IBM HR Analytics Employee Attrition & Performance

Jesus Gastañaduy

June 21, 2020

Introduction

It is well known that human capital is one of the main assets of any company, if not the most important one. For this reason, it is wise to lower the attrition rate as much as possible. To do this, we must first know who are the employees that are quitting and what are the variables which might be explaining this behavior.

With this purpose in mind, we wanted to create an algorithm that could predict employee attrition. On this occasion, we worked with a fictional open dataset created by IBM data scientists. We first started by exploring the dataset.

The whole database contains 35 variables with 1470 observations. Some of the variables included are "Attrition", "Age", "Business Travel", "Environment satisfaction", "Monthly income", "Performance rating", "Total working years" among the most salient. It is worth mentioning that all variables were transformed into factors or numerical vectors, as corresponded, for analysis purposes.

As said, the goal of our machine learning model was to predict employee attrition. To do that we tried different algorithms and tested which performed better according to the most relevant evaluation metrics given the data.

Methods

We first wanted to know if our data was biased in some way. After visualizing the proportion of employees that either continued or left the company, it could be easily seen that our dataset was biased; with nearly 84% of employee continuity and over 16% of employee attrition.

```r
prop.table(table(tb$Attrition))
```

```
##
##         0         1
## 0.8387755 0.1612245
```

This unbalance was taken into account when evaluating our machine learning models.

To start with the creation of our algorithms, we first partitioned the whole dataset into two: one train set (80% of the data) and one test set (20% of the data). We carried all of our analyses on the train set; the test set was only used for validation purposes.

```r
##create train and validation sets
set.seed(1)
index<- createDataPartition(y= tb$Attrition, times = 1, p = 0.8, list = FALSE)
train_set<- tb[index, ]
test_set<- tb[-index, ]
```

Before the creation of any machine learning model, we did some pre-processing of our data on the train set. As there were many predictors, we looked for the ones which had near zero variance to remove them. We ended up removing the variables “Employee Count”, “Over 18” and “Standard Hours”. We updated our train set to only include the variables which had statistical variation.

```
##preprocessing train_set
nearZeroVar(tb)
```

```
## [1]  9 22 27
```

```
train_set<- train_set%>%
  select(-c("EmployeeCount", "Over18", "StandardHours"))
```

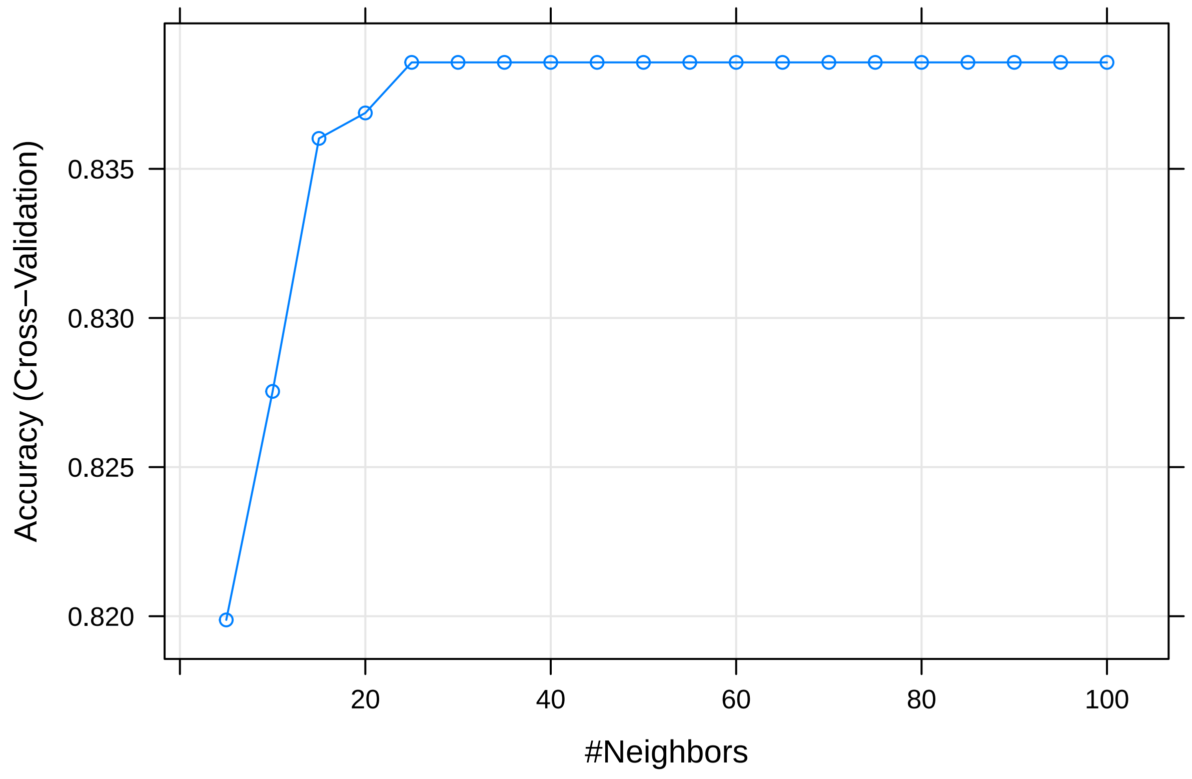
After doing the pre-processing of our data, we started by fitting some models.

k- nearest neighbors

```
##knn
set.seed(5)
train_knn<- train(Attrition ~ ., data = train_set,
                  method = "knn",
                  tuneGrid = data.frame(k = seq(5, 100, 5)),
                  trControl= trainControl(method = "cv", number = 10, p = .9))
```

We started with k-nearest neighbors. We chose attrition as our dependent variable. To optimize on our k parameter on the train set, we conducted ten-fold cross validation using 90% of the data. For replication purposes, we set a seed. The optimal k of this first model was 100. We did not include a larger number of k, as the plot below shows how the accuracy of the model depending on k remains constant as k is above 20 neighbors.

```
plot(train_knn)
```



By looking at the confusion matrix of this model we saw that any value was predicted as 0, which implied employee continuity.

```r
confusionMatrix(predict(train_knn, test_set), test_set$Attrition, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 246  47
##          1   0   0
##
##                Accuracy : 0.8396
##                  95% CI : (0.7925, 0.8797)
##     No Information Rate : 0.8396
##     P-Value [Acc > NIR] : 0.5388
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : 1.949e-11
##
##             Sensitivity : 0.0000
##             Specificity : 1.0000
##          Pos Pred Value :    NaN
##          Neg Pred Value : 0.8396
##              Prevalence : 0.1604
##          Detection Rate : 0.0000
##    Detection Prevalence : 0.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : 1
##
```

This may make sense, from a machine learning perspective, as the nearest neighbors of any point were employees who continued on their job (because of the high prevalence of this condition). However, a machine learning model that only predicts continuity on the job does not have any practical usage.

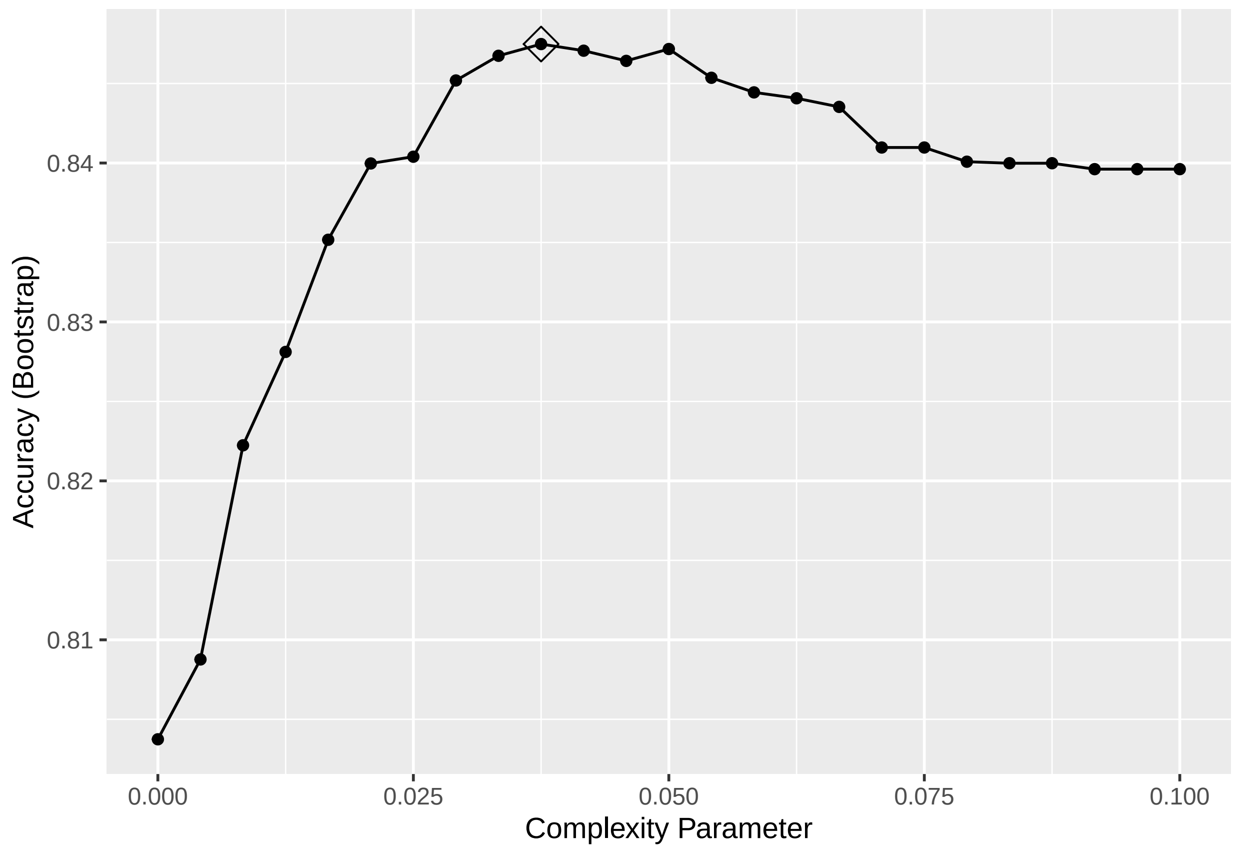
Classification tree

```
train_rpart<- train(Attrition ~ ., data= train_set,
                    method= "rpart",
                    tuneGrid = data.frame(cp = seq(0, 0.1, length.out = 25)))
```

We then tried to fit a classification tree to see if this model could make better predictions than the first one. In this case we optimized on the complexity parameter by creating a sequence of 25 numbers from 0 to 0.10, so we were able to identify the minimum improvement in accuracy for another partition to be made.

By plotting the complexity parameter values against their corresponding accuracy, we found that the parameter that optimized the classification tree was 0.0375. Higher values of the complexity parameter only diminished accuracy, so no more values were included in the optimization process.

```
ggplot(train_rpart, highlight = TRUE)
```



By looking at the confusion matrix however, we could observe that this model performed no much better than knn in accuracy terms.

```r
confusionMatrix(predict(train_rpart, test_set), test_set$Attrition, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 238  40
##          1   8   7
##
##                Accuracy : 0.8362
##                  95% CI : (0.7887, 0.8767)
##     No Information Rate : 0.8396
##     P-Value [Acc > NIR] : 0.6009
##
##                   Kappa : 0.1607
##
##  Mcnemar's Test P-Value : 7.66e-06
##
##             Sensitivity : 0.14894
##             Specificity : 0.96748
##          Pos Pred Value : 0.46667
##          Neg Pred Value : 0.85612
##              Prevalence : 0.16041
##          Detection Rate : 0.02389
##    Detection Prevalence : 0.05119
##       Balanced Accuracy : 0.55821
##
##        'Positive' Class : 1
##
```

Random forest

We then tried using a random forest approach to see whether it could perform better than its basic form, the classification tree.

```r
set.seed(51)
train_rf<- train(Attrition ~., data = train_set,
                 method= "Rborist",
                 nTree= 50,
                 trControl= trainControl(method = "cv", number = 5, p = .9),
                 tuneGrid= data.frame(predFixed = seq(2, 20, 2),
                                      minNode = c(12)))
```

To reduce computing time, we first created a forest of 50 trees using five-fold cross validation with 90% of the training data, so the random selection of features as well as the node size were optimized.

By looking at the confusion matrix of this model, we saw that it performed no better than the previous algorithms we tried.

```r
confusionMatrix(predict(train_rf, test_set), test_set$Attrition, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 245  46
##          1   1   1
##
##                Accuracy : 0.8396
##                  95% CI : (0.7925, 0.8797)
##     No Information Rate : 0.8396
##     P-Value [Acc > NIR] : 0.5388
##
##                   Kappa : 0.0281
##
##  Mcnemar's Test P-Value : 1.38e-10
##
##             Sensitivity : 0.021277
##             Specificity : 0.995935
##          Pos Pred Value : 0.500000
##          Neg Pred Value : 0.841924
##              Prevalence : 0.160410
##          Detection Rate : 0.003413
##    Detection Prevalence : 0.006826
##       Balanced Accuracy : 0.508606
##
##        'Positive' Class : 1
##
```

At this point, it was evident that the models were predicting to many negatives while zero positives. This was an important insight gained. In our view, for a machine learning algorithm to be of any use to an HR department, it should detect as many employees who will quit or likely quit as possible. This was not happening with the algorithms we were creating. This made us realize that, the models we tried to fit were not the best, we were also not evaluating on the best performance metrics given the data.

Logistic regression

```
train_glm<- glm(Attrition ~., data= train_set, family = "binomial")
p_hat_glm<- predict(train_glm, test_set, type = "response")
y_hat_glm<- ifelse(p_hat_glm > 0.5, "1", "0")%>%
  factor(levels = c("0", "1"))
```

As the task of our algorithm was to classify, we tried a very basic model in machine learning to see how it performed. We tried logistic regression and, to our surprise, its accuracy was better than any of the other models we used.

By looking at its confusion matrix, we saw that not only accuracy improved, but also sensitivity, the positive predictive value and the balanced accuracy.

```
confusionMatrix(y_hat_glm, test_set$Attrition, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 234  29
##          1  12  18
##
##                Accuracy : 0.8601
##                  95% CI : (0.815, 0.8977)
##     No Information Rate : 0.8396
##     P-Value [Acc > NIR] : 0.19174
##
##                   Kappa : 0.3915
##
##  Mcnemar's Test P-Value : 0.01246
##
##             Sensitivity : 0.38298
##             Specificity : 0.95122
##          Pos Pred Value : 0.60000
##          Neg Pred Value : 0.88973
##              Prevalence : 0.16041
##          Detection Rate : 0.06143
##    Detection Prevalence : 0.10239
##       Balanced Accuracy : 0.66710
##
##        'Positive' Class : 1
##
```

Given this model was better than the previous ones, we wanted to know whether we could optimize this model even more. That is why we looked at the significance of every predictor to see which ones to keep and which ones to drop.

```r
summary(train_glm)
```

```
##
## Call:
## glm(formula = Attrition ~ ., family = "binomial", data = train_set)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5429  -0.4693  -0.2339  -0.0793   3.4264
##
## Coefficients:
##                                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)                       3.062e+00  1.608e+00   1.904 0.056898 .
## Age                              -3.255e-02  1.523e-02  -2.138 0.032528 *
## BusinessTravelTravel_Rarely       1.064e+00  4.210e-01   2.527 0.011516 *
## BusinessTravelTravel_Frequently   2.031e+00  4.597e-01   4.418 9.96e-06 ***
## DailyRate                        -4.956e-04  2.504e-04  -1.980 0.047751 *
## DepartmentResearch & Development  4.799e-01  1.371e+00   0.350 0.726281
## DepartmentHuman Resources        -1.367e+01  6.993e+02  -0.020 0.984405
## DistanceFromHome                  4.459e-02  1.221e-02   3.651 0.000261 ***
## Education                         5.519e-02  9.977e-02   0.553 0.580115
## EducationFieldOther              -7.491e-02  4.812e-01  -0.156 0.876308
## EducationFieldMedical            -1.699e-01  2.509e-01  -0.677 0.498177
## EducationFieldMarketing           3.627e-01  3.642e-01   0.996 0.319250
## EducationFieldTechnical Degree    9.922e-01  3.542e-01   2.801 0.005094 **
## EducationFieldHuman Resources     1.254e+00  9.958e-01   1.259 0.207856
## EmployeeNumber                   -1.533e-04  1.742e-04  -0.880 0.378836
## EnvironmentSatisfaction          -4.369e-01  9.591e-02  -4.556 5.22e-06 ***
## GenderMale                        5.271e-01  2.137e-01   2.467 0.013640 *
## HourlyRate                       -2.591e-04  5.042e-03  -0.051 0.959021
## JobInvolvement                   -5.384e-01  1.393e-01  -3.864 0.000111 ***
## JobLevel                          2.783e-01  3.709e-01   0.750 0.453117
## JobRoleResearch Scientist        -7.801e-01  1.412e+00  -0.553 0.580552
## JobRoleLaboratory Technician      6.025e-02  1.402e+00   0.043 0.965724
## JobRoleManufacturing Director    -1.682e+00  1.467e+00  -1.146 0.251660
## JobRoleHealthcare Representative -1.288e+00  1.448e+00  -0.890 0.373644
## JobRoleManager                   -1.495e+00  1.317e+00  -1.135 0.256245
## JobRoleSales Representative       9.930e-01  4.687e-01   2.119 0.034131 *
## JobRoleResearch Director         -2.509e+00  1.851e+00  -1.356 0.175153
## JobRoleHuman Resources            1.393e+01  6.993e+02   0.020 0.984103
## JobSatisfaction                  -4.102e-01  9.235e-02  -4.441 8.94e-06 ***
## MaritalStatusMarried             -1.017e+00  2.943e-01  -3.454 0.000552 ***
## MaritalStatusDivorced            -1.370e+00  3.937e-01  -3.481 0.000499 ***
## MonthlyIncome                    -4.426e-05  9.754e-05  -0.454 0.649982
## MonthlyRate                       6.773e-06  1.434e-05   0.472 0.636659
## NumCompaniesWorked                1.931e-01  4.502e-02   4.290 1.78e-05 ***
## OverTimeYes                       2.142e+00  2.264e-01   9.460  < 2e-16 ***
## PercentSalaryHike                -4.657e-02  4.559e-02  -1.022 0.307007
## PerformanceRating                 3.305e-01  4.631e-01   0.714 0.475328
## RelationshipSatisfaction         -2.543e-01  9.447e-02  -2.692 0.007096 **
## StockOptionLevel                 -4.155e-02  1.751e-01  -0.237 0.812494
## TotalWorkingYears                -9.216e-02  3.497e-02  -2.635 0.008406 **
## TrainingTimesLastYear            -1.881e-01  8.457e-02  -2.224 0.026157 *
## WorkLifeBalance                  -3.286e-01  1.401e-01  -2.345 0.019009 *
## YearsAtCompany                    1.268e-01  4.891e-02   2.593 0.009514 **
## YearsInCurrentRole               -1.686e-01  5.301e-02  -3.180 0.001471 **
## YearsSinceLastPromotion           2.024e-01  4.846e-02   4.177 2.95e-05 ***
## YearsWithCurrManager             -1.480e-01  5.525e-02  -2.679 0.007391 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1040.54  on 1176  degrees of freedom
## Residual deviance:  665.51  on 1131  degrees of freedom
## AIC: 757.51
##
## Number of Fisher Scoring iterations: 15
```

By looking at all the predictors we were able to see which ones we could drop to improve the performance of our model. We ended up removing the predictors "Department", "Education", "Employee number", "Job level", "Hourly rate", "Monthly income", "Monthly rate", "Percent salary hike", "Performance rating" and "Stock option level" because none of them were statistically significant in predicting attrition.

```
train_glm2<- glm(Attrition ~ .-Department-Education-EmployeeNumber-JobLevel- HourlyRate
                 -MonthlyIncome-MonthlyRate - PercentSalaryHike- PerformanceRating- StockOptionLevel,
                 data= train_set, family = "binomial")
p_hat_glm2<- predict(train_glm2, test_set, type = "response")
y_hat_glm2<- ifelse(p_hat_glm2 > 0.5, "1", "0")%>%
  factor(levels = c("0", "1"))
```

After doing this, we realized we did not improve our accuracy; as it can be see in the confusion matrix.

```
confusionMatrix(y_hat_glm2, test_set$Attrition, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 234  29
##          1  12  18
##
##                Accuracy : 0.8601
##                  95% CI : (0.815, 0.8977)
##     No Information Rate : 0.8396
##     P-Value [Acc > NIR] : 0.19174
##
##                   Kappa : 0.3915
##
##  Mcnemar's Test P-Value : 0.01246
##
##             Sensitivity : 0.38298
##             Specificity : 0.95122
##          Pos Pred Value : 0.60000
##          Neg Pred Value : 0.88973
##              Prevalence : 0.16041
##          Detection Rate : 0.06143
##    Detection Prevalence : 0.10239
##       Balanced Accuracy : 0.66710
##
##        'Positive' Class : 1
##
```

However, we did improve the fit of our model. We were able to look at this by comparing the AIC of the first logistic model and the AIC of this second one. As it can be seen in the tables below, the AIC of the second logistic model was lower than of the first one, which implied a better model.

```
glance(train_glm)
```

```
## # A tibble: 1 x 7
##   null.deviance df.null logLik   AIC   BIC deviance df.residual
##           <dbl>   <int>  <dbl> <dbl> <dbl>    <dbl>       <int>
## 1         1041.    1176  -333.  758.  991.     666.        1131
```

```
glance(train_glm2)
```

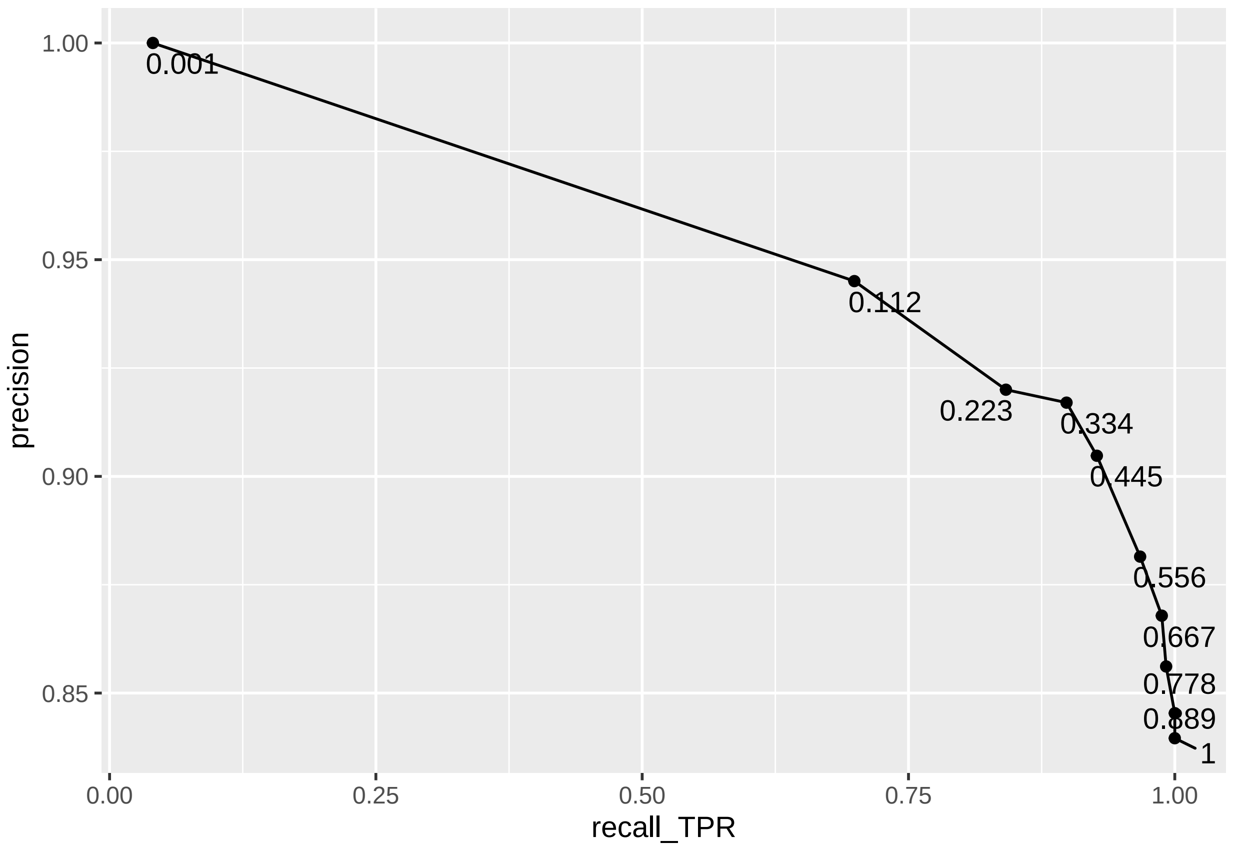
```
## # A tibble: 1 x 7
##   null.deviance df.null logLik   AIC   BIC deviance df.residual
##           <dbl>   <int>  <dbl> <dbl> <dbl>    <dbl>       <int>
## 1         1041.    1176  -335.  740.  917.     670.        1142
```

We did not stop here. As we realized, the goal of our machine learning model was to predict as many positives, while keeping a decent amount of true positives and negatives. Because of this reason, we wanted to know if lowering the probability below 50% of predicting either a positive or a negative, could improve the sensitivity of our model, as well as its precision.

```
##Precision-Recall(logistic regression)
probs <- seq(0.001, 1, length.out = 10)

method_logistic<- sapply(probs, function(p){
  y_hat<- ifelse(p_hat_glm2 > p, "1", "0")%>%
    factor(levels = c("0", "1"))
  list(method = "Logistic Regression",
       cutoff= p,
       FPR = 1-specificity(y_hat, test_set$Attrition),
       recall_TPR = sensitivity(y_hat, test_set$Attrition),
      precision = precision(y_hat, test_set$Attrition))})

tb_logistic<- data.frame(t(method_logistic))

pr_plot_logistic<-
  tb_logistic%>%
  unnest()%>%
  ggplot(aes(recall_TPR, precision, label = cutoff)) +
  geom_line() +
  geom_point()+
  geom_text_repel(nudge_x = 0.001, nudge_y = -0.001)
```

Given our data was biased, we knew that a precision-recall plot could help us visualize different probability values to optimize on our chosen parameters. The plot below shows different probability values with its respective recall and precision.

```
pr_plot_logistic
```



By looking at this plot, we saw that a probability of 0.334 gave us a good balance between good precision and a good enough true positive rate. We went a step further and and wanted to know whether a value below 0.35 was better at giving us a better balance between precision and recall. We will discuss our results of this model later on.

```
###optimizing probability cutoff in logistic regression
probs <- seq(0.001, 0.35, length.out = 12)

method_logistic<- map_df(probs, function(p){
```

```r
  y_hat<- ifelse(p_hat_glm2 > p, "1", "0")%>%
    factor(levels = c("0", "1"))
  list(method = "Logistic Regression",
       cutoff= round(p,3),
       FPR = 1-specificity(y_hat, test_set$Attrition),
       recall_TPR = sensitivity(y_hat, test_set$Attrition),
       precision = precision(y_hat, test_set$Attrition))})

tb_logistic<- data.frame(t(method_logistic))

pr_plot_logistic<-
  tb_logistic%>%
  unnest()%>%
  ggplot(aes(recall_TPR, precision, label = cutoff)) +
  geom_line() +
  geom_point()+
  geom_text_repel(nudge_x = 0.001, nudge_y = -0.001)
```

Linear Discriminant Analysis

As our logistic regression model performed well, we wanted to know whether a Linear Discriminant Analysis approach could be better or could ensemble well with our logistic regression model.

```r
##LDA
train_lda<- train(Attrition ~ ., data = train_set,
                  method = "lda")
p_hat_lda<- predict(train_lda, test_set, type = "prob")
y_hat_lda<- predict(train_lda, test_set)
```

By looking at its confusion matrix, we were able to see that this model had the best accuracy thus far. However, it did not improve by much the true positive cases detected in comparison with our logistic regression approach.

```r
confusionMatrix(y_hat_lda, test_set$Attrition, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 237  28
##          1   9  19
##
##                Accuracy : 0.8737
##                  95% CI : (0.8302, 0.9095)
##     No Information Rate : 0.8396
##     P-Value [Acc > NIR] : 0.061938
##
##                   Kappa : 0.4395
##
##  Mcnemar's Test P-Value : 0.003085
##
##             Sensitivity : 0.40426
##             Specificity : 0.96341
##          Pos Pred Value : 0.67857
##          Neg Pred Value : 0.89434
##              Prevalence : 0.16041
##          Detection Rate : 0.06485
##    Detection Prevalence : 0.09556
##       Balanced Accuracy : 0.68383
##
##        'Positive' Class : 1
##
```

To optimize on this, we did the same procedure as with logistic regression to see whether relaxing our probability value could give us better sensitivity, precision and recall.

```
##Precision-Recall (lda)
probs <- seq(0.001, 0.35, length.out = 12)

method_lda<- map_df(probs, function(p){
  y_hat<- ifelse(p_hat_lda[,2] > p, "1", "0")%>%
    factor(levels = c("0", "1"))
  list(method = "LDA",
       cutoff= round(p,3),
       FPR = 1-specificity(y_hat, test_set$Attrition),
       recall_TPR = sensitivity(y_hat, test_set$Attrition),
       precision = precision(y_hat, test_set$Attrition))})
```

Ensemble (logistic regression + lda)

```
##ensemble lda + logistic
p_ensemble<- (p_hat_glm2+p_hat_lda[,2])/2

probs <- seq(0.001, 0.35, length.out = 12)

method_ensemble<- map_df(probs, function(p){
  y_hat<- ifelse(p_ensemble> p, "1", "0")%>%
    factor(levels = c("0", "1"))
  list(method = "Ensemble",
       cutoff= round(p,3),
       FPR = 1-specificity(y_hat, test_set$Attrition),
       recall_TPR = sensitivity(y_hat, test_set$Attrition),
       precision = precision(y_hat, test_set$Attrition))})
```

Finally, we created an ensemble model, which averaged the probabilities of our logistic regression and of our linear discriminant analysis. For optimization purposes, we also relaxed our probability cutoff of 50% to see whether a lower value performed better according to our chosen evaluation metrics.

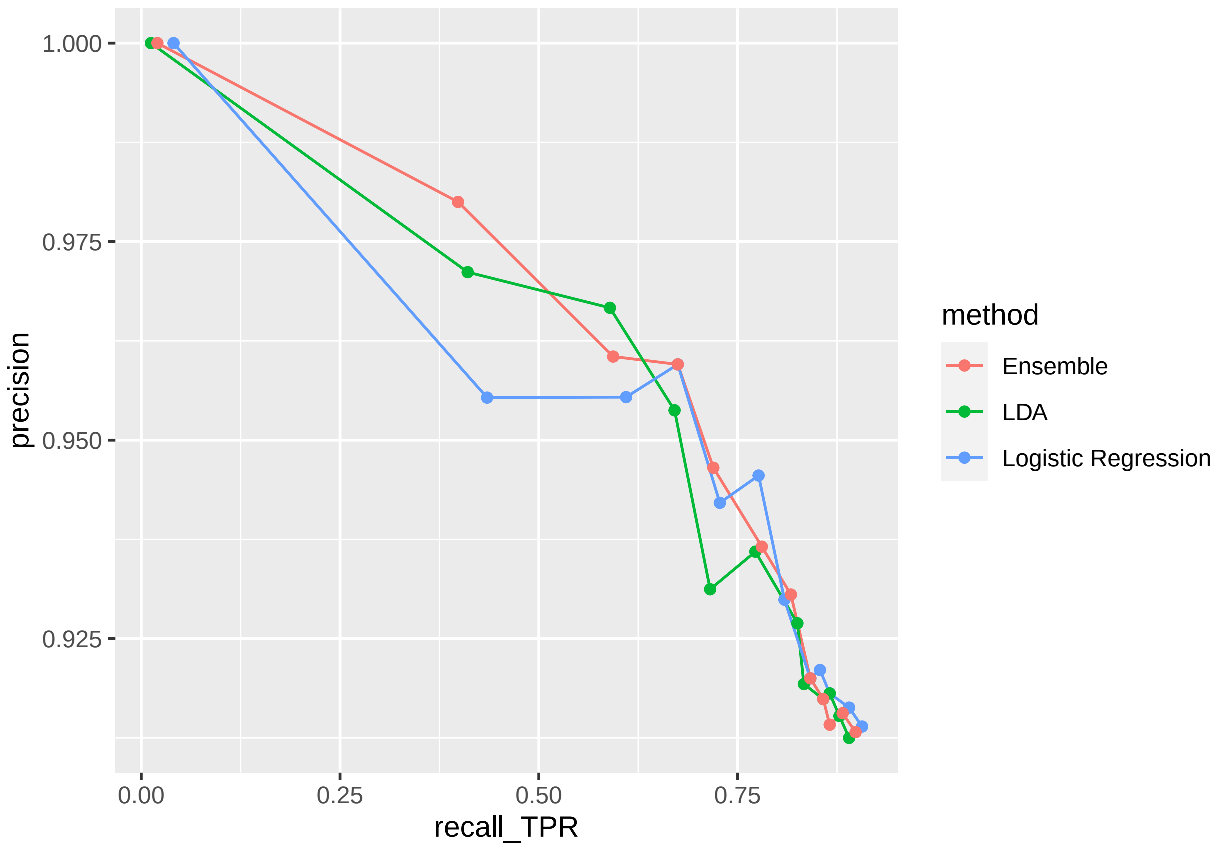
Results

```
pr_plots<-
bind_rows(method_logistic, method_lda, method_ensemble)%>%
  ggplot(aes(recall_TPR, precision, color= method))+
  geom_line() +
  geom_point()
```

After trying different machine learning algorithms to predict attrition, we ended up with three main ones which performed best according to our chose evaluation metrics : logistic regression, linear discriminant analysis and the ensemble model (logistic regression + lda).

The plot below shows the precision recall curves of these three models according to different probability values; all of which were below 35%. It is worth mentioning again that, in our view, choosing a probability value below 50% was correct, given making some false positive errors was preferable to making many false negative errors.

```
pr_plots
```



By looking at the plot, we were able to see that any of the three models could have best performance depending on the probability chosen value. We looked at different probability values that had the best relative performance according to the method used.

```
method_ensemble[4,]
```

```
## # A tibble: 1 x 5
##   method   cutoff   FPR recall_TPR precision
##   <chr>     <dbl> <dbl>      <dbl>     <dbl>
## 1 Ensemble  0.096 0.149      0.675     0.960
```

```
method_ensemble[5,]
```

```
## # A tibble: 1 x 5
##   method   cutoff   FPR recall_TPR precision
##   <chr>     <dbl> <dbl>      <dbl>     <dbl>
## 1 Ensemble  0.128 0.213      0.720     0.947
```

```
method_logistic[6,]
```

```
## # A tibble: 1 x 5
##   method              cutoff   FPR recall_TPR precision
##   <chr>                 <dbl> <dbl>      <dbl>     <dbl>
## 1 Logistic Regression   0.16 0.234      0.776     0.946
```

```r
method_lda[3,]
```

```
## # A tibble: 1 x 5
##   method cutoff   FPR recall_TPR precision
##   <chr>   <dbl> <dbl>      <dbl>     <dbl>
## 1 LDA     0.064 0.106      0.589     0.967
```

```r
method_lda[7,]
```

```
## # A tibble: 1 x 5
##   method cutoff   FPR recall_TPR precision
##   <chr>   <dbl> <dbl>      <dbl>     <dbl>
## 1 LDA     0.191 0.340      0.825     0.927
```

After examining these values, we considered that a probability value equal or below 10% would have great sensitivity, but would predict too many false positives, which is also not optimal. That is why, one of our criterion for selecting a probability value cutoff was that it should be above 10%.

```r
##evaluation metrics of final models
y_hat_glm2.3<- ifelse(p_hat_glm2 > 0.16, "1", "0")%>%
  factor(levels = c("0", "1"))
metrics_logistic<- confusionMatrix(y_hat_glm2.3, test_set$Attrition, positive = "1")

y_hat_lda2<- ifelse(p_hat_lda[2] > 0.191, "1", "0")%>%
  factor(levels = c("0", "1"))
metrics_lda<- confusionMatrix(y_hat_lda2, test_set$Attrition, positive = "1")

y_hat_ensemble<- ifelse(p_ensemble > 0.128, "1", "0")%>%
  factor(levels = c("0", "1"))
metrics_ensemble<- confusionMatrix(y_hat_ensemble, test_set$Attrition, positive = "1")
```

In the end, we chose different probability values for each of the methods considered. Thus, we chose the values which allowed one method to outperform the other two methods regarding precision and recall. So, we ended up choosing a probability cutoff of 16% for logistic regression; 19.1%, for linear discriminant analysis; and 12.8%, for the ensemble model (logistic regression + lda).

By looking at the confusion matrix of these methods, we were able to see that the method which had the best balanced accuracy was the logistic regression alone; it was .77. The lda approach, however, had the best positive predictive value metric above all; but it was the lowest in sensitivity.

```r
metrics_logistic
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 191  11
##          1  55  36
##
##                Accuracy : 0.7747
##                  95% CI : (0.7225, 0.8213)
##     No Information Rate : 0.8396
##     P-Value [Acc > NIR] : 0.9985
##
##                   Kappa : 0.3934
```

```
##
##  Mcnemar's Test P-Value : 1.204e-07
##
##             Sensitivity : 0.7660
##             Specificity : 0.7764
##          Pos Pred Value : 0.3956
##          Neg Pred Value : 0.9455
##              Prevalence : 0.1604
##          Detection Rate : 0.1229
##    Detection Prevalence : 0.3106
##       Balanced Accuracy : 0.7712
##
##        'Positive' Class : 1
##
```

```
metrics_lda
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 203  16
##          1  43  31
##
##                Accuracy : 0.7986
##                  95% CI : (0.7481, 0.843)
##     No Information Rate : 0.8396
##     P-Value [Acc > NIR] : 0.973995
##
##                   Kappa : 0.3934
##
##  Mcnemar's Test P-Value : 0.000712
##
##             Sensitivity : 0.6596
##             Specificity : 0.8252
##          Pos Pred Value : 0.4189
##          Neg Pred Value : 0.9269
##              Prevalence : 0.1604
##          Detection Rate : 0.1058
##    Detection Prevalence : 0.2526
##       Balanced Accuracy : 0.7424
##
##        'Positive' Class : 1
##
```

```
metrics_ensemble
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 177  10
##          1  69  37
##
##                Accuracy : 0.7304
##                  95% CI : (0.6757, 0.7803)
##     No Information Rate : 0.8396
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3361
##
##  Mcnemar's Test P-Value : 6.777e-11
##
##             Sensitivity : 0.7872
##             Specificity : 0.7195
##          Pos Pred Value : 0.3491
##          Neg Pred Value : 0.9465
##              Prevalence : 0.1604
##          Detection Rate : 0.1263
##    Detection Prevalence : 0.3618
##       Balanced Accuracy : 0.7534
##
##        'Positive' Class : 1
##
```

At this point it is important to take into account the accuracy of the Linear Discriminant Analysis which took 50% as the probability value cutoff. Although it had a much higher accuracy than our logistic regression model (.87 in comparison with .77), it had a much lower sensitivity, as well as balanced accuracy (.40 and .68 respectively).

As it was mentioned before, we did not mind much making false positive mistakes, while detecting a fair amount of true positives and negatives. For these reasons we believe our best performing model to predict attrition was the logistic regression alone. It took 16% as the probability value cutoff, and, although had a positive predictive value of .39; had a balanced accuracy of .77 and a sensitivity of .76.

Conclusion

The goal of this project was to predict employee attrition. We fitted several models and ended up choosing a logistic regression approach because we believed it would be the algorithm which would help best an HR department: it had one of the highest sensitivity, while keeping specificity high and maintaining the overall best balanced accuracy. Future work may try to fit unsupervised machine learning models to see whether they can outperform our logistic regression approach.