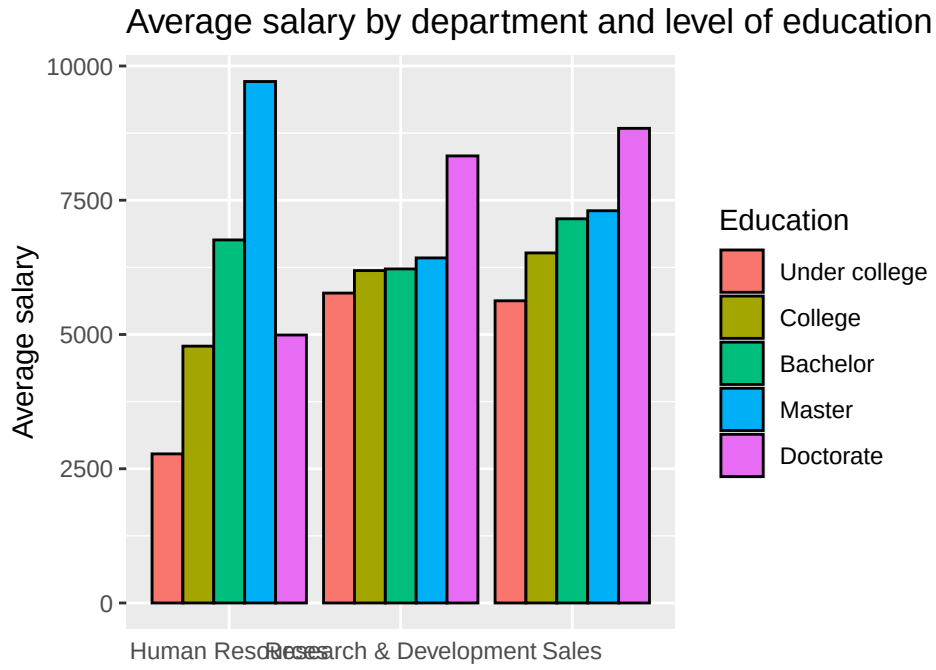
```r
# Years since last promotion and attrition percentage
dataset %>%
  group_by(YearsSinceLastPromotion) %>%
  summarise(Attrition_perc = mean(Attrition == "Yes")) %>%
  ggplot() +
  geom_col(aes(YearsSinceLastPromotion, Attrition_perc)) +
  scale_y_continuous(name = "Attrition percentage") +
  ggtitle("Years since last promotion and attrition percentage") +
  xlab("Years since last promotion")
```



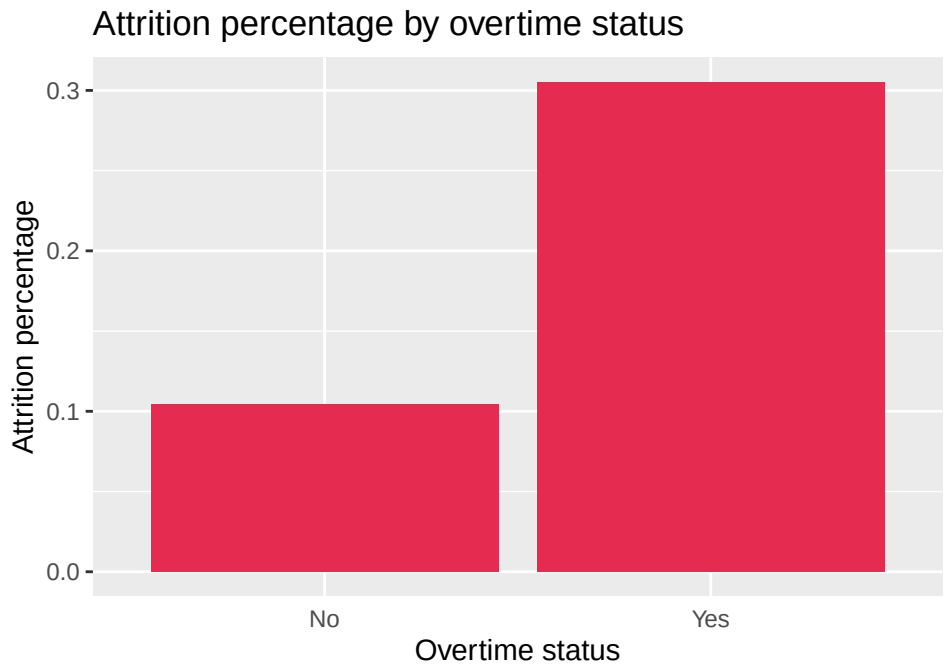

```r
# Attrition by department and salary
dataset %>%
  group_by(Attrition, Department) %>%
  summarise(Average_salary = mean(MonthlyIncome)) %>%
  ggplot(aes(Department, Average_salary, fill = Attrition)) +
  geom_col(position = "dodge", colour = "black") +
  ggtitle("Attrition levels by department and salary") +
  ylab("Average salary") +
  theme(axis.title.x = element_blank(),
        axis.ticks.x = element_blank()) +
  scale_fill_manual(values=c("#00CC00", "#CC0000"))
```




```
# Average salary by department and level of education
dataset %>%
  group_by(Education, Department) %>%
  summarise(Average_salary = mean(MonthlyIncome)) %>%
  ggplot(aes(Department, Average_salary, fill = Education)) +
  geom_col(position = "dodge", colour = "black") +
  ggtitle("Average salary by department and level of education") +
  ylab("Average salary") +
  theme(axis.title.x = element_blank(),
        axis.ticks.x = element_blank())
```




```
# Attrition percentage by overtime status
dataset %>%
  group_by(OverTime) %>%
  summarise(Attrition_perc = mean(Attrition == "Yes")) %>%
  ggplot(aes(OverTime, Attrition_perc)) +
  geom_col(fill = "#E52B50") +
  ggtitle("Attrition percentage by overtime status") +
  ylab("Attrition percentage") +
  xlab("Overtime status") +
  theme(axis.ticks.x = element_blank())
```

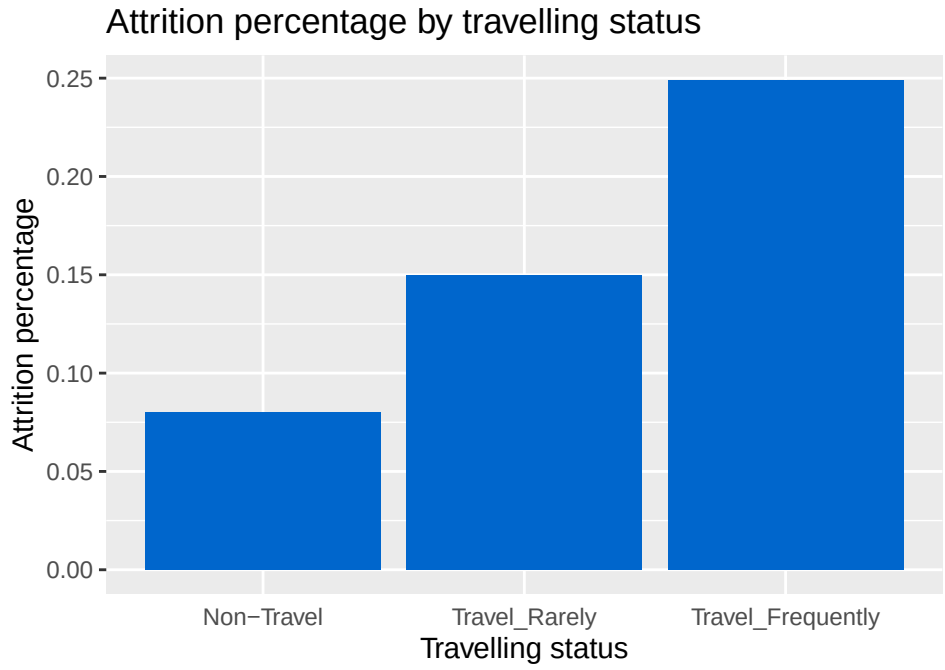


As stated above, attrition rapidly decline as salary increase and it is often close to 0% when it is greater than 10000$. Coherently indeed, attrition decreases as level of education grows because salary would increase accordingly. On the other hand the number of years since last promotion doesn't seem to affect attrition differently from the working department. In this case the sale sector, even with a greater average salary, presents greater attrition levels than the others as opposed to the statement that attrition falls with salary increase. We don't have data to infer reasons for this but it is a factor to take into account. Finally, the last plot probably shows the most important difference and factor to consider when looking to decreasing attrition: the overtime status. Employees working overtime displays three times the probability of attrition.

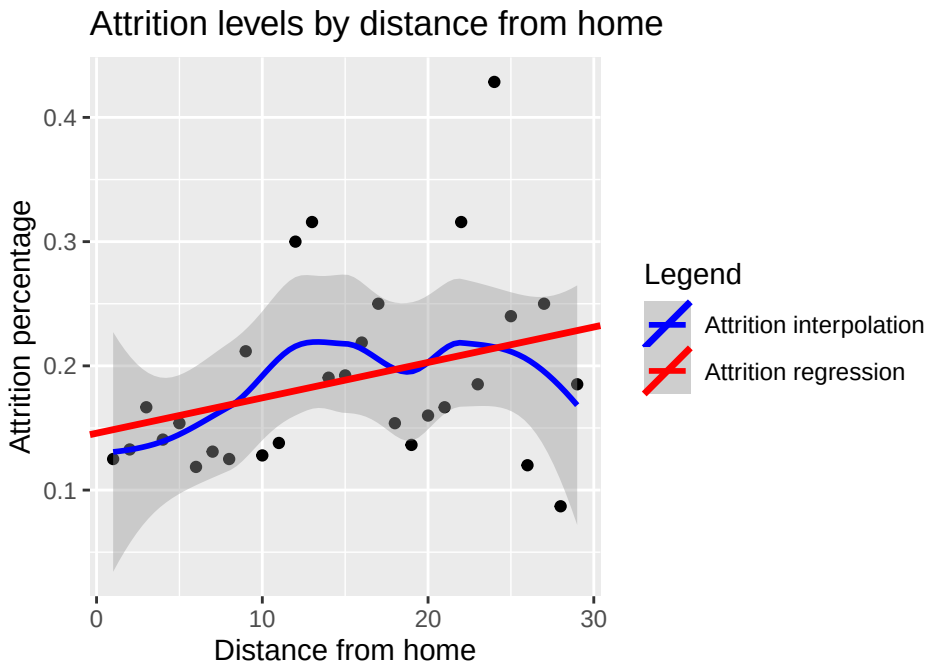
Personal factors

Finally, also travelling or personal factors could have their influence on the employee.

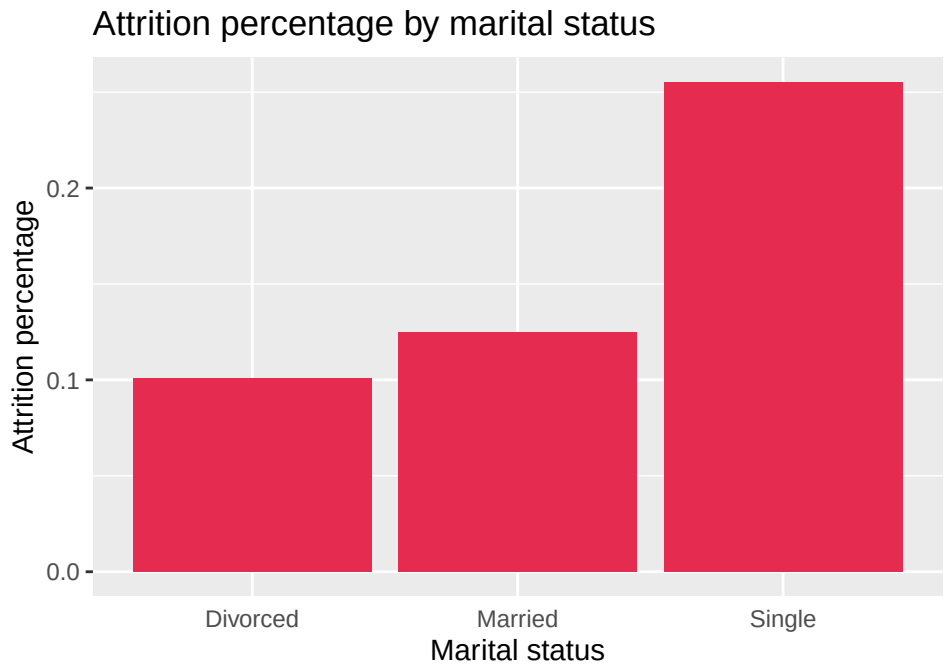
```
# Attrition percentage by travelling status
dataset %>%
  group_by(BusinessTravel) %>%
  summarise(Attrition_perc = mean(Attrition == "Yes")) %>%
  mutate(BusinessTravel = reorder(BusinessTravel, Attrition_perc)) %>%
  ggplot(aes(BusinessTravel, Attrition_perc)) +
  geom_col(fill = "#0066CC") +
  ggtitle("Attrition percentage by travelling status") +
  ylab("Attrition percentage") +
  xlab("Travelling status") +
  theme(axis.ticks.x = element_blank())
```



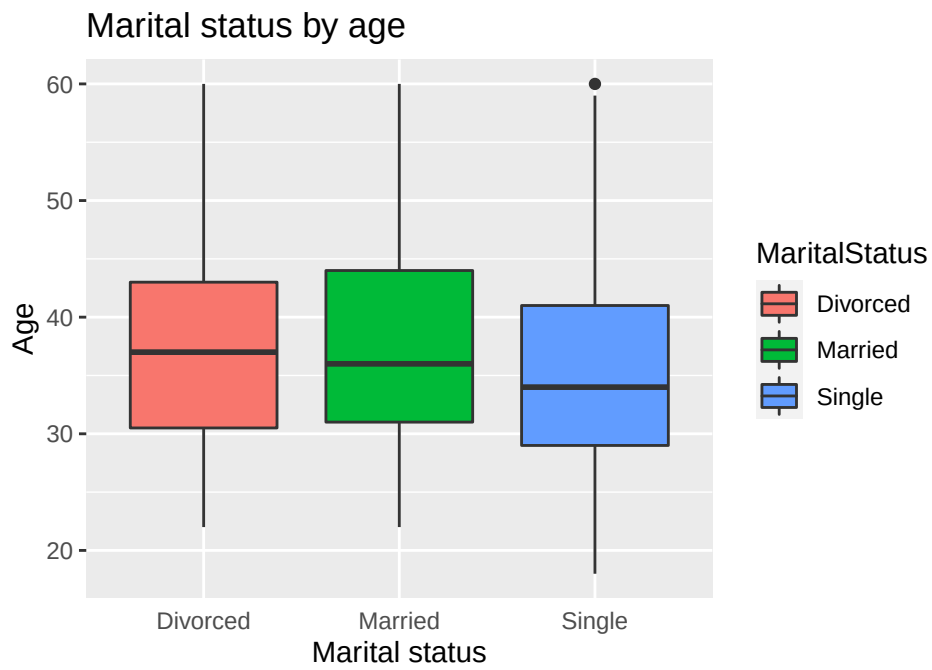

```r
# Compute attrition regression by distance from home
distanceRegression <- dataset %>%
  group_by(DistanceFromHome) %>%
  summarise(Attrition_perc = mean(Attrition == "Yes")) %>%
  lm(Attrition_perc ~ DistanceFromHome, data = .)

# Attrition levels by distance from home
cols <- c("Attrition interpolation" = "#0000FF",
          "Attrition regression" = "#FF0000")
dataset %>%
  group_by(DistanceFromHome) %>%
  summarise(Attrition_perc = mean(Attrition == "Yes"), Count = n()) %>%
  ggplot() +
  geom_point(aes(DistanceFromHome, Attrition_perc)) +
  geom_smooth(aes(DistanceFromHome, Attrition_perc, colour = "Attrition interpolation")) +
  geom_abline(aes(slope = distanceRegression$coefficients["DistanceFromHome"],
              intercept = distanceRegression$coefficients["(Intercept)"],
              colour = "Attrition regression"), size = 1.2) +
  scale_colour_manual(name="Legend", values=cols) +
  ggtitle("Attrition levels by distance from home") +
  xlab("Distance from home") +
  ylab("Attrition percentage")
```




```r
# Attrition percentage by marital status
dataset %>%
  group_by(MaritalStatus) %>%
  summarise(Attrition_perc = mean(Attrition == "Yes")) %>%
  ggplot(aes(MaritalStatus, Attrition_perc)) +
  geom_col(fill = "#E52B50") +
  ggtitle("Attrition percentage by marital status") +
  ylab("Attrition percentage") +
  xlab("Marital status") +
  theme(axis.ticks.x = element_blank())
```




```
# Marital status by age
dataset %>%
  group_by(MaritalStatus) %>%
  ggplot(aes(MaritalStatus, Age, fill = MaritalStatus)) +
  geom_boxplot() +
  ggtitle("Marital status by age") +
  ylab("Age") +
  xlab("Marital status") +
  theme(axis.ticks.x = element_blank())
```



In particular, as we can see in the fisrt two plot, travels negatively influence the attitude of the employee. As they travel more frequently, or as the commute distance rises, the more likely is for the worker to show attrition. In addition, also being single have its importance in feeling attrition, but this is only poorly correlated to the more likely younger age of the single employee, because median age difference by marital status is actually very little.

Training phase

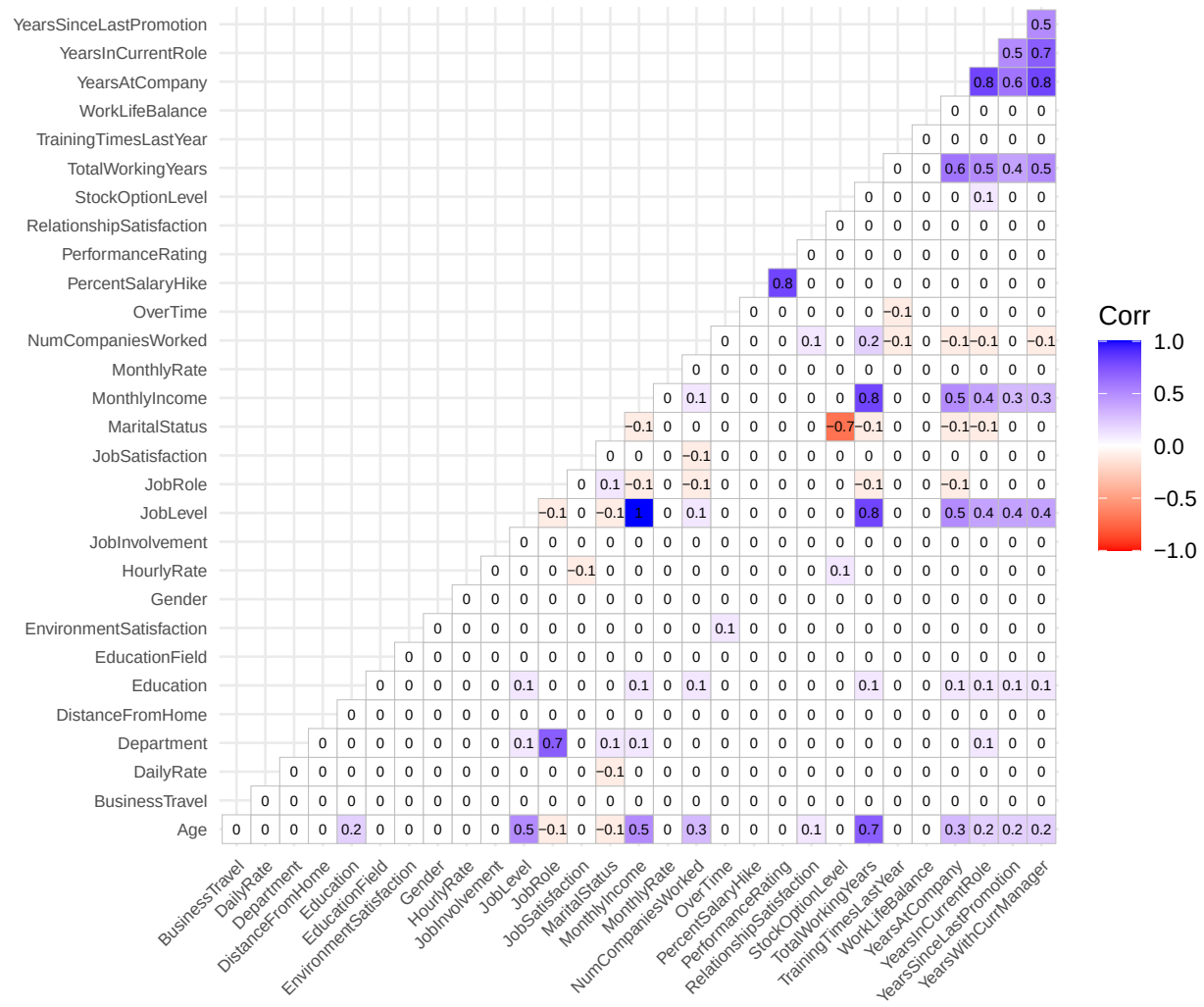
After having surmised important insights in the previous section, I am now going to start the predictive analysis in order to be able to predict attrition based on some feature. As seen, there are so many possible interactions between `Attrition` and the features (and many features are intercorrelated too) to make the selection of features very difficult by hand.

In this section I am going to show the steps done in order to select the most important features and the models checking to find the best classification algorithm.

Correlation matrix

Since, as said, there are so many different combinations, having a general idea of the correlation between features could be crucial to avoid training models with every of them, likely causing overfitting. In this regard the correlation matrix will be very useful.

```
# Get the correlation matrix by first converting factors
# to integer, and then using cor() rounding to the first
# decimal place.
correlationMatrix <- dataset %>% select(-Attrition)
correlationMatrix[,which(sapply(correlationMatrix, class) == "factor")] <-
  lapply(correlationMatrix[,which(sapply(correlationMatrix, class) == "factor")], as.integer)

correlationMatrix <- round(cor(correlationMatrix), 1)

# Plot the lower triangle correlation matrix.
ggcorrplot(correlationMatrix,
           type = "lower",
           colors = c("red", "white", "blue"),
           lab = TRUE,
           lab_size = 2,
           tl.cex=7)
```



Looking at the matrix, we can note many strong and weak correlations among features. It could be possible to use a threshold so as to arbitrarly decide what combinations are considered strongly correlated, and thus deleting one of the two features, or could be probably better to apply more suitable methods.

Features selection

The *caret* package comes in handy with such problems because it provides with so many different algorithms in this regard. I decided to use a filter-based feature selection.

Filter-based methods use statistical techniques to evaluate the relationship between each input variable and the target variable, using them for finding the best candidates for the model features. It is general less accurate than wrappers methods (see Kuhn (2019)), but they are so much more computationally efficient and risk of overfitting is much lower too (Kuhn 2019).

Feature selection is also useful to determine the percentage of data to be used in the training set (Guyon

1997), but to determine the number of features we would need the training set. To solve the problem I used the whole dataset to discern the best features and then I split it based on this result.

```
# Feature selection step using filter method
# Seed is for reproducibility.
set.seed(1, sample.kind = "Rounding")
filterControl <- sbfControl(functions = rfSBF,
                            method = "boot",
                            number = 10)
featureSelection <- sbf(Attrition ~ .,
                        data = dataset,
                        sbfControl = filterControl)

featureSelection$optVariables
```

```
##  [1] "Age"                                "BusinessTravelTravel_Frequently"
##  [3] "DailyRate"                          "DepartmentResearch & Development"
##  [5] "DepartmentSales"                    "DistanceFromHome"
##  [7] "EducationFieldMarketing"            "EducationFieldTechnical Degree"
##  [9] "JobInvolvementHigh"                 "JobInvolvementVery high"
## [11] "JobLevel"                           "JobRoleLaboratory Technician"
## [13] "JobRoleManager"                     "JobRoleManufacturing Director"
## [15] "JobRoleResearch Director"           "JobRoleSales Representative"
## [17] "JobSatisfactionVery high"           "MaritalStatusMarried"
## [19] "MaritalStatusSingle"                "MonthlyIncome"
## [21] "OverTimeYes"                        "StockOptionLevel"
## [23] "TotalWorkingYears"                  "TrainingTimesLastYear"
## [25] "WorkLifeBalanceBetter"              "YearsAtCompany"
## [27] "YearsInCurrentRole"                 "YearsWithCurrManager"
```

These are the features that will be used in the next model phase.

Model checking

After having selected the best candidates as model features, next step was choosing how to split the dataset. I thought of two different approach. First was to apply Pareto's principle ("Pareto's Principle," n.d.) and split the dataset with a 80/20 ratio, or second was to take advantage of the work by Guyon (Guyon 1997), who found an equation to take the best ratio according to the number of features. Such ratio is the inverse of the square root of the number of features. Luckily, in this case the number was very close to the 20% of Pareto, and thus I used such value. The dataset is not very big as we saw, but enough to let us take that percentage and have enough data points for predictions.

The last phase was to finally find the best model. I started with the simple ones and adding up from there. In this analysis I am reporting only the best ones and discuss them. First classification algorithm was the *Generalised Linear Regression* and a variant like *Bayesian Generalised Linear Model*. Then I wanted to try out more linear methods and I was thinking about using the *Linear Discriminant Analysis* when I finally opted for the *Regularised Discriminant Analysis* which is a genereralisation of it, and coincides for the regularisation parameter $\lambda = 1$ (Dalpiaz 2019). Finally I tried out more complex methods like *Random Forest* and *Neural Network*. Regarding the latter, I also chose to apply the best feature extraction via pca, having thus the *Neural Networks with Feature Extraction* technique.

So many different training algorithms were tried, but here, I will report only the most significant ones. My criterion was simple: in accordance with the Occam's Razor principle, if two model performed similarly I kept the simpler one.

Each model was tried with different tuning parameters and was validated with a bootstraps cross validation with ten straps. Metric used was area under ROC.

```r
# Create the training set (80%) and the validation set (20%)
# from the new dataset. More info in the report.
set.seed(1, sample.kind = "Rounding")
validation_index <- createDataPartition(dataset$Attrition, times = 1, p = 0.2, list = FALSE)
training <- dataset[-validation_index,]
validation <- dataset[validation_index,]

# Create bootstraps training control.
trainingCtrl <- trainControl(method = "boot",
                             number = 20,
                             summaryFunction = twoClassSummary,
                             classProbs = TRUE)

# Define weighs.
# An error on 'Yes' weights more than on 'No'
# Weights are defined proportionally to dataset imbalance
model_weights <- ifelse(training$Attrition == "Yes",
                        (1 / table(training$Attrition)[[1]]) * 0.5,
                        (1 / table(training$Attrition)[[2]]) * 0.5)

# Defining models to check
models <- c("glm", "bayesglm", "rda",
            "rf", "pcaNNet")

# Define tuning parameters
tuningFrames <- list("rda" =     expand.grid(gamma = seq(0, 1, 0.2),
                                             lambda = seq(0.2, 1, 0.2)),
                     "rf" =      expand.grid(mtry = seq(1, 9, 2)),
                     "pcaNNet" = expand.grid(size = seq(1, 4, 1),
                                             decay = seq(0.1, 1, 0.1)))

# Running model checking phase.
# Using seed for a matter of reproducibility
set.seed(1, sample.kind = "Rounding")
results <- lapply(models, function(mod){
  fit <- train(Attrition ~ .,
               data = training,
               method = mod,
               tuneGrid = tuningFrames[[mod]],
               trControl = trainingCtrl,
               weights = model_weights,
               metric = "ROC",
               trace = FALSE)
  return(c("Best ROC" = max(fit$results["ROC"]), fit$bestTune))
})
names(results) <- models
```

	glm	bayesglm	rda	rf	pcaNNet
Best ROC	0.7657026	0.766729	0.778272176924161	0.747847287965222	0.740417184284883
Parameter 1	NA	NA	gamma: 0	mtry: 3	size: 3
Parameter 2	NA	NA	lambda: 1	NA	decay: 0.3

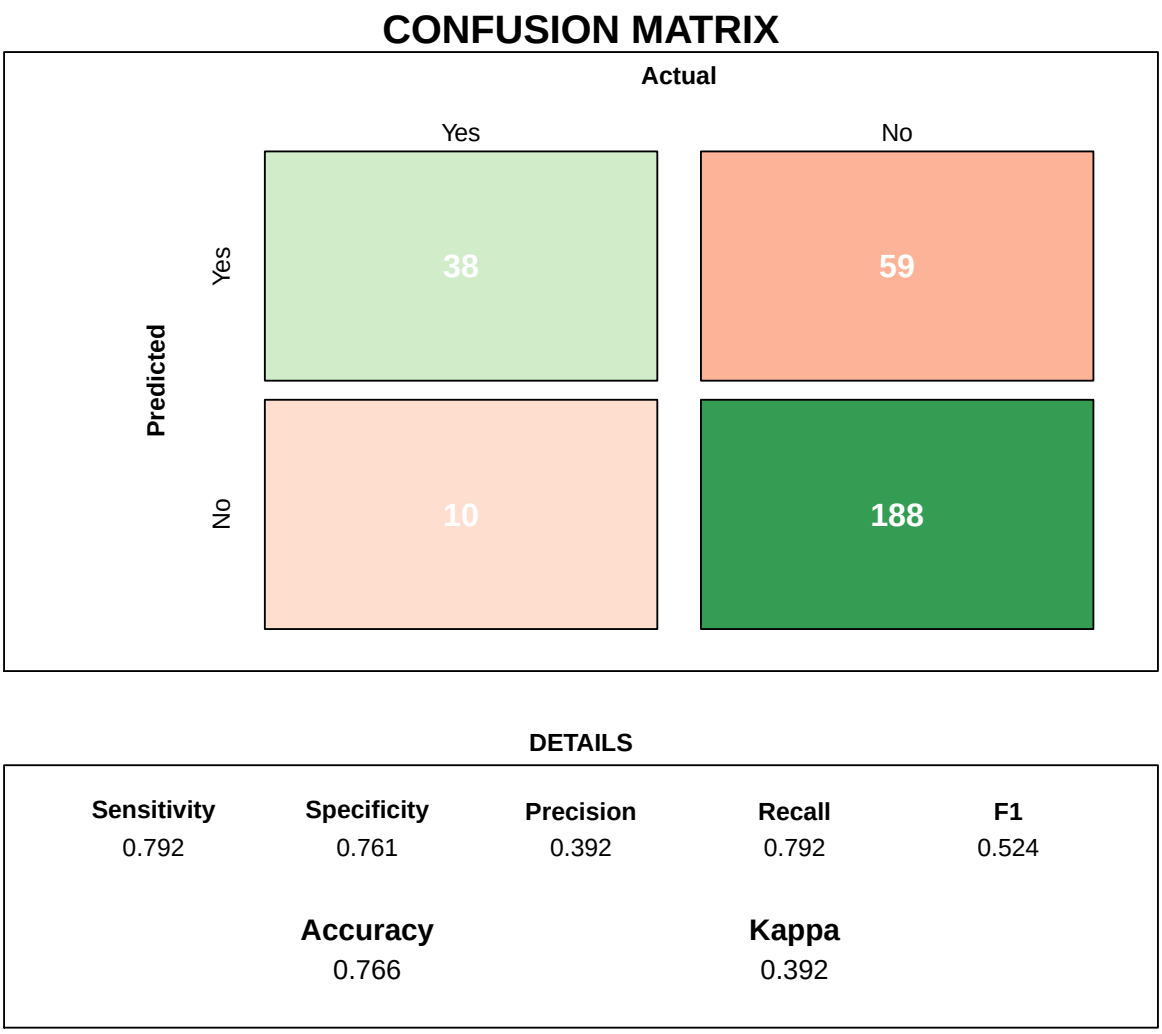
Such models perform approximately similarly but I reported them all for completeness. I could have just

picked the best model but the areas under ROC were so similar that differences were caused, mostly, by randomness. In fact I obtained slightly different results (around 1%) when seed was changed. So instead of picking just the best one for this seed, which might not be the best during test phase, I picked the simplest one to be coherent with the Occam's Razor. The candidate for final model building was then the *Generalised Linear Regression*.

Model building and results

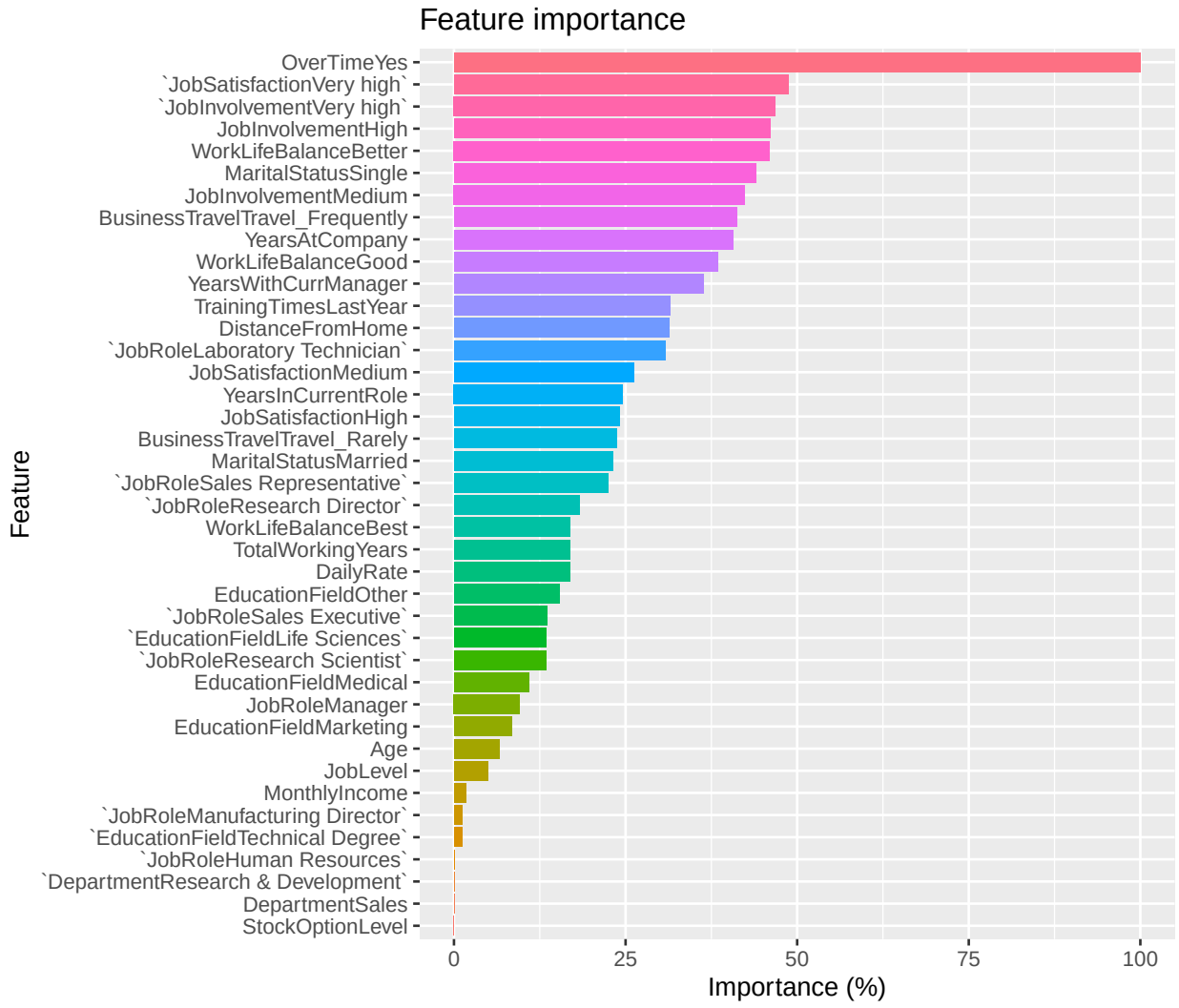
After having infered the best training algorithm we can use the whole training set to predict the validation one, and check the results.

```
# Model building based on previous results
set.seed(1, sample.kind = "Rounding")
model <- train(Attrition ~ .,
               data = training,
               weights = model_weights,
               method = "glm")

# Predict attrition
predictions <- predict(model, validation)

# Get and plot the confusion matrix.
cm <- confusionMatrix(predictions, validation$Attrition)
draw_confusion_matrix(cm)
```



To conclude plotting the features importance could bring further insights to understand this big problem for companies.

```
# Get feature importance
featuresImportance <- varImp(model)$importance
featuresImportance[,"Feature"] <- rownames(featuresImportance)
rownames(featuresImportance) <- NULL

# And plot them
featuresImportance %>%
  mutate(Feature = reorder(Feature, Overall)) %>%
  ggplot(aes(Feature, Overall, fill = Feature)) +
  geom_col() +
  coord_flip() +
  theme(legend.position = "none") +
  ggtitle("Feature importance") +
  ylab("Importance (%)")
```



Discussion

First thing to say, is the reason why I set up class weights. If one had to choose between a high FPR and a high FNR companies should tend towards the first from an economical point of view. Considered, in fact, the financial loss caused by attrition it would be better to erroneously predict `Yes` and intervene with no effect, than to predict `No`, doing nothing to fix the situation, and having the emplyoee resigning. For this the weight on `Yes` weights more, proportionally to the dataset imbalance. I could have decided the weights arbitrarly but I opted for this approach.

Results seems very good and encouraging. Both Sensitivity and Specificity are greater than 75%, and in particular we note that the False Positive Rate (FPR) is greater than False Negative Rate (FNR), too. And that is exactly what we wanted. Accordingly accuracy is also the the same range. On the other hand precision and F1 factors are not very good, but unless I had been able to have almost perfect specificity and sensitivity, I prefer to have lower precision and better sensitivity, than much greater precision but much lower sensitivity, considering the financial problem stated above.

Moving to the feature importance, we note a suprisingly high priority in overtime status that seems to be the most important factor to cause attrition (according to this model). That is actually very good because it one of the factors that could be most easily managed by the company. In the example below I converted every overtime `Yes` entry to `No`: the result is a very significant fall in attrition rate

```
## Attrition rate predicted with some overtime:  0.329
## Attrition rate predicted with no overtime:  0.22
```

Other main features are the ones related to personal factors such as work-life balance, job satisfaction and involvement and marital status. Even more suprising is the almost null importance of salary to attrition rates which is in contrast to what we find in the previous exploratory analysis

Conclusion and future development

In this papar I presented, inspected and analysed the *IBM HR Analytics Employee Attrition & Performance*. First step was to change some column type for optimisation, plotting, or predictive purposes. Then, I moved on to visually inspect and plot the main features combinations to gain insights and clues about how they were related each other. Correlogram was last used to get an overall index of correlation between features. Last step was the training phase. There, I tried different training models by using bootstrap cross validation and different weights for attrition given the financial problems that companies may run into. Best algorithm was the *Generalised Linear Model* that let us predict attrition with promising results and provided us with an overview about the most important factors a company should take into consideration when dealing with such problem.

To further improve the algorithm maybe a bigger dataset could be helpful. Unfortunately, many machine learning models were tried out and almost everyone had an area under ROC at around 77% making me think that complex algorithm are not needed and simple ones may be more than sufficient with this dataset. On the other hand, further improvement could be obtained with a better (and longer) feature selection process, using a wrapper method. Better estimates could be obtained if we had more data regarding workplace discriminations (of any kind) which is a really serious and though problem of modern society, whose data are hard to be gathered in a rigorous and thorough way, but it impacts people and economy very hardly ("How Workplace Discrimination Impedes Economic Growth" 2017).

Reference

Dalpiaz, David. 2019. "R for Statistical Learning." https://daviddalpiaz.github.io/r4sl/regularized-discriminant-analysis.html.

"Generation." n.d. Wikipedia. en.wikipedia.org/wiki/Generation.

Guyon, Isabelle. 1997. "A Scaling Law for the Validation-Set Training-Set Size Ratio."

"How Workplace Discrimination Impedes Economic Growth." 2017. Council on foreign relations. 2017. www.cfr.org/event/how-workplace-discrimination-impedes-economic-growth.

Kuhn, Max. 2019. "The Caret Package." topepo.github.io/caret/index.html.

Mariotti, Andrew, Sam Robinson, and Evren Esen. 2017. "Talent Acquisition Benchmarking Report." Society for Human Resource Management. www.shrm.org/hr-today/trends-and-forecasting/research-and-surveys/Documents/2017-Talent-Acquisition-Benchmarking.pdf.

"Pareto's Principle." n.d. Wikipedia. en.wikipedia.org/wiki/Pareto_principle.

Subhash, Pavan. 2017. "IBM Hr Analytics Employee Attrition & Performance." 2017. www.kaggle.com/pavansubhasht/ibm-hr-analytics-attrition-dataset.