USE CASE STUDY REPORT

Student Name: Joseph Lynch

Executive Summary:

The purpose of my study is to understand the factors that contribute to an employees decision to leave an organization and accurately predict the likelihood of individual employees leaving in the future. my data comes from the IBM Data Science team who created mock data from actual employee survey responses and personal data within their own organization. I began by categorizing my character data into indicators for preprocessing, then did an analysis of the outcome variable using a histogram. Next, I did a heatmap analysis to understand the correlations that each factor had with my outcome variable.

The problem is a classification problem at its core, so I used different supervised machine learning techniques that fit appropriately to my problem. I used a Feedforward Neural Network, Naive Bayes and Random Forest algorithm to attempt to accurately predict employees who were likely to leave from my validation data sets.

What I found was that there were certain factors that contribute significantly to an employees decision to leave more than the other predictors. These included job satisfaction, whether an employee works overtime, years in their current role, years since last promotion and number of companies worked previously among others.

My recommendation is that employees who have these particular qualities should be the main focus of any employee retention efforts, as these are the employees who are most likely to leave. Anticipating which employees are more likely to leave and devoting resources to making efforts for them to stay is far less expensive than hiring/onboarding/training new ones.

I. Background and Introduction

One of the most costly endeavors an organization goes through is hiring, onboarding and training new employees. Because of this, it's in every organization's best interest to increase their retention rates by as much as possible.

Oftentimes, an employee does not choose to leave an organization for one reason alone. It almost always derives from a confluence of factors. A long commute, lack of career direction and frequent travel among other things do not occur in a vacuum. It's important to understand what factors go into an employee making the decision to stay or leave. Whether it's a question of salary, job satisfaction, tenure, I would like to understand what are the most important factors that contribute to this type of decision.

Using a data set created by IBM's data science team to mirror actual employee Human Resource data, I will build a model that is able to predict whether an employee is going to leave. Based on the data I have available to us, I believe that I can create a model that will be able to predict whether an employee will likely stay or leave their current position.

II. Data Exploration and Visualization

To begin my analysis, I took a look at all of the data available using the "str" function in R to see which types of variables I can utilize. I noticed that my predictors were a mix of both numerical and categorical variables. I also noticed that I could filter out several variables that provided us with redundant information.

To simplify my analysis, I converted the categorical variables into numerical indicators using for loops so that my data exploration/correlation analysis would be simpler.

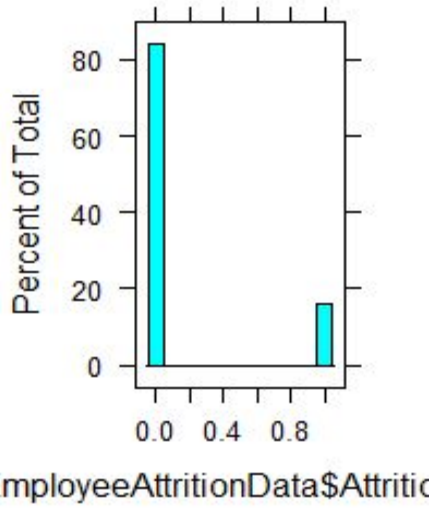
Additionally, I used a histogram to get a sense of the distribution of my output variable(ie employee attrition) to understand the overall distribution of my variable of interest.

III. Data Preparation and Preprocessing

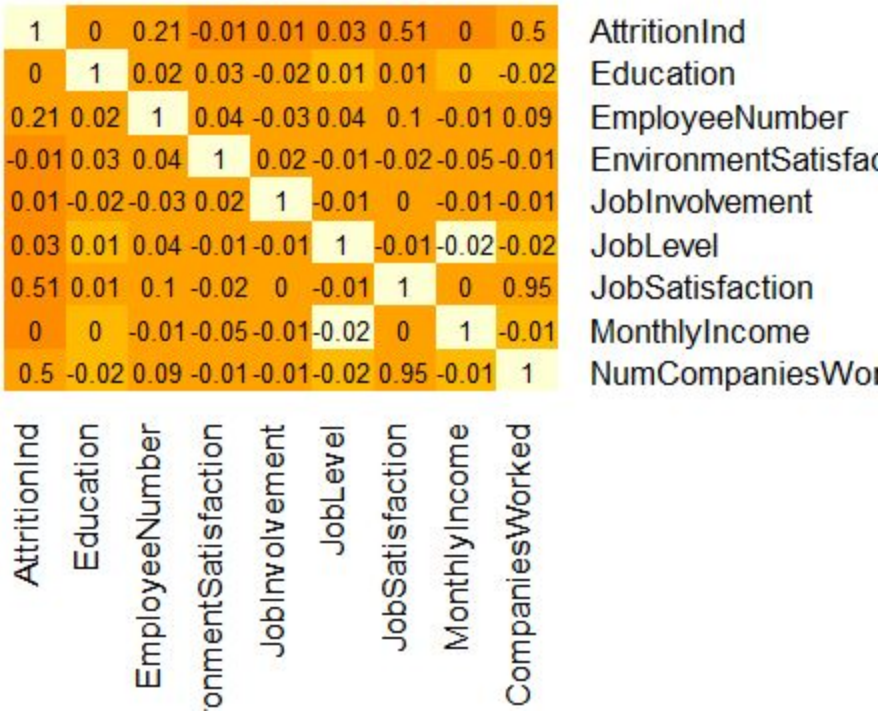
My ultimate goal to start is to understand the most important factors that contribute to an employee's decision to leave an organization in order to more accurately predict people who are likely to leave in order to convince them to stay. The reason being that hiring and onboarding employees is costly, whereas it is far less expensive to retain existing employees in addition to not losing their specific skills/knowledge.

I began my analysis by taking a comprehensive look at the data set. One of the first things I noticed was that there was a significant amount of redundant information in my set that could be removed. Monthly Income, Monthly Rate, Hmyly Rate and and Daily Rate all explain the same thing so I kept monthly pay rate. Employee count was always one, over18 was always Yes and standard hmys was always 40 so I removed them as well.

my main output variable is "Attrition". Namely, whether the employee left. Clearly, my problem is a classification problem based on the fact that I are attempting to protect whether an employee will "Stay" or "Leave". Additionally, my output of interest "Leave", is comparatively rare to employees who stay. (0=Stay, 1=Leave)



To gain a clearer picture of how my data interacts, I converted my character data into indicators using for loops and plotted each variable to Attrition using heat maps(example below).



Some of the stronger correlation variables include Job Satisfaction and interestingly the number of Companies Worked. I found that in addition, variables such as Overtime and Gender had strong correlations with attrition.

IV. Data Mining Techniques and Implementation

To summarize my problem, I would like to classify employees by their likelihood to leave an organization so that I can predict in the future who is likely to leave. This will be a supervised learning classification problem where my outcome of interest is comparatively rare.

Because of the nature of my problem, where classification is key, I also chose to implement a Random Forest algorithm for testing. The simplicity of the algorithm and the ability to lower the probability of any single classification make it an excellent fit for my problem. It also fits well due to the Random Forest's ability to tell us which variables decrease it's accuracy the most. In my case, this means I can understand which factors contribute to an employee's decision to Leave the most.

I also believed the best choices were the Naive Bayes algorithm and a Feedforward Neural Network. The Naive Bayes algorithm's simplicity and good classification performance are beneficial within the context of my problem because I need an algorithm

that can handle a categorical output variable(“Stay” or “Leave”) while also ranking each record's individual probability of belonging to a certain class(“Stay” or “Leave”). This is because in most cases people will be more likely to Stay rather than Leave, although the probability of leaving will be much higher for certain subsets of people.

Since there are many different factors that go into a decision to leave, they can often be interrelated. Nevertheless, it’s extremely important to be able to predict these decisions. This is why I chose the feedforward Neural Network. The Neural Network’s ability to cut through noisy data and identify complicated relationships along with its strong predictive performance are well suited to my problem.

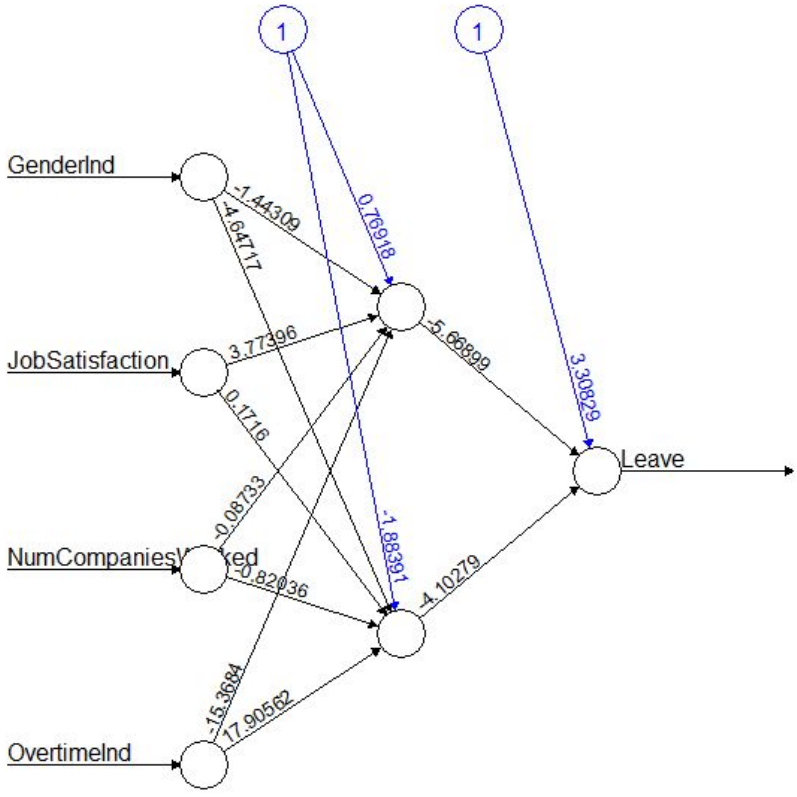
V. Performance Evaluation

I began testing the performance of my variables through a Feedforward Neural Network algorithm. I began by splitting the data into training and validation sets(75% training, 25% validation). Next, I set my Neural Network to have “Leave” as the output variable of interest and my categorical predictors to Gender, ‚Job Satisfaction, Number of Companies Worked and whether the employee works overtime.

```
library(dplyr)
EmployeeAttritionData$ID <- 1:nrow(EmployeeAttritionData)
train <- EmployeeAttritionData %>% dplyr::sample_frac(.75)
validation  <- dplyr::anti_join(EmployeeAttritionData, train, by = 'ID')

nn <- neuralnet(Leave ~ GenderInd+JobSatisfaction+NumCompaniesWorked +
OvertimeInd, linear.output = F ,data = train, hidden = 2)
```

I then plotted the weights and biases of my network as shown below -



Error: 66.193933 Steps: 3099

To get a better sense of the performance of my model, I tested it against the validation set. First, the output of "Leave" is far less likely than the likelihood of employees to "Stay". With this in mind, I decided on a cutoff over 10% for a propensity to leave, due to the fact that it is far more important to correctly classify people who leave than misclassifying people who stay as leaving.

```
library(caret)
predict <-
neuralnet::compute(nn,data.frame(validation$NumCompaniesWorked,validation$JobSati
sfaction,validation$GenderInd,validation$OvertimeInd))
predicted.class = ifelse(predict$net.result[,1]>.1,1,0)
```

I evaluated the performance of my prediction with a confusion matrix table seen here -

(Predicted class of 1= Leave)

my table has an accuracy rating of roughly 40% overall, but what I lack in overall accuracy I make up for in my successful categorization of my output variable of interest. The neural network model accurately predicts 69.84% of the overall employees who are likely to leave.

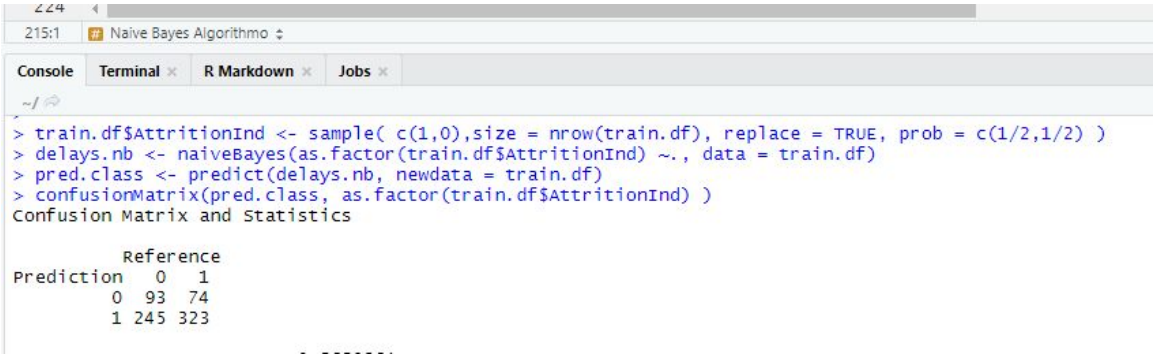
Next, I used the Naive Bayes algorithm to test my categorical predictors to Gender, Job Satisfaction, Number of Companies Worked. Since my outcome variable is not evenly split between "Stay" and "Leave", I will need to reweight my initial training set to be 50/50 to appropriately train my Naive Bayes algorithm.

```
selected.var <- c(22,10, 8,26)
train.index <- sample(c(1:dim(EmployeeAttritionData)[1]),
dim(EmployeeAttritionData)[1]*0.5)
train.df <- EmployeeAttritionData[train.index, selected.var]
valid.df <- EmployeeAttritionData[-train.index, selected.var]

train.df$AttritionInd <- sample( c(1,0),size = nrow(train.df), replace = TRUE, prob =
c(1/2,1/2) )
```

I then implemented my Naive Bayes algorithm on my training data set and measured it's performance.

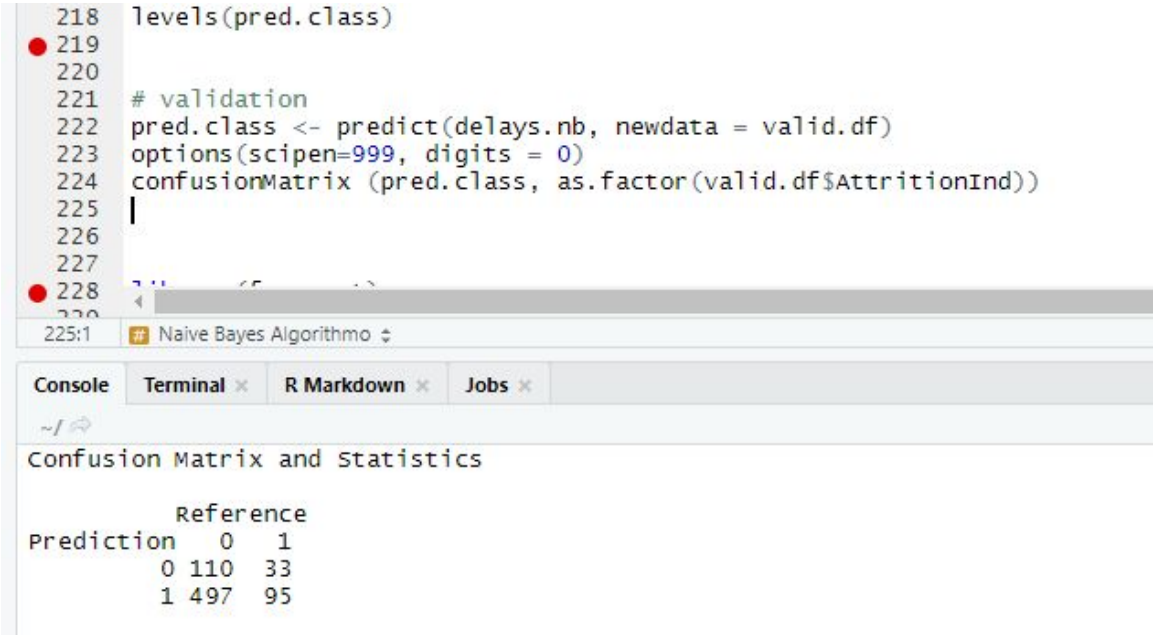
```
delays.nb <- naiveBayes(as.factor(train.df$AttritionInd) ~., data = train.df)
pred.class <- predict(delays.nb, newdata = train.df)
confusionMatrix(pred.class, as.factor(train.df$AttritionInd) )
```



My training model accurately categorizes 81% of the "Leave" instances, but was likely due to some overfitting of the training data as well as the fact that the training data oversampled "Leave" instances.

Since my validation model is unweighted, I tested my prediction model on that data set. -

pred.class <- predict(delays.nb, newdata = valid.df)
options(scipen=999, digits = 0)
confusionMatrix (pred.class, as.factor(valid.df$AttritionInd))



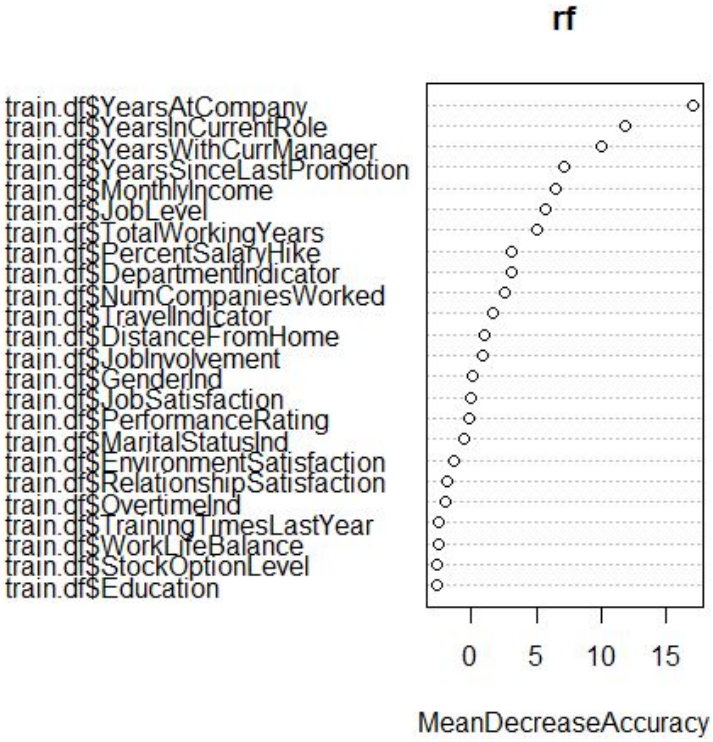
My prediction against the validation set categorizes 74.2% of the "Leave" instances accurately, which is worse than the training set but this to be expected due to the unweighted sample and non-fitted data sample.

Finally, I created a model for employee attrition using the Random Forest algorithm. First, I created a training and validation test data set and reweighted to training data to reflect a 50/50 split between Leave and Stay.

```
train.index <- sample(c(1:dim(EmployeeAttritionData)[1]),
dim(EmployeeAttritionData)[1]*0.5)
train.df <- EmployeeAttritionData[train.index, ]
valid.df <- EmployeeAttritionData[-train.index, ]
train.df$AttritionInd <- sample( c(1,0),size = nrow(train.df), replace = TRUE, prob =
c(1/2,1/2) )
train.df$Leave <- train.df$AttritionInd==1
train.df$Stay <- train.df$AttritionInd==0
valid.df$Leave <- valid.df$AttritionInd==1
valid.df$Stay <- valid.df$AttritionInd==0
```

Next, I ran the algorithm using every variable in my training set. From there, I wanted to understand which variables were the most important predictors and remove the rest to reduce dimensionality.

```
varImpPlot(rf, type = 1)
```



I then removed the variables after the cutoff of Total Working Years and ran the Random Forest algorithm again. Next, I tested my model against the validation set and created a confusion matrix to measure performance.

rf.pred <- predict(rf, valid.df$Leave)
confusionMatrix(rf.pred,as.factor(valid.df$Leave))

```
> rf.pred <- predict(rf, valid.df$Leave)
> confusionMatrix(rf.pred,as.factor(valid.df$Leave))
Confusion Matrix and Statistics

          Reference
Prediction FALSE TRUE
     FALSE   142   22
     TRUE    470  101

               Accuracy : 0.3306
                 95% CI : (0.2967, 0.3659)
    No Information Rate : 0.8327
    P-Value [Acc > NIR] : 1
```

(True=Leave,False=Stay)

I can see that my Random Forest algorithm predicts 82.11% of employees who are likely to leave based on the above factors(Years In Current Role, Years Since Last Promotion,Years At Company,Years With Current Manager,Monthly Income,Total Working Years) accurately.

VI. Discussion and Recommendation

I can understand from my analysis and results that while there is no singular determining factor that makes an employee decide to stay to leave, there are certainly strong indicators that employers should look for in order to identify potential attrition.

For example, my Random Forest algorithm, the strongest indicators of a person's propensity to leave an organization included their tenure in their current role and how long it has been since their last promotion. An organization can deduce from this information that employees who feel stuck in their current role are more likely to leave.

A potential solution would be to preemptively have conversations and take actions with those particular employees about steps they can take in order to grow their careers. Keeping this type of long standing employee will be beneficial to an organization.

In my Naive Bayes/Neural Network algorithms, I found that an employee's job satisfaction, number of companies worked and how much overtime worked were also significant indicators of employee attrition. Employees who feel overworked are more

likely than others to want to leave, and employees who have worked in many different organizations are also more likely to leave. An organization would benefit greatly by identifying which of their employees have these types of roles where significant overtime is required and who have worked at many different companies. This way, they can focus their time and effort on trying to retain employees more likely to leave, and less so on employees that are staying regardless.

VII. Summary

Given that employee hiring/onboarding/training are extremely expensive, it is important for organizations to understand which factors contribute to an employees decision to stay or leave. Using different supervised machine learning techniques, I attempted to solve this classification problem by utilizing different algorithms that could determine the key factors that contribute to the decision to leave and make accurate predictions as to whether an employee was likely to leave.

I used a Feedforward Neural Network, Naive Bayes and Random Forest algorithm to more accurately predict employee attrition for my output variable outcome of interest(ie Leave). What I found was there are several meaningful indicators such as Overtime, Years worked under current manager, years in current role, etc. that play a large role in an employees decision whether to leave. As such, organizations should focus their efforts more specifically on retaining these employees who I can predict are more likely to leave than those who are more likely to stay. In this way, an organization can spend its time and resmyces more efficiently and save on the substantial costs of hiring/onboarding/training.

Appendix: R Code for use case study

```
#Project Code Solution Design
library(readxl)
library(carData)
library(car)
library(caret)
library(bootstrap)

histogram(EmployeeAttritionData$AttritionInd)


EmployeeAttritionData <-
read.csv("C:/Users/Administrator/Documents/WA_Fn-UseC_-HR-Employee-Attrition.csv")
```

```
str(EmployeeAttritionData)

EmployeeAttritionData$AttritionInd <- rep(0, 1470)
for(i in 1:1470) {if (EmployeeAttritionData$Attrition[i] %in% c("Yes"))
EmployeeAttritionData$AttritionInd[i] <- 1}

EmployeeAttritionData$TravelIndicator <- rep(0, 1470)
for(i in 1:1470) {if (EmployeeAttritionData$BusinessTravel[i] %in% c("Non-Travel"))
EmployeeAttritionData$TravelIndicator[i] <- 0
if (EmployeeAttritionData$BusinessTravel[i] %in% c("Travel_Rarely"))
EmployeeAttritionData$TravelIndicator[i] <- 1
if (EmployeeAttritionData$BusinessTravel[i] %in% c("Travel_Frequently"))
EmployeeAttritionData$TravelIndicator[i] <- 2}


EmployeeAttritionData$DepartmentIndicator <- rep(0, 1470)
for(i in 1:1470) {if (EmployeeAttritionData$Department[i] %in% c("Sales"))
EmployeeAttritionData$DepartmentIndicator[i] <- 1
if (EmployeeAttritionData$Department[i] %in% c("Research & Development"))
EmployeeAttritionData$DepartmentIndicator[i] <- 2
if (EmployeeAttritionData$Department[i] %in% c("Human Resmyces"))
EmployeeAttritionData$DepartmentIndicator[i] <- 3}


EmployeeAttritionData$EducationFieldIndicator <- rep(0, 1470)
for(i in 1:1470) {if (EmployeeAttritionData$EducationField[i] %in% c("Life Sciences"))
EmployeeAttritionData$EducationFieldIndicator[i] <- 6
if (EmployeeAttritionData$EducationField[i] %in% c("Other"))
EmployeeAttritionData$EducationFieldIndicator[i] <- 5
if (EmployeeAttritionData$EducationField[i] %in% c("Medical"))
EmployeeAttritionData$EducationFieldIndicator[i] <- 4
if (EmployeeAttritionData$EducationField[i] %in% c("Marketing"))
EmployeeAttritionData$EducationFieldIndicator[i] <- 3
if (EmployeeAttritionData$EducationField[i] %in% c("Human Resmyces"))
EmployeeAttritionData$EducationFieldIndicator[i] <- 2
if (EmployeeAttritionData$EducationField[i] %in% c("Technical Degree"))
EmployeeAttritionData$EducationFieldIndicator[i] <- 1}


EmployeeAttritionData$JobRoleIndicator <- rep(0, 1470)
for(i in 1:1470) {if (EmployeeAttritionData$JobRole[i] %in% c("Sales Executive"))
EmployeeAttritionData$JobRoleIndicator[i] <- 9
if (EmployeeAttritionData$JobRole[i] %in% c("Research Scientist"))
EmployeeAttritionData$JobRoleIndicator[i] <- 8
```

```
if (EmployeeAttritionData$JobRole[i] %in% c("Manufacturing Director"))
EmployeeAttritionData$JobRoleIndicator[i] <- 7
if (EmployeeAttritionData$JobRole[i] %in% c("Healthcare Representative"))
EmployeeAttritionData$JobRoleIndicator[i] <- 6
if (EmployeeAttritionData$JobRole[i] %in% c("Manager"))
EmployeeAttritionData$JobRoleIndicator[i] <- 5
if (EmployeeAttritionData$JobRole[i] %in% c("Sales Representative"))
EmployeeAttritionData$JobRoleIndicator[i] <- 4
if (EmployeeAttritionData$JobRole[i] %in% c("Research Director"))
EmployeeAttritionData$JobRoleIndicator[i] <- 3
if (EmployeeAttritionData$JobRole[i] %in% c("Human Resmyces"))
EmployeeAttritionData$JobRoleIndicator[i] <- 2
if (EmployeeAttritionData$JobRole[i] %in% c("Laboratory Technician"))
EmployeeAttritionData$JobRoleIndicator[i] <- 1}

EmployeeAttritionData$GenderInd <- rep(0, 1470)
for(i in 1:1470) {if (EmployeeAttritionData$Gender[i] %in% c("Female"))
EmployeeAttritionData$GenderInd[i] <- 1
if (EmployeeAttritionData$Gender[i] %in% c("Male"))
EmployeeAttritionData$GenderInd[i] <- 2}

EmployeeAttritionData$MaritalStatusInd <- rep(0, 1470)
for(i in 1:1470) {if (EmployeeAttritionData$MaritalStatus[i] %in% c("Single"))
EmployeeAttritionData$MaritalStatusInd[i] <- 1
if (EmployeeAttritionData$MaritalStatus[i] %in% c("Divorced"))
EmployeeAttritionData$MaritalStatusInd[i] <- 2
if (EmployeeAttritionData$MaritalStatus[i] %in% c("Married"))
EmployeeAttritionData$MaritalStatusInd[i] <- 3}


EmployeeAttritionData$OvertimeInd <- rep(0, 1470)
for(i in 1:1470) {if (EmployeeAttritionData$OverTime[i] %in% c("Yes"))
EmployeeAttritionData$OvertimeInd[i] <- 1}



unique(EmployeeAttritionData$Department)
unique(EmployeeAttritionData$EducationField)
unique(EmployeeAttritionData$JobRole)
unique(EmployeeAttritionData$Gender)
unique(EmployeeAttritionData$MaritalStatus)
unique(EmployeeAttritionData$OverTime)

#MonthlyIncome, Monthly Rate, Hmyly Rate and  and Daily Rate all say the same thing
so I will just keep monthly rate
```

```
#Employee count is always one, over18 is always Yes and standard hmys is always 80 so
I will remove them as well
#I will also remove all the of the character data and replace them with indicators
str(EmployeeAttritionData)
ncol(EmployeeAttritionData)
EmployeeAttritionData <- subset(EmployeeAttritionData, select =
-c(2,3,4,5,8,9,12,13,16,18,20,22,23,27))

data.frame(colnames(EmployeeAttritionData))

summary(EmployeeAttritionData)



######Data analysis

scatterplotMatrix(EmployeeAttritionData[,c(2,9,10,11,12,13,14,15,16,17,18)],pch=19)

library(psych)


pairs.panels(EmployeeAttritionData[,c(2,9,10,11,12,13,14,15,16,17,18)])


cor(EmployeeAttritionData$AttritionInd,EmployeeAttritionData[,c(11,13,14,15)])

str(EmployeeAttritionData)

head(EmployeeAttritionData)


cor(EmployeeAttritionData$JobSatisfaction,EmployeeAttritionData$AttritionInd)
#I can see a clear negative correlation between job satisfaction and attrition, as more
satisfied employees are less likely to leave

cor(EmployeeAttritionData$TravelIndicator,EmployeeAttritionData$AttritionInd)
#I also notice a positive correlation between travel and attrition, as employees who are
required to travel more frequently are more likely to leave

head(EmployeeAttritionData)

ncol(EmployeeAttritionData)

heatmap(cor(EmployeeAttritionData), Rowv = NA, Colv = NA)
```

```
library(gplots)
heatmap.2(cor(EmployeeAttritionData[,c(22,3,4,5,6,7,8,9,10)]), Rowv = FALSE, Colv =
FALSE, dendrogram = "none",
        cellnote = round(cor(EmployeeAttritionData),2),
        notecol = "black", key = FALSE, trace = 'none', margins = c(10,10))


heatmap.2(cor(EmployeeAttritionData[,c(22,11,12,13,14,15,16,17,18,19)]), Rowv =
FALSE, Colv = FALSE, dendrogram = "none",
        cellnote = round(cor(EmployeeAttritionData),2),
        notecol = "black", key = FALSE, trace = 'none', margins = c(10,10))

heatmap.2(cor(EmployeeAttritionData[,c(22,20,21,23,24,25,26)]), Rowv = FALSE, Colv
= FALSE, dendrogram = "none",
        cellnote = round(cor(EmployeeAttritionData),2),
        notecol = "black", key = FALSE, trace = 'none', margins = c(10,10))

heatmap.2(cor(EmployeeAttritionData[,c(22,27,28,29)]), Rowv = FALSE, Colv =
FALSE, dendrogram = "none",
        cellnote = round(cor(EmployeeAttritionData),2),
        notecol = "black", key = FALSE, trace = 'none', margins = c(10,10))


#########Neural Network application########

library(neuralnet)
library(nnet)
library(caret)

data.frame(colnames(train.index))


set.seed(1)  # set seed for reproducing the partition
train.index <- sample(c(1:1470), 600)
selected.var <- c(22,10, 8,26)
train.df <- EmployeeAttritionData[train.index, selected.var]
valid.df <- EmployeeAttritionData[-train.index, selected.var]


trainData <-
(EmployeeAttritionData$GenderInd,EmployeeAttritionData$JobSatisfaction)

EmployeeAttritionData$AttritionInd
```

```
EmployeeAttritionData$Leave <- EmployeeAttritionData$AttritionInd==1
EmployeeAttritionData$Stay <- EmployeeAttritionData$AttritionInd==0

#Companiesworked,#yearsatcompany,yearsincurrentrole,maritalstatus

EmployeeAttritionData$JobSatisfaction

library(dplyr)
EmployeeAttritionData$ID <- 1:nrow(EmployeeAttritionData)
train <- EmployeeAttritionData %>% dplyr::sample_frac(.75)
validation  <- dplyr::anti_join(EmployeeAttritionData, train, by = 'ID')

nn <- neuralnet(Leave ~ GenderInd+JobSatisfaction+NumCompaniesWorked +
OvertimeInd, linear.output = F ,data = train, hidden = 2)

nn$weights
prediction(nn)
plot(nn, rep="best")

library(caret)
predict <-
neuralnet::compute(nn,data.frame(validation$NumCompaniesWorked,validation$JobSati
sfaction,validation$GenderInd,validation$OvertimeInd))
predicted.class = ifelse(predict$net.result[,1]>.1,1,0)

table(predicted.class,validation$AttritionInd)

#########Neural Network application########


####Naive Bayes Algorithmo#######

library(e1071)

library(dplyr)

install.packages("survey")
library(survey)

EmployeeAttritionData

str(EmployeeAttritionData)

data.frame(colnames(EmployeeAttritionData))
```

```
selected.var <- c(22,10, 8,26)
train.index <- sample(c(1:dim(EmployeeAttritionData)[1]),
dim(EmployeeAttritionData)[1]*0.5)
train.df <- EmployeeAttritionData[train.index, selected.var]
valid.df <- EmployeeAttritionData[-train.index, selected.var]


train.df$AttritionInd <- sample( c(1,0),size = nrow(train.df), replace = TRUE, prob =
c(1/2,1/2) )

#GenderInd+JobSatisfaction+NumCompaniesWorked + OvertimeInd

delays.nb <- naiveBayes(as.factor(train.df$AttritionInd) ~., data = train.df)



library(caret)
# training
pred.class <- predict(delays.nb, newdata = train.df)
confusionMatrix(pred.class, as.factor(train.df$AttritionInd) )

#.8960 specificity on people who will likely leave, when including  Gender, Number of
companies worked, job satisfaction and gender

levels(pred.class)


# validation
pred.class <- predict(delays.nb, newdata = valid.df)
options(scipen=999, digits = 0)
confusionMatrix (pred.class, as.factor(valid.df$AttritionInd))



library(forecast)
valid.df <- data.frame(valid.df)
NROW(forecast(pred.class))
NROW(valid.df$AttritionInd)
pred.class <- as.numeric(pred.class)
valid.df$AttritionInd <- as.numeric(valid.df$AttritionInd)
accuracy(pred.class,valid.df$AttritionInd)
options(scipen=999, digits = 10)
```

```
#######Random forest#######
install.packages("randomForest")
library(randomForest)


#selected.var <- c(22,10, 8,26)
train.index <- sample(c(1:dim(EmployeeAttritionData)[1]),
dim(EmployeeAttritionData)[1]*0.5)
train.df <- EmployeeAttritionData[train.index, ]
valid.df <- EmployeeAttritionData[-train.index, ]

train.df$AttritionInd <- sample( c(1,0),size = nrow(train.df), replace = TRUE, prob =
c(1/2,1/2) )


train.df$Leave <- train.df$AttritionInd==1
train.df$Stay <- train.df$AttritionInd==0
valid.df$Leave <- valid.df$AttritionInd==1
valid.df$Stay <- valid.df$AttritionInd==0


EmployeeAttritionData[,c(1,2,3)]

## random forest
rf <- randomForest(as.factor(train.df$Leave) ~
train.df$DistanceFromHome+train.df$Education+train.df$EnvironmentSatisfaction+train
.df$JobInvolvement+train.df$JobLevel+train.df$JobSatisfaction+train.df$MonthlyIncom
e+train.df$NumCompaniesWorked+train.df$PercentSalaryHike+train.df$PerformanceRa
ting+train.df$RelationshipSatisfaction

+train.df$StockOptionLevel+train.df$TotalWorkingYears+train.df$TrainingTimesLastY
ear+train.df$WorkLifeBalance+train.df$YearsAtCompany+train.df$YearsInCurrentRole
+train.df$YearsSinceLastPromotion+train.df$YearsWithCurrManager+train.df$TravelIn
dicator+train.df$DepartmentIndicator+train.df$DepartmentIndicator+train.df$GenderInd
+train.df$MaritalStatusInd+train.df$OvertimeInd
                , v = train.df, ntree = 1000,
                mtry = 10, nodesize = 5, importance = TRUE)



rf <- randomForest(as.factor(train.df$Leave) ~
```

```
train.df$YearsInCurrentRole+train.df$YearsSinceLastPromotion+train.df$YearsAtComp
any
+train.df$YearsWithCurrManager+train.df$MonthlyIncome+train.df$TotalWorkingYear
s, v = train.df, ntree = 500,
 mtry = 4, nodesize = 5, importance = TRUE)



## variable importance plot
varImpPlot(rf, type = 1)

NROW(valid.df)
## confusion matrix
rf.pred <- predict(rf, valid.df$Leave)
confusionMatrix(rf.pred,as.factor(valid.df$Leave))
```