Using Predictive Modeling to Analyze Employee Churn

Frederick T. Williams

April 30, 2020

Defining the Business Problem

Within the US, employers face an average employee churn of about 10%-15% annually which can prove costly to companies especially those in their early-stages of growth (Law, 2019). Kolowich (2018) wrote that when valued employees leave abruptly, it is estimated that it costs companies 30% from its other employees' annual salary to hire junior employees. However, that percentage can be increased to 400% when replace more senior position roles (Kolowich, 2018).

For companies finding a replacement difficult because the company is trying to find someone who is as productive as their former employee. The company also must consider the loss of knowledge and business acumen about the company, and time and resources needed to teach the new hire. As a result, this process can be a serious problem for companies that are facing high rates of attrition due to the extra load management being placed on other employees (Ashe-Edmunds, 2017). However, many companies today try to resolve this issue by creating programs that provide training and career development, and improved work-life balance to boost employee retention (Regan, 2020).

The fact that employee churn has and will continue be an issue that companies face, within this data science project I will be creating an model with the IBM dataset, provided by Kaggle (https://www.kaggle.com/pavansubhasht/ibm-hr-analytics-attrition-dataset). In this analysis, I hope to use this dataset to build a model to predict when employees are going to quit by understanding the main drivers of employee churn.

Importing the data

Let's import the dataset and make of a copy of the source file for this analysis.

```
setwd("C:/Users/frede/OneDrive - Regis University/MSDS_696/MSDS696_Data_Science_Practicum_II/")

# Read Excel file
df_source <- read.csv("Data/WA_Fn-UseC_-HR-Employee-Attrition.csv")
names(df_source)
```

```
##  [1] "ï..Age"                   "Attrition"
##  [3] "BusinessTravel"           "DailyRate"
##  [5] "Department"               "DistanceFromHome"
##  [7] "Education"                "EducationField"
##  [9] "EmployeeCount"            "EmployeeNumber"
## [11] "EnvironmentSatisfaction"  "Gender"
## [13] "HourlyRate"               "JobInvolvement"
## [15] "JobLevel"                 "JobRole"
## [17] "JobSatisfaction"          "MaritalStatus"
## [19] "MonthlyIncome"            "MonthlyRate"
## [21] "NumCompaniesWorked"       "Over18"
## [23] "OverTime"                 "PercentSalaryHike"
## [25] "PerformanceRating"        "RelationshipSatisfaction"
## [27] "StandardHours"            "StockOptionLevel"
## [29] "TotalWorkingYears"        "TrainingTimesLastYear"
## [31] "WorkLifeBalance"          "YearsAtCompany"
## [33] "YearsInCurrentRole"       "YearsSinceLastPromotion"
## [35] "YearsWithCurrManager"
```

```r
colnames(df_source)[1] <- "Age" # Renaming the column
```

```r
# Making copy of the dataset
library(data.table)
HR_data <- copy(df_source)
```

Exploratory Data Analysis

Let's look at the data and see how it is formatted before performing analysis

```r
str(HR_data)
```

```
## 'data.frame':    1470 obs. of  35 variables:
##  $ Age                     : int  41 49 37 33 27 32 59 30 38 36 ...
##  $ Attrition               : Factor w/ 2 levels "No","Yes": 2 1 2 1 1 1 1 1 1 1 ...
##  $ BusinessTravel          : Factor w/ 3 levels "Non-Travel","Travel_Frequently",..: 3 2 3 2 3 2 3 3 2 3
...
##  $ DailyRate               : int  1102 279 1373 1392 591 1005 1324 1358 216 1299 ...
##  $ Department              : Factor w/ 3 levels "Human Resources",..: 3 2 2 2 2 2 2 2 2 2 ...
##  $ DistanceFromHome        : int  1 8 2 3 2 2 3 24 23 27 ...
##  $ Education               : int  2 1 2 4 1 2 3 1 3 3 ...
##  $ EducationField          : Factor w/ 6 levels "Human Resources",..: 2 2 5 2 4 2 4 2 2 4 ...
##  $ EmployeeCount           : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ EmployeeNumber          : int  1 2 4 5 7 8 10 11 12 13 ...
##  $ EnvironmentSatisfaction : int  2 3 4 4 1 4 3 4 4 3 ...
##  $ Gender                  : Factor w/ 2 levels "Female","Male": 1 2 2 1 2 2 1 2 2 2 ...
##  $ HourlyRate              : int  94 61 92 56 40 79 81 67 44 94 ...
##  $ JobInvolvement          : int  3 2 2 3 3 3 4 3 2 3 ...
##  $ JobLevel                : int  2 2 1 1 1 1 1 1 3 2 ...
##  $ JobRole                 : Factor w/ 9 levels "Healthcare Representative",..: 8 7 3 7 3 3 3 3 5 1 ...
##  $ JobSatisfaction         : int  4 2 3 3 2 4 1 3 3 3 ...
##  $ MaritalStatus           : Factor w/ 3 levels "Divorced","Married",..: 3 2 3 2 2 3 2 1 3 2 ...
##  $ MonthlyIncome           : int  5993 5130 2090 2909 3468 3068 2670 2693 9526 5237 ...
##  $ MonthlyRate             : int  19479 24907 2396 23159 16632 11864 9964 13335 8787 16577 ...
##  $ NumCompaniesWorked      : int  8 1 6 1 9 0 4 1 0 6 ...
##  $ Over18                  : Factor w/ 1 level "Y": 1 1 1 1 1 1 1 1 1 1 ...
##  $ OverTime                : Factor w/ 2 levels "No","Yes": 2 1 2 2 1 1 2 1 1 1 ...
##  $ PercentSalaryHike       : int  11 23 15 11 12 13 20 22 21 13 ...
##  $ PerformanceRating       : int  3 4 3 3 3 3 4 4 4 3 ...
##  $ RelationshipSatisfaction: int  1 4 2 3 4 3 1 2 2 2 ...
##  $ StandardHours           : int  80 80 80 80 80 80 80 80 80 80 ...
##  $ StockOptionLevel        : int  0 1 0 0 1 0 3 1 0 2 ...
##  $ TotalWorkingYears       : int  8 10 7 8 6 8 12 1 10 17 ...
##  $ TrainingTimesLastYear   : int  0 3 3 3 3 2 3 2 2 3 ...
##  $ WorkLifeBalance         : int  1 3 3 3 3 2 2 3 3 2 ...
##  $ YearsAtCompany          : int  6 10 0 8 2 7 1 1 9 7 ...
##  $ YearsInCurrentRole      : int  4 7 0 7 2 7 0 0 7 7 ...
##  $ YearsSinceLastPromotion : int  0 1 0 3 2 3 0 0 1 7 ...
##  $ YearsWithCurrManager    : int  5 7 0 0 2 6 0 0 8 7 ...
```

From the data, we can see that there are 1,470 observations and 35 variables with various information about the employees.

Now let us have a glimpse of the data but instead of using the `glimpse()` or `summary()` functions, lets use the `skim()` function. The reason why is because can provide more detail about the data, such as the missing rate, complete rate, and a mini histogram of each variable (Quinn & Waring, 2019).

```r
#install.packages('skimr')
library(skimr)
skim(HR_data)
```

Data summary

Name	HR_data
Number of rows	1470
Number of columns	35
Column type frequency:	
factor	9
numeric	26
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
Attrition	0	1	FALSE	2	No: 1233, Yes: 237
BusinessTravel	0	1	FALSE	3	Tra: 1043, Tra: 277, Non: 150
Department	0	1	FALSE	3	Res: 961, Sal: 446, Hum: 63
EducationField	0	1	FALSE	6	Lif: 606, Med: 464, Mar: 159, Tec: 132
Gender	0	1	FALSE	2	Mal: 882, Fem: 588
JobRole	0	1	FALSE	9	Sal: 326, Res: 292, Lab: 259, Man: 145
MaritalStatus	0	1	FALSE	3	Mar: 673, Sin: 470, Div: 327
Over18	0	1	FALSE	1	Y: 1470
OverTime	0	1	FALSE	2	No: 1054, Yes: 416

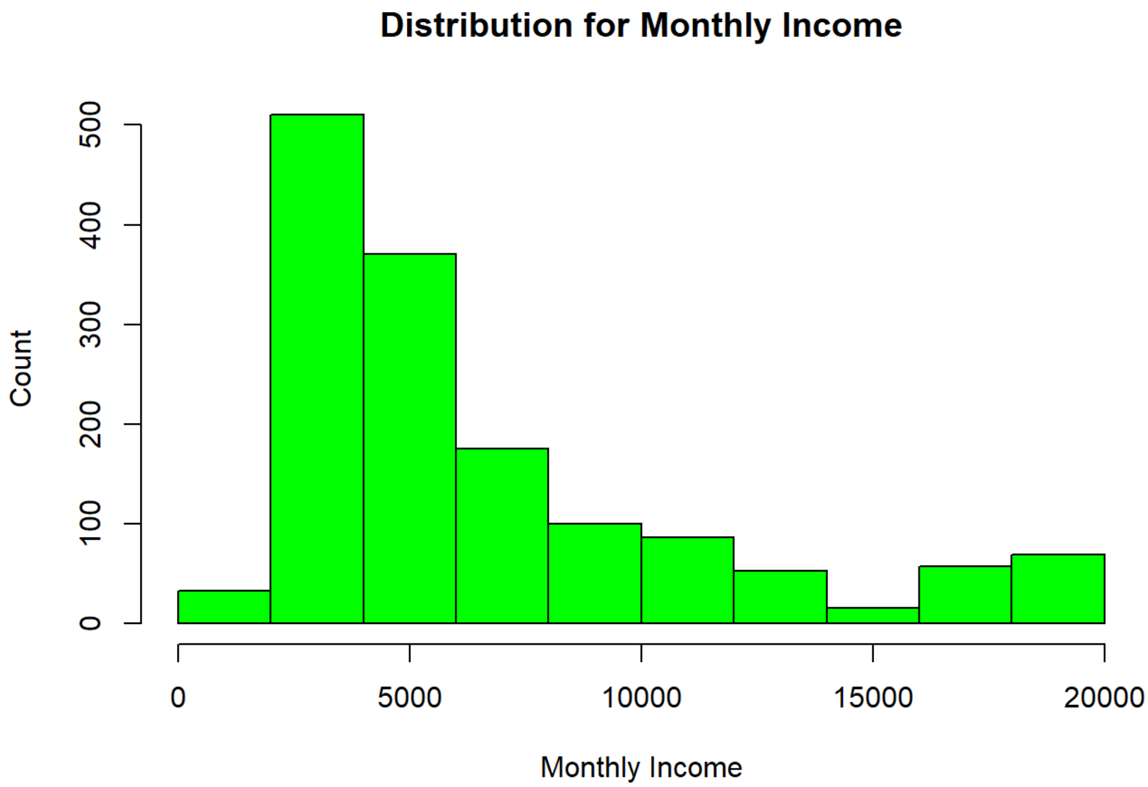
Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
Age	0	1	36.92	9.14	18	30.00	36.0	43.00	60	
DailyRate	0	1	802.49	403.51	102	465.00	802.0	1157.00	1499	
DistanceFromHome	0	1	9.19	8.11	1	2.00	7.0	14.00	29	
Education	0	1	2.91	1.02	1	2.00	3.0	4.00	5	
EmployeeCount	0	1	1.00	0.00	1	1.00	1.0	1.00	1	
EmployeeNumber	0	1	1024.87	602.02	1	491.25	1020.5	1555.75	2068	
EnvironmentSatisfaction	0	1	2.72	1.09	1	2.00	3.0	4.00	4	
HourlyRate	0	1	65.89	20.33	30	48.00	66.0	83.75	100	
JobInvolvement	0	1	2.73	0.71	1	2.00	3.0	3.00	4	
JobLevel	0	1	2.06	1.11	1	1.00	2.0	3.00	5	
JobSatisfaction	0	1	2.73	1.10	1	2.00	3.0	4.00	4	

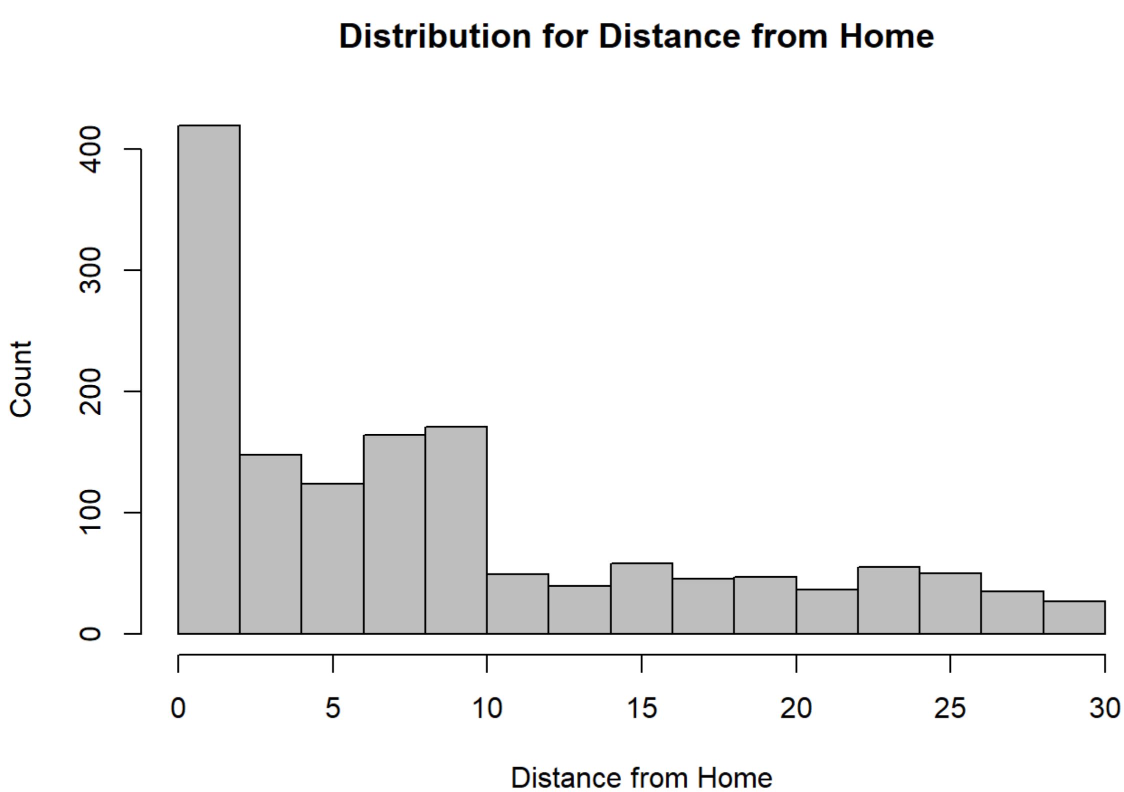
skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
MonthlyIncome	0	1	6502.93	4707.96	1009	2911.00	4919.0	8379.00	19999	
MonthlyRate	0	1	14313.10	7117.79	2094	8047.00	14235.5	20461.50	26999	
NumCompaniesWorked	0	1	2.69	2.50	0	1.00	2.0	4.00	9	
PercentSalaryHike	0	1	15.21	3.66	11	12.00	14.0	18.00	25	
PerformanceRating	0	1	3.15	0.36	3	3.00	3.0	3.00	4	
RelationshipSatisfaction	0	1	2.71	1.08	1	2.00	3.0	4.00	4	
StandardHours	0	1	80.00	0.00	80	80.00	80.0	80.00	80	
StockOptionLevel	0	1	0.79	0.85	0	0.00	1.0	1.00	3	
TotalWorkingYears	0	1	11.28	7.78	0	6.00	10.0	15.00	40	
TrainingTimesLastYear	0	1	2.80	1.29	0	2.00	3.0	3.00	6	
WorkLifeBalance	0	1	2.76	0.71	1	2.00	3.0	3.00	4	
YearsAtCompany	0	1	7.01	6.13	0	3.00	5.0	9.00	40	
YearsInCurrentRole	0	1	4.23	3.62	0	2.00	3.0	7.00	18	
YearsSinceLastPromotion	0	1	2.19	3.22	0	0.00	1.0	3.00	15	
YearsWithCurrManager	0	1	4.12	3.57	0	2.00	3.0	7.00	17	

From a glace of the mini histograms, it seems that several variables are tail-heavy. So let’s use the `hist()` function to have a better look at some of these variables.

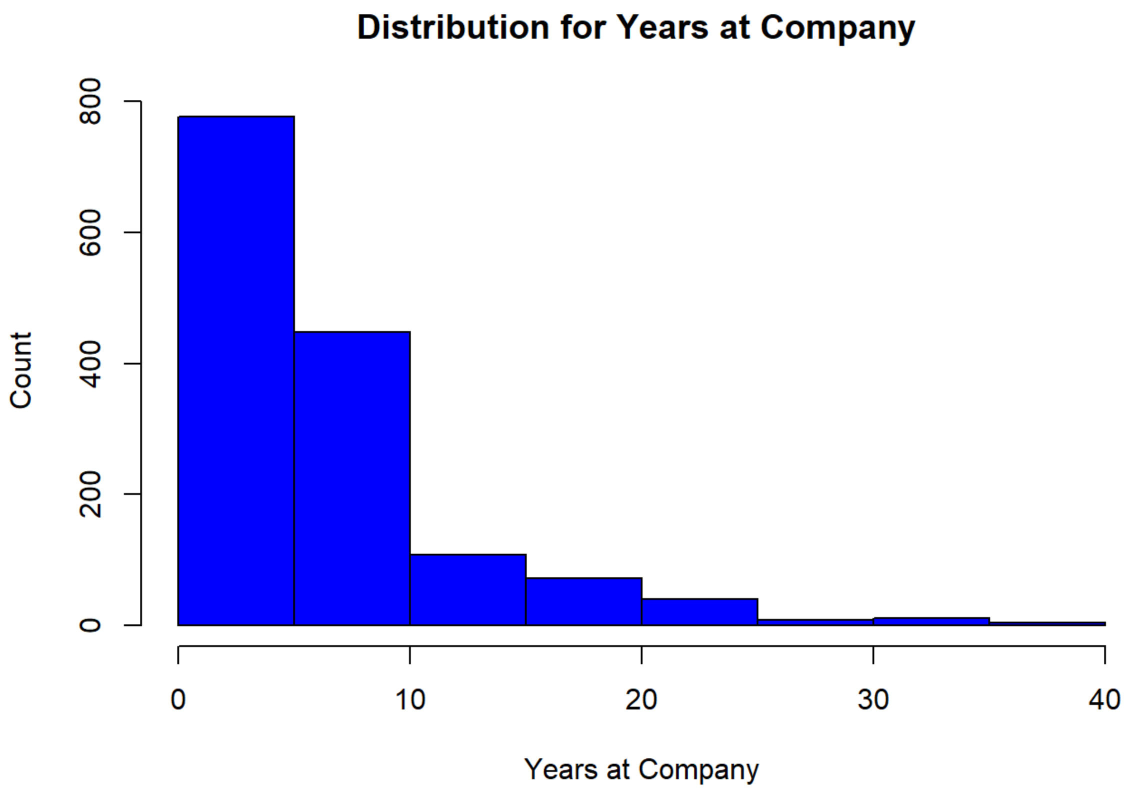
```
hist(HR_data$MonthlyIncome,main="Distribution for Monthly Income",xlab="Monthly Income",ylab="Count",col="g
reen")
```



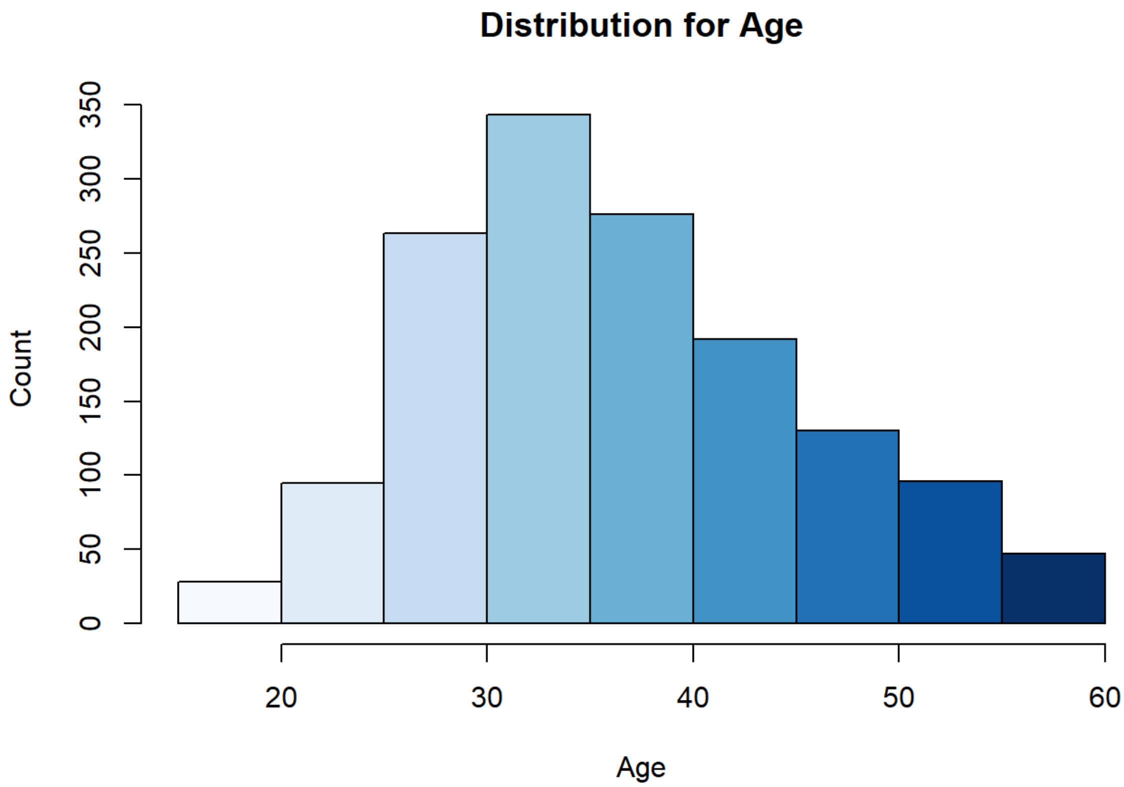

```
hist(HR_data$DistanceFromHome,main="Distribution for Distance from Home ",xlab="Distance from Home",ylab="Count",col="grey")
```



```
hist(HR_data$YearsAtCompany,main="Distribution for Years at Company ",xlab="Years at Company",ylab="Count",
col="blue")
```




```
hist(HR_data$Age,main="Distribution for Age",xlab="Age",ylab="Count",col=blues9)
```



After looking at

the MonthlyIncome, DistanceFromHome, and YearsAtCompany they do show a right-skewed in their distributions. The distribution for Age also has normal distribution that looks slightly right-skewed with majority being in the age range of 30 to 40.

```
prop.table(table(HR_data$Gender)) #Percentage of Gender
```

```
##
## Female   Male
##    0.4    0.6
```

The table above show that 60% of the dataset gender is male.

Within our exploratory analysis, the Attrition column will be used as our target variable. Before continuing out analysis of the data, we should find out the distribution and percentage of the Attrition variable.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:data.table':
##
##     between, first, last
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

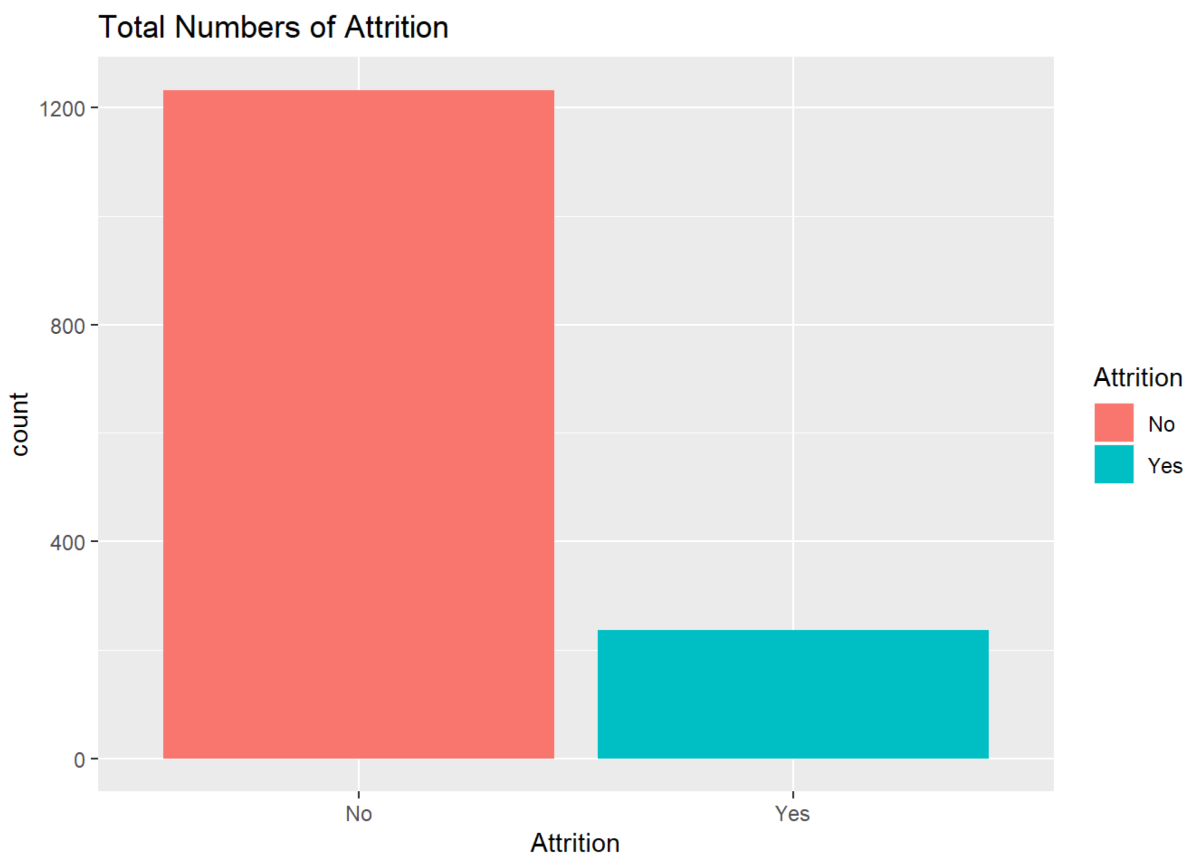
```
library(magrittr)

HR_data %>% group_by(Attrition) %>% summarise(Total = n()) %>% print()
```

```
## # A tibble: 2 x 2
##   Attrition Total
##   <fct>     <int>
## 1 No         1233
## 2 Yes         237
```

```
library(ggplot2)

ggplot(HR_data,aes(Attrition,fill=Attrition))+geom_bar() + ggtitle("Total Numbers of Attrition")
```




```
prop.table(table(HR_data$Attrition)) #Percentage of Attrition
```

```
##
##        No       Yes
## 0.8387755 0.1612245
```

From the table above, we see approximately 16% of IBM employees are leaving

Now that we have set the Attrition as our target variable, we can see how it affects the other variables in the dataset. In order to reduce time producing single graphs for these variables, we are going to use the `grid()` and `gridExtra()` functions to help arrange multiple grid-based plots on a page (Phiri, 2013).

```
library(grid)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
age_graph <- ggplot(HR_data,aes(Age,fill=Attrition))+geom_density()+facet_grid(~Attrition)
gender_graph <- ggplot(HR_data,aes(Gender,fill=Attrition))+geom_bar()
marital_graph <- ggplot(HR_data,aes(MaritalStatus,fill=Attrition))+geom_bar()
business_graph <- ggplot(HR_data,aes(BusinessTravel,fill=Attrition))+geom_bar()
grid.arrange(age_graph,gender_graph,marital_graph,business_graph,ncol=2, bottom = "Figure 1")
```

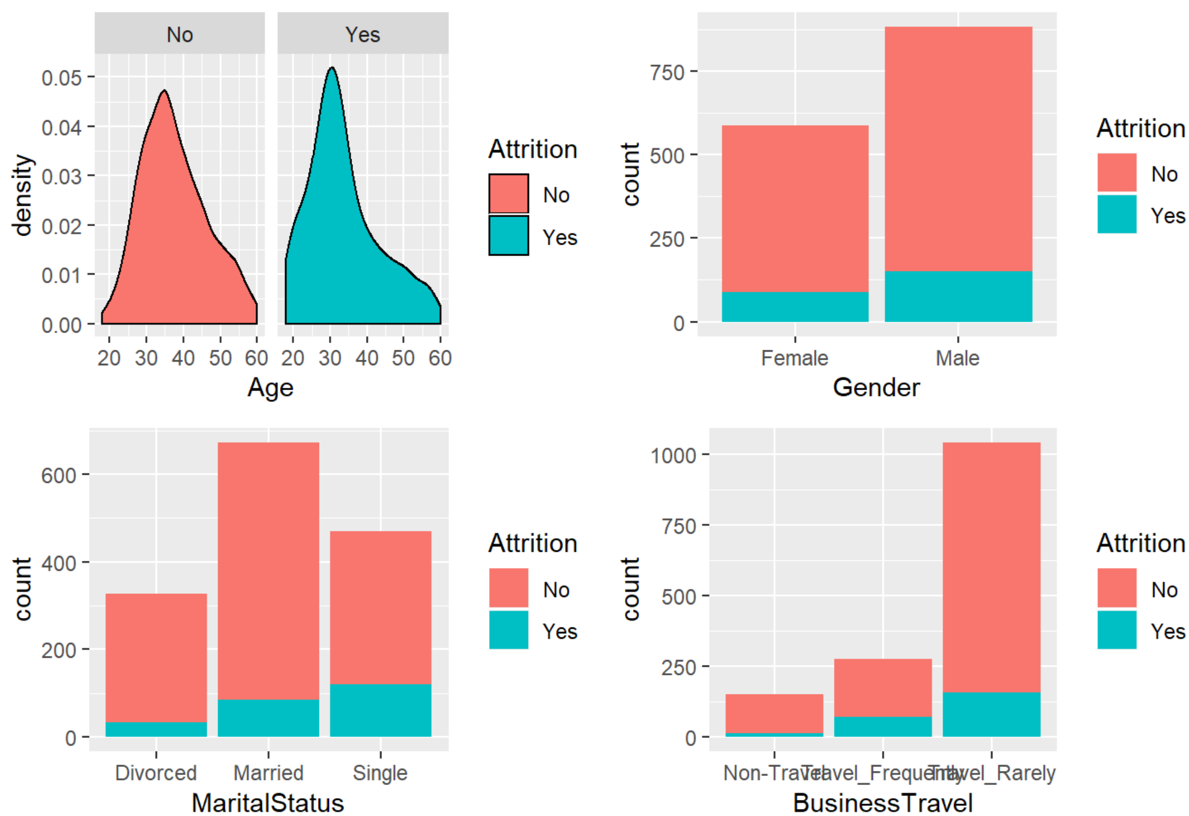


Figure 1

In Figure 1, we see the following:

1. Age: Most employees that leave IBM are around 30 years old.
2. Gender: We see that majority of separated employees are Male and that is due to our dataset being comprised of 60% Male.
3. Marital Status: Employees that are Single show the highest signs of Attrition, while Divorced employees are the lowest.
4. Business Travel: Among employee who leave IBM, most travel.

```
YAC_graph <- ggplot(HR_data,aes(YearsAtCompany,fill = Attrition))+geom_bar()
YSP_graph <- ggplot(HR_data,aes(YearsSinceLastPromotion,fill = Attrition))+geom_bar()
YCM_graph <- ggplot(HR_data,aes(YearsWithCurrManager,fill = Attrition))+geom_bar()
MTHincome_graph <- ggplot(HR_data,aes(MonthlyIncome,fill=Attrition))+geom_density()
OT_graph<- ggplot(HR_data,aes(OverTime,fill=Attrition))+geom_bar()
grid.arrange(YAC_graph,YSP_graph,YCM_graph,MTHincome_graph,OT_graph,ncol=2, bottom = "Figure 2")
```

Figure 2

In Figure 2, we see the following:

5. Years at Company: Employees who have been with IBM for <3 years make up a larger proportion of those quitting the company.
6. Years Since Last Promotion: Employees that have been recently promoted are making up a larger proportion of those who quit IBM.
7. Years With Current Manager: Newly hired managers are also a reason for employees to quit.
8. Monthly Income: We see higher levels of attrition among the lower segment of monthly income.
9. Over Time: Employees who work overtime also have a larger proportion that are quitting.

Preprocessing the Data

Before we start modelling the data, we should check if there are any missing values in the data which interfer which with the predictive model.

```
sum(is.na(HR_data))
```

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

We see that there are no missing values in the data after checking it, but we also would like to perform some data transformation. That is convert the type of some columns into a proper format.

```
HR_data$Education <- as.factor(HR_data$Education)
HR_data$EnvironmentSatisfaction <- as.factor(HR_data$EnvironmentSatisfaction)
HR_data$JobInvolvement <- as.factor(HR_data$JobInvolvement)
HR_data$JobLevel <- as.factor(HR_data$JobLevel)
HR_data$JobSatisfaction <- as.factor(HR_data$JobSatisfaction)
HR_data$StockOptionLevel <- as.factor(HR_data$StockOptionLevel)
HR_data$PerformanceRating <- as.factor(HR_data$PerformanceRating)
HR_data$RelationshipSatisfaction <- as.factor(HR_data$RelationshipSatisfaction)
HR_data$WorkLifeBalance <- as.factor(HR_data$WorkLifeBalance)
```

Due to some columns only having a single value in their columns, we will remove them.

```
HR_data <- HR_data %>% select(-EmployeeCount, -StandardHours, -Over18)
```

Feature Engineering

Feature engineering can be defined as the science of extracting more information from existing data. This newly extracted information can be used as input to our prediction model (Bock, 2017). Thereby, creating the outcome to have more impact than the model. Now based on my assumptions, we can create two features with existing variables.

1. Tenure per job: People who worked at several companies but only for a short period time usually leave the company early maybe for a change of pace or building up enough experience through these companies to help them land a job at a major company.
2. Years without Change: People who went through role or job level changes probably enjoy the thought of taking on more responsible task as they gain seniority within a company. This variable will see the years a employee went without kind of change using the Role, Job Change and Promotion, as the metrics to determine change.

```
HR_data_feng <- HR_data

HR_data_feng$TenurePerJob <- ifelse(HR_data_feng$NumCompaniesWorked!=0, HR_data_feng$TotalWorkingYears/HR_d
ata_feng$NumCompaniesWorked,0)
HR_data_feng$YearWithoutChange <- HR_data_feng$YearsInCurrentRole - HR_data_feng$YearsSinceLastPromotion
HR_data_feng$YearsWithoutChange2 <- HR_data_feng$TotalWorkingYears - HR_data_feng$YearsSinceLastPromotion

TPJ_grapn <- ggplot(HR_data_feng,aes(TenurePerJob))+geom_density()+facet_grid(~Attrition)
YWC_graph <- ggplot(HR_data_feng,aes(YearWithoutChange))+geom_density()+facet_grid(~Attrition)
YWC2_graph <- ggplot(HR_data_feng,aes(YearsWithoutChange2))+geom_density()+facet_grid(~Attrition)
grid.arrange(TPJ_grapn,YWC_graph,YWC2_graph,ncol=2,bottom = "Figure 3")
```

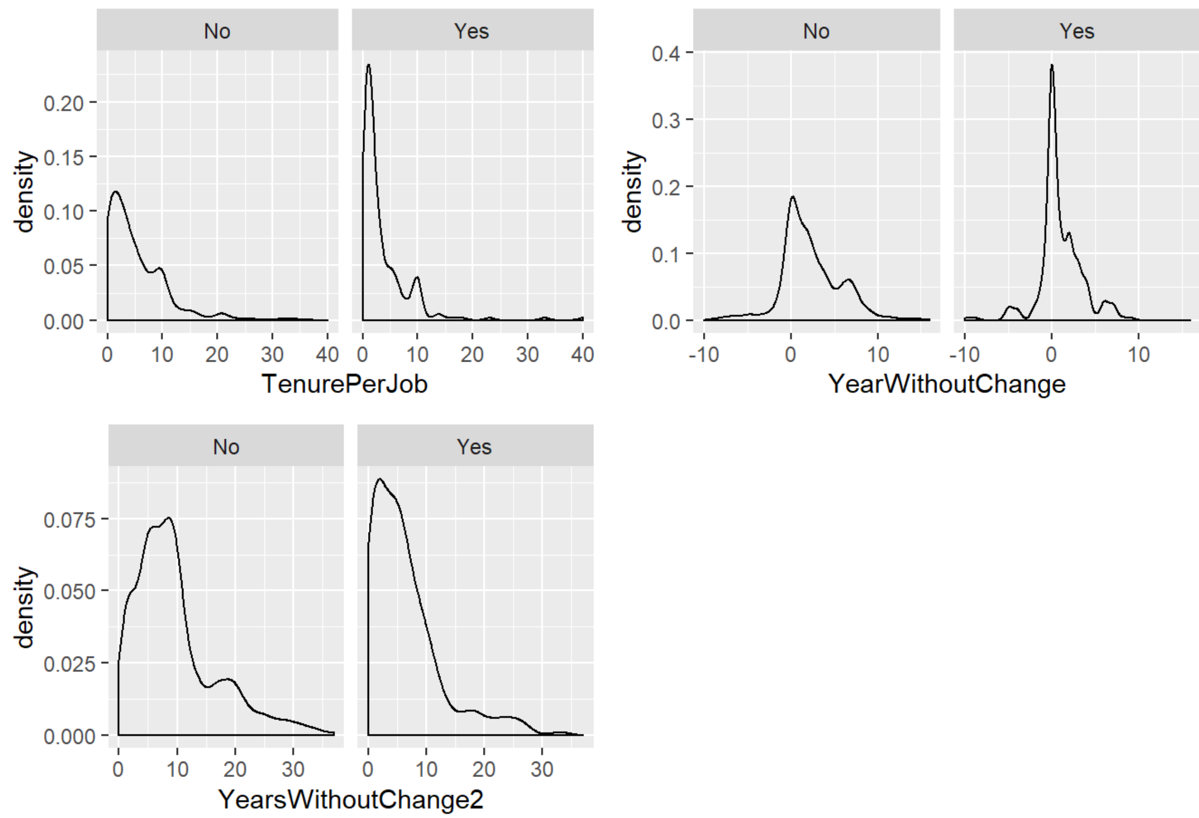


Figure 3

In figure 3, we see that the Attrition variable does have an affect on these new features.

Logistic Regression

Now we split the data into a 20% testing set and 80% training set with the `sample()` function which takes a vector as input; then you tell it how many samples to draw from that list (‘R Function’, 2019). We also use the `set.seed()` function which produces the same sample again and again. The purpose of creating two sets of data is so the training set is the one on which we train and fit our model basically to fit the parameters whereas test data is used only to assess performance of model (Shah, 2017).

```
library(caret)
```

```
## Loading required package: lattice
```

```
# Spliting the data
set.seed(18)
attr_training <- sample(nrow(HR_data), nrow(HR_data)*.8)
train_attr <- HR_data %>% slice(attr_training)
test_attr <- HR_data %>% slice(-attr_training)
```

```
# Check the portion and percentage of Attrition in train data
table(train_attr$Attrition)
```

```
##
##  No Yes
## 996 180
```

```
prop.table(table(train_attr$Attrition))
```

```
##
##        No       Yes
## 0.8469388 0.1530612
```

From the infomation displayed, we see that the data is imbalanced with Yes cases at 15%.However, we could to try to fix this imbalance sample by using up-sampling or down-sampling techniques. Keep in mind, there are pros and cons when using those techniques (Altini, 2015). With that in mind, we will try to make it balanced by using an upsampling technique with `ovun.sample()` function from ROSE package (Analytics, 2019).

```
#install.packages("ROSE")
library(ROSE)
```

```
## Loaded ROSE 0.0-3
```

```
library(brglm2)
balanced_attr <- ovun.sample(Attrition ~ ., data = train_attr, method = "over",
                              N = 996*2, seed = 1)$data
table(balanced_attr$Attrition)
```

```
##
##  No Yes
## 996 996
```

After balancing the data, we will perform our first Logistic Regression model by using all the predictors in the formula.

```
log_regress <- glm(Attrition ~ ., family = "binomial", data = balanced_attr)
summary(log_regress)
```

```
##
## Call:
## glm(formula = Attrition ~ ., family = "binomial", data = balanced_attr)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8672  -0.5173   0.0246   0.5847   3.4560
##
## Coefficients:
##                                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)                      -9.801e+00  3.256e+02  -0.030 0.975983
## Age                              -3.858e-02  1.031e-02  -3.741 0.000183
## BusinessTravelTravel_Frequently   2.260e+00  3.203e-01   7.054 1.74e-12
## BusinessTravelTravel_Rarely       1.224e+00  2.896e-01   4.226 2.38e-05
## DailyRate                        -4.255e-04  1.751e-04  -2.429 0.015120
## DepartmentResearch & Development  1.521e+01  3.256e+02   0.047 0.962735
## DepartmentSales                   1.505e+01  3.256e+02   0.046 0.963137
## DistanceFromHome                  4.566e-02  8.898e-03   5.132 2.87e-07
## Education2                       -1.167e-02  2.495e-01  -0.047 0.962693
## Education3                       -9.487e-02  2.207e-01  -0.430 0.667304
## Education4                        2.541e-01  2.422e-01   1.049 0.294043
## Education5                        2.890e-01  4.320e-01   0.669 0.503559
## EducationFieldLife Sciences      -1.471e+00  6.599e-01  -2.229 0.025842
## EducationFieldMarketing          -6.816e-01  6.877e-01  -0.991 0.321555
## EducationFieldMedical            -1.346e+00  6.509e-01  -2.068 0.038676
## EducationFieldOther              -8.496e-01  6.986e-01  -1.216 0.223902
## EducationFieldTechnical Degree    1.691e-01  6.741e-01   0.251 0.801992
## EmployeeNumber                   -2.245e-04  1.224e-04  -1.834 0.066652
## EnvironmentSatisfaction2         -1.244e+00  2.258e-01  -5.509 3.60e-08
## EnvironmentSatisfaction3         -1.090e+00  2.059e-01  -5.296 1.18e-07
## EnvironmentSatisfaction4         -1.111e+00  2.029e-01  -5.473 4.41e-08
## GenderMale                        4.716e-01  1.448e-01   3.258 0.001124
## HourlyRate                       -1.655e-03  3.419e-03  -0.484 0.628366
## JobInvolvement2                  -1.240e+00  3.224e-01  -3.846 0.000120
## JobInvolvement3                  -1.580e+00  3.078e-01  -5.133 2.85e-07
## JobInvolvement4                  -1.678e+00  3.808e-01  -4.408 1.05e-05
## JobLevel2                        -1.025e+00  2.986e-01  -3.433 0.000596
## JobLevel3                         1.023e+00  5.256e-01   1.946 0.051666
## JobLevel4                        -2.091e+00  9.785e-01  -2.137 0.032572
## JobLevel5                         3.386e+00  1.189e+00   2.848 0.004400
## JobRoleHuman Resources            1.642e+01  3.256e+02   0.050 0.959764
## JobRoleLaboratory Technician      2.041e+00  4.764e-01   4.284 1.83e-05
## JobRoleManager                   -3.228e-01  8.494e-01  -0.380 0.703927
## JobRoleManufacturing Director     1.068e+00  4.715e-01   2.264 0.023555
## JobRoleResearch Director         -2.529e+00  9.117e-01  -2.774 0.005531
## JobRoleResearch Scientist         1.107e+00  4.789e-01   2.312 0.020776
## JobRoleSales Executive            2.459e+00  1.007e+00   2.442 0.014620
## JobRoleSales Representative       3.035e+00  1.061e+00   2.862 0.004213
## JobSatisfaction2                 -1.168e+00  2.168e-01  -5.387 7.17e-08
## JobSatisfaction3                 -8.921e-01  1.891e-01  -4.718 2.38e-06
## JobSatisfaction4                 -1.750e+00  2.043e-01  -8.564  < 2e-16
## MaritalStatusMarried              5.933e-01  2.022e-01   2.935 0.003338
## MaritalStatusSingle               6.732e-01  2.930e-01   2.297 0.021591
## MonthlyIncome                    -1.466e-04  7.005e-05  -2.093 0.036321
## MonthlyRate                       9.625e-06  9.522e-06   1.011 0.312118
## NumCompaniesWorked                2.380e-01  3.096e-02   7.687 1.50e-14
## OverTimeYes                       2.020e+00  1.534e-01  13.166  < 2e-16
## PercentSalaryHike                -6.368e-02  3.011e-02  -2.115 0.034450
## PerformanceRating4                2.976e-01  3.191e-01   0.933 0.351078
```

```
## RelationshipSatisfaction2        -1.149e+00  2.183e-01  -5.264 1.41e-07
## RelationshipSatisfaction3        -1.045e+00  1.955e-01  -5.347 8.93e-08
## RelationshipSatisfaction4        -1.094e+00  1.970e-01  -5.552 2.82e-08
## StockOptionLevel1                -1.274e+00  2.405e-01  -5.299 1.16e-07
## StockOptionLevel2                -1.131e+00  3.153e-01  -3.586 0.000336
## StockOptionLevel3                -3.910e-01  3.662e-01  -1.068 0.285618
## TotalWorkingYears                -3.083e-02  2.078e-02  -1.483 0.138032
## TrainingTimesLastYear            -1.209e-01  5.217e-02  -2.318 0.020468
## WorkLifeBalance2                 -9.685e-01  2.946e-01  -3.288 0.001008
## WorkLifeBalance3                 -1.549e+00  2.815e-01  -5.503 3.73e-08
## WorkLifeBalance4                 -6.003e-01  3.325e-01  -1.805 0.071037
## YearsAtCompany                    2.011e-01  3.016e-02   6.670 2.56e-11
## YearsInCurrentRole               -2.519e-01  4.021e-02  -6.266 3.70e-10
## YearsSinceLastPromotion           1.024e-01  3.173e-02   3.225 0.001258
## YearsWithCurrManager             -1.281e-01  3.721e-02  -3.444 0.000573
##
## (Intercept)
## Age                              ***
## BusinessTravelTravel_Frequently  ***
## BusinessTravelTravel_Rarely      ***
## DailyRate                        *
## DepartmentResearch & Development
## DepartmentSales
## DistanceFromHome                 ***
## Education2
## Education3
## Education4
## Education5
## EducationFieldLife Sciences      *
## EducationFieldMarketing
## EducationFieldMedical            *
## EducationFieldOther
## EducationFieldTechnical Degree
## EmployeeNumber                   .
## EnvironmentSatisfaction2         ***
## EnvironmentSatisfaction3         ***
## EnvironmentSatisfaction4         ***
## GenderMale                       **
## HourlyRate
## JobInvolvement2                  ***
## JobInvolvement3                  ***
## JobInvolvement4                  ***
## JobLevel2                        ***
## JobLevel3                        .
## JobLevel4                        *
## JobLevel5                        **
## JobRoleHuman Resources
## JobRoleLaboratory Technician     ***
## JobRoleManager
## JobRoleManufacturing Director    *
## JobRoleResearch Director         **
## JobRoleResearch Scientist        *
## JobRoleSales Executive           *
## JobRoleSales Representative      **
## JobSatisfaction2                 ***
## JobSatisfaction3                 ***
## JobSatisfaction4                 ***
## MaritalStatusMarried             **
## MaritalStatusSingle              *
## MonthlyIncome                    *
```

```
## MonthlyRate
## NumCompaniesWorked              ***
## OverTimeYes                     ***
## PercentSalaryHike               *
## PerformanceRating4
## RelationshipSatisfaction2       ***
## RelationshipSatisfaction3       ***
## RelationshipSatisfaction4       ***
## StockOptionLevel1               ***
## StockOptionLevel2               ***
## StockOptionLevel3
## TotalWorkingYears
## TrainingTimesLastYear           *
## WorkLifeBalance2                **
## WorkLifeBalance3                ***
## WorkLifeBalance4                .
## YearsAtCompany                  ***
## YearsInCurrentRole              ***
## YearsSinceLastPromotion         **
## YearsWithCurrManager            ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2761.5  on 1991  degrees of freedom
## Residual deviance: 1511.3  on 1928  degrees of freedom
## AIC: 1639.3
##
## Number of Fisher Scoring iterations: 14
```

```
glm(formula = Attrition ~ ., family = "binomial", data = train_attr,
    method = "detect_separation", linear_program = "dual")
```

```
## Separation: FALSE
## Existence of maximum likelihood estimates
##                     (Intercept)                              Age
##                            -Inf                                0
##  BusinessTravelTravel_Frequently      BusinessTravelTravel_Rarely
##                               0                                0
##                       DailyRate DepartmentResearch & Development
##                               0                              Inf
##                 DepartmentSales                 DistanceFromHome
##                             Inf                                0
##                      Education2                       Education3
##                               0                                0
##                      Education4                       Education5
##                               0                                0
##     EducationFieldLife Sciences          EducationFieldMarketing
##                               0                                0
##           EducationFieldMedical              EducationFieldOther
##                               0                                0
##   EducationFieldTechnical Degree                   EmployeeNumber
##                               0                                0
##        EnvironmentSatisfaction2         EnvironmentSatisfaction3
##                               0                                0
##        EnvironmentSatisfaction4                       GenderMale
##                               0                                0
##                      HourlyRate                  JobInvolvement2
##                               0                                0
##                 JobInvolvement3                  JobInvolvement4
##                               0                                0
##                       JobLevel2                        JobLevel3
##                               0                                0
##                       JobLevel4                        JobLevel5
##                               0                                0
##           JobRoleHuman Resources     JobRoleLaboratory Technician
##                             Inf                                0
##                  JobRoleManager    JobRoleManufacturing Director
##                               0                                0
##        JobRoleResearch Director         JobRoleResearch Scientist
##                               0                                0
##          JobRoleSales Executive     JobRoleSales Representative
##                               0                                0
##                JobSatisfaction2                 JobSatisfaction3
##                               0                                0
##                JobSatisfaction4             MaritalStatusMarried
##                               0                                0
##             MaritalStatusSingle                    MonthlyIncome
##                               0                                0
##                     MonthlyRate               NumCompaniesWorked
##                               0                                0
##                     OverTimeYes                PercentSalaryHike
##                               0                                0
##              PerformanceRating4        RelationshipSatisfaction2
##                               0                                0
##       RelationshipSatisfaction3        RelationshipSatisfaction4
##                               0                                0
##               StockOptionLevel1                StockOptionLevel2
##                               0                                0
##               StockOptionLevel3                TotalWorkingYears
##                               0                                0
##           TrainingTimesLastYear                  WorkLifeBalance2
```

```
##                          0                          0
##               WorkLifeBalance3               WorkLifeBalance4
##                          0                          0
##                 YearsAtCompany             YearsInCurrentRole
##                          0                          0
##        YearsSinceLastPromotion           YearsWithCurrManager
##                          0                          0
## 0: finite value, Inf: infinity, -Inf: -infinity
```

The separation in the our model returned False so there extist no perfect separation.

Next, we will be doing a feature engineering by applying the `step()` function which is used for stepwise regression in order to fit regression models in which the choice of predictive variables are carried out by an automatic procedure (Kassambara, 2018). We will set the direction to backward so that it will iterate over the model, and remove the least contributive predictors.

```
bw_reg_model<- step(log_regress, direction = "backward", trace = FALSE)
summary(bw_reg_model)
```

```
##
## Call:
## glm(formula = Attrition ~ Age + BusinessTravel + DailyRate +
##     Department + DistanceFromHome + EducationField + EmployeeNumber +
##     EnvironmentSatisfaction + Gender + JobInvolvement + JobLevel +
##     JobRole + JobSatisfaction + MaritalStatus + MonthlyIncome +
##     NumCompaniesWorked + OverTime + PercentSalaryHike + RelationshipSatisfaction +
##     StockOptionLevel + TotalWorkingYears + TrainingTimesLastYear +
##     WorkLifeBalance + YearsAtCompany + YearsInCurrentRole + YearsSinceLastPromotion +
##     YearsWithCurrManager, family = "binomial", data = balanced_attr)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8611  -0.5238   0.0286   0.5907   3.4512
##
## Coefficients:
##                                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)                     -1.020e+01  3.216e+02  -0.032 0.974706
## Age                             -3.617e-02  1.003e-02  -3.607 0.000310
## BusinessTravelTravel_Frequently  2.247e+00  3.185e-01   7.054 1.73e-12
## BusinessTravelTravel_Rarely      1.235e+00  2.873e-01   4.298 1.72e-05
## DailyRate                       -4.102e-04  1.721e-04  -2.384 0.017135
## DepartmentResearch & Development  1.535e+01  3.216e+02   0.048 0.961932
## DepartmentSales                  1.518e+01  3.216e+02   0.047 0.962360
## DistanceFromHome                 4.663e-02  8.788e-03   5.307 1.12e-07
## EducationFieldLife Sciences     -1.530e+00  6.345e-01  -2.411 0.015903
## EducationFieldMarketing         -7.559e-01  6.685e-01  -1.131 0.258195
## EducationFieldMedical           -1.436e+00  6.248e-01  -2.299 0.021518
## EducationFieldOther             -8.717e-01  6.693e-01  -1.302 0.192746
## EducationFieldTechnical Degree   4.438e-02  6.511e-01   0.068 0.945657
## EmployeeNumber                  -2.060e-04  1.209e-04  -1.703 0.088483
## EnvironmentSatisfaction2        -1.228e+00  2.241e-01  -5.481 4.22e-08
## EnvironmentSatisfaction3        -1.073e+00  2.041e-01  -5.257 1.47e-07
## EnvironmentSatisfaction4        -1.081e+00  2.009e-01  -5.381 7.42e-08
## GenderMale                       4.882e-01  1.436e-01   3.400 0.000673
## JobInvolvement2                 -1.221e+00  3.185e-01  -3.832 0.000127
## JobInvolvement3                 -1.556e+00  3.013e-01  -5.163 2.44e-07
## JobInvolvement4                 -1.678e+00  3.747e-01  -4.478 7.54e-06
## JobLevel2                       -1.005e+00  2.963e-01  -3.390 0.000698
## JobLevel3                        1.020e+00  5.201e-01   1.961 0.049846
## JobLevel4                       -2.127e+00  9.762e-01  -2.179 0.029357
## JobLevel5                        3.288e+00  1.176e+00   2.796 0.005179
## JobRoleHuman Resources           1.645e+01  3.216e+02   0.051 0.959207
## JobRoleLaboratory Technician     1.983e+00  4.728e-01   4.195 2.73e-05
## JobRoleManager                  -1.912e-01  8.395e-01  -0.228 0.819847
## JobRoleManufacturing Director    1.051e+00  4.682e-01   2.245 0.024773
## JobRoleResearch Director        -2.265e+00  8.983e-01  -2.522 0.011679
## JobRoleResearch Scientist        1.058e+00  4.758e-01   2.224 0.026168
## JobRoleSales Executive           2.461e+00  1.011e+00   2.434 0.014931
## JobRoleSales Representative      2.974e+00  1.065e+00   2.794 0.005207
## JobSatisfaction2                -1.168e+00  2.138e-01  -5.463 4.68e-08
## JobSatisfaction3                -8.703e-01  1.861e-01  -4.677 2.92e-06
## JobSatisfaction4                -1.731e+00  2.005e-01  -8.635  < 2e-16
## MaritalStatusMarried             5.959e-01  1.996e-01   2.986 0.002826
## MaritalStatusSingle              6.371e-01  2.893e-01   2.203 0.027626
## MonthlyIncome                   -1.574e-04  6.925e-05  -2.272 0.023069
## NumCompaniesWorked               2.340e-01  3.066e-02   7.632 2.31e-14
## OverTimeYes                      2.012e+00  1.516e-01  13.270  < 2e-16
## PercentSalaryHike               -4.092e-02  1.929e-02  -2.121 0.033902
```

```
## RelationshipSatisfaction2        -1.112e+00  2.130e-01  -5.222 1.77e-07
## RelationshipSatisfaction3        -1.010e+00  1.918e-01  -5.267 1.39e-07
## RelationshipSatisfaction4        -1.046e+00  1.948e-01  -5.371 7.83e-08
## StockOptionLevel1                -1.356e+00  2.350e-01  -5.771 7.87e-09
## StockOptionLevel2                -1.197e+00  3.086e-01  -3.879 0.000105
## StockOptionLevel3                -4.014e-01  3.607e-01  -1.113 0.265790
## TotalWorkingYears                -2.977e-02  2.054e-02  -1.449 0.147289
## TrainingTimesLastYear            -1.282e-01  5.158e-02  -2.486 0.012915
## WorkLifeBalance2                 -9.798e-01  2.906e-01  -3.372 0.000745
## WorkLifeBalance3                 -1.531e+00  2.757e-01  -5.553 2.81e-08
## WorkLifeBalance4                 -6.113e-01  3.278e-01  -1.865 0.062216
## YearsAtCompany                    2.056e-01  3.004e-02   6.845 7.65e-12
## YearsInCurrentRole               -2.527e-01  4.009e-02  -6.303 2.92e-10
## YearsSinceLastPromotion           1.046e-01  3.127e-02   3.345 0.000824
## YearsWithCurrManager             -1.277e-01  3.683e-02  -3.466 0.000528
##
## (Intercept)
## Age                              ***
## BusinessTravelTravel_Frequently  ***
## BusinessTravelTravel_Rarely      ***
## DailyRate                        *
## DepartmentResearch & Development
## DepartmentSales
## DistanceFromHome                 ***
## EducationFieldLife Sciences      *
## EducationFieldMarketing
## EducationFieldMedical            *
## EducationFieldOther
## EducationFieldTechnical Degree
## EmployeeNumber                   .
## EnvironmentSatisfaction2         ***
## EnvironmentSatisfaction3         ***
## EnvironmentSatisfaction4         ***
## GenderMale                       ***
## JobInvolvement2                  ***
## JobInvolvement3                  ***
## JobInvolvement4                  ***
## JobLevel2                        ***
## JobLevel3                        *
## JobLevel4                        *
## JobLevel5                        **
## JobRoleHuman Resources
## JobRoleLaboratory Technician     ***
## JobRoleManager
## JobRoleManufacturing Director    *
## JobRoleResearch Director         *
## JobRoleResearch Scientist        *
## JobRoleSales Executive           *
## JobRoleSales Representative      **
## JobSatisfaction2                 ***
## JobSatisfaction3                 ***
## JobSatisfaction4                 ***
## MaritalStatusMarried             **
## MaritalStatusSingle              *
## MonthlyIncome                    *
## NumCompaniesWorked               ***
## OverTimeYes                      ***
## PercentSalaryHike                *
## RelationshipSatisfaction2        ***
## RelationshipSatisfaction3        ***
```

```
## RelationshipSatisfaction4        ***
## StockOptionLevel1                ***
## StockOptionLevel2                ***
## StockOptionLevel3
## TotalWorkingYears
## TrainingTimesLastYear            *
## WorkLifeBalance2                 ***
## WorkLifeBalance3                 ***
## WorkLifeBalance4                 .
## YearsAtCompany                   ***
## YearsInCurrentRole               ***
## YearsSinceLastPromotion          ***
## YearsWithCurrManager             ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2761.5  on 1991  degrees of freedom
## Residual deviance: 1517.1  on 1935  degrees of freedom
## AIC: 1631.1
##
## Number of Fisher Scoring iterations: 14
```

Now, we will check if multicollinearity exists in our model using the `vif()` function, which ‘calculates variance-inflation and generalized variance-inflation factors for linear, generalized linear, and other models’ (Fox & Weisberg, 2019). In an article by Kassambara (2018), he explains that a high value of VIF indicates the existence of multicollinearity and suggests dropping the highest value of VIF.

```
car::vif(bw_reg_model)
```

```
##                                   GVIF Df GVIF^(1/(2*Df))
## Age                       1.987430e+00  1        1.409763
## BusinessTravel            1.338507e+00  2        1.075611
## DailyRate                 1.175317e+00  1        1.084120
## Department                5.743169e+07  2       87.053832
## DistanceFromHome          1.228154e+00  1        1.108221
## EducationField            4.666025e+00  5        1.166527
## EmployeeNumber            1.185489e+00  1        1.088801
## EnvironmentSatisfaction   1.636246e+00  3        1.085529
## Gender                    1.147067e+00  1        1.071012
## JobInvolvement            1.598506e+00  3        1.081315
## JobLevel                  7.672646e+01  4        1.720355
## JobRole                   1.243779e+09  8        3.701873
## JobSatisfaction           1.595519e+00  3        1.080978
## MaritalStatus             3.569071e+00  2        1.374481
## MonthlyIncome             1.533085e+01  1        3.915463
## NumCompaniesWorked        1.523547e+00  1        1.234321
## OverTime                  1.308205e+00  1        1.143768
## PercentSalaryHike         1.128670e+00  1        1.062389
## RelationshipSatisfaction  1.508066e+00  3        1.070870
## StockOptionLevel          4.275539e+00  3        1.273987
## TotalWorkingYears         5.195686e+00  1        2.279405
## TrainingTimesLastYear     1.135342e+00  1        1.065524
## WorkLifeBalance           1.602410e+00  3        1.081755
## YearsAtCompany            7.559767e+00  1        2.749503
## YearsInCurrentRole        4.249090e+00  1        2.061332
## YearsSinceLastPromotion   2.307515e+00  1        1.519051
## YearsWithCurrManager      3.736256e+00  1        1.932940
```

```
#Recreating a formula with the highest VIF is dropped.
bw_reg_model <- glm(formula = Attrition ~ Age + BusinessTravel + DailyRate +
                      DistanceFromHome + EducationField + EmployeeNumber +
                      EnvironmentSatisfaction + Gender + JobInvolvement + JobLevel +
 JobRole + JobSatisfaction + MaritalStatus + MonthlyIncome +
                      NumCompaniesWorked + OverTime + PercentSalaryHike +
                      RelationshipSatisfaction +
                      StockOptionLevel + TotalWorkingYears + TrainingTimesLastYear +
 WorkLifeBalance + YearsAtCompany + YearsInCurrentRole +
                      YearsSinceLastPromotion +
                      YearsWithCurrManager, family = "binomial", data=
                      balanced_attr)
```

```
#Recalculate the VIF value.
car::vif(bw_reg_model)
```

```
##                              GVIF Df GVIF^(1/(2*Df))
## Age                      1.983295  1        1.408295
## BusinessTravel           1.329186  2        1.073733
## DailyRate                1.175157  1        1.084047
## DistanceFromHome         1.216231  1        1.102829
## EducationField           4.315324  5        1.157448
## EmployeeNumber           1.182660  1        1.087502
## EnvironmentSatisfaction  1.614912  3        1.083157
## Gender                   1.138396  1        1.066956
## JobInvolvement           1.549342  3        1.075700
## JobLevel                73.886948  4        1.712265
## JobRole                 77.216664  8        1.312145
## JobSatisfaction          1.574728  3        1.078618
## MaritalStatus            3.560918  2        1.373696
## MonthlyIncome           15.480245  1        3.934494
## NumCompaniesWorked       1.516519  1        1.231470
## OverTime                 1.298094  1        1.139339
## PercentSalaryHike        1.124487  1        1.060418
## RelationshipSatisfaction 1.497423  3        1.069607
## StockOptionLevel         4.255169  3        1.272974
## TotalWorkingYears        5.173397  1        2.274510
## TrainingTimesLastYear    1.134146  1        1.064963
## WorkLifeBalance          1.598007  3        1.081259
## YearsAtCompany           7.271288  1        2.696533
## YearsInCurrentRole       4.092361  1        2.022959
## YearsSinceLastPromotion  2.249532  1        1.499844
## YearsWithCurrManager     3.700395  1        1.923641
```

Model Performance (Logistic Regression)

After the highest value of VIF was dropped, our model is ready for predicting the `test_attr` dataset.

```
pd_model <- predict(bw_reg_model, test_attr, type = "response")

pdm <- as.factor(ifelse(pd_model >= 0.5, "Yes", "No"))
confusionMatrix(pdm, test_attr$Attrition, positive = "Yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  186  13
##        Yes  51  44
##
##                Accuracy : 0.7823
##                  95% CI : (0.7307, 0.8281)
##     No Information Rate : 0.8061
##     P-Value [Acc > NIR] : 0.8651
##
##                   Kappa : 0.4443
##
##  Mcnemar's Test P-Value : 3.746e-06
##
##             Sensitivity : 0.7719
##             Specificity : 0.7848
##          Pos Pred Value : 0.4632
##          Neg Pred Value : 0.9347
##              Prevalence : 0.1939
##          Detection Rate : 0.1497
##    Detection Prevalence : 0.3231
##       Balanced Accuracy : 0.7784
##
##        'Positive' Class : Yes
##
```

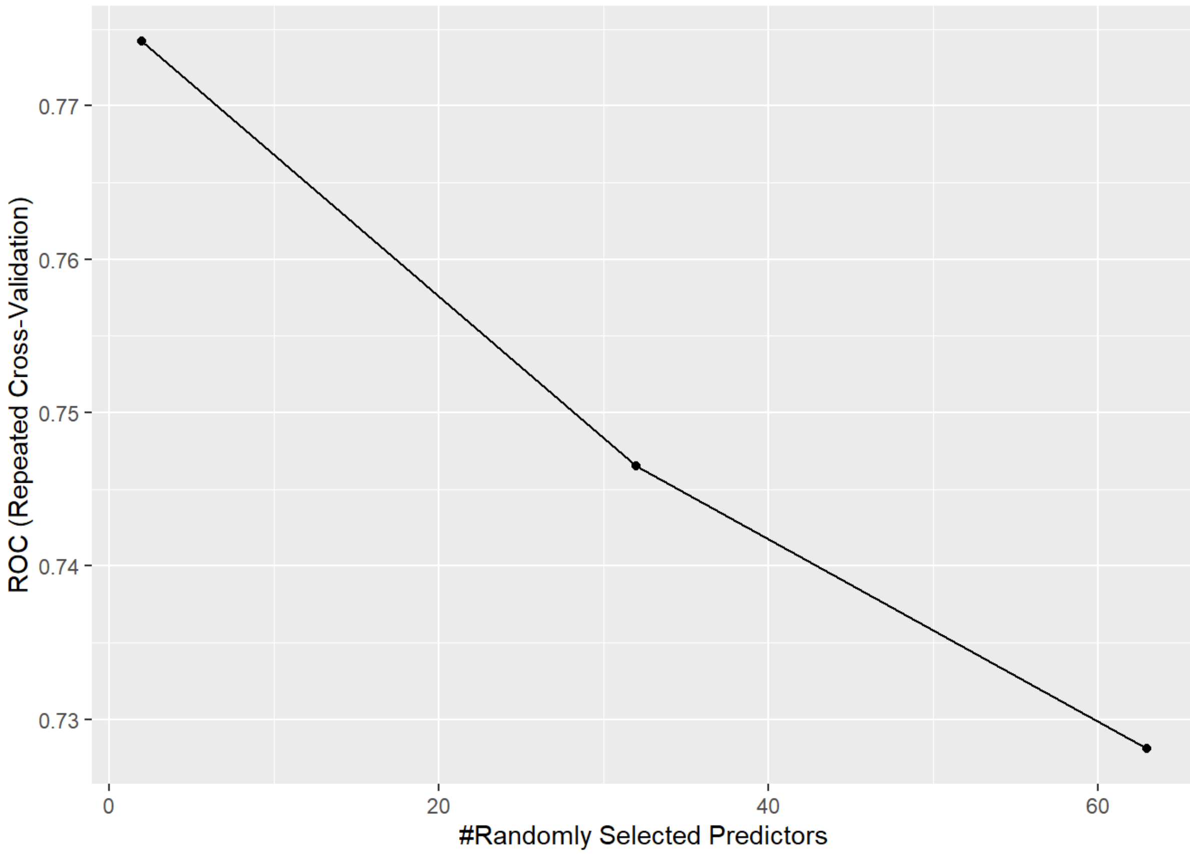
Random Forest

Next, we will look at Random Forest which is a ML Algorithm based on Decision Trees. Random Trees lies in one of those Class of ML Algorithms which does ensemble classification (Koehrsen, 2017). However, we first need to handle the data imbalance by using the `trainControl()` function which resamples a specific number of training choices required by the `train()` function (Dalpiaz, 2019).Then, we will use the k-Fold Cross-Validation method in the `trainControl()` function by setting the method to `repeatedcv` and set the number to 5 and repeats to 3 (Dalpiaz, 2019).

```
upsample_data <- train(
  Attrition ~., data = train_attr, method = "rf",
  trControl = trainControl(method = "repeatedcv",
                           number = 5,
                           repeats = 3,
                           sampling = "up",
                           summaryFunction=twoClassSummary,
                           classProbs=T)
)
```

Afterwards, we can now visualize the `mtry` which refers to number of variables available for splitting at each tree node (Brownlee, 2019).

```
upsample_data %>% ggplot()
```



Let us look at the detail and plot the `upsample_data`.

```
upsample_data
```

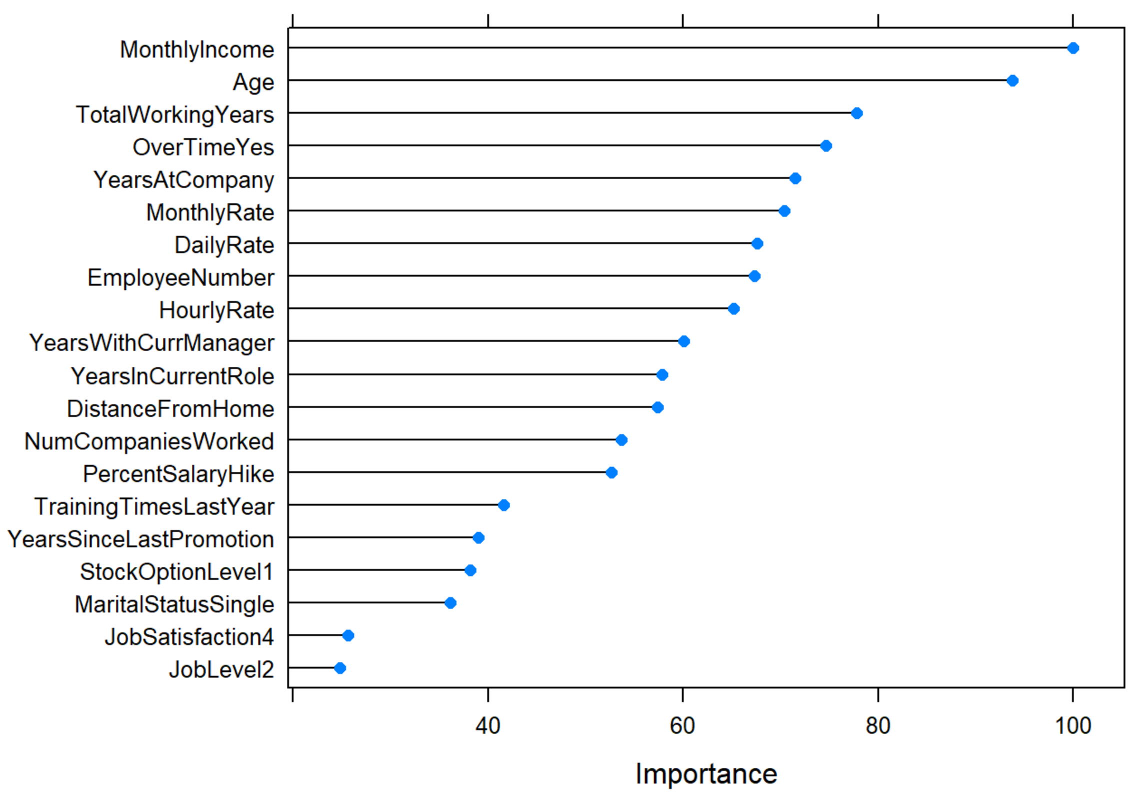
```
## Random Forest
##
## 1176 samples
##   31 predictor
##    2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 3 times)
## Summary of sample sizes: 941, 940, 941, 941, 941, 941, ...
## Addtional sampling using up-sampling
##
## Resampling results across tuning parameters:
##
##   mtry  ROC        Sens       Spec
##    2    0.7742121  0.9685360  0.2425926
##   32    0.7465151  0.9595092  0.2518519
##   63    0.7280674  0.9491441  0.2592593
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

Here is a list of the top 20 important variables and a plot to show them.

```
varImp(upsample_data)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 63)
##
##                         Overall
## MonthlyIncome            100.00
## Age                       93.72
## TotalWorkingYears         77.78
## OverTimeYes               74.62
## YearsAtCompany            71.44
## MonthlyRate               70.36
## DailyRate                 67.58
## EmployeeNumber            67.29
## HourlyRate                65.17
## YearsWithCurrManager      60.08
## YearsInCurrentRole        57.86
## DistanceFromHome          57.37
## NumCompaniesWorked        53.67
## PercentSalaryHike         52.61
## TrainingTimesLastYear     41.61
## YearsSinceLastPromotion   39.02
## StockOptionLevel1         38.17
## MaritalStatusSingle       36.13
## JobSatisfaction4          25.68
## JobLevel2                 24.76
```

```
varImp(upsample_data) %>% plot(20)
```



Model Performance (Random Forest)

Using the `upsample_data` model we made, we will predict using both `raw` as its typein the `predict()` function.

```
upsample_raw <- predict(upsample_data, test_attr, type = "raw")

confusionMatrix(upsample_raw,test_attr$Attrition, positive = "Yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  231  40
##        Yes   6  17
##
##                Accuracy : 0.8435
##                  95% CI : (0.7969, 0.8831)
##     No Information Rate : 0.8061
##     P-Value [Acc > NIR] : 0.0579
##
##                   Kappa : 0.3529
##
##  Mcnemar's Test P-Value : 1.141e-06
##
##             Sensitivity : 0.29825
##             Specificity : 0.97468
##          Pos Pred Value : 0.73913
##          Neg Pred Value : 0.85240
##              Prevalence : 0.19388
##          Detection Rate : 0.05782
##    Detection Prevalence : 0.07823
##       Balanced Accuracy : 0.63646
##
##        'Positive' Class : Yes
##
```

Results

We are going to focus on the sensitivity metic, which tells us when an employee is going to leave/attrition. So let us see how accurately my classifier can predict. Looking at confusion matrices in both models show that Logistic Regression is better to use. Meaning that the attrition for the sensitivity metric is 77% and accuracy is 78%, which is better than Random Forest's sensitivity metric which is 28%, although its accuracy is 84%

Modeling Employee Attrition with H2O

We are going to use the `h2o.automl()` function from the H2O platform which is an open-source, distributed in-memory machine learning platform with linear scalability (Cook, 2017). H2O also supports the most commonly used statistical and machine learning algorithms.

```
#initializign the JVM that H2O uses locally.
#install.packages("h2o")
library(h2o)
```

```
##
## ----------------------------------------------------------------------
##
## Your next step is to start H2O:
##     > h2o.init()
##
## For H2O package documentation, ask for help:
##     > ??h2o
##
## After starting H2O, you can use the Web UI at http://localhost:54321
## For more information visit http://docs.h2o.ai
##
## ----------------------------------------------------------------------
```

```
##
## Attaching package: 'h2o'
```

```
## The following objects are masked from 'package:data.table':
##
##     hour, month, week, year
```

```
## The following objects are masked from 'package:stats':
##
##     cor, sd, var
```

```
## The following objects are masked from 'package:base':
##
##     %*%, %in%, &&, ||, apply, as.factor, as.numeric, colnames,
##     colnames<-, ifelse, is.character, is.factor, is.numeric, log,
##     log10, log1p, log2, round, signif, trunc
```

```
h2o.init()
```

```
##  Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         1 hours 28 minutes
##     H2O cluster timezone:       America/Chicago
##     H2O data parsing timezone:  UTC
##     H2O cluster version:        3.30.0.1
##     H2O cluster version age:    29 days
##     H2O cluster name:           H2O_started_from_R_frede_ilh295
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   6.53 GB
##     H2O cluster total cores:    12
##     H2O cluster allowed cores:  12
##     H2O cluster healthy:        TRUE
##     H2O Connection ip:          localhost
##     H2O Connection port:        54321
##     H2O Connection proxy:       NA
##     H2O Internal Security:      FALSE
##     H2O API Extensions:         Amazon S3, Algos, AutoML, Core V3, TargetEncoder, Core V4
##     R Version:                  R version 3.5.1 (2018-07-02)
```

```
#Turn off output of progress bars
h2o.no_progress()
```

Next, in order for the h2o package to function on the data, we need change our data by splitting the data into a train, validation, and test sets. Cook's (2017) suggested that when training data in H2O we should split the data into three parts 70%(train), 15% (validation), 15%(test).

```
# Split data into Train/Validation/Test Sets
h2o_data <- as.h2o(HR_data)
set.seed(123)
h2o_split <- h2o.splitFrame(h2o_data, c(0.7, 0.15), seed = 1234 )
h2o_train <- h2o.assign(h2o_split[[1]], "train" )
h2o_valid <- h2o.assign(h2o_split[[2]], "valid" )
h2o_test  <- h2o.assign(h2o_split[[3]], "test" )
```

Modeling

Now we in order to ready the model, we will set the target as Attrition which we want to predict and set feature names every other column that we will use to model our prediction.

```
# Set names for h2o
target <- "Attrition"
features <- setdiff(names(h2o_train), target)
```

Now we can use the `h2o.automl()` function with its respected arguments set in place to run the models against.

1. x = features: Feature columns.
2. y = target: Target column.
3. training_frame = h2o_train: The 70% of data that will be used for training.
4. leaderboard_frame = h2o_valid:The 15% of data that will be used for validation. Also to ensure that overfitting does not occur in the model, H2O uses the validation_set to solve that issue (Cook,2017).
5. max_runtime_secs = 30: Due to the algoritm having a large number of complex models, we set this runtime to 30 so that it can speed up the process at the expense of some accuracy.

```
# Run the automated machine learning
h2o_automl_models <- h2o.automl(
  x = features,
  y = target,
  training_frame    = h2o_train,
  leaderboard_frame = h2o_valid,
  max_runtime_secs  = 30
)
```

```
##
## 14:28:33.875: AutoML: XGBoost is not available; skipping it.
```

The `h2o_automl_models` will store all the models, but primary focus will be on the best model in terms of accuracy on the validation set. Then have that models object extracted.

```
# View the AutoML Leaderboard
leaderboard_1 <- h2o_automl_models@leaderboard
leaderboard_1
```

```
##                                              model_id       auc   logloss
## 1                        GLM_1_AutoML_20200503_142833 0.8364350 0.3461573
## 2 DeepLearning_grid__1_AutoML_20200503_142833_model_1 0.8284385 0.4352318
## 3 StackedEnsemble_BestOfFamily_AutoML_20200503_142833 0.8261122 0.3520441
## 4    StackedEnsemble_AllModels_AutoML_20200503_142833 0.8245129 0.3522909
## 5 DeepLearning_grid__1_AutoML_20200503_142833_model_2 0.8202966 0.3991124
## 6 DeepLearning_grid__2_AutoML_20200503_142833_model_3 0.8176796 0.4590100
##       aucpr mean_per_class_error      rmse       mse
## 1 0.5664478            0.2492003 0.3241355 0.1050638
## 2 0.5343350            0.2297179 0.3517678 0.1237406
## 3 0.5864403            0.2470922 0.3266879 0.1067250
## 4 0.5882948            0.2366967 0.3270538 0.1069642
## 5 0.5728671            0.2907822 0.3344829 0.1118788
## 6 0.5635296            0.2233934 0.3568209 0.1273211
##
## [27 rows x 7 columns]
```

```
lead_model<-h2o_automl_models@leader
```

Predicting

Now prediction can be made on the test set, which is not seen during the modeling process. In order to make predictions, we will use the `h2o.predict()` function to get a true test of performance.

```
# Predict on hold-out set, test_h2o
h2o_prediction <- h2o.predict(object = lead_model, newdata = h2o_test)
```

Performance

Let us on evaluate `lead_model` by reformatting the test set and adding the predictions as column. That way we can see both the actual column and prediction column.

```
#predictions on test set
h2o_pdict<- predict(lead_model, h2o_test)
head(h2o_pdict)
```

```
##   predict        No         Yes
## 1      No 0.6842836 0.315716365
## 2      No 0.9654499 0.034550097
## 3     Yes 0.2042892 0.795710820
## 4      No 0.9876785 0.012321475
## 5      No 0.7323095 0.267690504
## 6      No 0.9922056 0.007794423
```

```
#perfomance on test set
h2o_perform <- h2o.performance(lead_model, h2o_test)
print(h2o_perform)
```

```
## H2OBinomialMetrics: glm
##
## MSE:  0.07203801
## RMSE:  0.268399
## LogLoss:  0.2489955
## Mean Per-Class Error:  0.2178856
## AUC:  0.9062145
## AUCPR:  0.7263954
## Gini:  0.812429
## R^2:  0.3923448
## Residual Deviance:  105.0761
## AIC:  265.0761
##
## Confusion Matrix (vertical: actual; across: predicted) for F1-optimal threshold:
##         No Yes    Error     Rate
## No     178   4 0.021978   =4/182
## Yes     12  17 0.413793   =12/29
## Totals 190  21 0.075829  =16/211
##
## Maximum Metrics: Maximum metrics at their respective thresholds
##                         metric threshold      value idx
## 1                       max f1  0.495752   0.680000  20
## 2                       max f2  0.376682   0.718954  36
## 3                 max f0point5  0.538474   0.752688  15
## 4                 max accuracy  0.495752   0.924171  20
## 5                max precision  0.889522   1.000000   0
## 6                   max recall  0.031027   1.000000 158
## 7              max specificity  0.889522   1.000000   0
## 8             max absolute_mcc  0.495752   0.648939  20
## 9   max min_per_class_accuracy  0.237678   0.818681  56
## 10 max mean_per_class_accuracy  0.376682   0.838102  36
## 11                     max tns  0.889522 182.000000   0
## 12                     max fns  0.889522  28.000000   0
## 13                     max fps  0.000377 182.000000 210
## 14                     max tps  0.031027  29.000000 158
## 15                     max tnr  0.889522   1.000000   0
## 16                     max fnr  0.889522   0.965517   0
## 17                     max fpr  0.000377   1.000000 210
## 18                     max tpr  0.031027   1.000000 158
##
## Gains/Lift Table: Extract with `h2o.gainsLift(<model>, <data>)` or `h2o.gainsLift(<model>, valid=<T/F>,
xval=<T/F>)`
```

```r
library(tibble)
#Prepping for performance assessment
performance_test <- h2o_test %>%
  tibble::as_tibble() %>%
  select(Attrition) %>%
  add_column(predictions = as.vector(h2o_pdict$predict)) %>%
  mutate_if(is.character, as.factor)
performance_test
```

```
## # A tibble: 211 x 2
##    Attrition predictions
##    <fct>     <fct>
##  1 No        No
##  2 No        No
##  3 Yes       Yes
##  4 No        No
##  5 No        No
##  6 No        No
##  7 Yes       Yes
##  8 No        No
##  9 No        No
## 10 Yes       No
## # ... with 201 more rows
```

```
#Building confusion matrix for test set
h2o_cmtx <- h2o.confusionMatrix(h2o_perform)
print(h2o_cmtx)
```

```
## Confusion Matrix (vertical: actual; across: predicted)  for max f1 @ threshold = 0.49575199442704:
##         No Yes    Error     Rate
## No     178   4 0.021978   =4/182
## Yes     12  17 0.413793   =12/29
## Totals 190  21 0.075829  =16/211
```

```
h2o.precision(h2o_perform)
```

```
##   threshold precision
## 1 0.8895218         1
## 2 0.8790361         1
## 3 0.8754062         1
## 4 0.8533827         1
## 5 0.8057816         1
##
## ---
##        threshold precision
## 206 0.0027456723 0.1407767
## 207 0.0024295353 0.1400966
## 208 0.0017465319 0.1394231
## 209 0.0011489459 0.1387560
## 210 0.0010812618 0.1380952
## 211 0.0003767416 0.1374408
```

```
h2o.accuracy(h2o_perform)
```

```
##   threshold  accuracy
## 1 0.8895218 0.8672986
## 2 0.8790361 0.8720379
## 3 0.8754062 0.8767773
## 4 0.8533827 0.8815166
## 5 0.8057816 0.8862559
##
## ---
##         threshold  accuracy
## 206 0.0027456723 0.1611374
## 207 0.0024295353 0.1563981
## 208 0.0017465319 0.1516588
## 209 0.0011489459 0.1469194
## 210 0.0010812618 0.1421801
## 211 0.0003767416 0.1374408
```

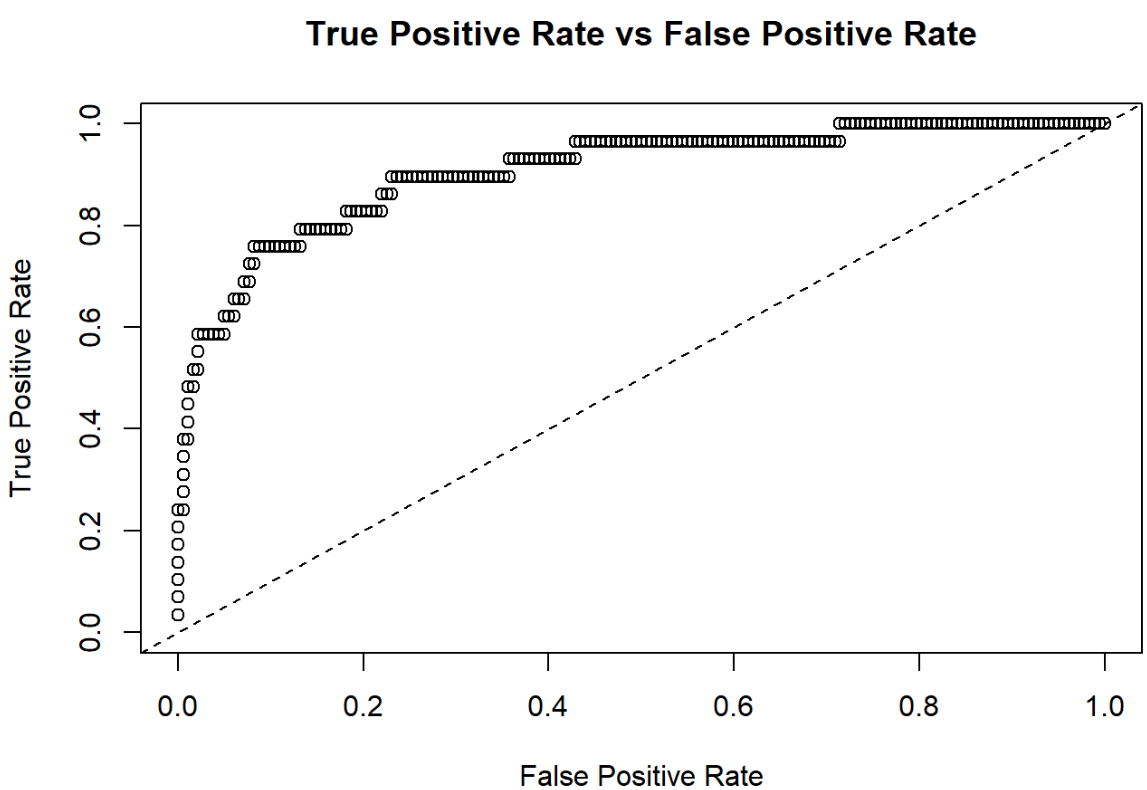
```
h2o.auc(h2o_perform)
```

```
## [1] 0.9062145
```

```
h2o.auc(h2o_perform)
```

```
## [1] 0.9062145
```

```
#Plot ROC for test set
plot(h2o_perform,type="roc")
```



Then using the `table()` function, we are able to get a quick look at the results via a confusion matrix.

```
# Confusion table counts
c_mtx <- performance_test %>%
  table()
c_mtx
```

```
##          predictions
## Attrition  No Yes
##       No  168  14
##       Yes   9  20
```

From the table, we see the `lead_model` was not the best. Although, the tack of trying to identify which employees that are likely to quit, it did a decent job of that.

Now we will see this model's performance by running a binary classification analysis.

```
# Performance analysis
tn <- c_mtx[1]
tp <- c_mtx[4]
fp <- c_mtx[3]
fn <- c_mtx[2]

accuracy <- (tp + tn) / (tp + tn + fp + fn)
misclassification_rate <- 1 - accuracy
sensitivity <- tp / (tp + fn)
precision <- tp / (tp + fp)
null_error_rate <- tn / (tp + tn + fp + fn)

tibble(
  paste("accuracy: ",accuracy),
  paste("misclassification_rate: ", misclassification_rate),
  paste("sensitivity: ", sensitivity),
  paste("precision: ", precision),
  paste("null_error_rate: ", null_error_rate)
) %>%
  transpose()
```

```
##                                        V1
## 1               accuracy:  0.890995260663507
## 2 misclassification_rate:  0.109004739336493
## 3            sensitivity:  0.689655172413793
## 4              precision:  0.588235294117647
## 5        null_error_rate:  0.796208530805687
```

Conclusion

The autoML algorithm from worked well for classifying attrition with an accuracy around the high 80 percentile on the unmodeled dataset.

References

Altini, M. (2015, August 17). Dealing with imbalanced data: undersampling, oversampling and proper cross-validation. Retrieved April 14, 2020, from https://www.marcoaltini.com/blog/dealing-with-imbalanced-data-undersampling-oversampling-and-proper-cross-validation (https://www.marcoaltini.com/blog/dealing-with-imbalanced-data-undersampling-oversampling-and-proper-cross-validation)

Analytics , V. (Ed.). (2019, June 24). Practical Guide to deal with Imbalanced Classification Problems in R. Retrieved April 14, 2020, from https://www.analyticsvidhya.com/blog/2016/03/practical-guide-deal-imbalanced-classification-problems/ (https://www.analyticsvidhya.com/blog/2016/03/practical-guide-deal-imbalanced-classification-problems/)

Ashe-Edmunds, S. (2017, November 21). Retention vs. Replacement for Employees. Retrieved April 07, 2020, from https://work.chron.com/retention-vs-replacement-employees-22865.html (https://work.chron.com/retention-vs-replacement-employees-22865.html)

Bendemra, H. (2019, March 11). Building an Employee Churn Model in Python to Develop a Strategic Retention Plan. Retrieved March 16, 2020, from https://towardsdatascience.com/building-an-employee-churn-model-in-python-to-develop-a-strategic-retention-plan-57d5bd882c2d (https://towardsdatascience.com/building-an-employee-churn-model-in-python-to-develop-a-strategic-retention-plan-57d5bd882c2d)

Bock, T. (2018, October 25). What is Feature Engineering? Retrieved April 10, 2020, from https://www.displayr.com/what-is-feature-engineering/ (https://www.displayr.com/what-is-feature-engineering/)

Brownlee, J. (2019, August 08). A Gentle Introduction to k-fold Cross-Validation. Retrieved May 26, 2020, from https://machinelearningmastery.com/k-fold-cross-validation/ (https://machinelearningmastery.com/k-fold-cross-validation/)

Brownlee, J. (2019, August 22). Tune Machine Learning Algorithms in R (random forest case study). Retrieved April 27, 2020, from https://machinelearningmastery.com/tune-machine-learning-algorithms-in-r/ (https://machinelearningmastery.com/tune-machine-learning-algorithms-in-r/)

Cook, D. (2017). Practical machine learning with H2O powerful, scalable techniques for deep learning and AI. Seabastopol, CA: O'Reilly Media.

Dalpiaz, D. (2019, March 22). R for Statistical Learning. Retrieved April 26, 2020, from https://daviddalpiaz.github.io/r4sl/the-caret-package.html (https://daviddalpiaz.github.io/r4sl/the-caret-package.html)

Fox, J., & Weisberg, S. (2019). An R companion to applied regression (3rd ed.). Thousand Oaks, CA: SAGE Publications.

Kassambara, A. (2018, March 11). Stepwise Regression Essentials in R. Retrieved April 27, 2020, from http://www.sthda.com/english/articles/37-model-selection-essentials-in-r/154-stepwise-regression-essentials-in-r/ (http://www.sthda.com/english/articles/37-model-selection-essentials-in-r/154-stepwise-regression-essentials-in-r/)

Kassambara, A. (2018, March 11). Multicollinearity Essentials and VIF in R. Retrieved April 27, 2020, from http://www.sthda.com/english/articles/39-regression-model-diagnostics/160-multicollinearity-essentials-and-vif-in-r/ (http://www.sthda.com/english/articles/39-regression-model-diagnostics/160-multicollinearity-essentials-and-vif-in-r/)

Koehrsen, W. (2017, December 27). Random Forest Simple Explanation. Retrieved April 26, 2020, from https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d (https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d)

Kolowich, L. (2018, February 26). Why Are Your Employees Leaving The Organization (And How to Make Them Stay). Retrieved April 07, 2020, from https://blog.hubspot.com/agency/why-employees-leave (https://blog.hubspot.com/agency/why-employees-leave)

Kosmidis, I., & Schumacher, D. (2020, January 5). Detect/check for separation and infinite maximum likelihood estimates in logistic regression. Retrieved April 19, 2020, from https://cran.r-project.org/web/packages/detectseparation/vignettes/separation.html (https://cran.r-project.org/web/packages/detectseparation/vignettes/separation.html)

Kuhn, M. (2019, March 27). The caret Package. Retrieved April 13, 2020, from https://topepo.github.io/caret/ (https://topepo.github.io/caret/)

Law, R. (2019, May 28). Churn Rate: How High is Too High? A Meta-Analysis of Churn Studies. Retrieved April 07, 2020, from https://www.cobloom.com/blog/churn-rate-how-high-is-too-high (https://www.cobloom.com/blog/churn-rate-how-high-is-too-high)

Phiri, L. (2013, February 13). Ggplot2 - Multiple Plots in One Graph Using gridExtra. Retrieved April 09, 2020, from https://lightonphiri.org/blog/ggplot2-multiple-plots-in-one-graph-using-gridextra (https://lightonphiri.org/blog/ggplot2-multiple-plots-in-one-graph-using-gridextra)

Quinn, M., & Waring, E. (2019, October 29). (Re)introducing skimr v2 - A year in the life of an open source R project - rOpenSci - open tools for open science. Retrieved April 08, 2020, from https://ropensci.org/blog/2019/10/29/skimrv2/ (https://ropensci.org/blog/2019/10/29/skimrv2/)

R Function of the Day: sample. (2010, May 23). Retrieved April 16, 2020, from https://www.r-bloggers.com/r-function-of-the-day-sample-2/ (https://www.r-bloggers.com/r-function-of-the-day-sample-2/)

Regan, R. (2020, March 17). 10 Clever Employee Retention Strategies in 2020. Retrieved April 07, 2020, from https://connecteam.com/employee-retention-strategies/ (https://connecteam.com/employee-retention-strategies/)

Shah, T. (2017, December 10). About Train, Validation and Test Sets in Machine Learning. Retrieved April 14, 2020, from https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7 (https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7)

Sirohi, K. (2018, December 29). Simply Explained Logistic Regression with Example in R. Retrieved April 19, 2020, from https://towardsdatascience.com/simply-explained-logistic-regression-with-example-in-r-b919acb1d6b3 (https://towardsdatascience.com/simply-explained-logistic-regression-with-example-in-r-b919acb1d6b3)

Xu, N. [Data Science Dojo]. (2018, January 24). Feature Engineering | Introduction to dplyr Part 4 [Video File]. Retrieved from https://youtu.be/nVnAcE-BGvA (https://youtu.be/nVnAcE-BGvA)