Mathematician in Data Science

Wednesday, January 11, 2017

Machine Prediction for Human Resources Analytics

Maiia Bakhova

Content

Introduction

Here I show analysis, feature engineering and decision tree construction with the result explanation for a csv data set.

I work on kaggle data set "Human Resources Analytics", which can be found at url

https://www.kaggle.com/ludobenistant/hr-analytics.

There we see that a question for the data set is the following:

Why are our best and most experienced employees leaving prematurely?

There you can find a data file in csv format and download it. Each record (a row) represents an employee. Fields in the data set include:

- Employee satisfaction level
- Last evaluation
- Number of projects
- Average monthly hours
- Time spent at the company
- Whether they have had a work accident
- Whether they have had a promotion in the last 5 years
- Department
- Salary
- Whether the employee has left

Data Load

I start with reading data set and looking at its properties: dimensions, column names and types of data. Let us assume that the data file lies in our working directory.

```
dt=read.csv("HR_comma_sep.csv", stringsAsFactors=F)
options(wide=120)
str(dt)
```

```
## 'data.frame': 14999 obs. of  10 variables:
##  $ satisfaction_level   : num  0.38 0.8 0.11 0.72 0.37 0.41 0.1 0.92 0.89 0.42 ...
##  $ last_evaluation      : num  0.53 0.86 0.88 0.87 0.52 0.5 0.77 0.85 1 0.53 ...
##  $ number_project       : int  2 5 7 5 2 2 6 5 5 2 ...
##  $ average_montly_hours : int  157 262 272 223 159 153 247 259 224 142 ...
##  $ time_spend_company   : int  3 6 4 5 3 3 4 5 5 3 ...
##  $ Work_accident        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ left                 : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ promotion_last_5years: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ sales                : chr  "sales" "sales" "sales" "sales" ...
##  $ salary               : chr  "low" "medium" "medium" "low" ...
```


About Me

Maiia Bak

Follow

View my comp

```
# Checking for missing values:
sapply(dt, function(x) sum(is.na(x)))
```

```
##    satisfaction_level       last_evaluation        number_project
##                     0                     0                     0
##  average_montly_hours    time_spend_company         Work_accident
##                     0                     0                     0
##                  left promotion_last_5years                 sales
##                     0                     0                     0
##               salary
##                     0
```

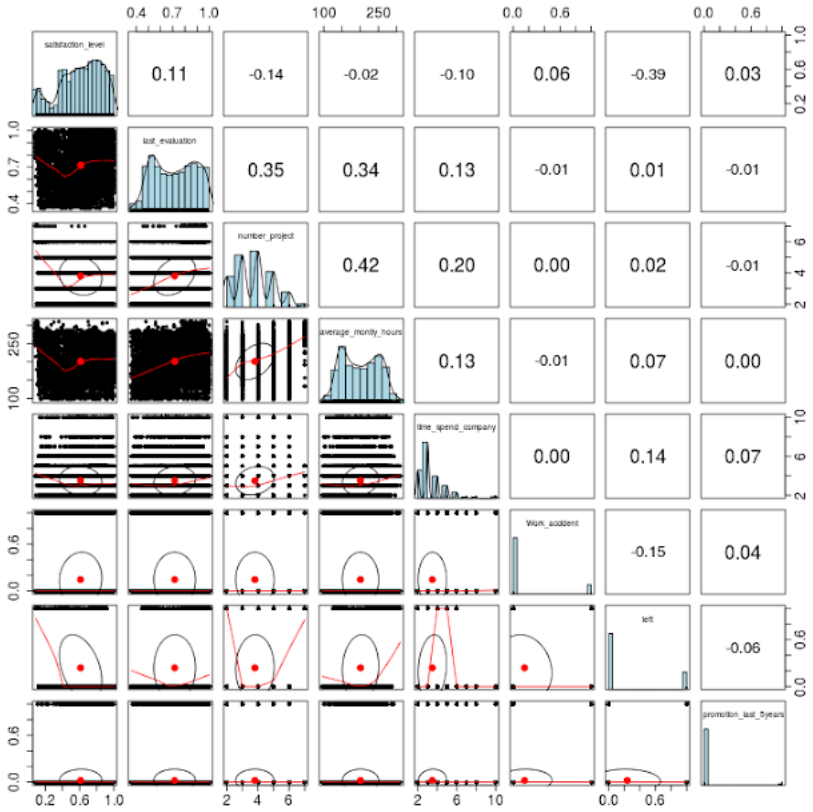
```
mean(dt$left)
```

```
## [1] 0.238082539
```

As we see all data columns (variables) except for the last 2 ones have numeric and integer types. Our target variable is marked as "left" and it is 7th. In addition, here are no missing values, which is convenient. We learnt that a rate of leaving the company 23.8% in accordance to this data set.

We can check pairwise plots, correlations and densities for first 8 columns.

```
library(psych)
pairs.panels(dt[,1:8], hist.col="lightblue",
             cex.axis=1.5, cex.cor=0.5)
```



This picture shows that the greatest absolute correlations of the "left" column are "satisfaction_level", "time_spend_company" and "Work_accident".

Creating Numeric Variables from Character Columns

Now let us look at what the values in last 2 columns.

```
unique(dt$salary)
```

```
## [1] "low"    "medium" "high"
```

```
unique(dt$sales)
```

```
##  [1] "sales"       "accounting"  "hr"          "technical"   "support"
##  [6] "management"  "IT"          "product_mng" "marketing"   "RandD"
```

In "salary" column we see 3 levels for salary values and it looks like "sales" column represents 10 departments. Let us see how uniformly data records are distributed between salary levels and departments:

```
table(dt$salary)
```

```
##
##   high    low medium
##   1237   7316   6446
```

```
table(dt$sales)
```

```
##
##  accounting          hr          IT  management   marketing product_mng
##         767         739        1227         630         858         902
##       RandD       sales     support   technical
##         787        4140        2229        2720
```

It is not very uniform, but at least here are a few hundred records for every salary type and for each department. I would like to replace "salary" column with three other columns: "high_salary", "low_salary", "medium_salary". Each will have values 0 and 1, denoting "no" and "yes", respectfully. In a package called "dummies" we find a function which produces such columns from a text variable.

```
library(dummies)
```

```
## dummies-1.5.6 provided by Decision Patterns
```

```
dumdt=dummy(dt$salary)
dumdt=as.data.frame(dumdt)
names(dumdt)
```

```
## [1] "HumanResoursesAnalytics.Rhtmlhigh"
## [2] "HumanResoursesAnalytics.Rhtmllow"
## [3] "HumanResoursesAnalytics.Rhtmlmedium"
```

Remark: I got these names because I was using RStudio to create this text. The names may be different if you work with a command line interface.

The names are not good because they are too long. I will replace them.

```
names(dumdt)=c("high_salary", "low_salary", "medium_salary")
```

I will attach my new variables to existing data frame and remove now redundant "salary".

```
dt=cbind(dt,dumdt)
dt$salary=NULL
str(dt)
```

```
## 'data.frame': 14999 obs. of  12 variables:
##  $ satisfaction_level   : num  0.38 0.8 0.11 0.72 0.37 0.41 0.1 0.92 0.89 0.42 ...
##  $ last_evaluation      : num  0.53 0.86 0.88 0.87 0.52 0.5 0.77 0.85 1 0.53 ...
##  $ number_project       : int  2 5 7 5 2 2 6 5 5 2 ...
##  $ average_montly_hours : int  157 262 272 223 159 153 247 259 224 142 ...
##  $ time_spend_company   : int  3 6 4 5 3 3 4 5 5 3 ...
##  $ Work_accident        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ left                 : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ promotion_last_5years: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ sales                : chr  "sales" "sales" "sales" "sales" ...
##  $ high_salary          : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ low_salary           : int  1 0 0 1 1 1 1 1 1 1 ...
##  $ medium_salary        : int  0 1 1 0 0 0 0 0 0 0 ...
```

I will go through the similar steps with "sales" column. It has 10 values, so I'll create new columns, rename them and remove the "sales" column.

```
dumdt=dummy(dt$sales)
dumdt=as.data.frame(dumdt)
names(dumdt)
```

```
##  [1] "HumanResoursesAnalytics.Rhtmlaccounting"
##  [2] "HumanResoursesAnalytics.Rhtmlhr"
##  [3] "HumanResoursesAnalytics.RhtmlIT"
##  [4] "HumanResoursesAnalytics.Rhtmlmanagement"
##  [5] "HumanResoursesAnalytics.Rhtmlmarketing"
##  [6] "HumanResoursesAnalytics.Rhtmlproduct_mng"
##  [7] "HumanResoursesAnalytics.RhtmlRandD"
##  [8] "HumanResoursesAnalytics.Rhtmlsales"
```

```
##  [9] "HumanResoursesAnalytics.Rhtmlsupport"
## [10] "HumanResoursesAnalytics.Rhtmltechnical"
```

```
names(dumdt)=c("accounting","hr","IT","management","marketing","product_mng","RandD",
               "sales","support", "technical")
dt=cbind(dt,dumdt)
dt$sales=NULL
dim(dt)
```

```
## [1] 14999    21
```

```
# Removing the object we do not need anymore
rm(dumdt)
```

We got 13 new columns. We can check how they are correlated. In this case all our variables are categorical, and plotting them is not helpful. We will check numerical correlations.

```
options(wide=110)
cor(dt[,c(7,9:14)])
```

```
##                       left   high_salary     low_salary  medium_salary
## left           1.0000000000 -0.1209294638  0.13472197414 -0.06883296809
## high_salary   -0.1209294638  1.0000000000 -0.29256037558 -0.26027352121
## low_salary     0.1347219741 -0.2925603756  1.00000000000 -0.84714420898
## medium_salary -0.0688329681 -0.2602735212 -0.84714420898  1.00000000000
## accounting     0.0152011507  0.0118213193 -0.00975882741  0.00328479594
## hr             0.0282487481 -0.0178579572 -0.01568984972  0.02576547182
## IT            -0.0109248273 -0.0160889786  0.00511558982  0.00377497434
##                   accounting            hr             IT
## left           0.01520115067  0.0282487481 -0.01092482732
## high_salary    0.01182131932 -0.0178579572 -0.01608897862
## low_salary    -0.00975882741 -0.0156898497  0.00511558982
## medium_salary  0.00328479594  0.0257654718  0.00377497434
## accounting     1.00000000000 -0.0528478314 -0.06929286031
## hr            -0.05284783141  1.0000000000 -0.06794949459
## IT            -0.06929286031 -0.0679494946  1.00000000000
```

```
cor(dt[,c(7,15:21)])
```

```
##                       left    management       marketing   product_mng
## left          1.000000000000 -0.0460353907 -0.000859304044 -0.0110291521
## management   -0.046035390706  1.0000000000 -0.051577535319 -0.0529659689
## marketing    -0.000859304044 -0.0515775353  1.000000000000 -0.0623079556
## product_mng -0.011029152078 -0.0529659689 -0.062307955590  1.0000000000
## RandD       -0.046595651167 -0.0492738809 -0.057964667577 -0.0595250386
## sales        0.009923407034 -0.1292892024 -0.152092863576 -0.1561871042
## support      0.010700118013 -0.0874815739 -0.102911324656 -0.1056816303
## technical    0.020076104934 -0.0985507548 -0.115932856081 -0.1190536929
##                     RandD          sales       support     technical
## left        -0.0465956512  0.00992340703  0.0107001180  0.0200761049
## management  -0.0492738809 -0.12928920242 -0.0874815739 -0.0985507548
## marketing   -0.0579646676 -0.15209286358 -0.1029113247 -0.1159328561
## product_mng -0.0595250386 -0.15618710420 -0.1056816303 -0.1190536929
## RandD        1.0000000000 -0.14529980143 -0.0983149024 -0.1107548413
## sales       -0.1452998014  1.00000000000 -0.2579674078 -0.2906084287
## support     -0.0983149024 -0.25796740777  1.0000000000 -0.1966357767
## technical   -0.1107548413 -0.29060842875 -0.1966357767  1.0000000000
```

As we see our new variables may have lower correlations with our target variable "left" than between themselves. It happens because 1) a person should have some salary, so if it is not high or low it must be medium, 2) a person should belong to one of the departments as well. Hence variables in these groups are correlated.

I will remove a variable in each group which yields the lowest correlation with my target variable: "medium_salary" and "marketing".

```
dt$medium_salary =NULL
dt$marketing =NULL
```

Choosing data for work

As we remember we are supposed to answer a specific question: Why are our best and most experienced employees leaving prematurely? Regretfully we are not told if our data set already contains the employees, and we need to check it out. Let us compute what are "last_evaluation" and "time_spend_company" ranges.

```
range(dt$last_evaluation)
```

```
## [1] 0.36 1.00
```

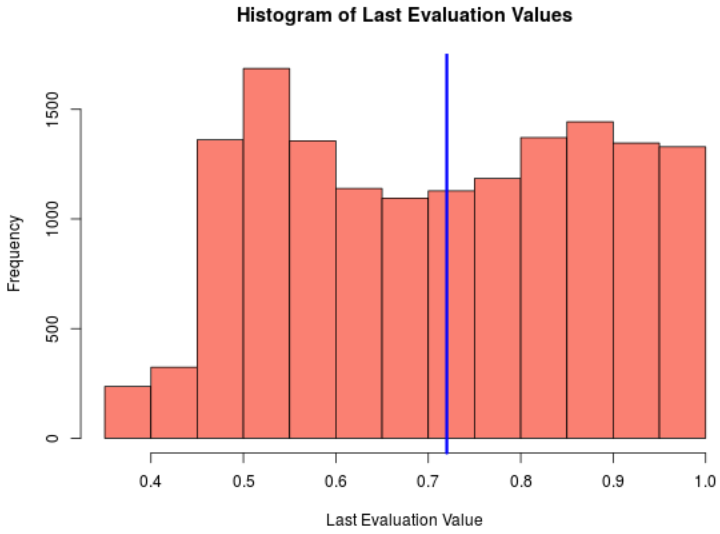
```
range(dt$time_spend_company)
```

```
## [1]  2 10
```

As we see evaluations may be rather low. Therefore our data set has other employees which we do not need for our analysis.

I will plot a histogram for "last_evaluation" column" and mark the median for its values as a blue vertical line.

```
hist(dt$last_evaluation, col="salmon",
     xlab="Last Evaluation Value",
 main="Histogram of Last Evaluation Values")
abline(v=median(dt$last_evaluation),
    col=4,lwd=3)
```

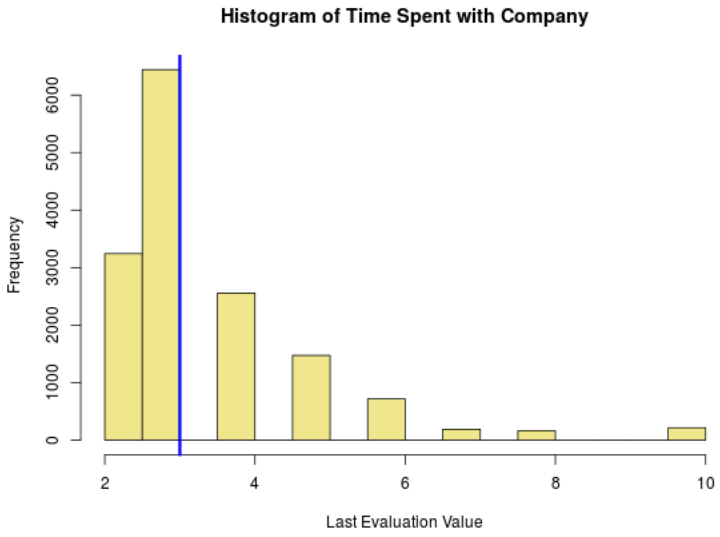



```
median(dt$last_evaluation)
```

```
## [1] 0.72
```

Doing going through similar steps for time spent with company:

```
hist(dt$time_spend_company, col="khaki",
     xlab="Last Evaluation Value",
 main="Histogram of Time Spent with Company")
abline(v=median(dt$time_spend_company),
    col=4,lwd=3)
```




```
median(dt$time_spend_company)
```

```
## [1] 3
```

We can consider as best and most experienced employees people who have evaluation above 0.72 and spent more than 3 years with the company as the "best and most experienced employees". We can compute what is a rate of leaving for such employees.

```
dt=dt[dt$last_evaluation>0.72 & dt$time_spend_company>3,]
mean(dt$left)
```

```
## [1] 0.523085747
```

The rate of leaving was 23.8%, and now it is more than doubled. It is a cause for concern.

Let us now repeat the same pairwise plots with correlations we did at first.

```
library(psych)
pairs.panels(dt[,1:8], hist.col="lightblue",
             cex.axis=1.5, cex.cor=0.5)
```

As we see the data behavior is a bit different now. Now columns "average_montly_hours", "number_project" and "satisfaction_level" yield highest correlations with "left" variable.

My data frame is ready for prediction.

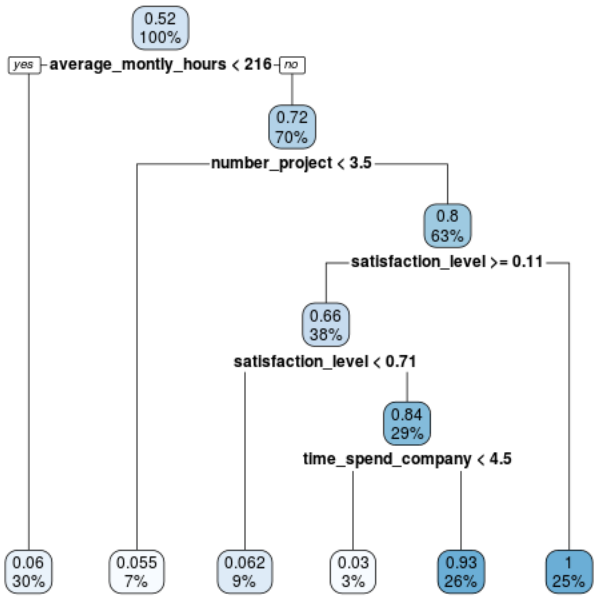
Machine Learning

Because we need to explain why employees are leaving then a simple decision tree algorithm is the best.

Decision Tree with Result Explanation

The option called "maxdepth" defines a maximal depth of a tree.

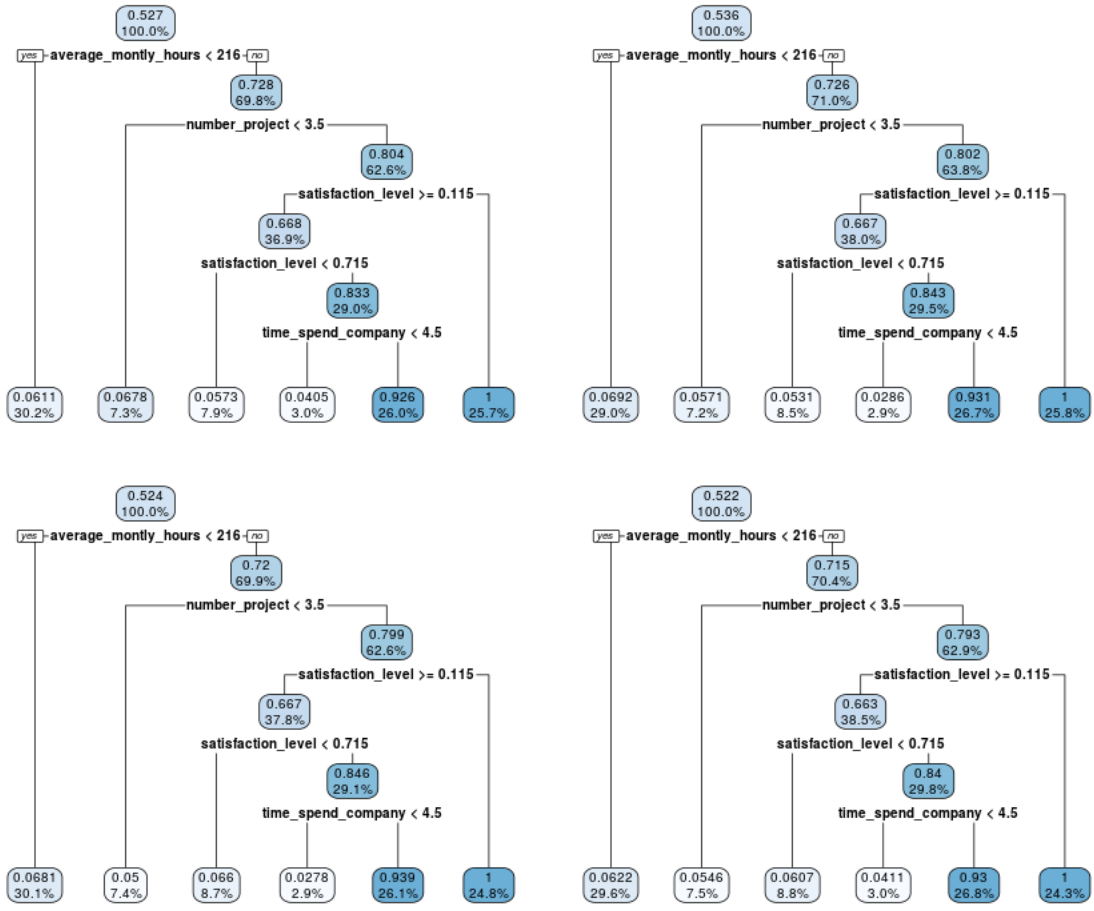
```
library(rpart)
library(rpart.plot )
cart_mod=rpart(left~., data=dt, maxdepth=5)
rpart.plot(cart_mod)
```



So, what do we see on our tree graph? That majority of valuable employees work overtime: 70%. People with normal time load tend to stay, with rate of 94%. Among those who work too many hours several can stay if they have no more than 3.5 projects. There are a few enthusiasts who do not leave with too much work and their satisfaction is above 0.71. But most of overworked people are leaving, and they are the 54% of all valuable workers. Even high salary does not help. As we see the longer overworked people stay at the company, the more they are inclined to leave.

I want to see how reliable is my result and I will split the data set into 2 sets: for training and for testing. We will compute a decision tree model for a train set and check what accuracy we get predicting with the model on test set. To make sure that our accuracy number is not accidental it makes sense to repeat it a few times. In practice it is done at least 10 times (and called cross-validation), but I will limit it to 4. I will set up random seeds to make sure I have no repetitions.

```
par(mfrow=c(2,2))
seed=c(89,132,765,4)
accuracy=numeric(length=4)
for (i in 1:4){
    set.seed(seed[i])
    indeces=sample(1:dim(dt)[1], round(.7*dim(dt)[1]))
    train=dt[indeces,]
    test=dt[-indeces,]
    cart_mod=rpart(left~., data=train, maxdepth=5)
    rpart.plot(cart_mod, digits=3)
# In this case our predictions are returned as probabilities,
#  and we need 0 and 1.
    predictionsAs0and1=sapply(predict(cart_mod,test[,-7]),
                              function(nu) ifelse(nu>.5, 1, 0))
          accuracy[i]=
sum(test$left==predictionsAs0and1)/length(test$left)
}
```




```
accuracy
```

```
## [1] 0.959847036 0.958891013 0.956022945 0.958891013
```

As you see our accuracy here fluctuates somewhere between 95.5% and 96%. The decision trees change slightly as well, but not much.

If we are not limiting our predictions with methods which are to explain to a layman then we can get better accuracy.

Random Forest Method

For random forest method we construct many trees and default option is 500 of them. For every tree and at each tree level we pick up only a subset of variables. We need to choose a size for such subset and in the package below it is called "mtry". As before to check method reliability I will examine its work on different subsets of the data. In addition I want to see what variables are important for the method, so I create a data frame for "mtry" number, corresponding accuracy and for 3 most important variables.

```
library(randomForest)
# For classification I'm to declare the target variable as factor.
dt$left=as.factor(dt$left)
resultTable=data.frame(N=1:4, accuracy=0,
                       importance1=character(length=4L),
                       importance2=character(length=4L),
                       importance3=character(length=4L), stringsAsFactors=FALSE)
```

```
for (i in 1:4) {
    set.seed(seed[i])
    indeces=sample(1:dim(dt)[1], round(.7*dim(dt)[1]))
    train=dt[indeces,]
    test=dt[-indeces,]
   rf_model=randomForest(left~., data=train, mtry=6,importance=TRUE)
   resultTable[i,"accuracy"]=
   sum(test$left==predict(rf_model,test[,-7]))/length(test$left)
   resultTable[i,3:5]=
   rownames(rf_model$importance)[1:3]
}
resultTable
```

```
##   N    accuracy        importance1     importance2    importance3
## 1 1 0.985659656 satisfaction_level last_evaluation number_project
## 2 2 0.984703633 satisfaction_level last_evaluation number_project
## 3 3 0.991395793 satisfaction_level last_evaluation number_project
## 4 4 0.990439771 satisfaction_level last_evaluation number_project
```

We can get accuracy 98.5%-99% choosing "mtry=6". The most significant variable is "satisfaction_level", then "last_evaluation" and "number_project". As we saw previously the fist variable is responsible for majority of leaving workers.

Linear and Quadratic Discriminant Analysis

These methods require some specific properties of data. Our variables are supposed to have a normal distribution, and by the look of their histograms they do not. Thus the methods is not likely not perform well and we see as a result.

Linear discriminant analysis

```
library(MASS)
lda_model=lda(left~., data=train)
pred=predict(lda_model, test)$class
sum(test$left==pred)/length(test$left)
```

```
## [1] 0.86998088
```

Quadratic discriminant analysis in addition suggests that variable correlations are consistent, which clearly does not help.

```
qda_model=qda(left~., data=train)
pred=predict(qda_model, test)$class
sum(test$left==pred)/length(test$left)
```

```
## [1] 0.863288719
```

Nearest Neighbors

With nearest neighbors we check for each test value if it is close to some values in train set if we consider each record as a point in multidimensional space. For this we need values for each variable to be approximately of the same scale. We can look at our plot and see that it is not the case. Majority of our data values are 0 and 1, but some are not. Let us see it in more detail:

```
 sapply(dt[,1:6], range)
```

```
##      satisfaction_level last_evaluation number_project
## [1,]               0.09            0.73              2
## [2,]               1.00            1.00              7
##      average_montly_hours time_spend_company Work_accident
## [1,]                   96                  4             0
## [2,]                  310                 10             1
```

I will shift and scale variables which have large numbers.

```
dt$number_project=(dt$number_project-2)/5
dt$average_montly_hours=(dt$average_montly_hours-90)/200
dt$time_spend_company=dt$time_spend_company/10
```

Number of neighbors in the method is "k".

```
library(class)
knn_accuracy=numeric(length=19)
for (i in 1:19){
   pred=knn(train[, -7], test[, -7], train$left, k=i)
   knn_accuracy[i]=sum(test$left==pred)/length(test$left)
}
knn_accuracy
```

```
##  [1] 0.913001912 0.885277247 0.875717017 0.874760994 0.875717017
##  [6] 0.874760994 0.871892925 0.871892925 0.869980880 0.872848948
## [11] 0.871892925 0.869024857 0.870936902 0.864244742 0.861376673
## [16] 0.859464627 0.858508604 0.859464627 0.852772467
```

The method does not yield good accuracy for any number of neighbors even in comparison with the decision tree method and it does not make sense to investigate how reliable is the resulting number.

Support Vector Machines

We can try another method which is based on measuring distances: support vector machines. For this method we are looking for linear borders between geometrical clusters of our data.

```
library(e1071)
svm_model=svm(left~., data=dt, kernel="linear", type="C")
sum(test$left==predict(svm_model, test[,-7]))/length(test$left)
```

```
## [1] 0.526768642
```

It is the worst. It means that our data do not form two clusters which we can mark as "left" and "stayed". You can try other kernels which represent different kind of borders and verify that it is no help. I am not going to check out the resulting accuracy as well.

Logistic Regression

Now let us use logistic regression. It is not likely to produce a good result, because for it to work properly we need a number of assumptions. Still we can check it out.

```
seed=c(89,132,765,4)
accuracy=numeric(length=4)
for (i in 1:4){
    set.seed(seed[i])
    indeces=sample(1:dim(dt)[1], round(.7*dim(dt)[1]))
    train=dt[indeces,]
    test=dt[-indeces,]
    log_reg_model=glm(left~., data=train, family="binomial")
## Values for logistic regression predictions are
##  not limited to 0 and 1.
    predictionsAs0and1=sapply(predict(log_reg_model,test[,-7]), function(nu) ifelse(nu>0, 1, 0))
    accuracy[i]=sum(test$left==predictionsAs0and1)/length(test$left)
}
accuracy
```

```
## [1] 0.860420650 0.869980880 0.855640535 0.875717017
```

It is not one of the best as well. So our variables are not very good fit for linear additive model. Although sometimes 85% is all you can get from data, but here we've seen better accuracy.

Conclusion

Random Forest method yields the best accuracy, although it does not explain much. The Decision Tree is much more useful for staff policy recommendations. Other method mostly tell us what our data are not: we can not divide records into 2 well defined clusters which we can name "left" and "stayed", variables are not mutlinormally distributed, and linear additive model does not work well predicting desired outcome.

Remark: I would like to note that I used simple variations of the methods and did not explore all options.

As you see I have a lot of repetitions in my code. There is a way to avoid it: look at "caret" package!

Posted by Maiia Bakhova at 12:57:00 PM

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)

R bloggers