Employee Turnover Prediction

Albesa Istrefaj, Antonia Durisch, Philipp Drebes

08.06.2023

Introduction

Our team, consisting of the prospective Data Scientists Philipp Drebes, Albesa Istrefaj and Antonia Durisch, has taken on the topic of employee turnover prediction.

In this project, we will be using various machine learning algorithms to analyze our chosen dataset containing employee turnover data. We will develop predictive models and analyze our results in depth.

Employee turnover is a critical issue for companies, as its results are immense recurring costs in areas such as recruitment, training, and lost productivity.

«Some studies predict that every time a business replaces a salaried employee, it costs 6 to 9 months' salary on average.» (Peoplekeep, 2023)

Our main idea has been to identify factors that contribute to employee turnover, to be able to develop accurate predictive models which help companies better understand employee behavior and improve employee retention.

Data set

The dataset we will be using is the Employee Turnover data set. This data set contains the following information about the employees of a company.

Column	Description
stag	Experience (time)
event	Employee turnover
gender	Employee's gender, female(f), or male(m)
age	Employee's age (year)
industry	Employee's Industry
profession	Employee's profession
traffic	From what pipelene candidate came to the company
coach	Presence of a coach (training) on probation
head_gender	head (supervisor) gender
greywage	Greywage in Russia or Ukraine means that the employer (company) pays just a tiny bit amount of salary above the white-wage (white-wage means minimum wage)
way	how an employee gets to workplace (by feet, by bus etc)
extraversion	result from Big Five personality test
independ	result from Big Five personality test
selfcontrol	result from Big Five personality test
anxiety	result from Big Five personality test
novator	result from Big Five personality test

Preparation and Exploration

In a first step we are loading all the libraries:

```
library(tidyverse, quietly = T)
library(vioplot, quietly = T)
library(caret, quietly = T)
library(vtreat, quietly = T)
library(ggcorrplot, quietly = T)
library(readr, quietly = T)
library(ggplot2, quietly = T)
library(gridExtra, quietly = T)
library(mgcv, quietly = T)
library(e1071, quietly = T)
library(dplyr, quietly = T)
library(mlbench, quietly = T)
library(caret, quietly = T)
library(arm, quietly = T)
library(doParallel, quietly = T)
library(randomForest, quietly = T)
library(neuralnet, quietly = T)
library(lattice, quietly = T)
library(caTools, quietly = T)
library(nnet, quietly = T)
```

Then we are loading and inspecting our data:

```
turnover <- read_csv('data/turnover.csv', col_names=TRUE)
```

```
## Rows: 1129 Columns: 16
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (8): gender, industry, profession, traffic, coach, head_gender, greywage...
## dbl (8): stag, event, age, extraversion, independ, selfcontrol, anxiety, nov...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# summary(turnover)

# Having a look at the structure.
str(turnover)
```

```
## spc_tbl_ [1,129 x 16] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ stag        : num [1:1129] 7.03 22.97 15.93 15.93 8.41 ...
##  $ event       : num [1:1129] 1 1 1 1 1 1 1 1 1 1 ...
##  $ gender      : chr [1:1129] "m" "m" "f" "f" ...
##  $ age         : num [1:1129] 35 33 35 35 32 42 42 28 29 30 ...
##  $ industry    : chr [1:1129] "Banks" "Banks" "PowerGeneration" "PowerGeneration" ...
##  $ profession  : chr [1:1129] "HR" "HR" "HR" "HR" ...
##  $ traffic     : chr [1:1129] "rabrecNErab" "empjs" "rabrecNErab" "rabrecNErab" ...
##  $ coach       : chr [1:1129] "no" "no" "no" "no" ...
##  $ head_gender : chr [1:1129] "f" "m" "m" "m" ...
##  $ greywage    : chr [1:1129] "white" "white" "white" "white" ...
##  $ way         : chr [1:1129] "bus" "bus" "bus" "bus" ...
##  $ extraversion: num [1:1129] 6.2 6.2 6.2 5.4 3 6.2 6.2 3.8 8.6 5.4 ...
##  $ independ    : num [1:1129] 4.1 4.1 6.2 7.6 4.1 6.2 6.2 5.5 6.9 5.5 ...
##  $ selfcontrol : num [1:1129] 5.7 5.7 2.6 4.9 8 4.1 4.1 8 2.6 3.3 ...
##  $ anxiety     : num [1:1129] 7.1 7.1 4.8 2.5 7.1 5.6 5.6 4 4 7.9 ...
##  $ novator     : num [1:1129] 8.3 8.3 8.3 6.7 3.7 6.7 6.7 4.4 7.5 8.3 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   stag = col_double(),
##   ..   event = col_double(),
##   ..   gender = col_character(),
##   ..   age = col_double(),
##   ..   industry = col_character(),
##   ..   profession = col_character(),
##   ..   traffic = col_character(),
##   ..   coach = col_character(),
##   ..   head_gender = col_character(),
##   ..   greywage = col_character(),
##   ..   way = col_character(),
##   ..   extraversion = col_double(),
##   ..   independ = col_double(),
##   ..   selfcontrol = col_double(),
##   ..   anxiety = col_double(),
##   ..   novator = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

The "stag" column represents the numerical variable of Experience (time) and ranges from 0.39 to 179.

The numerical column "event" indicates whether an event occurred for each individual. The value 1 suggests that an event took place and 0 that no event took place. Through the numerical notation it is not possible to determine, what kind of event took place.

The categorical column "gender" represents the gender of each individual. The values "m - 24%" and "f - 67%" indicate male and female, respectively.

The numerical column "age" represents the age of each individual in the age span between 18 and 58 years.

The categorical column "industry" represents the industry in which each individual is employed. The values indicate different industries such as "retail - 26%", "manufacture - 13%", "other - 62%" like "Banks" or "PowerGeneration".

The categorical column "profession" represents the profession of each individual. The values "HR - 67%", "IT - 7%", "other - 26%" provide information about the specific professions within each industry.

The categorical column "traffic" represents from what pipelene the employee came to the company. The values suggest different conditions such as "youjs - 28%", "empjs - 22%," or "other - 50%".

The categorical column "coach" with the values "no - 60%", "my head - 28%" and "other - 12%" indicates whether each individual has a coach and training or not during probation.

The categorical column "head_gender" represents the gender of the head of the individual. The values "m - 52%" and "f - 48%" indicate male and female gender.

The categorical column "greywage" indicates with "white - 89%" or "grey - 11%" the wage type for each individual, where the salary does not seem to the tax authorities.

The categorical column "way" represents the way of transportation "bus - 60%", "car - 29%" or "other - 10%" for each individual.

"extraversion", "independ", "selfcontrol", "anxiety", and "novator": These numerical columns represent personality traits of each individual. The values range from low to high, providing information about the individual's level of extraversion, independence, self-control, anxiety, and innovation. Higher values indicate a higher presence of each trait.

Extraversion: 25x:1 - 1.9 29x: 1.9 - 2.8 84x: 2.8 - 3.7 293x: 3.7 - 4.6 172x: 4.6 - 5.5 199x: 5.5 - 6.4 135x: 6.4 - 7.3 97x: 7.3 - 8.2 55x: 8.2 - 9.1 40x: 9.1 - 10

Independent: 26x: 1 - 1.9 61x: 1.9 - 2.8 102x: 2.8 - 3.7 120x: 3.7 - 4.6 172x: 4.6 - 5.5 333x: 5.5 - 6.4 142x: 6.4 - 7.3 102x: 7.3 - 8.2 38x: 8.2 - 9.1 33x: 9.1 - 10

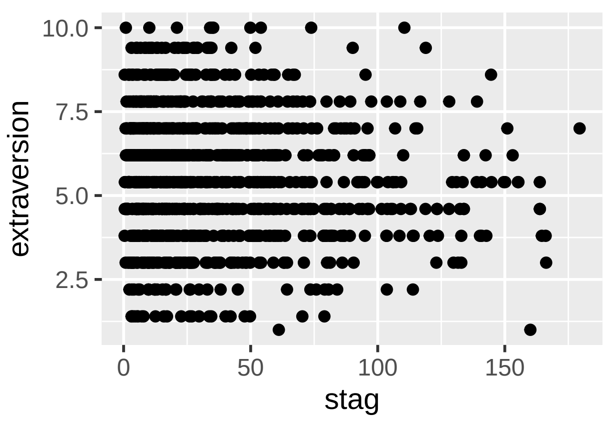
Selfcontrol: 37x: 1 - 1.9 76x: 1.9 - 2.8 92x: 2.8 - 3.7 156x: 3.7 - 4.6 140x: 4.6 - 5.5 153x: 5.5 - 6.4 287x: 6.4 - 7.3 85x: 7.3 - 8.2 59x: 8.2 - 9.1 44x: 9.1 - 10

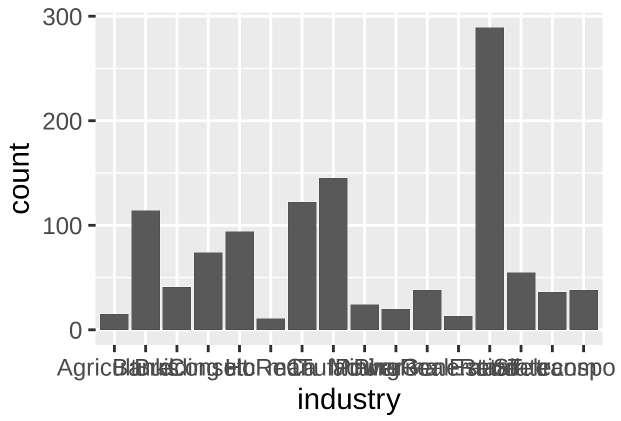
Anxiety: 52x: 1.70 - 2.53 92x: 2.53 - 3.36 130x: 3.36 - 4.19 180x: 4.19 - 5.02 200x: 5.02 - 5.85 173x: 5.85 - 6.68 129x: 6.68 - 7.51 87x: 7.51 - 8.34 58x: 8.34 - 9.17 28x: 9.17 - 10

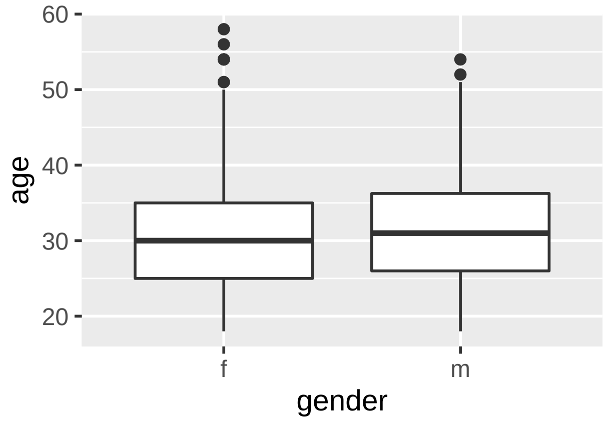
Novator: 17x: 1 - 1.9 35x: 1.9 - 2.8 71x: 2.8 - 3.7 186x: 3.7 - 4.6 165x: 4.6 - 5.5 168x: 5.5 - 6.4 173x: 6.4 - 7.3 144x: 7.3 - 8.2 151x: 8.2 - 9.1 19x: 9.1 - 10

Exploration

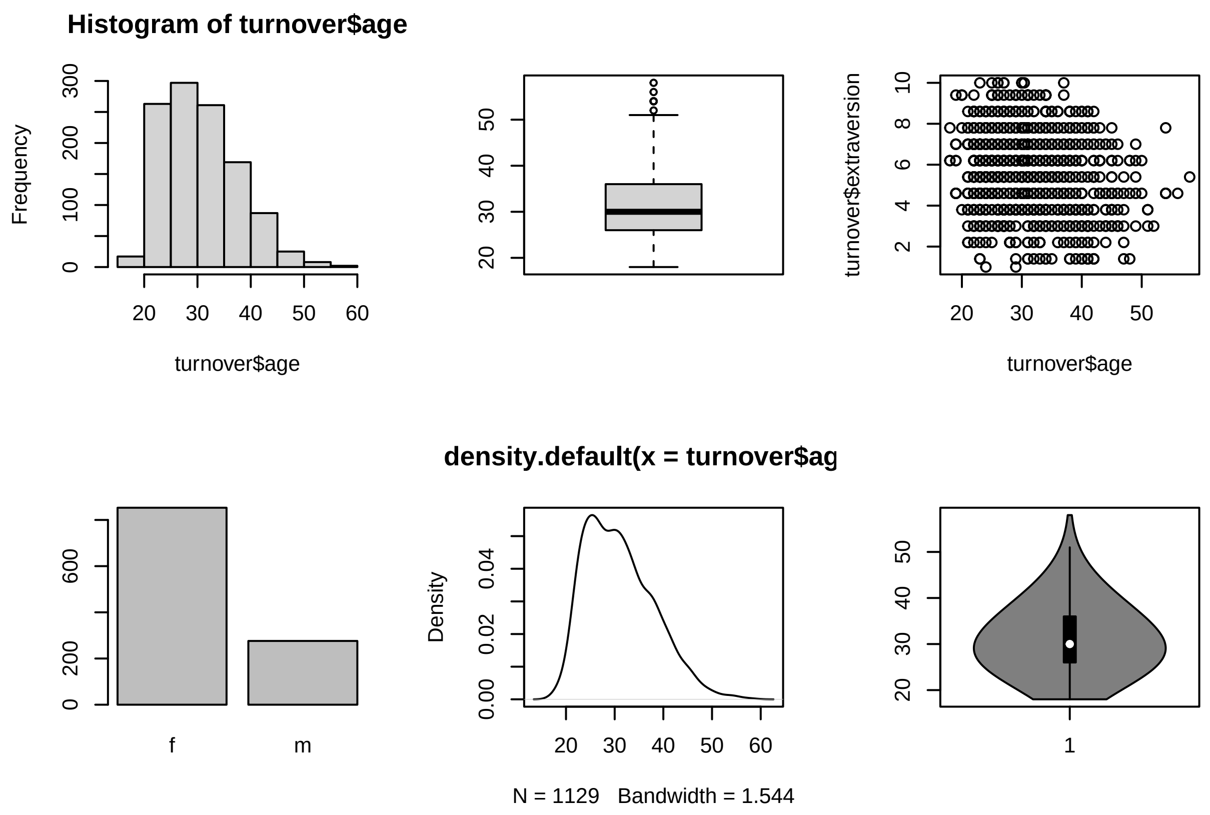
```r
# scatterplot of stag vs. extraversion
plot.scatter <- ggplot(data = turnover, aes(x = stag, y = extraversion)) +
  geom_point()

# visualizing the distribution of a categorical variable - distribution of industry
plot.distca <- ggplot(data = turnover, aes(x = industry)) +
  geom_bar()

# distribution of a continuous variable - box plot of age by gender
plot.distco <- ggplot(data = turnover, aes(x = gender, y = age)) +
  geom_boxplot()

grid.arrange(plot.scatter, plot.distca, plot.distco, ncol=2)
```







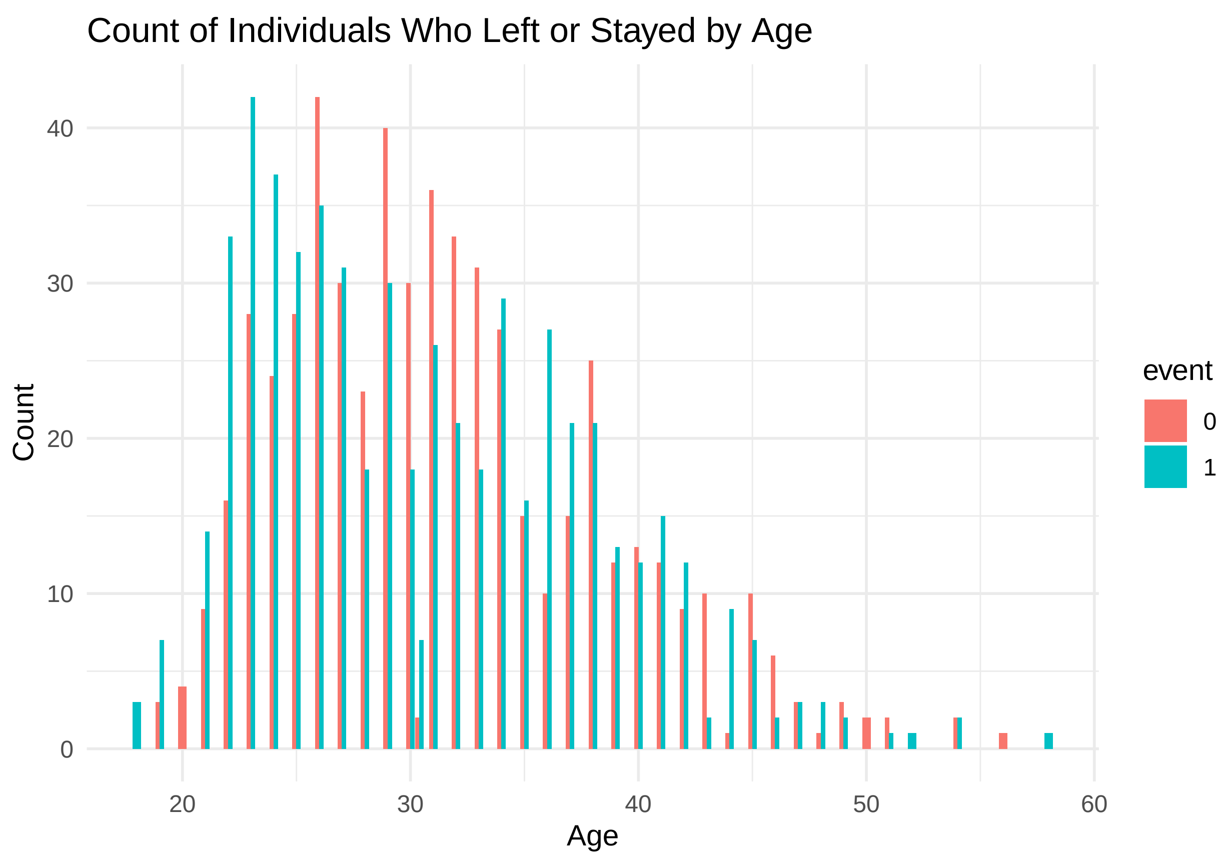
```
par(mfrow = c(2,3))

# histogram of the age
hist(turnover$age)

# boxplot of the age
boxplot(turnover$age)

# plot of age and extraversion
plot(turnover$age, turnover$extraversion)

# barplot of gender - male and female
barplot(table(turnover$gender))

# density plots
plot(density(turnover$age))

# violin plot
vioplot(turnover$age)
```



The interpretation of the data shows immediately, that the dataset contains more young people than older. It peaks from 25 - 30 years old, has as well much 20 to 35 years old, and declines steadily from 35 to 60 years old. Also you see, that there are about three times as many female individuals, than male.

We then thought, that there might be the possibility, that due to the new generational behavior from generation (Y), Z and A, there is a higher churn rate as their generations tend to have shorter periods of employment. So we plotted the plot below, to analyze this hypothesis:

```r
# Convert 'event' to factor if it's not already
turnover2 <- turnover
turnover2$event <- as.factor(turnover2$event)

# Create a bar chart showing count of individuals who left or stayed by age
ggplot(turnover2, aes(x = age, fill = event)) +
  geom_bar(position = "dodge") +
  xlab("Age") +
  ylab("Count") +
  ggtitle("Count of Individuals Who Left or Stayed by Age") +
  theme_minimal()
```



From here we decided to build our models, to be able to back the further analysis with a profound scientific methodology.

One-Hot encoding

In a next step we performed one-hot encoding on the categorical variables in the dataset for several reasons. This, as it was necessary to ensure compatibility with our machine learning algorithms which typically require numerical input. By converting categorical variables into binary vectors using one-hot encoding, we could represent the categorical information in a numerical format that could then be processed by these algorithms.

Another crucial reason for employing one-hot encoding was to eliminate any assumptions of ordinality among the categorical variables. Because unlike numerical variables, categorical variables normally lack a natural order or numerical relationship between their values. By using one-hot encoding, we avoided imposing any ordinality assumptions and created separate binary variables for each category. This approach ensured that the algorithm did not misinterpret any non-existent numerical patterns or relationships.

The one-hot encoding helped us to address the issue of magnitude bias. Since categorical variables have different levels or categories without a natural numerical relationship, assigning arbitrary numerical values to these categories can introduce bias. The one-hot encoding overcame this problem by assigning equal weight to each category, thereby preventing any inherent bias from influencing the analysis.

Through this step we were able to enhance the interpretability of the data. The resulting binary vectors provided clear and interpretable information. Each category within a variable became its own feature, and the presence or absence of a category was represented by 1 or 0. This allowed us easier interpretation of the impact or importance of each category in the analysis.

```r
dummy <- dummyVars(" ~ .", data=turnover)
```

```
#perform one-hot encoding on data frame
turnover.oh <- data.frame(predict(dummy, newdata=turnover))

str(turnover.oh)
```

```
## 'data.frame':    1129 obs. of  59 variables:
##  $ stag                       : num  7.03 22.97 15.93 15.93 8.41 ...
##  $ event                      : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ genderf                    : num  0 0 1 1 0 1 1 1 1 1 ...
##  $ genderm                    : num  1 1 0 0 1 0 0 0 0 0 ...
##  $ age                        : num  35 33 35 35 32 42 42 28 29 30 ...
##  $ industryAgriculture        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ industryBanks              : num  1 1 0 0 0 0 0 0 1 0 ...
##  $ industryBuilding           : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ industryConsult            : num  0 0 0 0 0 0 0 0 0 1 ...
##  $ industryetc                : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ industryHoReCa             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ industryIT                 : num  0 0 0 0 0 0 0 0 0 0 ...
##   ... output truncated ...
```

Then in a next step we performed normalization on selected scalar variables in the dataset. Our goal of the normalization has been to rescale the values of these variables to a common scale, between 0 and 1. This process helped us to eliminate the influence of different scales and units, allowing for fairer comparisons and accurate analysis.

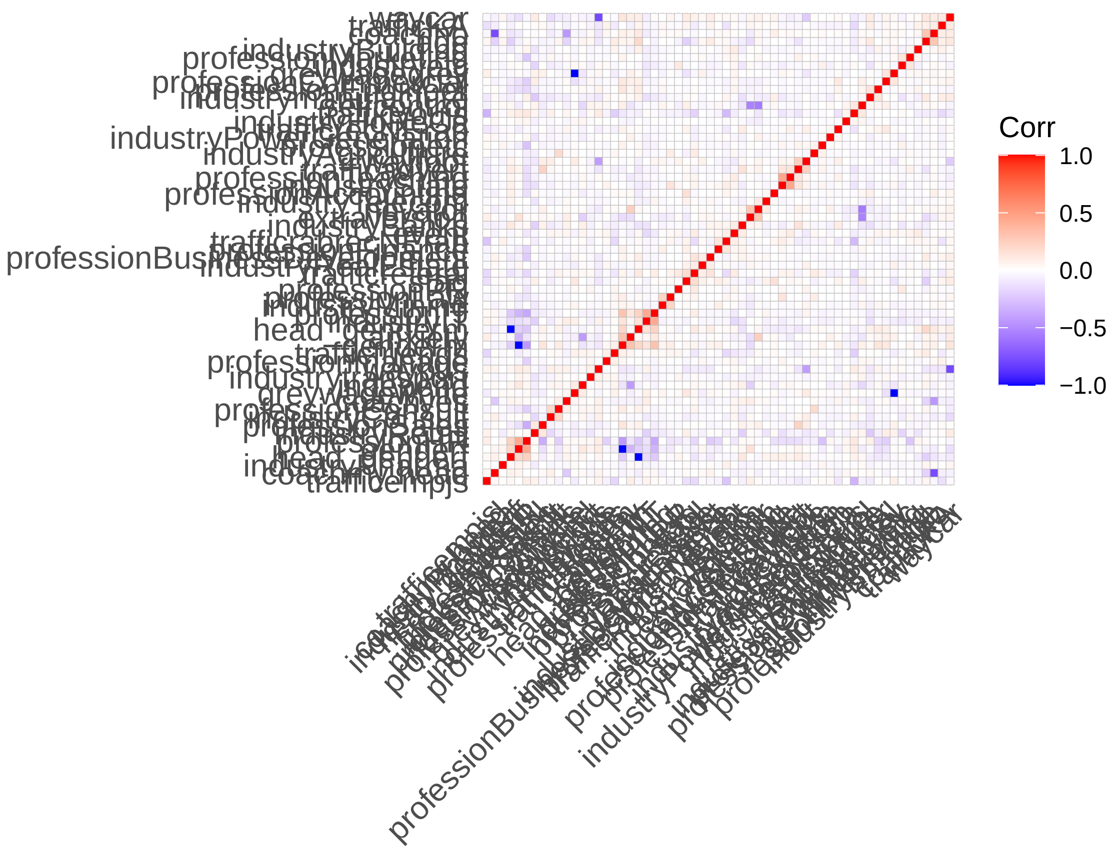
The variables selected for normalization were "extraversion", "independ", "selfcontrol", "anxiety", "novator", and "stag". As these variables represented different attributes of measurements of individuals within the dataset.

```
# Normalize selected columns
columns_to_normalize <- c("extraversion", "independ", "selfcontrol",
                          "anxiety", "novator", "stag")
turnover.oh[columns_to_normalize] <- scale(turnover.oh[columns_to_normalize],
                                           center = FALSE,
                                           scale = apply(turnover.oh[columns_to_normalize], 2, range)[2
```

Then we wanted to identify and remove highly correlated variables from our dataset. This, because highly correlated variables will introduce redundancy and multicollinearity issues, negatively affecting the accuracy and interpretability of our models.

To achieve this, we performed the following steps: We computed the correlation matrix to measure the pairwise correlation between variables in the dataset. Then we created a visualization of the correlation matrix to examine the correlation patterns and identify highly correlated variables. Highly correlated variables were determined by us based on a specified threshold, indicating a strong correlation between two variables. The indices or names of the highly correlated variables were identified and a reduced version of the dataset was created by removing the highly correlated variables. Then a new correlation matrix was generated for the reduced dataset to confirm that the highly correlated variables had been successfully removed. By removing highly correlated variables, our analysis aimed to enhance the quality and reliability of the data, ensuring that the model built upon this dataset would not be influenced by redundant or collinear information.

```
correlationMatrix <- cor(turnover.oh)
ggcorrplot(correlationMatrix, hc.order = TRUE)
```



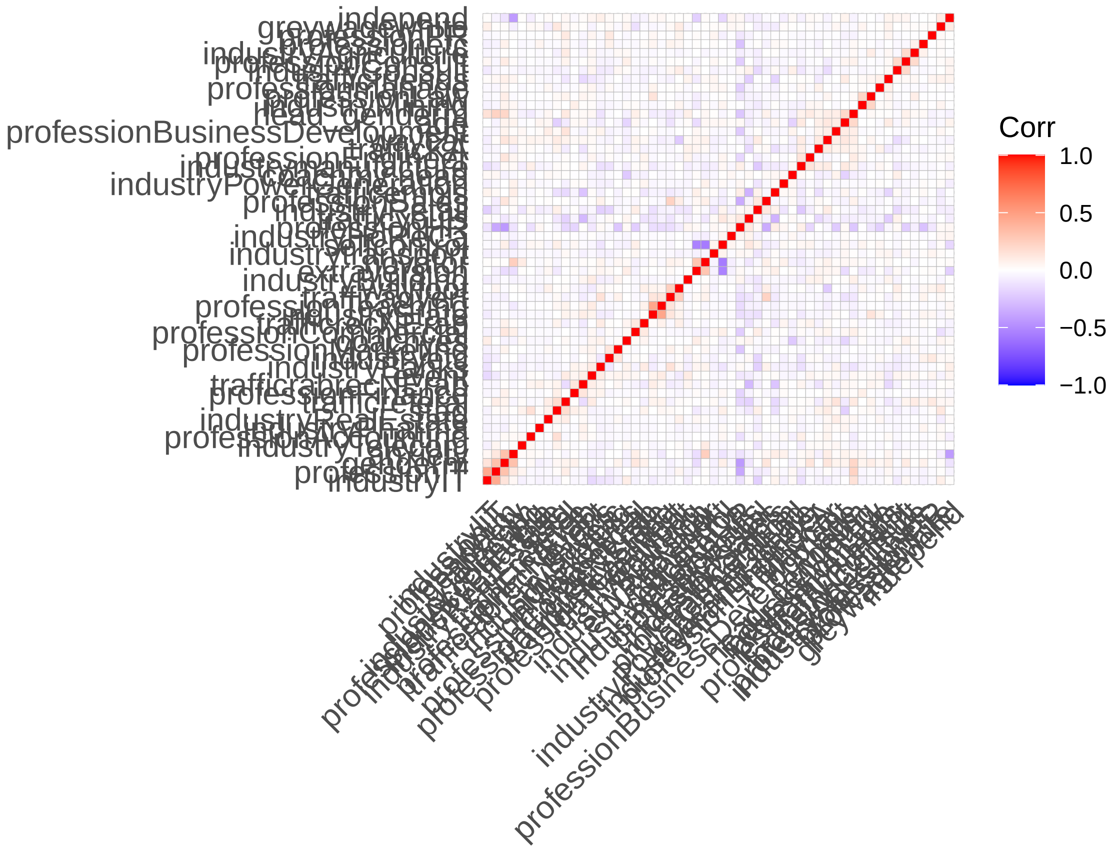
```r
# find attributes that are highly corrected ( > 0.75 )
highlyCorrelated <- findCorrelation(correlationMatrix, cutoff=0.75)
highlyCorrelated <- sort(highlyCorrelated)

# print indexes of highly correlated attributes
print(highlyCorrelated)
```

```
## [1]  3 46 48 50 52
```

```r
names(turnover.oh[highlyCorrelated])
```

```
## [1] "genderf"      "coachno"      "head_genderf" "greywagegrey" "waybus"
```

```r
turnover.reduced = turnover.oh[,-c(highlyCorrelated)]

ggcorrplot(cor(turnover.reduced), hc.order = TRUE)
```



In our next step the first modification involved converting the variable named event into a factor. This transformation has been performed to treat event as a categorical variable, allowing for the distinct categories or groups it represents to be properly recognized and utilized in subsequent analyses.

The second adjustment pertained to the age variable. To ensure uniformity and integrity in the representation of ages, all values within this variable were converted to integers. This conversion ensures that fractional or decimal components were removed, aligning the data type with the discrete nature of age values.

```r
turnover.reduced$event <- as.factor(turnover.reduced$event) # Convert event as factor
turnover.oh$event <- as.factor(turnover.oh$event)
turnover.reduced$age <- as.integer(turnover.reduced$age) # Make sure all ages are integers
```

Linear Models

Next, we explored linear models and tried to use them to predict the likelihood of an employee leaving the company. Philipp Drebes took the lead on this Linear Model section.

Our primary research topic has been employee turnover, which has been recorded by the binomial variable `event` in our data set. Later we analyzed this dependent variable with the use of GLMs, GAMs, SVM and ANN.

First we tried to predict the experience (time) of an employee, given their industry, profession, age and gender, as we expected them to have an effect.

```r
fit <- lm(stag ~ industry + profession + age + gender, data = turnover)
summary(fit)
```

```
##
## Call:
```

```
## lm(formula = stag ~ industry + profession + age + gender, data = turnover)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -63.850 -23.378  -7.259  15.329 135.837
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  102.7384    14.1189   7.277 6.50e-13 ***
## industryBanks                 12.8273     9.2123   1.392 0.164082
## industryBuilding              20.9947    10.0403   2.091 0.036754 *
## industryConsult                8.9464     9.3307   0.959 0.337864
## industryetc                   17.8158     9.3023   1.915 0.055727 .
## industryHoReCa                23.9261    13.1265   1.823 0.068615 .
## industryIT                    12.8831     9.2621   1.391 0.164523
## industrymanufacture           22.5787     9.1416   2.470 0.013667 *
## industryMining                20.7469    10.9587   1.893 0.058597 .
##  ... output truncated ...
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 32.51 on 1097 degrees of freedom
## Multiple R-squared:  0.1158, Adjusted R-squared:  0.09082
## F-statistic: 4.635 on 31 and 1097 DF,  p-value: 3.035e-15
```

The residuals were representing the differences between the predicted and actual values of employee experience (stag). The values ranged from -63.850 to 135.837, with a median residual of -7.259. These values indicated the model's prediction errors.

The coefficients estimated the impact of each predictor variable on employee experience. For example, the coefficient for the variable industryRealEstate has been 45.2553. This suggested that being in the real estate industry is associated with an increase in employee experience by 45.2553 units, compared to the reference category.

The multiple R-squared value of 0.1158 indicated that the predictors included in the model explained approximately 11.58% of the variability in employee experience. Which indicates, that there might be other relevant factors, not included in the model.

The F-statistic of 4.635 with a small p-value (3.035e-15) indicated that the model, as a whole, has been statistically significant. This suggested that the predictors collectively had a significant impact on predicting employee experience.

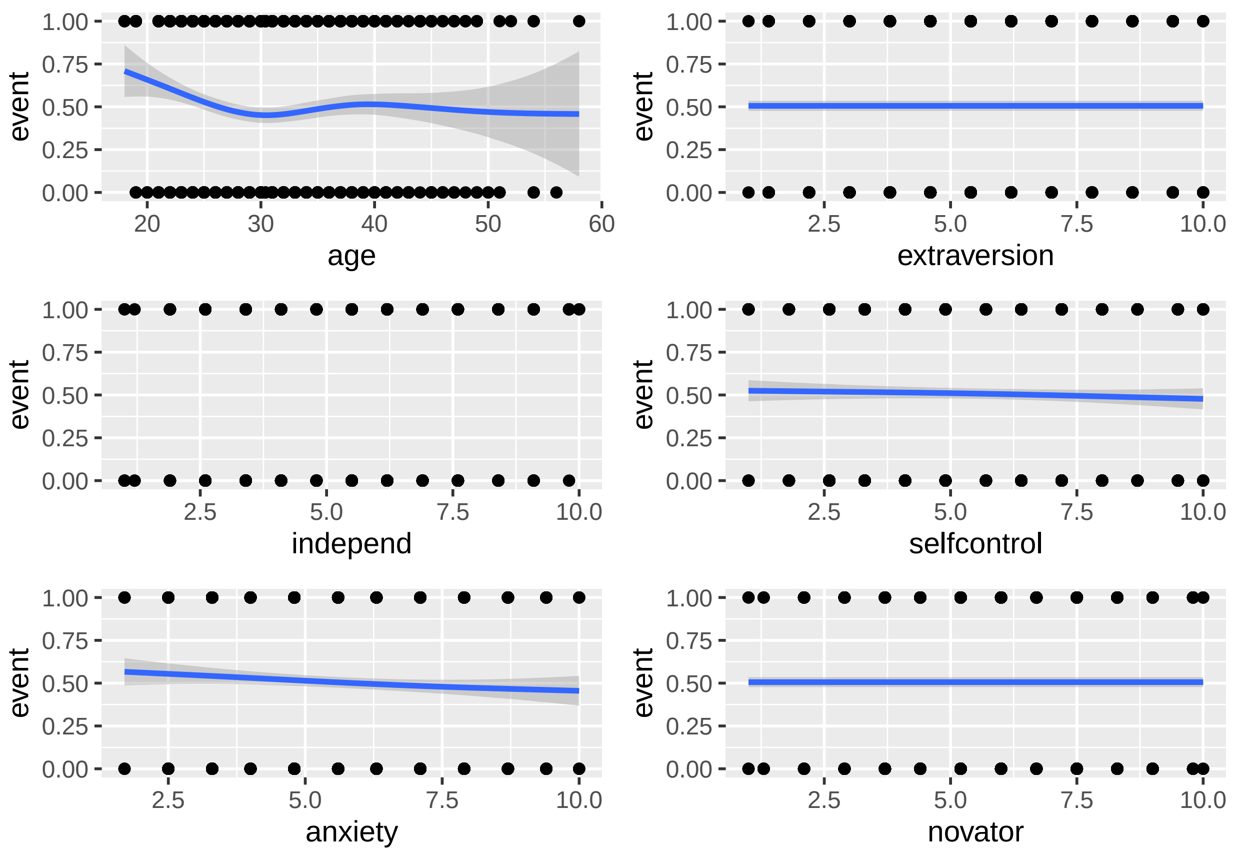
Non-linearities

In the next step of our analysis, we checked the relationships between numerical variables and employee turnover to find potential non-linear patterns. We focused on variables such as age, extraversion, independence, self-control, anxiety, and innovator.

```r
## Age
gg.age <- ggplot(data = turnover, mapping = aes(y = event, x = age)) +
  geom_point()

plot.age <- gg.age + geom_smooth()

## Extraversion
gg.extraversion <- ggplot(data = turnover, mapping = aes(y = event, x = extraversion)) +
  geom_point()
```

```r
plot.extraversion <- gg.extraversion +  geom_smooth()

## Independence
gg.independence <- ggplot(data = turnover, mapping = aes(y = event, x = independ)) +
  geom_point()

plot.independence <- gg.independence + geom_smooth()

## Self-control
gg.selfcontrol <- ggplot(data = turnover, mapping = aes(y = event, x = selfcontrol)) +
  geom_point()

plot.selfcontrol <- gg.selfcontrol + geom_smooth()

## Anxiety
gg.anxiety <- ggplot(data = turnover, mapping = aes(y = event, x = anxiety)) +
  geom_point()

plot.anxiety <- gg.anxiety + geom_smooth()

## Innovator
gg.innovator <- ggplot(data = turnover, mapping = aes(y = event, x = novator)) +
  geom_point()

plot.innovator <- gg.innovator + geom_smooth()

grid.arrange(plot.age, plot.extraversion, gg.independence, plot.selfcontrol, plot.anxiety, plot.innovat
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



It appeared that only the factor age has a strong non-linear behavior, where as anxiety and selfcontrol had a minimal non-linear behavior. We will use a GAM to find its complexity.

Training the model Performing feature selection before training a model is a good practice to improve model performance and reduce overfitting. So here we used recursive feature elimination (RFE) before training the model. We used only the one-hot encoded data set with correlating features included, as we expected the feature selection process to take care of it.

```
set.seed(12)

# Define the control parameters for feature selection
control <- rfeControl(functions = rfFuncs, method = "cv", number = 10)

# Specify the formula for the GAM model
formula <- event ~ s(age, bs = "tp", k = -1) + .

# Define a smaller range of subset sizes
subset_sizes <- c(1:10)

# Reduce the number of values in the tuning grid
tune_grid <- data.frame(nselect = 1:10)

unregister_dopar <- function() {
  env <- foreach:::.foreachGlobals
  rm(list=ls(name=env), pos=env)
}
```

```r
# Enable parallel processing
cl <- makeCluster(4)
registerDoParallel(cl)

# Run the feature selection algorithm
rfe_result <- rfe(turnover.oh[, -2], turnover.oh$event,
                  sizes = subset_sizes, rfeControl = control,
                  method = "gam", tuneGrid = tune_grid, verbose = FALSE)

# Stop the parallel processing
stopCluster(cl)

unregister_dopar()

# Get the selected features
selected_features <- names(turnover.reduced[, -2])[rfe_result$optVariables]
print(selected_features)
```

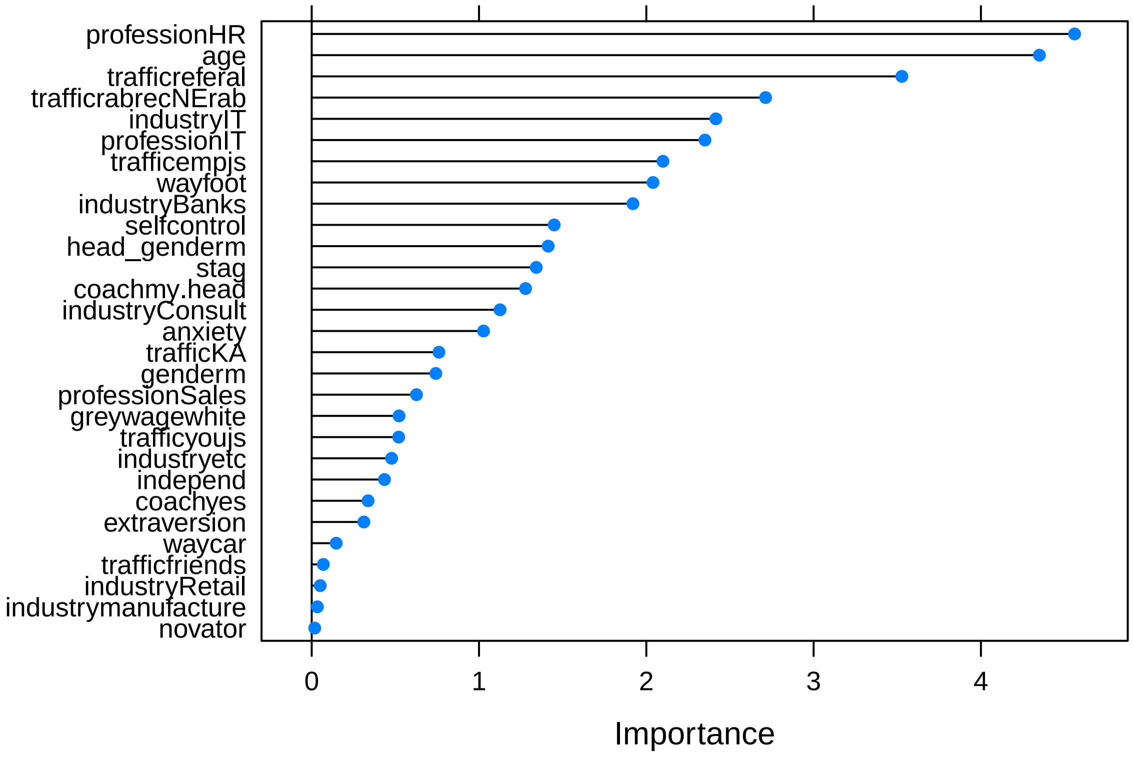
```
##  [1] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
## [26] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
## [51] NA NA NA NA NA NA NA NA
```

No features where selected. Therefore, we used again the `turnover.reduced` data set, where we tried a different approach.

```r
set.seed(12)
indices <- createDataPartition(turnover.reduced$event, p=.85, list = F)

train <- turnover.reduced %>%
  slice(indices[, 1])
test_in <- turnover.reduced %>%
  slice(-indices[, 1]) %>%
  dplyr::select(-event)
test_truth <- turnover.reduced %>%
  slice(-indices[, 1]) %>%
  pull(event)
```

```r
# ensure results are repeatable
set.seed(12)

# prepare training scheme
control <- trainControl(method="repeatedcv", number=10, repeats=3)

# Enable parallel processing
cl <- makeCluster(4)
registerDoParallel(cl)

# train the model
model.gam <- train(event ~ ., data=train, family = "binomial",
               method="gam", preProcess=c("scale", "center"), trControl=control)

# Stop the parallel processing
stopCluster(cl)
unregister_dopar()
```

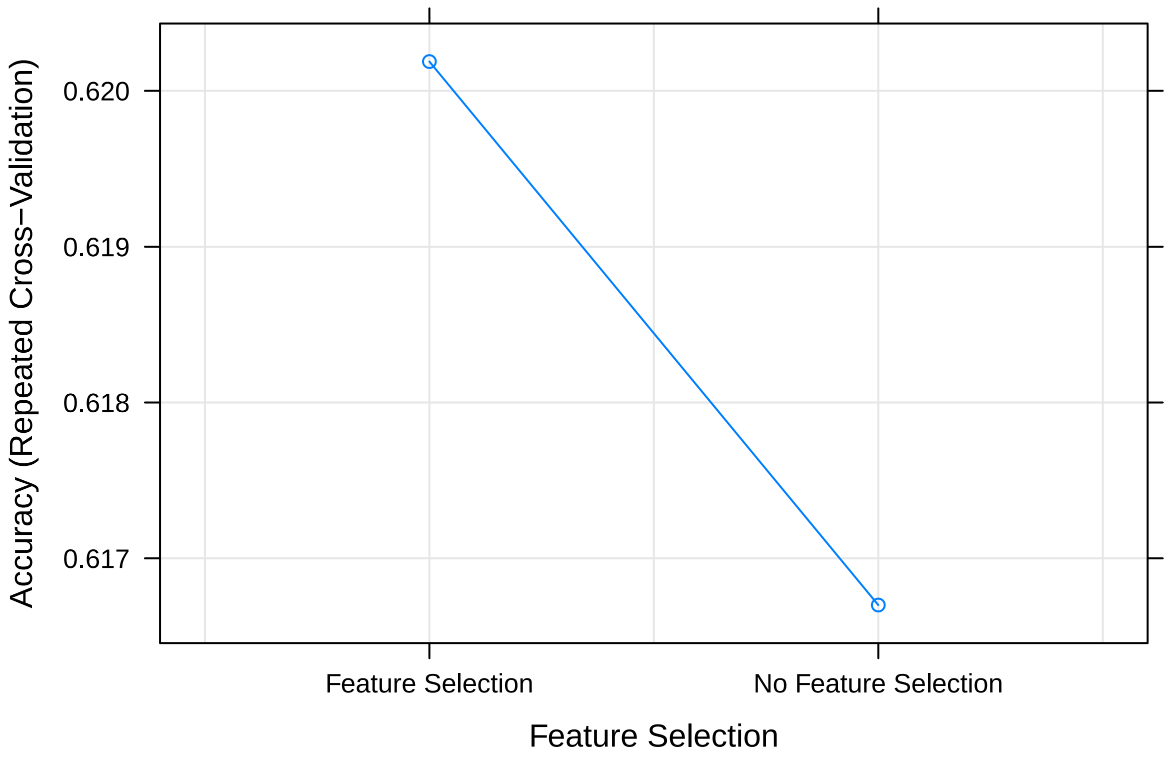
```
# estimate variable importance
importance <- varImp(model.gam, scale=FALSE)

# summarize importance
# print(importance)
plot(importance)
```



```
summary(model.gam)
```

```
##
## Family: binomial
## Link function: logit
##
## Formula:
## .outcome ~ genderm + industryBanks + industryConsult + industryetc +
##     industryIT + industrymanufacture + industryRetail + professionHR +
##     professionIT + professionSales + trafficempjs + trafficfriends +
##     trafficKA + trafficrabrecNErab + trafficreferal + trafficyoujs +
##     coachmy.head + coachyes + head_genderm + greywagewhite +
##     waycar + wayfoot + s(anxiety) + s(extraversion) + s(selfcontrol) +
##     s(novator) + s(independ) + s(age) + s(stag)
##
## Parametric coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)          0.030777   0.070151   0.439 0.660860
## genderm             -0.117336   0.087753  -1.337 0.181186
## industryBanks        0.206490   0.082224   2.511 0.012028 *
## industryConsult      0.139688   0.078422   1.781 0.074873 .
## industryetc          0.074999   0.077455   0.968 0.332903
## industryIT          -0.263636   0.091190  -2.891 0.003840 **
## industrymanufacture  0.007698   0.081643   0.094 0.924880
## industryRetail       0.012064   0.086261   0.140 0.888772
##  ... output truncated ...
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                       edf Ref.df Chi.sq  p-value
## s(anxiety)      5.257e+00      9  9.216   0.0940 .
## s(extraversion) 1.448e-05      9  0.000   0.4877
## s(selfcontrol)  4.455e+00      9 10.160   0.0355 *
## s(novator)      6.915e-06      9  0.000   0.9592
## s(independ)     1.324e-03      9  0.001   0.3674
## s(age)          4.280e+00      9 23.027 4.47e-05 ***
## s(stag)         7.374e-01      9  2.915   0.0455 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.128   Deviance explained = 12.5%
## UBRE = 0.29123  Scale est. = 1         n = 961
```

```
plot(model.gam, residuals = TRUE)
```



```
AIC(model.gam$finalModel)
```

```
## [1] 1240.868
```

```
# Make predictions on the testing data
test_pred <- predict(model.gam, newdata = test_in)

# Convert test_truth to a factor with the same levels as test_pred
test_truth <- factor(test_truth, levels = levels(test_pred))

# Evaluate the performance of the model using confusion matrix
conf_matrix <- confusionMatrix(test_pred, test_truth)
conf_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 49 34
##          1 34 51
##
##                Accuracy : 0.5952
##                  95% CI : (0.5169, 0.6702)
##     No Information Rate : 0.506
##     P-Value [Acc > NIR] : 0.01244
##
##                   Kappa : 0.1904
##
##  Mcnemar's Test P-Value : 1.00000
##
##             Sensitivity : 0.5904
##             Specificity : 0.6000
##          Pos Pred Value : 0.5904
##          Neg Pred Value : 0.6000
##              Prevalence : 0.4940
##          Detection Rate : 0.2917
##    Detection Prevalence : 0.4940
##       Balanced Accuracy : 0.5952
##
##        'Positive' Class : 0
##
```

All of the suspected factors are smooth terms and are indeed higher order polynomials.

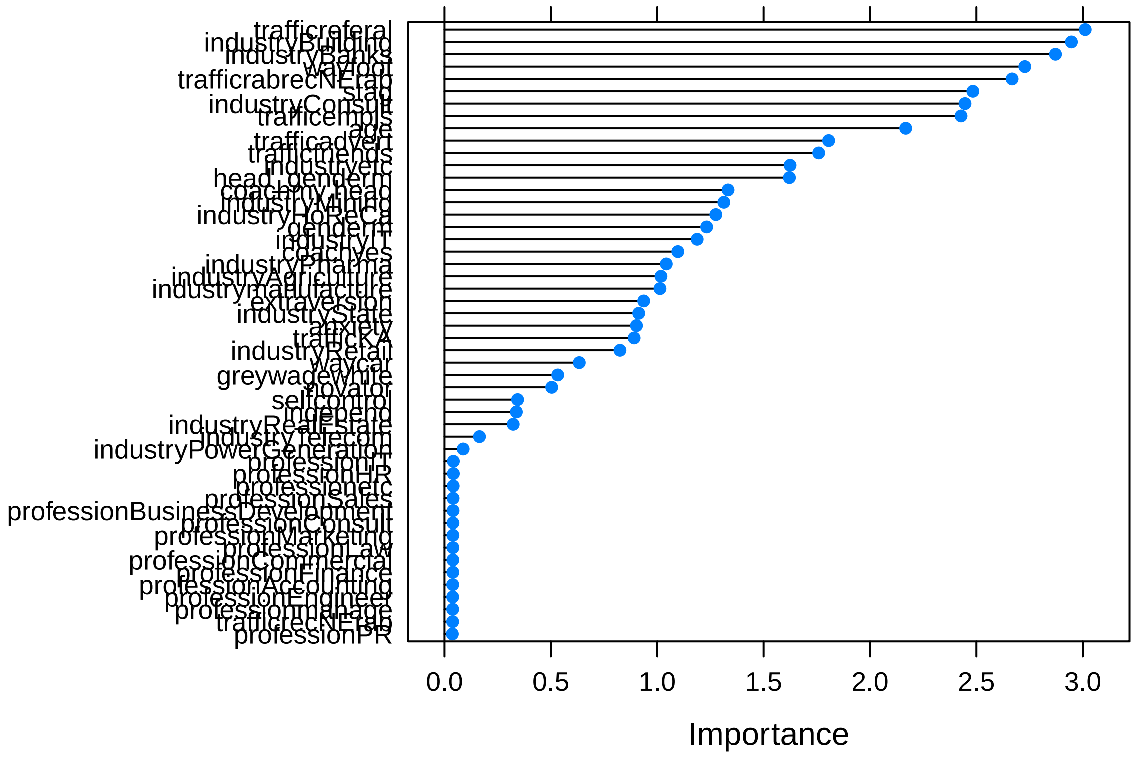
It has been still not a good fit, with only around 12.8% of the variation of the model explained. The accuracy of 59.52%, the Sensitivity of 59.04% and the Specificity of 60% indicate already, that the model still can be improved and is only slightliy above the no information rate. We also saw, that the feature selection only had a slight, almost negligible, impact on accuracy.

Generalized Linear Models - binomial / poisson

Next we have fit a generalized linear model in order to predict employee turnover.

```
# ensure results are repeatable
set.seed(12)
```

```r
# prepare training scheme
control <- trainControl(method="repeatedcv", number=10, repeats=3)

# Enable parallel processing
cl <- makeCluster(4)
registerDoParallel(cl)

# train the model
model.glm <- train(event ~ ., data=turnover.reduced, family = "binomial",
                method="glm", preProcess=c("scale", "center"), trControl=control)

# Stop the parallel processing
stopCluster(cl)
unregister_dopar()

# estimate variable importance
importance <- varImp(model.glm, scale=FALSE)

# summarize importance
# print(importance)
plot(importance)
```



```r
summary(model.glm)
```

```
##
## Call:
```

```
## NULL
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.01388  -1.05583   0.00056   1.04303   2.10600
##
## Coefficients: (3 not defined because of singularities)
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)                   0.173342   4.080159   0.042  0.96611
## stag                         -0.179826   0.072408  -2.483  0.01301 *
## genderm                      -0.099987   0.081131  -1.232  0.21780
## age                          -0.163337   0.075346  -2.168  0.03017 *
## industryAgriculture           0.082211   0.080793   1.018  0.30889
## industryBanks                 0.361355   0.125830   2.872  0.00408 **
## industryBuilding              0.286348   0.097176   2.947  0.00321 **
## industryConsult               0.264162   0.107974   2.447  0.01442 *
## industryetc                   0.187252   0.115287   1.624  0.10433
## industryHoReCa                0.089240   0.069964   1.276  0.20212
## industryIT                   -0.155816   0.131192  -1.188  0.23495
##  ... output truncated ...
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1565.0  on 1128  degrees of freedom
## Residual deviance: 1390.2  on 1078  degrees of freedom
## AIC: 1492.2
##
## Number of Fisher Scoring iterations: 14
```

```
# Make predictions on the testing data
test_pred <- predict(model.glm, newdata = test_in)
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
# Convert test_truth to a factor with the same levels as test_pred
test_truth <- factor(test_truth, levels = levels(test_pred))

# Evaluate the performance of the model using confusion matrix
conf_matrix <- confusionMatrix(test_pred, test_truth)
conf_matrix
```

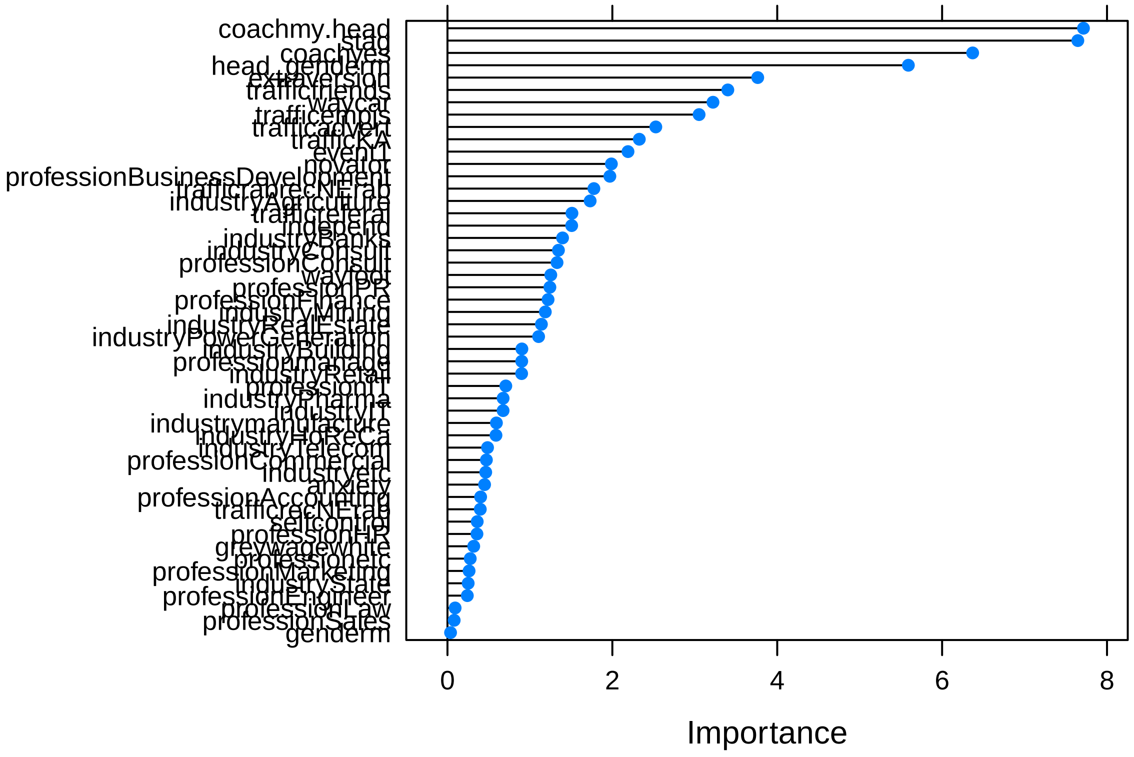
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 56 31
##          1 27 54
##
##                Accuracy : 0.6548
##                  95% CI : (0.5776, 0.7263)
##     No Information Rate : 0.506
##     P-Value [Acc > NIR] : 6.928e-05
##
```

```
##                  Kappa : 0.3098
##
##  Mcnemar's Test P-Value : 0.6936
##
##             Sensitivity : 0.6747
##             Specificity : 0.6353
##          Pos Pred Value : 0.6437
##          Neg Pred Value : 0.6667
##              Prevalence : 0.4940
##          Detection Rate : 0.3333
##    Detection Prevalence : 0.5179
##       Balanced Accuracy : 0.6550
##
##        'Positive' Class : 0
##
```

The accuracy improves to 65.48%, compared to the 59.52% of the GAM which we fitted prior. So far, we would prefer the GLM for further analysis and prediction.

We also wanted to try to predict an employees age by their amount of experience and multiple other factors. We took the Poisson regression model and fitted it to the 'age' variable using the `glm()` function.

```
set.seed(2)

# prepare training scheme
control <- trainControl(method="repeatedcv", number=10, repeats=3)

# Enable parallel processing
cl <- makeCluster(4)
registerDoParallel(cl)

# train the model
model.glm <- train(age ~ ., data=turnover.reduced, family = "quasipoisson",
                   method="glm", trControl=control)

# Stop the parallel processing
stopCluster(cl)
unregister_dopar()

# estimate variable importance
importance <- varImp(model.glm, scale=FALSE)

# summarize importance
# print(importance)
plot(importance)
```



```
summary(model.glm)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7425  -0.8199  -0.1287   0.6732   4.0602
##
## Coefficients: (3 not defined because of singularities)
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 3.4569007  0.1111363  31.105  < 2e-16 ***
## stag                       -0.2593545  0.0339196  -7.646 4.57e-14 ***
## event1                     -0.0273118  0.0124725  -2.190 0.028755 *
## genderm                     0.0006345  0.0168674   0.038 0.970000
## industryAgriculture        -0.1063573  0.0614503  -1.731 0.083776 .
## industryBanks              -0.0528852  0.0378846  -1.396 0.163015
## industryBuilding           -0.0401104  0.0444295  -0.903 0.366841
## industryConsult            -0.0534449  0.0396977  -1.346 0.178490
## industryetc                -0.0175779  0.0378767  -0.464 0.642682
## industryHoReCa              0.0386159  0.0656116   0.589 0.556285
##  ... output truncated ...
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for quasipoisson family taken to be 1.179111)
##
##     Null deviance: 1728.1  on 1128  degrees of freedom
## Residual deviance: 1245.9  on 1078  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 4
```

Our deviance residuals ranged from -2.7425 to 4.0602, indicating some variability between the observed and predicted values. Positive residuals indicated that the predicted age is higher than the observed age, while negative residuals indicated that the predicted age has been lower than the observed age. These values suggested that the model has some variability in its fit to the data, but the median residual is closer to zero, indicating a reasonably good fit.

Support Vector Machines

Next we have been using Support Vector Machines (SVM) to predict the employee turnover. Antonia Durisch took the lead on the Support Vector Machine section.

We started with setting a random seed to ensure reproducibility of the results. Next, we created data partitions using the createDataPartition() function. We specifyed that we want to partition the data based on the "event" variable from the "turnover.reduced" dataset, and we set the proportion of the data to be included in the training set as 85%. The list = F argument indicated that we wanted to obtain a vector of indices rather than a list of partitioned data. We then created the training set by subsetting the "turnover.reduced" dataset using the indices obtained from the data partition. This ensured that the training set contained 85% of the data. The remaining data has been used to create the test set. We subset the "turnover.reduced" dataset using the negative indices (i.e., excluding the indices used for the training set), and we removed the "event" variable from the test set using the dplyr::select() function. We also created a separate vector called "test_truth" that contains the actual "event" values for the observations in the test set. We used this later for evaluating our model's performance.

In the end, we had divided the data into a training set (85% of the data) and a test set (15% of the data) for the purpose of training and evaluating the SVM model to predict employee turnover.

```r
set.seed(123)
indices <- createDataPartition(turnover.reduced$event, p=.85, list = F)

train <- turnover.reduced %>%
  slice(indices[, 1])
test_in <- turnover.reduced %>%
  slice(-indices[, 1]) %>%
  dplyr::select(-event)
test_truth <- turnover.reduced %>%
  slice(-indices[, 1]) %>%
  pull(event)
```

Train the SVM

In this chapter, the objective has been to build and evaluate a support vector machine (SVM) model for predicting employee turnover. Also here, a specific seed value has been set, but this time a seed value of 100 has been set using set.seed(100). This ensured again that the random number generation process produced the same results every time the code has been executed, promoting reproducibility. Then, the tune() function has been employed to tune the parameters of the SVM model. The specific goal was to tune the cost parameter for a linear SVM model with C-classification. The tuning has been conducted on the training data (train), and a range of values was specified for the cost parameter using the ranges argument. The tune() function did output the best-tuned model based on the specified parameter ranges. This best-tuned model has been assigned to the variable svm.linear using the line svm.linear <- tune.out$best.model. This model was later used for predictions.

To evaluate the performance of our model, our predictions were made on the testing data (test_in) using the best-tuned SVM model. The predict() function has been used to generate these predictions, and the predicted values were stored in the test_pred variable. To facilitate accurate evaluation, the test_truth variable, which contained the actual event values for the observations in the test set, were converted to a factor with the same levels as the predicted values (test_pred). This ensured that the predicted and actual values had the same levels, enabling proper comparison.

Finally, the confusionMatrix() function has been employed to calculate various performance metrics, such as accuracy, precision, recall, and the F1 score. It compared the predicted values (test_pred) with the actual values (test_truth) and generated a confusion matrix. The resulting confusion matrix has been stored in the conf_matrix variable, providing a comprehensive evaluation of our SVM model's performance.

```r
set.seed(100)
# Fit the SVM model on the training data
#svm.linear <- svm(event ~ ., data = train, kernel = "linear",
#                  type = "C-classification", scale = TRUE, cost = 10)

tune.out <- tune(
  svm, event ~ ., data = train, kernel = "linear", type = "C-classification", scale = TRUE,
  ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100))
)
svm.linear <- tune.out$best.model

# Make predictions on the testing data
test_pred <- predict(svm.linear, newdata = test_in)

# Convert test_truth to a factor with the same levels as test_pred
test_truth <- factor(test_truth, levels = levels(test_pred))

# Evaluate the performance of the model using confusion matrix
conf_matrix <- confusionMatrix(test_pred, test_truth)
conf_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 54 32
##          1 29 53
##
##                Accuracy : 0.6369
##                  95% CI : (0.5593, 0.7096)
##     No Information Rate : 0.506
##     P-Value [Acc > NIR] : 0.0004222
##
##                   Kappa : 0.274
##
##  Mcnemar's Test P-Value : 0.7978939
##
##             Sensitivity : 0.6506
##             Specificity : 0.6235
##          Pos Pred Value : 0.6279
##          Neg Pred Value : 0.6463
##               Prevalence : 0.4940
##           Detection Rate : 0.3214
##     Detection Prevalence : 0.5119
##        Balanced Accuracy : 0.6371
##
##         'Positive' Class : 0
##
```

The confusion matrix and statistics provided an assessment of the performance of the SVM model. The confusion matrix showed the counts of true positive 54x, false positive 32x, false negative 29x, and true negative 53x predictions. The accuracy of the model was calculated to be 63.69%, indicating that it correctly predicted the outcome in approximately 63.69% of cases. The 95% confidence interval for the accuracy was estimated to be between 0.5593 and 0.7096. Comparing the model's performance to the no-information rate, which represents the accuracy of a naive model that always predicts the majority class, the p-value has been determined to be 0.0004222. This suggested that the SVM model significantly outperformed the naive approach. The kappa coefficient, a measure of agreement beyond chance, was computed to be 0.274. This indicated to us that only a fair level of agreement between the predicted and actual outcomes. Mcnemar's test has been conducted to evaluate if there has been a significant difference in the number of errors made by the model for different classes. The resulting p-value was 0.7978939, suggesting no significant difference in error rates between the classes. When we started analyzing the sensitivity and specificity of our model, the sensitivity -> true positive rate, was calculated to be 0.6506, indicating the model's ability to correctly identify the positive class. The specificity -> true negative rate, was determined to be 0.6235, reflecting our model's ability to correctly identify the negative class. The positive predictive value, which represented the proportion of positive predictions that were correct, was calculated to be 0.6279. And the negative predictive value, representing the proportion of negative predictions that are correct, was 0.6463. The prevalence of the positive class in the dataset has been 0.4940. The detection rate, which was the proportion of actual positive cases that were correctly predicted, has been found to be 0.3214. And last but not least, the detection prevalence, which represented the proportion of predicted positive cases, has been 0.5119.
Overall, the balanced accuracy of the model, calculated as the average of sensitivity and specificity, was 0.6371. Next we used the `tune` function again and we also tried different kernels for the SVM algorithm.

```r
set.seed(99)
tune.out <- tune(
  svm, event ~ ., data = train, type = "C-classification", scale = TRUE,
  ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100),
                kernel = c("linear", "radial", "polynomial", "sigmoid"))
)
print(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost     kernel
##    10 polynomial
##
## - best performance: 0.354854
```

```r
svm.best <- tune.out$best.model

# Make predictions on the testing data
test_pred <- predict(svm.best, newdata = test_in)

# Convert test_truth to a factor with the same levels as test_pred
```

```r
test_truth <- factor(test_truth, levels = levels(test_pred))

# Evaluate the performance of the model using confusion matrix
conf_matrix <- confusionMatrix(test_pred, test_truth)
conf_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 53 30
##          1 30 55
##
##                Accuracy : 0.6429
##                  95% CI : (0.5654, 0.7152)
##     No Information Rate : 0.506
##     P-Value [Acc > NIR] : 0.0002369
##
##                   Kappa : 0.2856
##
##  Mcnemar's Test P-Value : 1.0000000
##
##             Sensitivity : 0.6386
##             Specificity : 0.6471
##          Pos Pred Value : 0.6386
##          Neg Pred Value : 0.6471
##              Prevalence : 0.4940
##          Detection Rate : 0.3155
##    Detection Prevalence : 0.4940
##       Balanced Accuracy : 0.6428
##
##        'Positive' Class : 0
##
```

After using the `tune` function again and also trying different kernels for the SVM algorithm, we received the following values:

- 53 instances -> predicted as 0 and actually belong to class 0
- 30 instances -> predicted as 0 but actually belong to class 1
- 30 instances -> predicted as 1 but actually belong to class 0
- 55 instances -> predicted as 1 and actually belong to class 1

The accuracy has been calculated as 0.6429, which means that the model correctly predicted the outcome for approximately 64.29% of the instances. Based on these metrics, we could observe that the performance in terms of accuracy, sensitivity, specificity, and negative predictive value is slightly worse in the current input, so after using the tune function again, compared to the previous input. However, the positive predictive value is slightly better with 63.86% in the current input.

Artificial Neural Networks

In this chapter, we explored Artificial Neural Networks (ANN). ANN is a machine learning algorithm inspired by the biological structure of the human brain. We will use it to predict employee turnover. Albesa Istrefaj took the lead on the Artificial Neural Networks section.

In a first step we wanted to examine the structure of our data frame ‘turnover’, to receive information about the variables included, their data types and to see a short preview of the data. We determined the dimensions of our data frame, to receive an idea of its size -> rows and colums. Last but not least, our goal has been to identify missing values.

```
# Data overview
turnover <- read.csv("data/turnover.csv")

# View the structure of the data frame
str(turnover)
```

```
## 'data.frame':    1129 obs. of  16 variables:
##  $ stag        : num  7.03 22.97 15.93 15.93 8.41 ...
##  $ event       : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ gender      : chr  "m" "m" "f" "f" ...
##  $ age         : num  35 33 35 35 32 42 42 28 29 30 ...
##  $ industry    : chr  "Banks" "Banks" "PowerGeneration" "PowerGeneration" ...
##  $ profession  : chr  "HR" "HR" "HR" "HR" ...
##  $ traffic     : chr  "rabrecNErab" "empjs" "rabrecNErab" "rabrecNErab" ...
##  $ coach       : chr  "no" "no" "no" "no" ...
##  $ head_gender : chr  "f" "m" "m" "m" ...
##  $ greywage    : chr  "white" "white" "white" "white" ...
##  $ way         : chr  "bus" "bus" "bus" "bus" ...
##  $ extraversion: num  6.2 6.2 6.2 5.4 3 6.2 6.2 3.8 8.6 5.4 ...
##  $ independ    : num  4.1 4.1 6.2 7.6 4.1 6.2 6.2 5.5 6.9 5.5 ...
##  $ selfcontrol : num  5.7 5.7 2.6 4.9 8 4.1 4.1 8 2.6 3.3 ...
##  $ anxiety     : num  7.1 7.1 4.8 2.5 7.1 5.6 5.6 4 4 7.9 ...
##  $ novator     : num  8.3 8.3 8.3 6.7 3.7 6.7 6.7 4.4 7.5 8.3 ...
```

```
# Get the dimensions of the data frame (number of rows and columns)
dim(turnover)
```

```
## [1] 1129   16
```

```
# Check for missing values
sum(is.na(turnover))
```

```
## [1] 0
```

We converted in a first step categorical variables like “gender”, “industry”, “profession”, “traffic”, “coach”, “head_gender”, “greywage”, and “way” to factors. Next we wanted to ensure reproducibilty, why we have set the seed value to 42. We split then our dataset into training and testing sets. We did the split based on the “event” column, where we assigned 70% to the training and the remaining 30% to the testing set. This division enabled model training on the training set and evaluation on the unseen testing set. The numeric variables in both the training and testing sets, including “stag,” “age,” “extraversion,” “independ,” “selfcontrol,” “anxiety,” and “novator,” were being normalized. This normalization process rescaled the values of these variables to a common scale, between 0 and 1.

```
# Data Preprocessing

# Convert categorical variables to factors
turnover$gender <- as.factor(turnover$gender)
turnover$industry <- as.factor(turnover$industry)
turnover$profession <- as.factor(turnover$profession)
turnover$traffic <- as.factor(turnover$traffic)
turnover$coach <- as.factor(turnover$coach)
turnover$head_gender <- as.factor(turnover$head_gender)
turnover$greywage <- as.factor(turnover$greywage)
```

```
turnover$way <- as.factor(turnover$way)

# Set seed for reproducibility
set.seed(42)

# Split the dataset into training and testing sets (70% training, 30% testing)
partition <- createDataPartition(turnover$event, p = 0.7, list = FALSE)
train_data <- turnover[partition, ]
test_data <- turnover[-partition, ]

# Normalize numeric variables in the training and testing sets
train_data[, c("stag", "age", "extraversion", "independ", "selfcontrol", "anxiety", "novator")] <- scal
test_data[, c("stag", "age", "extraversion", "independ", "selfcontrol", "anxiety", "novator")] <- scale
```

Our next code block, where we have built the ANN model, started again with setting the seed to 42. We did this to ensure that any random processes involved in subsequent steps produced the same results when the code would have been run again. Then we needed to specify the relationship between the target variable "event" and the predictor variables in the data, which is why we needed to define a formula for this, but we excluded "stag" from it. Next we wanted to treat the target variable as a categorical variable with distinct levels, so we converted the "event" variable in both, training and testing set, to a factor. Then we finally could build our ANN model, where it took in the defined formula, the training data, and other parameters such as the number of neurons in the hidden layer, maximum number of iterations, weight decay parameter, and activation function for the output layer. Tracing has been disabled to reduce output verbosity. Our trained ANN model has then been used to make predictions on the test data. Last but not least, we needed to calculate the accuracy of the predictions, by comparing our predicted values with actual values in the test data.

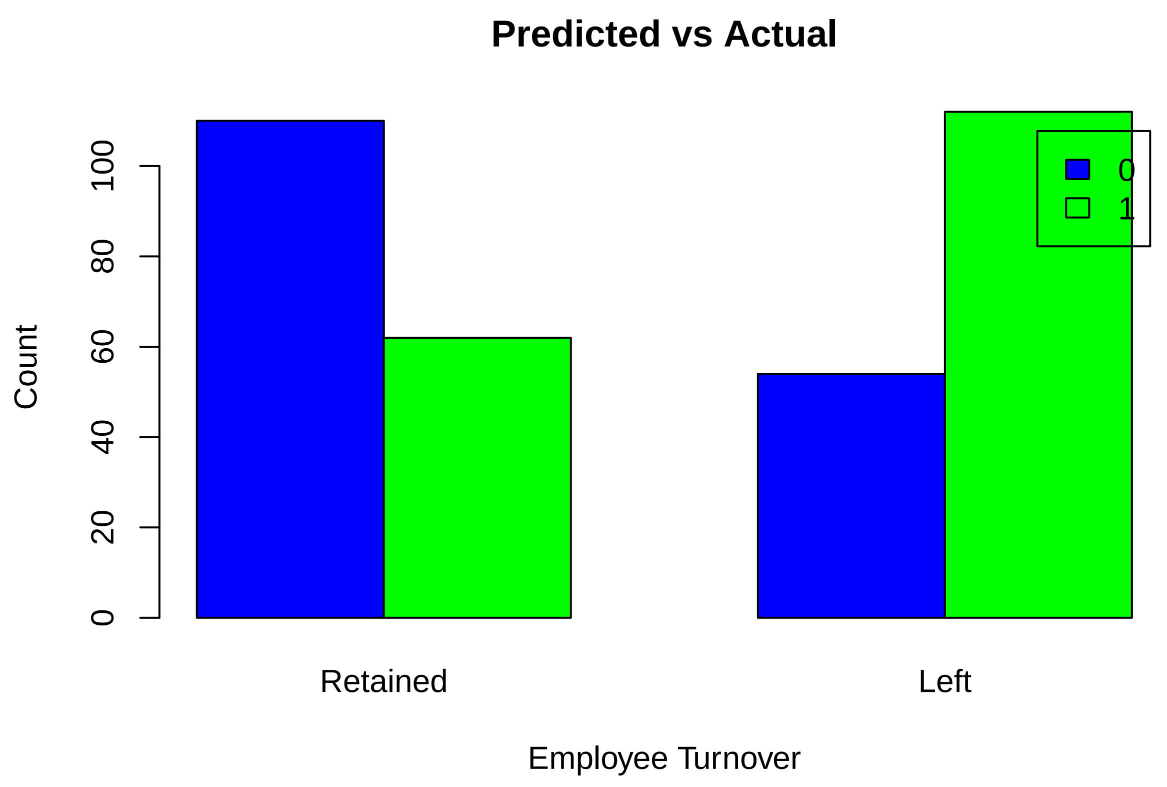
```
# Build the ANN Model

# Set the random seed for reproducibility
set.seed(42)

# Define the formula for the model
formula <- event ~ . - stag

# Convert the target variable to a factor
train_data$event <- as.factor(train_data$event)
test_data$event <- as.factor(test_data$event)

# Build the ANN model
ann_model <- nnet(
  formula,
  data = train_data,
  size = 10,  # Specify the number of neurons in the hidden layer
  maxit = 2000,  # Maximum number of iterations
  decay = 0.01,  # Weight decay parameter
  linout = FALSE,  # Use a non-linear activation function for the output layer
  trace = FALSE  # Disable tracing to reduce output
)

# Make predictions on the test data
predicted_values <- predict(ann_model, newdata = test_data, type = "class")

# Calculate accuracy
```

```
accuracy <- sum(predicted_values == test_data$event) / nrow(test_data)
cat("Accuracy: ", accuracy, "\n")
```

```
## Accuracy:  0.6568047
```

Next we created a data frame where we showed the predicted and actual values. We then counted the occurrences of each unique combination of predicted and actual values in the "prediction_data" data frame. This created a table that showed the frequency of each combination. Lastly, we created a bar plot, to be able to visualize the comparison between the predicted and actual values.

```
# Create a data frame with predicted and actual values
prediction_data <- data.frame(Predicted = predicted_values, Actual = test_data$event)

# Count the occurrences of each combination of predicted and actual values
prediction_counts <- table(prediction_data$Predicted, prediction_data$Actual)

# Plot the bar plot
barplot(prediction_counts, beside = TRUE, legend = TRUE,
        main = "Predicted vs Actual", xlab = "Employee Turnover",
        ylab = "Count", col = c("blue", "green"), names.arg = c("Retained", "Left"))
```



```
# Calculate the confusion matrix
confusion_matrix <- table(prediction_data$Actual, prediction_data$Predicted)

# Calculate true positives (TP)
TP <- confusion_matrix[2, 2]
```

```
# Calculate true negatives (TN)
TN <- confusion_matrix[1, 1]

# Calculate false positives (FP)
FP <- confusion_matrix[1, 2]

# Calculate false negatives (FN)
FN <- confusion_matrix[2, 1]

# Display the results
cat("True Positives (TP): ", TP, "\n")
```

```
## True Positives (TP):  112
```

```
cat("True Negatives (TN): ", TN, "\n")
```

```
## True Negatives (TN):  110
```

```
cat("False Positives (FP): ", FP, "\n")
```

```
## False Positives (FP):  62
```

```
cat("False Negatives (FN): ", FN, "\n")
```

```
## False Negatives (FN):  54
```

```
# Calculate sensitivity (true positive rate or recall)
sensitivity <- TP / (TP + FN)
cat("Sensitivity: ", sensitivity, "\n")
```

```
## Sensitivity:  0.6746988
```

```
# Calculate specificity (true negative rate)
specificity <- TN / (TN + FP)
cat("Specificity: ", specificity, "\n")
```

```
## Specificity:  0.6395349
```

```
# Calculate precision (positive predictive value)
precision <- TP / (TP + FP)
cat("Precision: ", precision, "\n")
```

```
## Precision:  0.6436782
```

The employee turnover has been represented by the 'event' variable, where the accuracy of our model has been 0.6568047, which meant that the model correctly predicted the employee turnover status for approximately 65.68% of the cases in the test data. However, accuracy alone has not been providing a complete picture of the model's performance, especially in imbalanced datasets where the number of employees who leave the company -> churn, is much smaller than those who stay -> non-churn. The models sensitivity has been 67.46%, its specificity 63.95% and its precision has been 64.36%. Our bar plot visualized the count of predicted and actual values for employee turnover. The blue bars represented the predicted "Retained" employees, and the green bars represented the predicted "Left" employees. The sensitivity value indicated that the model correctly identified 67.46% of the employees who actually left the company. The specificity value indicated that the model correctly identified 63.95% of the employees who actually remained with the company. The precision value suggested that around 64.36% of the employees predicted as "Left" by the model actually left the company. Our model showed moderate performance in predicting employee turnover, as it correctly identified a considerable number of employees who left the company, sensitivity, but it also had a notable number of false positives, FP, and false negatives, FN.

Conclusion

We were able to gain some insights and suggestions for your employee turnover problem.

Our data exploration identified that the most represented age group in your workforce has been between 20 and 35, where we first considered that the age might be one reason for a higher turnover rate, as especially generation (Y), Z and A have a shorter employment relationship than generation X. So we generated a plot which you have in detail in the "exploration" chapter, to understand this matter in depth.

We then explored various machine learning models to predict employee turnover and found varying degrees of accuracy across different models. The linear and generalized linear models could explain about 12.8% of the variation in employee turnover with factors like age, profession, and industry showing varying degrees of significance. Whilst the generalized additive model (GAM) had an accuracy of 59.52%, a Sensitivity of 59.04% and the Specificity of 60%, the generalized linear model showed an accuracy of 65.48%, a sensitivity of 67.47% and a specificity of 63.53%.

We also evaluated the application of support vector machines and artificial neural networks. The support vector machine (SVM) model correctly predicted approximately 63.69% of the cases, significantly better than simply predicting the most common class. Although it had a fair level of agreement between predictions and actual values, there's room for improvement. When we tried different kernels and tuning for the SVM model, the accuracy slightly improved to around 64.29%. The artificial neural network (ANN) model, on the other hand, showed a promising result with an accuracy of around 65.68%.

Given the current results, we suggest continuing with the Generalized Linear Model or the Artificial Neural Network (ANN) as they achieved one of the highest accuracies and especially the ANN is known for the ability to handle complex relationships between variables. We suggest to run in a next step a random forest model, to see if it outperforms the GLM and ANN results. We also advice to use more data for any further analysis, as the performance of the model is dependent on the input data as well. Once the analysis of the random forest model, and other suitable models has been done, we recommend a profound analysis of precision, recall and accuracy, to make a final decision on with which model to proceed.

Given the considerable role of profession, industry, and age in predicting employee turnover, we recommend taking these into account in your HR strategies. Implementing focused retention strategies targeting high-risk groups should prove beneficial in the long term.

Quotes

(Peoplekeep, 2023)

https://www.peoplekeep.com/blog/employee-retention-the-real-cost-of-losing-an-employee