HR-Employee-Attrition

Gladys Teeson

6/20/2020

Contents

Introduction

Employee attrition is defined as the natural process by which employees leave the workforce – for example, through resignation for personal reasons or retirement – and are not immediately replaced. But when attrition crosses a particular threshold, it becomes a cause for concern. For example, attrition among younger employees can affect the financial status of an organization via spending much costs for their training programs. Or, attrition among senior leaders can lead to a significant gap in organizational leadership. The attrition can also reduce the strength of workforce and increase the work load for remaining employees. So it is important to know where an organization stands on the employee attrition curve.

For this project, we will be uncovering the factors that lead to employee attrition using the dataset created by IBM data scientists. The objective of this project is to model machine learning algorithms that could generate new insights for the business on what drives attrition and the leading factors. We will finally evaluate the performance of the models using AUC(Area under the ROC curve).

IBM Employee Attrtion Dataset

Load Libraries

```r
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(arules)) install.packages("arules", repos = "http://cran.us.r-project.org")
if(!require(corrplot)) install.packages("corrplot", repos = "http://cran.us.r-project.org")
if(!require(dplyr)) install.packages("dplyr", repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(pROC)) install.packages("pROC", repos = "http://cran.us.r-project.org")
if(!require(xgboost)) install.packages("xgboost", repos = "http://cran.us.r-project.org")
if(!require(DMwR)) install.packages("DMwR", repos = "http://cran.us.r-project.org")
```

Import Dataset

```r
#IBM Dataset
#https://www.kaggle.com/pavansubhasht/ibm-hr-analytics-attrition-dataset

## Import the dataset
```

```r
data = read.csv("https://raw.githubusercontent.com/gladysteeson/HR-Employee-Attrition/master/HR-Employe

# structure of the dataset
str(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
##  $ 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 ...
```

```r
# number of rows and columns in the dataset
dim(data)
```

```
## [1] 1470   35
```

The IBM dataset contains 1470 rows and 35 columns with each row representing information about employees.

Data Pre-procesing

We will check for any missing values in the dataset.

```
# check for NA/missing values
sum(is.na(data))
```

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

The sum comes out to be zero which indicates that there are no NA values.

We will check for any duplicate entries in the dataset too.

```
# check for Duplicate entries
nrow(data[!(duplicated(data)),])
```

```
## [1] 1470
```

```
nrow(data)
```

```
## [1] 1470
```

We can see that all the rows are unique here.

In the dataset, we notice that some columns have the same value for all employees. We can delete these columns and some irrelevant columns too.

```
# remove columns which have same value for all
data$EmployeeCount <- NULL
data$StandardHours <- NULL
data$Over18 <- NULL

# remove irrelevant columns
data$EmployeeNumber <- NULL
data$DailyRate <- NULL
```

Now we have a dataset with 30 columns.

```
# current number of rows and columns in the dataset
dim(data)
```

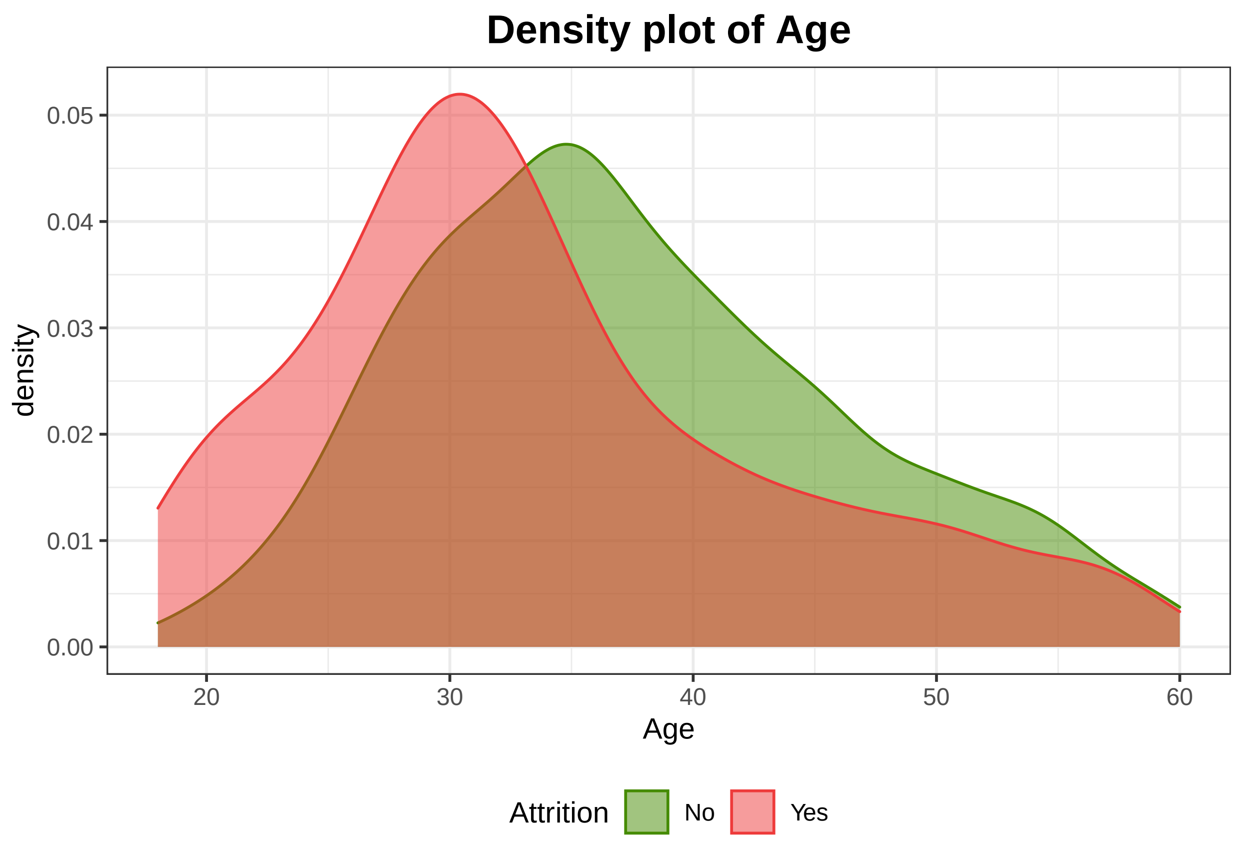
```
## [1] 1470   30
```

Exploratory Data Analysis

We can explore the relationships between Attrition and other variables and try to understand if any particular pattern exists.

Attrition by Age

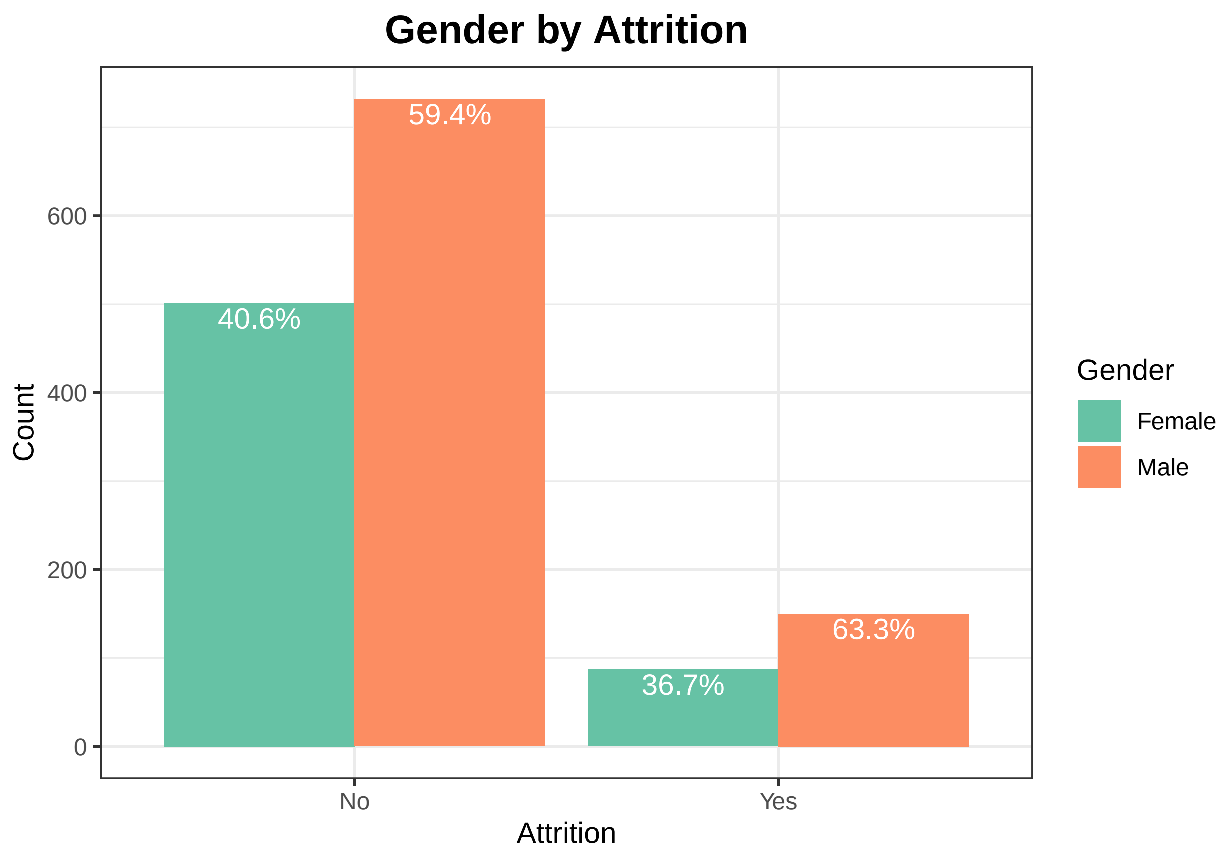
```r
# Attrition by Age
ggplot(data, aes(x=Age, fill=Attrition, color=Attrition)) +
  geom_density(position="identity", alpha=0.5) +
  theme_bw() +
  scale_fill_manual(values=c("chartreuse4", "brown2")) +
  scale_color_manual(values=c("chartreuse4", "brown2")) +
  ggtitle("Density plot of Age") +
  theme(plot.title = element_text(face = "bold", hjust = 0.5, size = 15), legend.position="bottom") +
  labs(x = "Age")
```



We can see that the attrition among the young employees are more than the senior employees.

Gender by Attrition

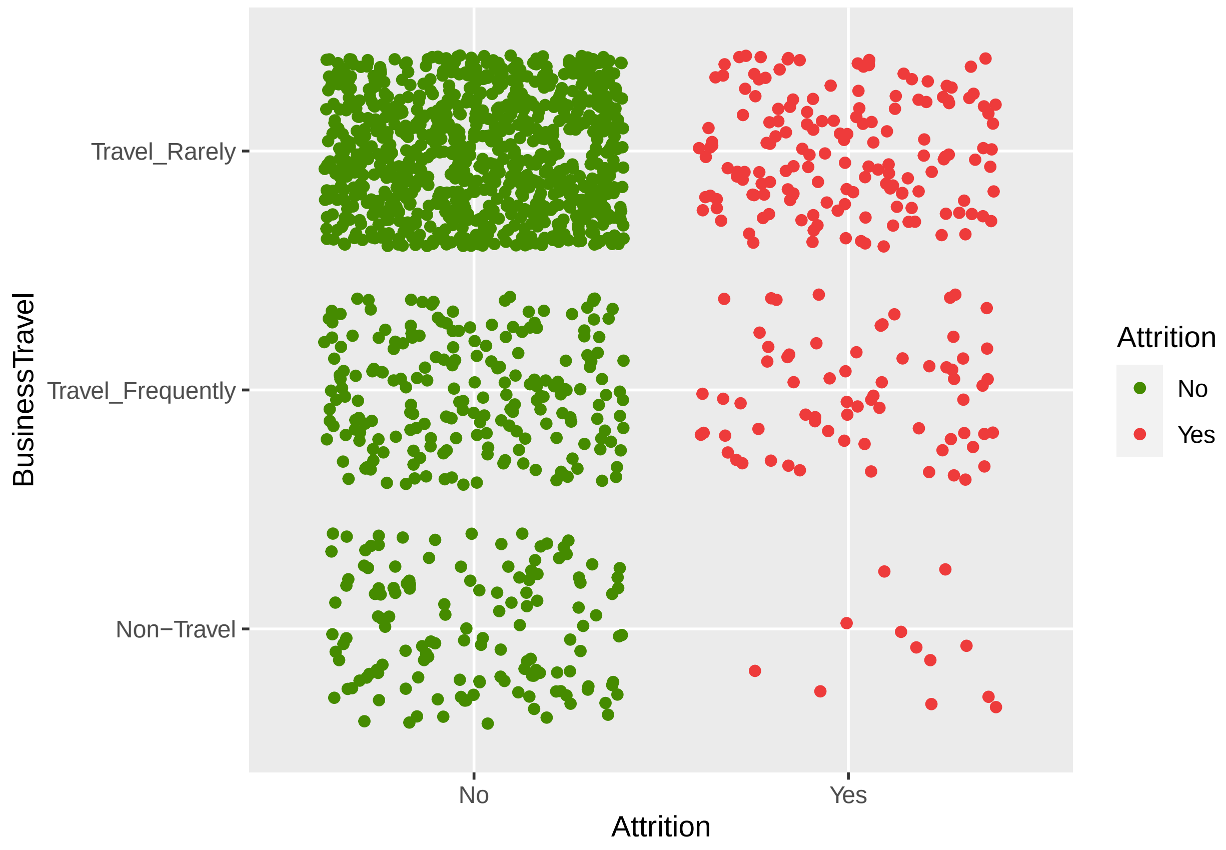
```r
# Gender distribution by Attrition
data %>% group_by(Attrition, Gender) %>% summarize(N = n()) %>% mutate(countT = sum(N)) %>%
  group_by(Attrition, Gender, add=TRUE) %>% mutate(per=paste0(round(100*N/countT,1),'%')) %>%
  ggplot(aes(x=Attrition, y=N, fill=Gender)) +
  geom_bar(stat="identity", position=position_dodge()) +
  theme_bw() +
  scale_fill_brewer(palette="Set2") +
  geom_text(aes(label = per), size = 4, vjust = 1.2, color = "#FFFFFF", position = position_dodge(0.9))
  ggtitle("Gender by Attrition") +
  theme(plot.title = element_text(face = "bold", hjust = 0.5, size = 15)) +
  labs(x = "Attrition", y = "Count")
```



We can see that males are leaving the company the most.

Business Travel by Attrition

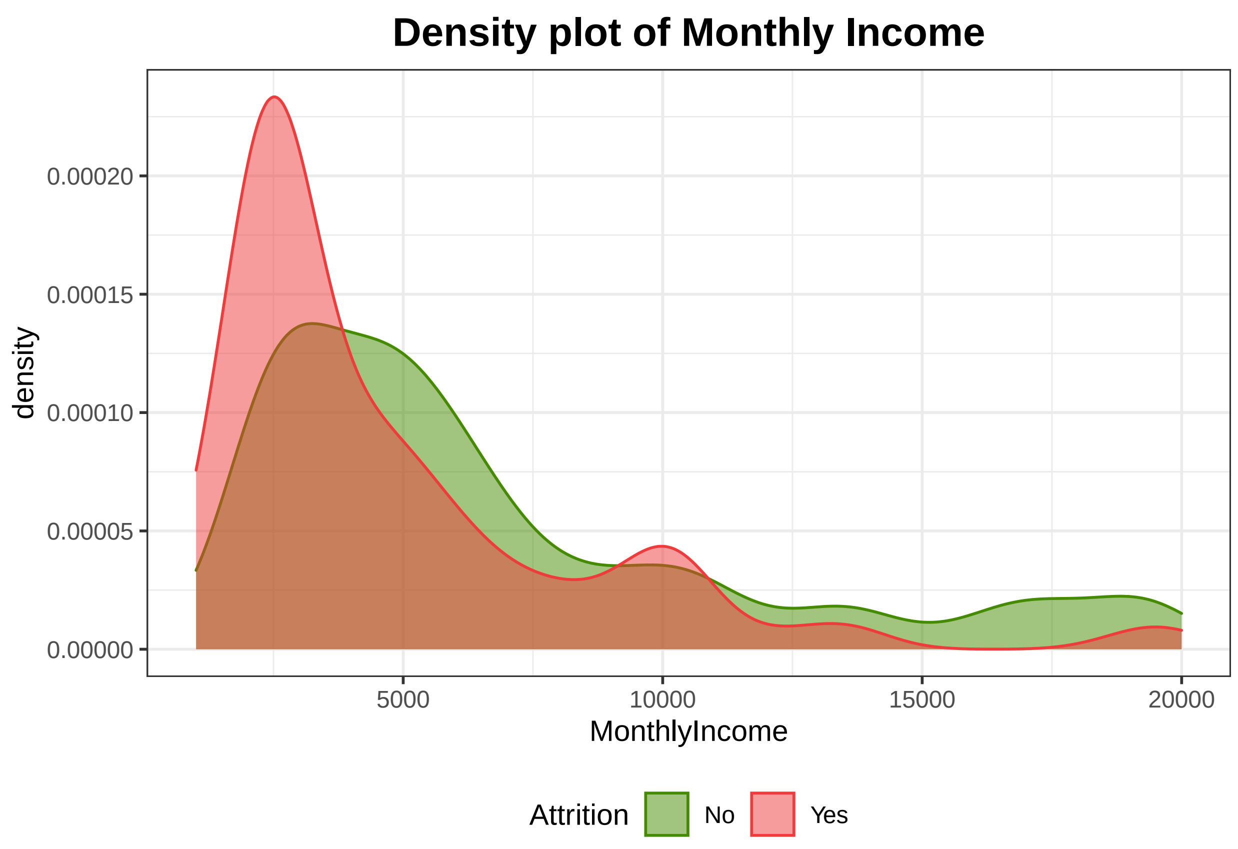
```r
# Business Travel frequency distribution by Attrition
ggplot(data,aes(Attrition,BusinessTravel,color=Attrition))+geom_jitter() + scale_color_manual(values=c(
```



It seems that Travel dosent have a high impact on Attrtion rates.

Monthly Income by Attrition

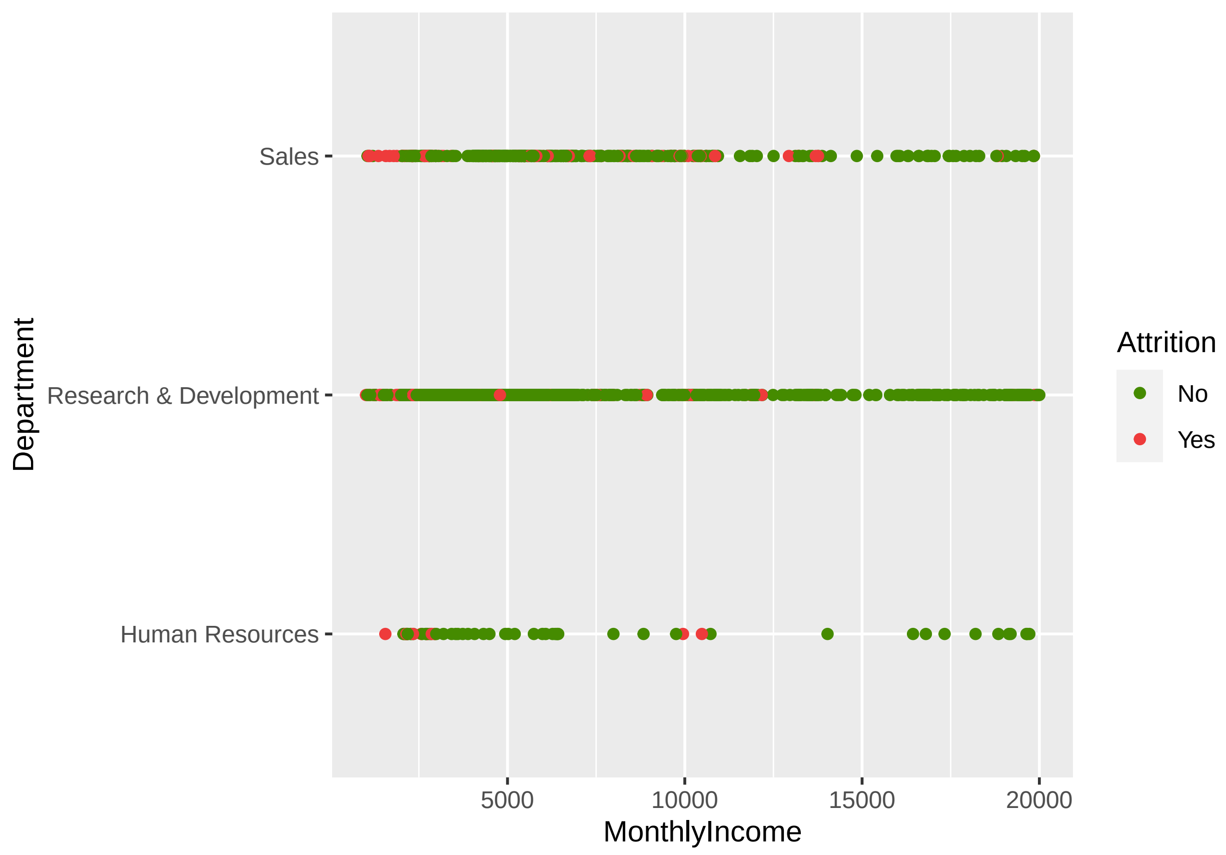
```r
# Monthly Income distribution by Attrition
ggplot(data, aes(x=MonthlyIncome, fill=Attrition, color=Attrition)) +
  geom_density(position="identity", alpha=0.5) +
  theme_bw() +
  scale_fill_manual(values=c("chartreuse4", "brown2")) +
  scale_color_manual(values=c("chartreuse4", "brown2")) +
  ggtitle("Density plot of Monthly Income") +
  theme(plot.title = element_text(face = "bold", hjust = 0.5, size = 15), legend.position="bottom") +
  labs(x = "MonthlyIncome")
```



Employees with low monthly income tend to leave the company the most.

Attrition by Monthly Income and Department

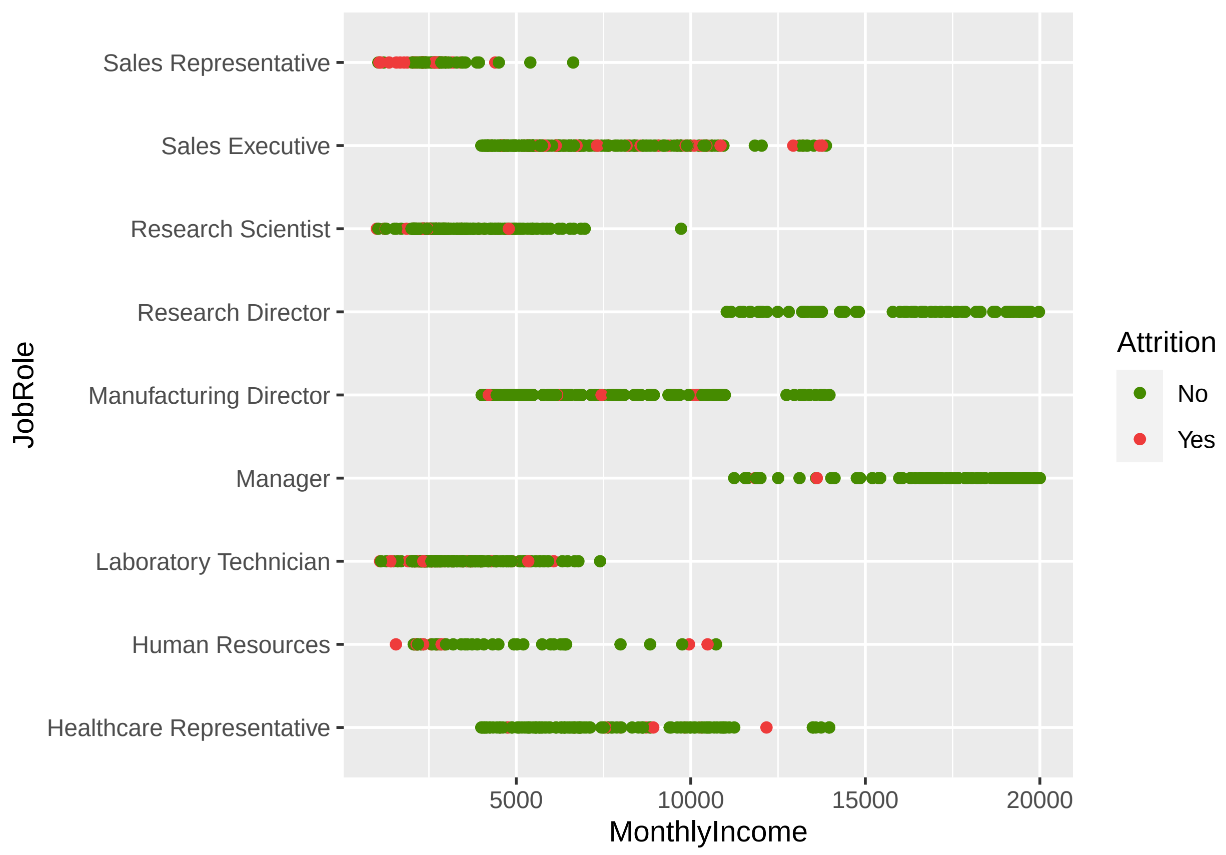
```
# Attrition by Monthly Income and Department
ggplot(data,aes(MonthlyIncome,Department, color=Attrition))+geom_point() + scale_color_manual(values=c(
```



Sales department shows the high attrition rate.

Attrition by Monthly Income and Job Role

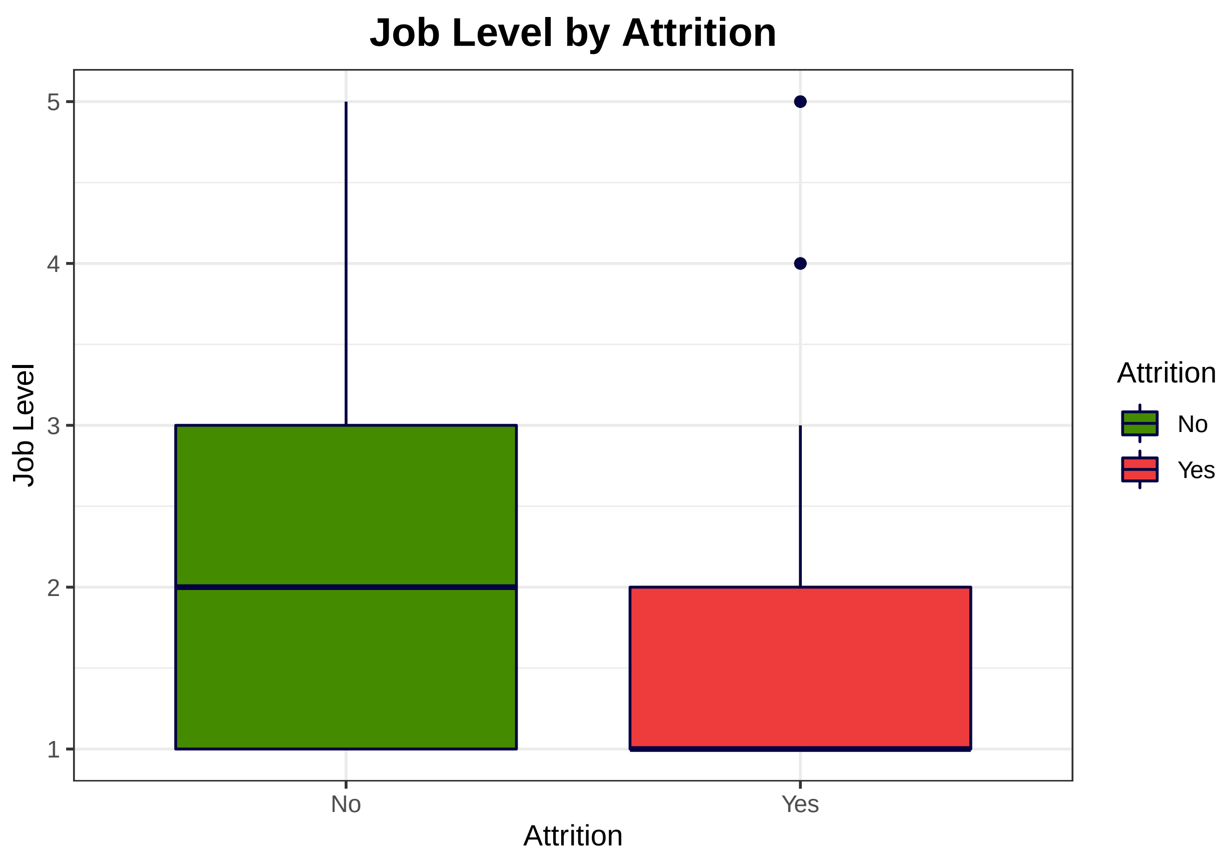
```
# Attrition by Monthly Income and Job Role
ggplot(data,aes(MonthlyIncome,JobRole, color=Attrition))+geom_point() + scale_color_manual(values=c("ch
```



Sales Employees with low monthly income seems to leave the company the most.

Job Levels by Attrition

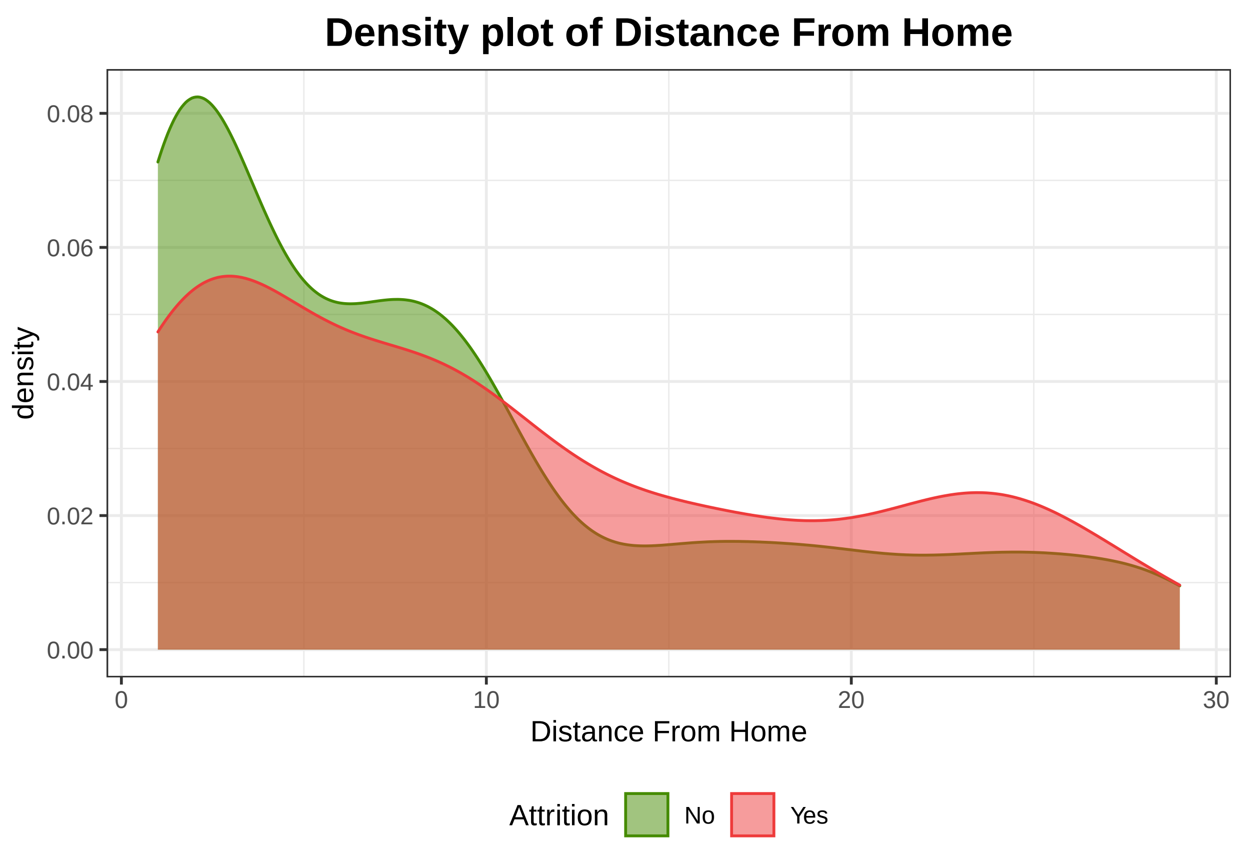
```r
# Job Levels by Attrition
ggplot(data, aes(x=Attrition, y=JobLevel, color=Attrition, fill=Attrition)) +
  geom_boxplot() +
  theme_bw() +
  scale_fill_manual(values=c("chartreuse4", "brown2")) +
  scale_color_manual(values=c("#040242", "#040242")) +
  ggtitle("Job Level by Attrition") +
  theme(plot.title = element_text(face = "bold", hjust = 0.5, size = 15)) +
  labs(x = "Attrition", y = "Job Level")
```



Employees with job level 4 & 5 show the least attrition.

Attrition by Distance From Home

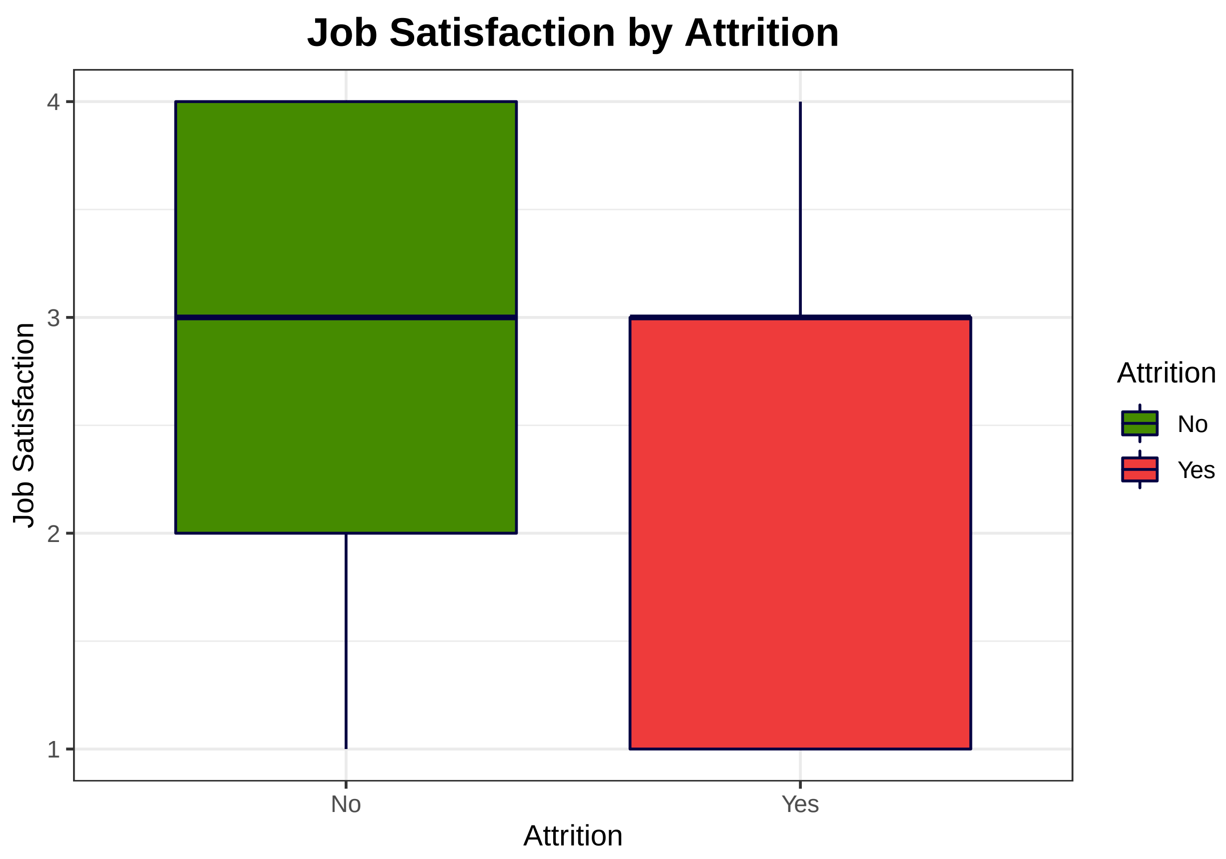
```
# Attrition by Distance From Home
ggplot(data, aes(x=DistanceFromHome, fill=Attrition, color=Attrition)) +
  geom_density(position="identity", alpha=0.5) +
  theme_bw() +
  scale_fill_manual(values=c("chartreuse4", "brown2")) +
  scale_color_manual(values=c("chartreuse4", "brown2")) +
  ggtitle("Density plot of Distance From Home") +
  theme(plot.title = element_text(face = "bold", hjust = 0.5, size = 15), legend.position="bottom") +
  labs(x = "Distance From Home")
```



Employees with less travelling distance from home seem to quit the job more.

Job Satisfaction by Attrition

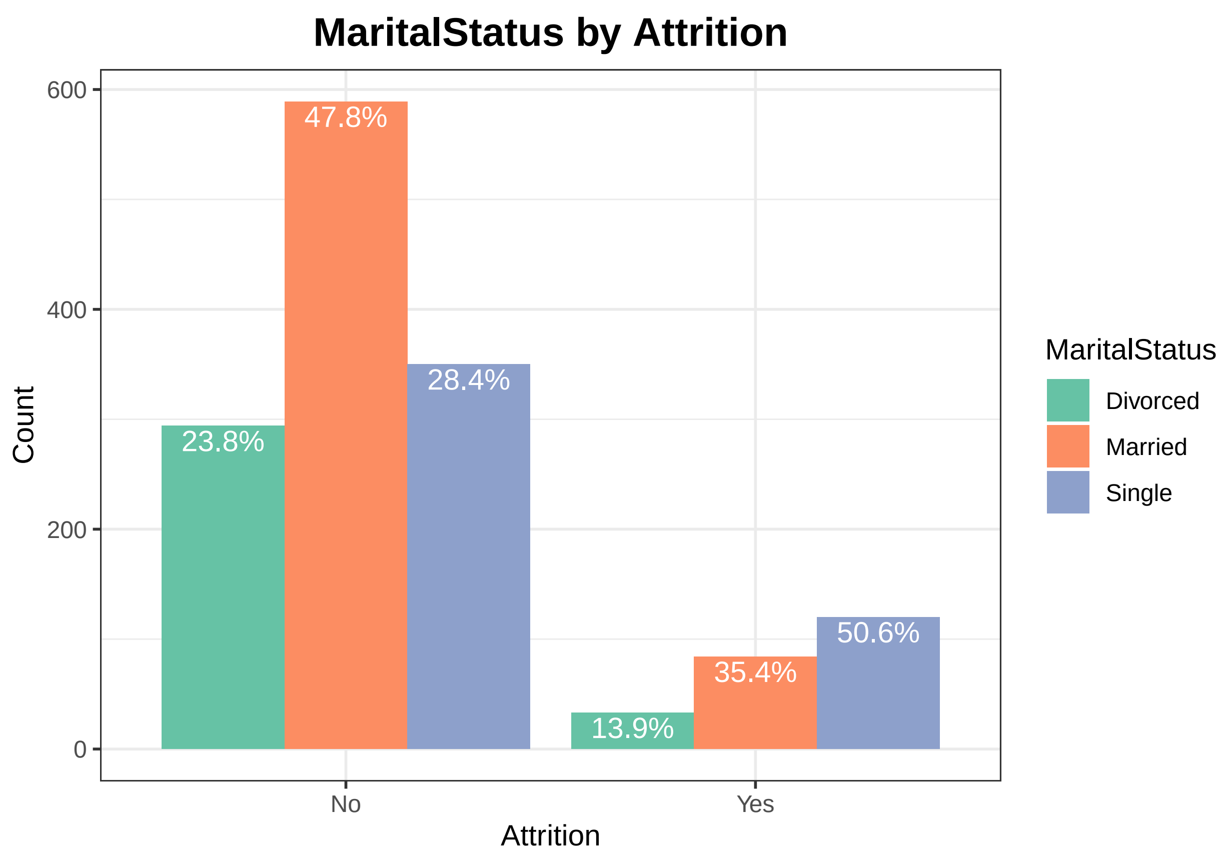
```r
# Job Satisfaction by Attrition
ggplot(data, aes(x=Attrition, y=JobSatisfaction, color=Attrition, fill=Attrition)) +
  geom_boxplot() +
  theme_bw() +
  scale_fill_manual(values=c("chartreuse4", "brown2")) +
  scale_color_manual(values=c("#040242", "#040242")) +
  ggtitle("Job Satisfaction by Attrition") +
  theme(plot.title = element_text(face = "bold", hjust = 0.5, size = 15)) +
  labs(x = "Attrition", y = "Job Satisfaction")
```



Employees with lower job satisfaction left the company the most.

Marital Status by Attrition

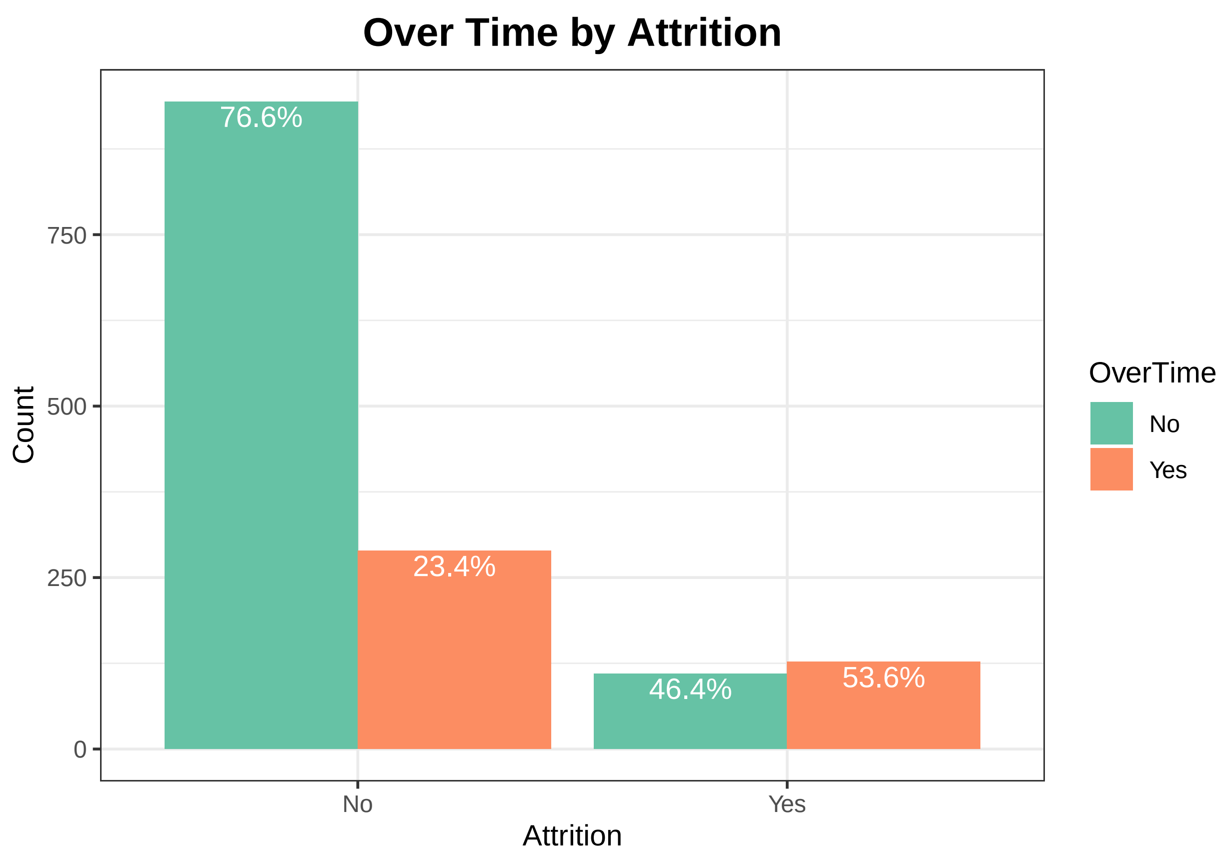
```
# Marital Status by Attrition
data %>% group_by(Attrition, MaritalStatus) %>% summarize(N = n()) %>% mutate(countT = sum(N)) %>%
  group_by(Attrition, MaritalStatus, add=TRUE) %>% mutate(per=paste0(round(100*N/countT,1),'%')) %>%
  ggplot(aes(x=Attrition, y=N, fill=MaritalStatus)) +
  geom_bar(stat="identity", position=position_dodge()) +
  theme_bw() +
  scale_fill_brewer(palette="Set2") +
  geom_text(aes(label = per), size = 4, vjust = 1.2, color = "#FFFFFF", position = position_dodge(0.9))
  ggtitle("MaritalStatus by Attrition") +
  theme(plot.title = element_text(face = "bold", hjust = 0.5, size = 15)) +
  labs(x = "Attrition", y = "Count")
```



We can see that employees who are single left the company the most.

Over Time by Attrition

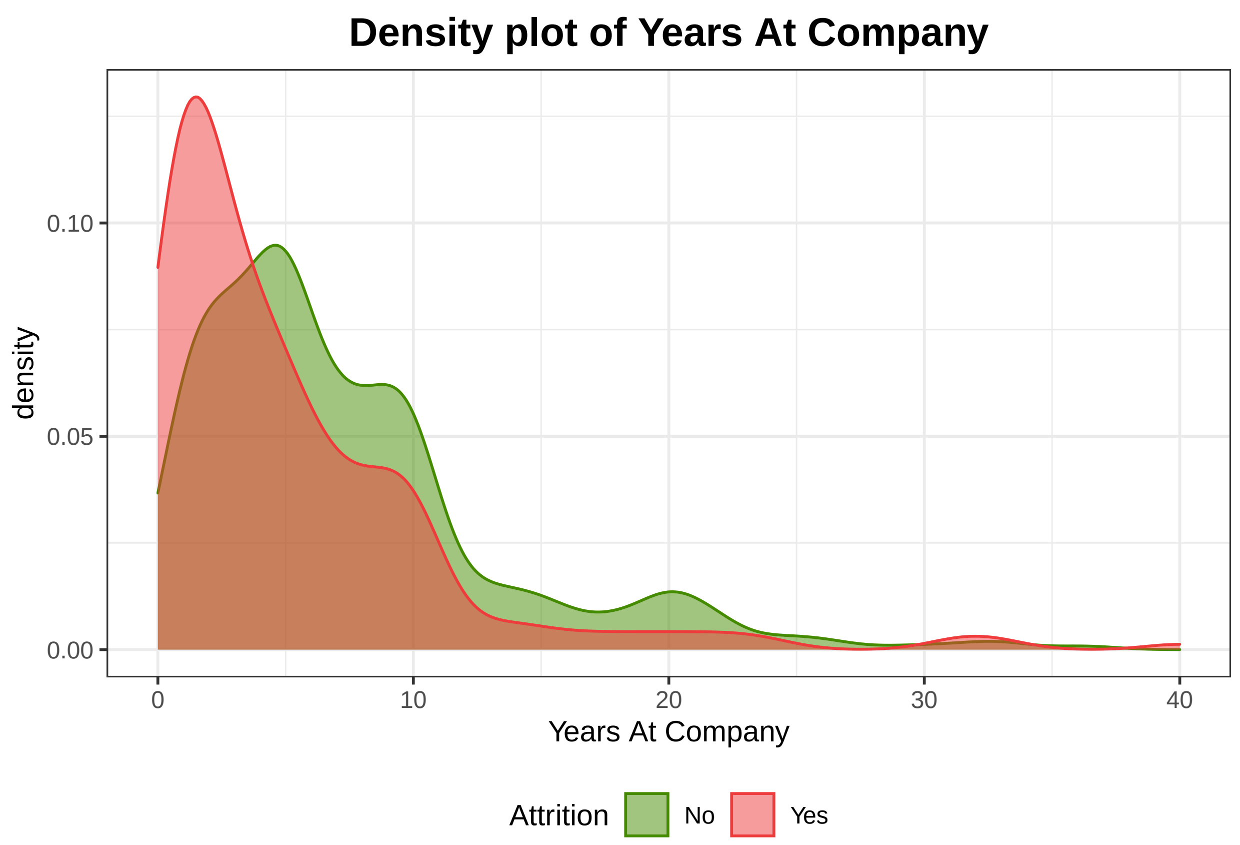
```
# Over Time worked by Attrition
data %>% group_by(Attrition, OverTime) %>% summarize(N = n()) %>% mutate(countT = sum(N)) %>%
  group_by(Attrition, OverTime, add=TRUE) %>% mutate(per=paste0(round(100*N/countT,1),'%')) %>%
  ggplot(aes(x=Attrition, y=N, fill=OverTime)) +
  geom_bar(stat="identity", position=position_dodge()) +
  theme_bw() +
  scale_fill_brewer(palette="Set2") +
  geom_text(aes(label = per), size = 4, vjust = 1.2, color = "#FFFFFF", position = position_dodge(0.9))
  ggtitle("Over Time by Attrition") +
  theme(plot.title = element_text(face = "bold", hjust = 0.5, size = 15)) +
  labs(x = "Attrition", y = "Count")
```



A larger proportion of employees who worked overtime has left the company.

Attrition by Years At Company

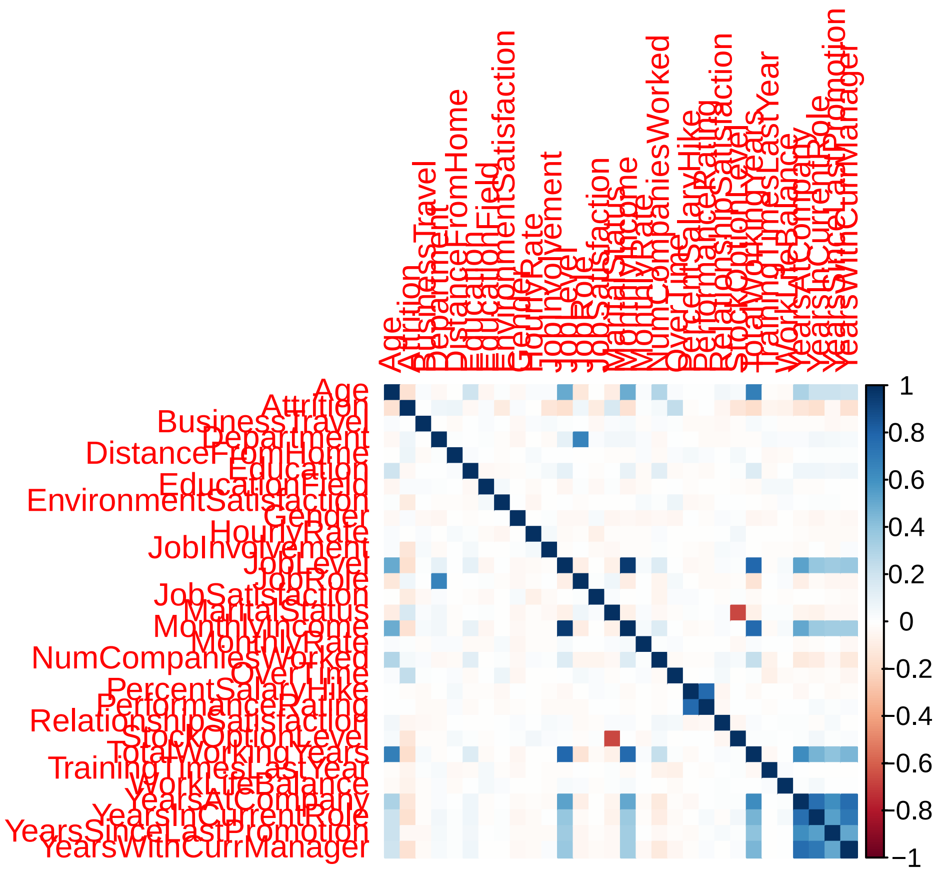
```r
# Attrition by Years At Company
ggplot(data, aes(x=YearsAtCompany, fill=Attrition, color=Attrition)) +
  geom_density(position="identity", alpha=0.5) +
  theme_bw() +
  scale_fill_manual(values=c("chartreuse4", "brown2")) +
  scale_color_manual(values=c("chartreuse4", "brown2")) +
  ggtitle("Density plot of Years At Company") +
  theme(plot.title = element_text(face = "bold", hjust = 0.5, size = 15), legend.position="bottom") +
  labs(x = "Years At Company")
```



Employees who have spent 10 years or more in the company have the least tendency of leaving.

A correlation plot is also plotted using the corrplot function to visualize the correlation among the attributes.

```r
# correlation between the attributes
corrplot(cor(sapply(data,as.integer)),method = "color")
```



Data Transformation

We will convert all the variables to numerical but keeping the target variable Attrition as factor.

```
# convert every variables into numeric
data <- data %>% mutate_if(is.factor, as.integer)

# convert the target variable Attrition to factor
data$Attrition=ifelse(data$Attrition==1,"No","Yes")
data$Attrition <- factor(data$Attrition)
```

We can see the types of all the attributes as:

```
# list types for each attribute
sapply(data, class)
```

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

It can be observed that all the attributes are converted into numerical type except Attrition.

Split Data into Train and Test Sets

We will split our dataset into train set and test set. 80% of our data will be the train set and the rest 20% will be our test set.

```
## Create data parition with 80% as training and 20% as testing
set.seed(1, sample.kind="Rounding")
train_indices <- createDataPartition(data$Attrition
                                    ,p = 0.8
                                    ,list = F)
data_train <- data[train_indices,]
data_test <- data[-train_indices,]
```

Machine Learning Data Models

AUC - Area Under Curve

We are going to plot the ROC curve and calculate the AUC (area under the curve) which are typical performance measurements for a binary classifier. The ROC is a curve generated by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings while the AUC is the area under the ROC curve. As a rule of thumb, a model with good predictive ability should have an AUC closer to 1 (1 is ideal) than to 0.5

We'll use caret::train() to fit the models and provide a method to implement the corresponding model. To modify the resampling method, a trainControl function is used. The option method controls the type of resampling. Here we will use 5 fold cross validation method.

To test the data we will use the inbuilt predict function.

Model 1: Logistic Regression

Logistic regression is a method for fitting a regression curve, y = f(x), when y is a categorical variable. The typical use of this model is predicting y given a set of predictors x. The predictors can be continuous, categorical or a mix of both. the glm() function, which is generally used to fit generalized linear models, will be used to fit the logistic regression model.

```r
# MODEL 1: Logistic Regression

# define the cross validation
tr_control <- trainControl(method = 'cv', number = 5, classProbs = T)

# fit Logistic regression Model
logR_model = train(Attrition~., data=data_train
                     , method="glm"
                     , family = "binomial"
                     , trControl=tr_control)

# predict using the Logistic Regression Model
predict_logR <- predict(logR_model
                         ,newdata = data_test
                         ,type = 'raw')

# evaluate the model
confusionMatrix(as.factor(predict_logR), as.factor(data_test$Attrition))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  236  30
##        Yes  10  17
##
##                Accuracy : 0.8635
##                  95% CI : (0.8188, 0.9006)
##     No Information Rate : 0.8396
##     P-Value [Acc > NIR] : 0.149962
##
##                   Kappa : 0.3878
##
##  Mcnemar's Test P-Value : 0.002663
##
##             Sensitivity : 0.9593
##             Specificity : 0.3617
##          Pos Pred Value : 0.8872
##          Neg Pred Value : 0.6296
##              Prevalence : 0.8396
##          Detection Rate : 0.8055
##    Detection Prevalence : 0.9078
##       Balanced Accuracy : 0.6605
##
##        'Positive' Class : No
##
```

```r
LogR_Model_auc <- auc(as.numeric(data_test$Attrition), as.numeric(predict_logR))
LogR_Model_auc
```

```
## Area under the curve: 0.6605
```

We will now create a results table to add this AUC. We will continue to add the AUC's of different models to this same results table.

```r
# add auc results in a table
auc_results <- data.frame(method = "LogR_Model", auc = as.numeric(LogR_Model_auc))
auc_results %>% knitr::kable()
```

method	auc
LogR_Model	0.6605259

We have obtained AUC of 0.66 with this model. Let's look into other models for a better AUC.

Model 2: Random Forest

Random forests are a modification of bagged decision trees that build a large collection of de-correlated trees to further improve predictive performance.

```r
# MODEL 2: Random Forest

# define the cross validation
tr_control <- trainControl(method = 'cv', number = 5, classProbs = T)

# fit the Basic Random Forest Model
RF_model_1 = train(Attrition~., data=data_train
                   , method="rf"
                   , trControl=tr_control)


# predict using the Basic Random Forest Model
predict_RF1 <- predict(RF_model_1
                       ,newdata = data_test
                       ,type = 'raw')

# evaluate the model
confusionMatrix(as.factor(predict_RF1), as.factor(data_test$Attrition))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  243  32
##        Yes   3  15
##
##                Accuracy : 0.8805
##                  95% CI : (0.8378, 0.9154)
##     No Information Rate : 0.8396
##     P-Value [Acc > NIR] : 0.03008
##
##                   Kappa : 0.409
##
##  Mcnemar's Test P-Value : 2.214e-06
##
##             Sensitivity : 0.9878
##             Specificity : 0.3191
##          Pos Pred Value : 0.8836
##          Neg Pred Value : 0.8333
##              Prevalence : 0.8396
##          Detection Rate : 0.8294
##    Detection Prevalence : 0.9386
##       Balanced Accuracy : 0.6535
##
##        'Positive' Class : No
##
```

```
RF_Model_1_auc <- auc(as.numeric(data_test$Attrition), as.numeric(predict_RF1))
RF_Model_1_auc
```

```
## Area under the curve: 0.6535
```

```
# add auc results in a table
auc_results <- bind_rows(auc_results,
                         data_frame(method="RF_Model",
                                    auc = as.numeric(RF_Model_1_auc)))
auc_results %>% knitr::kable()
```

method	auc
LogR_Model	0.6605259
RF_Model	0.6534769

The AUC with this model is around 0.65. The AUC is not improved from the previous model.

When we look the attrition distribution in the dataset, we see that more people stayed in the company than the people who left the company. That means, our dataset is an imbalanced dataset.

```
# Attrition Distribution
#1. Get the attrition class count
attrition_count <- table(data$Attrition)
attrition_count
```

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

A data set that exhibits an unequal distribution between its classes is considered to be imbalanced. An imbalanced dataset will bias the prediction model towards the more common class. So we will make our dataset balanced to improve the performance of the models.

SMOTE to create balanced dataset

A combination of over- and under-sampling is often successful to balance a dataset and a common approach is Synthetic Minority Over-Sampling Technique, or SMOTE. So, we will use the function SMOTE to make our imbalanced dataset to a balanced one.

```r
## use SMOTE to create balanced dataset

#1. Get the attrition class count
attrition_count <- table(data$Attrition)
attrition_count
```

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

```r
#2. Compute oversampling
over_count <- ((0.6 * max(attrition_count)) - min(attrition_count)) / min(attrition_count)

#3. Compute under sampling
under_count <- (0.4 * max(attrition_count)) / (min(attrition_count) * over_count)

over <- round(over_count,1) * 100
under <- round(under_count, 1) * 100

smote_data <-   SMOTE(Attrition ~.
                              ,data
                              ,perc.over = over
                              , k = 5
                              , perc.under = under)
```

We can see that our dataset is now balanced.

```r
# check the balanced dataset
table(smote_data$Attrition)
```

```
##
##  No Yes
## 474 711
```

Split Balanced Data into Train and Test Sets

```r
# create data parition with 80% as training and 20% as testing
set.seed(1, sample.kind="Rounding")
smote_train_indices <- createDataPartition(smote_data$Attrition
                                     ,p = 0.8
                                     ,list = F)
smote_data_train <- smote_data[smote_train_indices,]
smote_data_test <- smote_data[-smote_train_indices,]
```

Model 3: Random Forest using balanced dataset

```r
# MODEL 3: Random Forest using balanced dataset

# define the cross validation
tr_control <- trainControl(method = 'cv', number = 5, classProbs = T)

# fit the Random Forest Model
RF_model_2 = train(Attrition~., data=smote_data_train
                                    , method="rf"
                                    , trControl=tr_control)


# predict using the Random Forest Model
predict_RF2 <- predict(RF_model_2
          ,newdata = smote_data_test
          ,type = 'raw')

# evaluate the model
confusionMatrix(as.factor(predict_RF2), as.factor(smote_data_test$Attrition))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No   80  17
##        Yes  14 125
##
##                Accuracy : 0.8686
##                  95% CI : (0.8188, 0.909)
##     No Information Rate : 0.6017
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7274
##
##  Mcnemar's Test P-Value : 0.7194
##
##             Sensitivity : 0.8511
##             Specificity : 0.8803
##          Pos Pred Value : 0.8247
##          Neg Pred Value : 0.8993
##              Prevalence : 0.3983
##          Detection Rate : 0.3390
##    Detection Prevalence : 0.4110
##       Balanced Accuracy : 0.8657
##
##        'Positive' Class : No
##
```

```r
RF_Model_2_auc <- auc(as.numeric(smote_data_test$Attrition), as.numeric(predict_RF2))
RF_Model_2_auc
```

```
## Area under the curve: 0.8657
```

```
# add auc results in a table
auc_results <- bind_rows(auc_results,
                         data_frame(method="RF_Model_balanced",
                                    auc = as.numeric(RF_Model_2_auc)))
auc_results %>% knitr::kable()
```

method	auc
LogR_Model	0.6605259
RF_Model	0.6534769
RF_Model_balanced	0.8656728

The AUC of the Random Forest model has increased tremendously after making the dataset balanced.

Model 4: Extreme Gradient Boosting using balanced dataset

Extreme Gradient Boosting (xgboost) is similar to gradient boosting framework but more efficient. It has both linear model solver and tree learning algorithms and a capacity to do parallel computation on a single machine.

```
# MODEL 4: Extreme Gradient Boosting using balanced dataset

# define the cross validation
tr_control <- trainControl(method = 'cv', number = 5, classProbs = T)
xgbm_grid <- expand.grid(
  max_depth = 12
  , subsample = 0.9
  , nrounds   = 200
  , gamma   = 0.01
  , colsample_bytree = 0.5
  , min_child_weight = 1
  , eta = 0.02
)

# fit the Extreme GBM Model
XGBM_model = train(Attrition~., data=smote_data_train
                                    , method="xgbTree"
                                    , verbose = F
                                    , trControl=tr_control
                                    , tuneGrid = xgbm_grid)

# predict using the XGBM Model
predict_XGBM <- predict(XGBM_model
          ,newdata = smote_data_test
  )

# evaluate the model
confusionMatrix(as.factor(predict_XGBM), as.factor(smote_data_test$Attrition))
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction  No Yes
##        No   82  16
##        Yes  12 126
##
##                Accuracy : 0.8814
##                  95% CI : (0.8331, 0.9197)
##     No Information Rate : 0.6017
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7542
##
##  Mcnemar's Test P-Value : 0.5708
##
##             Sensitivity : 0.8723
##             Specificity : 0.8873
##          Pos Pred Value : 0.8367
##          Neg Pred Value : 0.9130
##              Prevalence : 0.3983
##          Detection Rate : 0.3475
##    Detection Prevalence : 0.4153
##       Balanced Accuracy : 0.8798
##
##        'Positive' Class : No
##
```

```
xgbm_auc <- auc(as.numeric(smote_data_test$Attrition), as.numeric(predict_XGBM))
xgbm_auc
```

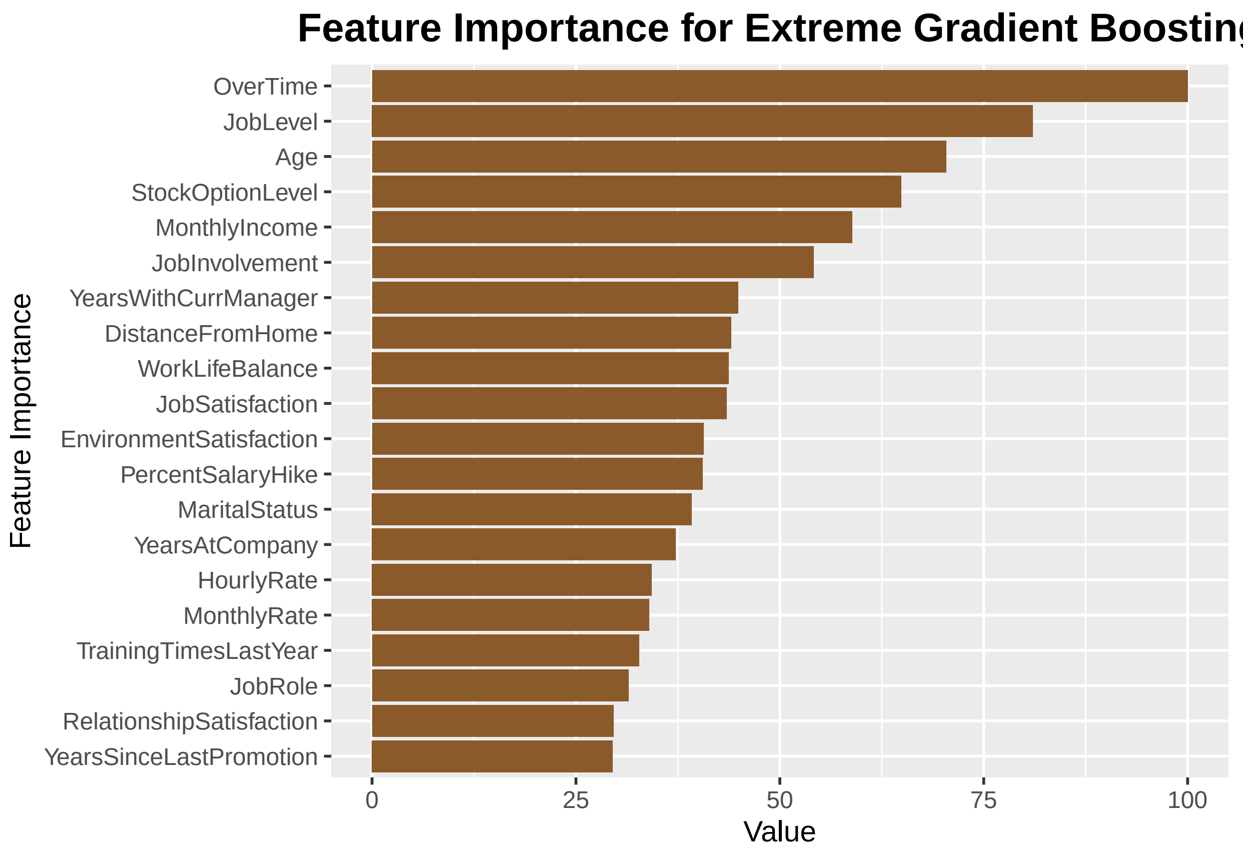
```
## Area under the curve: 0.8798
```

```
# add auc results to the table
auc_results <- bind_rows(auc_results,
                         data_frame(method="XGBM_Model",
                                    auc = as.numeric(xgbm_auc)))
auc_results %>% knitr::kable()
```

method	auc
LogR_Model	0.6605259
RF_Model	0.6534769
RF_Model_balanced	0.8656728
XGBM_Model	0.8798322

We have attained an AUC of 0.879 with this model.

Further, the Extreme Gradient Boosting model provides the estimates of feature importance from the trained predictive model. The model can retrieve importance scores for each attribute.

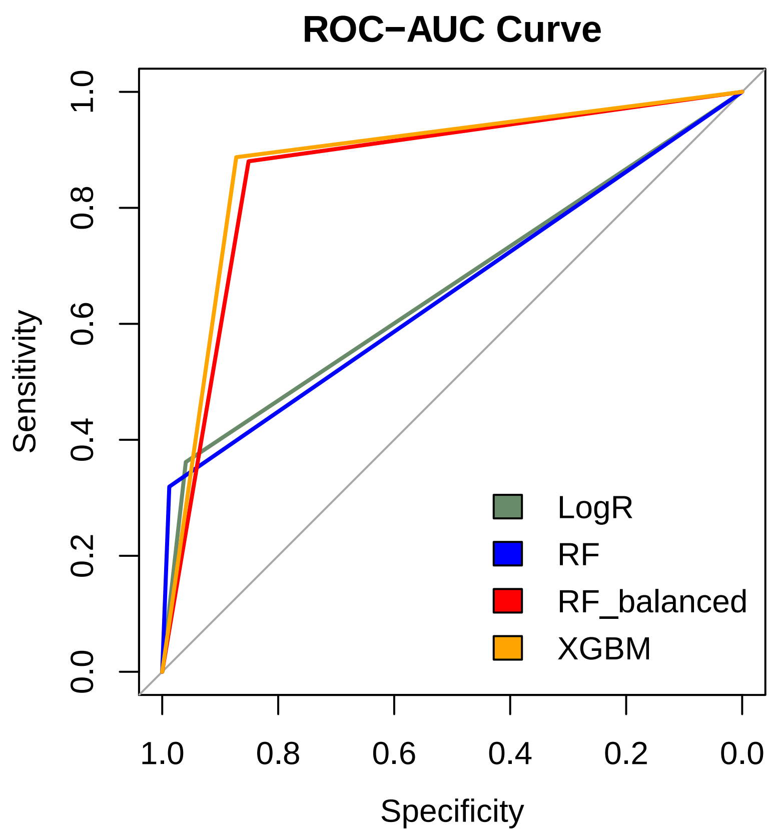


The Feature Importance of Extreme Gradient Boosting indicates that the attributes OverTime, JobLevel, Age, StockOptionLevel and MonthlyIncome played an important role in determining the attrition.

Results

We can inspect the AUCs for the various models trained as follows:

method	auc
LogR_Model	0.6605259
RF_Model	0.6534769
RF_Model_balanced	0.8656728
XGBM_Model	0.8798322



From this, we can infer that the AUCs of each model is an improvement from the previous models and the Extreme Gradient Boosting model has the highest AUC of about approximately 0.88.

Conclusion

We implemented four data models to uncover the factors that lead to employee attrition. The highest AUC was achieved with the Extreme Gradient Boosting model and the AUCs of the models improved as we balanced the data. Further, the Extreme Gradient Boosting model also helped listing out the driving factors for attrition. An organization can thus reduce the employee attrition rate by focusing their attention on these key factors.

Future Impovement

Further improvements in the AUCs can be achieved by ensembling methods through combining the results from well performing models. Also, the inclusion of attrition timeline, demographic information etc. can further help to draw additional inferences.

References

- https://rafalab.github.io/dsbook/introduction-to-machine-learning.html
- https://www.kaggle.com/pavansubhasht/ibm-hr-analytics-attrition-dataset