Employee Attrition Modelling: Part 2

SFL Scientific

August 2017

Part 2: Data Modeling with Machine Learning

In this study, we use several algorithms to model employee attrition: extreme gradient boosting (XGBoost), support vector machines (SVM), and logistic regression.

- XGBoost is a decision tree based algorithm. Multiple trees are ensembled to improve the predictive power of the model.
- An SVM is a discriminative classifier that takes labeled training data and constructs a hyperplane to categorize new examples.
- Finally, logistic regression is a simple classifier used to estimate the probability of a binary outcome based on several predictors and a logit function.

All three algorithms are supervised learning methods; these take in a set of labelled data, in this case a historic records of people who have either left the company or stayed on, and learns the underlying patterns in the available data - in this case, from features such as Age, Job Role, Overtime worked etc.

Preprocessing and Feature Generation

```
# Load workspace from part1
load("part1.RData")
```

After cleaning up the dataset (filling in missing values, removing spurious data found through EDA etc) we proceed to generate features. Features should capture the underlying information we are trying to model with machine learning. When generating features, we take the data and either decompose or aggregate it in order to answer an underlying question. In this particular case, since we want to know why employees attrite, we build features that we expect will help explain this phenomenon.

```
# load packages
library("xgboost")
library("e1071")
library("MASS")
library("xtable")
```

When it comes to building the model, the data is partitioned into three sets: training, validation and testing:

- The training set is responsible for initially teaching the model the causal relationship between all information and the attrition probability. The features are fed into the model and the machine learns how the patterns that correpond to employee attrition.
- The validation set is then used to estimate how well the model has been trained and fine tune the parameters to develop the best model. Once those two steps have been completed,
- The completed model is applied to the testing set in order to get accurate results on how the model would perform on real-world data. In this case, we can predict the likelihood for any employee to attrite in the future based solely on the quantifiable data that an HR department has access.

```
# Creat data for training and test
set.seed(0)
```

```r
tr.number<-sample(nrow(d),nrow(d)*2/3)
# we split whole dataset into 2/3 training data and 1/3 testing data
train<-d[tr.number,]
test<-d[-tr.number,]

column_names = names(test)

# split dataset
train_Y = as.numeric(train$Attrition)
train$Attrition<-NULL
test_Y = test$Attrition
test$Attrition<-NULL

# numericize training and testing data
train[] <- lapply(train, as.numeric)
test[] <- lapply(test, as.numeric)
```

Model fitting

XGBoost model

The first model we fit is a extreme gradient boosting (XGBoost) model. It is a boosted decision tree model where the parameters control the model complexity and avoid overfitting. For more detail, http://xgboost.readthedocs.io/en/latest/parameter.html, there is a detail explanation for the parameter setting. Here, we choose maximum depth of tree as 3, step size shrinkage parameter as 0.3, use logistic regression for binary classification with area under the curve(auc) as a evalation matric for validation data.

```r
# Construct xgb.DMatrix object from training matrix
dtrain <- xgb.DMatrix(as.matrix(train), label = train_Y)

# Creat a list of parameters for XGBoost model
param <- list(max_depth=3,
              silent=1,
              eta = 0.3,
              objective='binary:logistic',
              eval_metric = 'auc')

# Training a XGBoost model using training dataset and chosen parameters
bst <- xgb.train(param, nrounds = 82, dtrain)

# Predicting the results using testing dataset
pred.xgb <- predict(bst, as.matrix(test))

# Creat a table indicating the most important features of XGBoost model
importance <- xgb.importance(feature_names = column_names, model = bst)
```

```r
bst
```

```
## ##### xgb.Booster
## raw: 52.9 Kb
## call:
##   xgb.train(params = param, data = dtrain, nrounds = 82)
## params (as set within xgb.train):
##   max_depth = "3", silent = "1", eta = "0.3", objective = "binary:logistic", eval_metric = "auc", si
```

```
## xgb.attributes:
##   niter
## callbacks:
##   cb.print.evaluation(period = print_every_n)
## niter: 82
```

The XGBoost object gained from training the model is a list that contains basic information for the model training, e.g. the parameters setting, etc. We can use ***predict*** function to use our trained XGBoost model to make a prediction with testing dataset.

```
head(pred.xgb)
```

```
## [1] 0.61055940 0.59795433 0.01765974 0.03849135 0.91354728 0.06819534
```

The prediction result shows us predicted probabilities for testing dataset. For example, the first obs prediction is 0.61, base on our model, that employee will have a 61% to attrite.

After fitting the model, we use testing dataset to predict the attrition. We will save the prediction results from all three models in this result, evaluate their perfermances and visulize it by the Receiver operating characteristic(ROC) curve which is a very useful diagonistic plot for binary classification in later blogs.

Feature Importance

A great advantage of using XGBoost model is it can help us generate a feature importance table. The importance metric provides a score indicating how valuable each factor was in the construction of the boosted decision trees. Higher relative importance indicates a larger impact on the algorithm and final prediction. Not only does this feature ranking tell us about how important certain features cause for employee attrition but also, we can use it to manually eliminate redundant features. Performing these tasks provide huge benefits such as the speed and simplification of the model, whilst also ensuring greater generalization by reducing the potential for overfitting. Since the importance is calculated explicitly for each attribute, these attributes can be ranked and compared to each other. In table1, we select the 5 most important features of XGBoost model. ***Gain*** is a measurement of contribution of the feature, or we can say it is the improvement in accuracy brought by a feature to the branches it is on. ***Cover*** is a metric of the number of observation related to this feature. ***Frequency*** counts the number of times a feature is used in all generated trees. In Figure 1, we visualize the top 10 important features in histogram using ***Grin*** metric.

	Feature	Gain	Cover	Frequency
1	MaritalStatus	0.15	0.12	0.13
2	NumCompaniesWorked	0.08	0.09	0.04
3	Age	0.07	0.05	0.08
4	EducationField	0.07	0.08	0.09
5	JobInvolvement	0.06	0.06	0.07
6	StockOptionLevel	0.06	0.04	0.04

Table 1: The 5 most important features of XGBoost model

```
xgb.plot.importance(importance_matrix = importance, top_n = 10)
```

When the model is run on the entire dataset, the results show that Marital Status, Number of Companies Worked For, and Age are the dominant drivers of employee attrition for this dataset. By looking at the plots from Part 1, we can determine how these features impact attrition directly.

In terms of HR adjustable parameters, we note that adjustments to Job Involvement, Stock Option and Monthly Income might be used as incentives for high value employees. In addition to this, we can use similar feature importance methods to create an employee specific ranking of controllable factors that can then be used to target employees on an individual-by-individual basis.

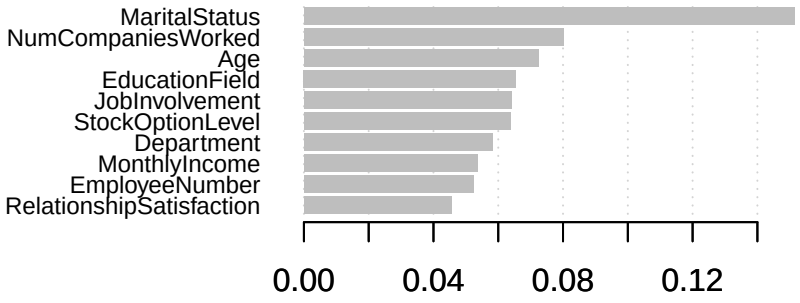

Figure 1: Shows the feature importance for the top 10 features. Marital status, the number of companies worked for and an employees age are the three top drivers of employee attrition.

SVM model

The second model we fit is a support vector machine(SVM) model. Basically, SVM constructs a hyperplane or a set of hyperplanes which has the largest distance to the nearest training data points of classes. Additionally, SVM can enlarge the feature space using kernels and provide more computational convenience. The gamma parameter defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'. The cost parameter is a measurement of 'tolorance' for SVM model, low values meaning 'high tolorance' for misclassification and high values meaning 'low tolorance' for misclassification. We choose radial kernel with proper gamma and cost values here to optimize the performance of SVM.

```
train$Attrition<-train_Y

# Training a SVM
svm_model<-svm(Attrition~.,                #set model formula
               type="C-classification",        #set classification machine
               gamma=0.001668101,              #set gamma parameter
               cost=35.93814,                  #set cost parameter
               data=train,
               cross=3,                        #3-fold cross validation
               probability = TRUE              #allow for probability prediction
)

# Predicting the results using testing dataset
# Obtain the predicted class 0/1
svm_model.predict<-predict(svm_model, test, probability=TRUE)
# Obtain the predicted probability for class 0/1
svm_model.prob <-attr(svm_model.predict,"probabilities")
```

```
svm_model
```

```
##
```

```
## Call:
## svm(formula = Attrition ~ ., data = train, type = "C-classification",
##     gamma = 0.001668101, cost = 35.93814, cross = 3, probability = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  35.93814
##       gamma:  0.001668101
##
## Number of Support Vectors:  339
```

Similar to XGBoost object, svm model object is a list presenting basic information about the parameters, number of support vectors, etc. The number of support vectors depends on how much slack we allow when training the model. If we allow a large amount of slack, we will have a large number of support vectors. A good way to interpret SVM result is to use ***plot.svm***, for the multi-dimensional data matrix here, we have to choose the slice for the data so that the plot can show a 2-Dimension visualization for the model.

Logistic regression model

The third model we fit is a logistic regression model. It uses a logit link from data matrix to class probability. The logit function can always make the predicted value between 0 and 1 which is actually a probability. The logistic regression is famously used for classification problems because of it is easy to be interpreted just like linear model.

```
# Training a logistic regression model
LR_model <- glm(Attrition ~.,family=binomial(link='logit'),data=train)

# Predicting the results using testing dataset
LR_model.predict <- predict(LR_model, test, type = "response")
```

```
coef(LR_model)[2]
```

```
##        Age
## -0.0298873
```

The coefficient for variable "Age" is around -0.03 which is interpreted as the expected change in log odds for an one-unit increase in the employees' age. The odds ratio can be calculated by exponentiating this value to get 0.98 which means we expect to see about 2% decrease in the odds of being an attrition, for a one-unit increase in employee's age. That is to say, if other variables are kept unchanged, the expected probability of attrition is lower when the employee is elder than others.

Now we have finished the model fitting and model intepretation parts, we will save our work for our later blog about model evaluation and feature discussion.

```
# save part2 workspace
save.image("part2.RData")
```

Remarks

We took a simple train/test splitting and fit three models using the training dataset. With a brief interpretation of those models output, we can get a general idea of what the model looks like and how can I interpret those numerical outputs after fitting a model.

The final output of the classifier can be both a ranked list of individuals most likely to leave and also a ranked list of contributing factors that govern each employee's attrition probability. Both are highly useful for any HR department aiming to minimize talent loss and ultimately save company dollars.

For more details on this or any potential analyses, please visit us at (http://sflscientific.com or contact mluk@sflscientific.com.