HR Analytics - Predicting Employee Turnover

# Summary

An organization is only as good as its employees, and as such, this post is about predicting employee turnover. The blog has been inspired by an article on the [Business Science](http://www.business-science.io/business/2017/09/18/hr_employee_attrition.html) website. I thought using an automated machine learning algorithm to find the best model was pretty interesting, so I wanted to give it a try on some data I found on [Kaggle](https://www.kaggle.com/ludobenistant/hr-analytics).

The data and R code can be found on my [GitHub](https://github.com/chrisselig/HR-Analytics) page.

## The Data

**Note: The data is simulated since companies do not like to release their sensitive HR data.**

### Data Features

1. satisfactionLevel: Level of satisfaction (0-1). 1 = highly satisfied.
2. lastEvaluation: Time since last performance evaluation (in Years).
3. numberProject: Number of projects completed while at work.
4. averageMonthlyHours: Average monthly hours worked.
5. tenure: Number of years spent at the company.
6. accident: Whether the employee had a workplace accident or not (0 no, 1 yes).
7. left: Whether the employee left the workplace or not (0 or 1). 0 = stayed.
8. promotion: Whether the employee was promoted in the last five years (0 or 1). 0 = not promoted.
9. department: Department employee works in.
10. salary: Relative level of salary (low, medium, high).

## Exploratory Data Analysis

The data consists of 10 features and 14,999 observations with no missing values. I have turned the data into 3 sets: training, validation, and test. This was done because it is wise not to explore an entire data set because you run the risk of overfitting your model, which means it will not perform well on unseen data. What follows is an exploration of the training data set.

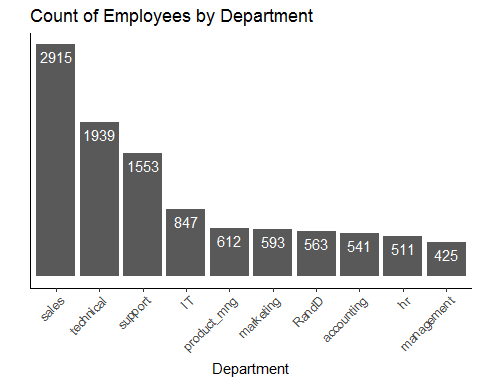
The training data set is a random sample of 70% (10,499 rows) of the total dataset. Some summary statistics for each of the numeric fields are listed below. One immediate observation is the different scales for each feature, especially averageMonthlyHours. Depending on which model I use, I may need to normalize the features. Other interesting information is that the satisfaction level averages 0.61, which a pretty large standard deviation, and finally, the means and medians are very close to each other implying that the distributions are approximately normal.

Summary Statistics for Features

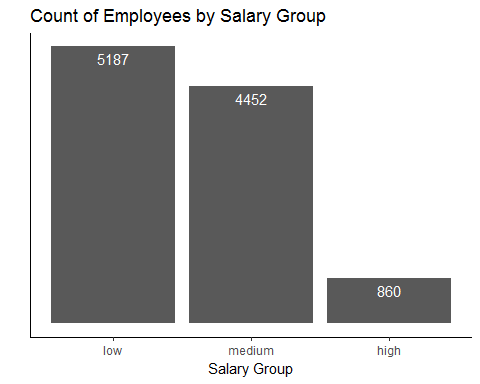
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | mean | median | sd | min | max |
| satisfactionLevel | 0.61 | 0.64 | 0.25 | 0.09 | 1 |
| lastEvaluation | 0.71 | 0.72 | 0.17 | 0.36 | 1 |
| numberProject | 3.81 | 4.00 | 1.24 | 2.00 | 7 |
| averageMonthlyHours | 201.10 | 200.00 | 49.92 | 96.00 | 310 |
| tenure | 3.49 | 3.00 | 1.44 | 2.00 | 10 |
| accident | 0.15 | 0.00 | 0.35 | 0.00 | 1 |
| promotion | 0.02 | 0.00 | 0.14 | 0.00 | 1 |

Moving on, let's look at the two non-numeric features: department employees work in and salary group.

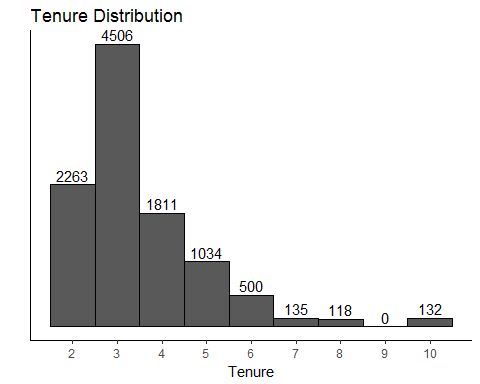
Reviewing the count of employees per department, it looks like the company develops new products and has a large sales force. It looks like the organization may have too many IT and HR people, and probably management.



Salary is grouped into "low", "medium", and "high" categories. Not suprisingly, the "low" category has the majority of people. I would imagine that the "high" category includes most management and top sales people.



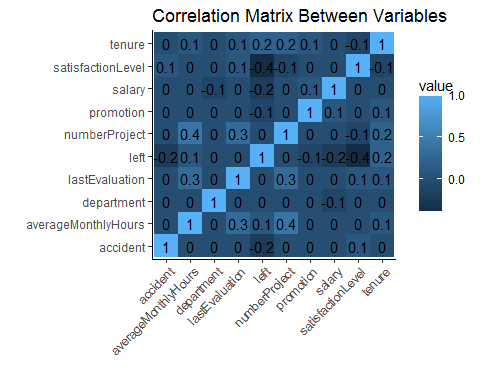
The last employee information we will look at is a histogram of tenure.



It appears the company has a hiring freeze for the last 2 years since there is no one in this data set with under two years of service. There is a huge spike of employees with three years of service and a decreasing amount per year after that.

## Modelling

Moving onto the interesting part, I thought it would be wise to take a quick look at correlations to see if any of the features were highly correlated. If they are, we can probably delete one of them. As you can see below, nothing is really correlated with each other, but I am a bit surprised that the number of projects (numberProject) and average monthly hours (averageMonthlyHours) are not more highly correlated. I guess this company has good project managers that keep everything running smoothly.



Now onto the model selection process. Writing a few lines of code to initialize the package, turn the three datasets into the appropriate format, and we end up with the "best" model. I did not even do any typical data processing, like turning each feature into a binary number, instead thinking I would just let the auto machine learning do its magic.

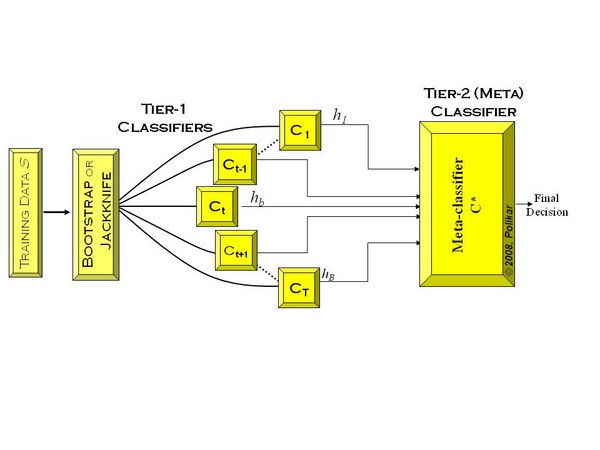
In this case, the model selected was a "stacked ensemble" model (see below).

## Model Details:  
## ==============  
##   
## H2ORegressionModel: stackedensemble  
## Model ID: StackedEnsemble\_0\_AutoML\_20171031\_222838   
## NULL  
##   
##   
## H2ORegressionMetrics: stackedensemble  
## \*\* Reported on training data. \*\*  
##   
## MSE: 0.002479176  
## RMSE: 0.04979132  
## MAE: 0.01848228  
## RMSLE: 0.03353755  
## Mean Residual Deviance : 0.002479176  
##   
##   
## H2ORegressionMetrics: stackedensemble  
## \*\* Reported on validation data. \*\*  
##   
## MSE: 0.01226582  
## RMSE: 0.1107511  
## MAE: 0.03429859  
## RMSLE: 0.07795474  
## Mean Residual Deviance : 0.01226582

I am not going to go over any of the information in the above graphic, because the model statistics are not really that important for the training and validation sets. The one thing I should mention though is that the machine learning algorithm actually uses the training set, and validates the model on the validation set to ensure the model is not overfitting.

Instead of trying to explain the model using my own words, here is an explanation I like. It is from Dr. Polikar from [Scholarpedia](http://www.scholarpedia.org/article/Ensemble_learning).

"In Wolpert's stacked generalization (or stacking), an ensemble of classifiers is first trained using bootstrapped samples of the training data, creating Tier 1 classifiers, whose outputs are then used to train a Tier 2 classifier (meta-classifier) (Wolpert 1992). The underlying idea is to learn whether training data have been properly learned. For example, if a particular classifier incorrectly learned a certain region of the feature space, and hence consistently misclassifies instances coming from that region, then the Tier 2 classifier may be able to learn this behavior, and along with the learned behaviors of other classifiers, it can correct such improper training. Cross validation type selection is typically used for training the Tier 1 classifiers: the entire training dataset is divided into T blocks, and each Tier-1 classifier is first trained on (a different set of) T-1 blocks of the training data. Each classifier is then evaluated on the Tth (pseudo-test) block, not seen during training. The outputs of these classifiers on their pseudo-training blocks, along with the actual correct labels for those blocks constitute the training dataset for the Tier 2 classifier (see Figure 7)."



Neat.

Now that we have that out of the way, how well did the model do on an unseen data set (test set)?

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| accuracy | true\_positive\_rate | precision | null\_error\_rate | true\_negative\_rate |
| 0.99 | 0.95 | 1 | 0.77 | 1 |

The model is highly accurate, predicting 99% of the values correctly. In comparison, if we just picked that the person stays (0) every time, we would have a 77% accuracy, so the model is better than guessing. Even the true positive and true negative rates are pretty high, 95% and 100% respectively. True positive in this case is that the model predicts the employee leaves, when they actually leave and true negative is the model predicts the employee stays when he/she actually stays. Human resources would rather predict that an employee is high risk of leaving when they don't leave than vice versa, so, having a true positive rate of only 95% is a bit of a disappointment. There is normally a tradeoff between true positive and true negative rates, so, it would be interesting to tweak the model to see if we could improve the true positive rate.

# Final Thoughts

The next step would be to delve into the features that actually made the model so accurate and figure out which ones are most important. Knowing what the important features are would provide human resources with a starting point on how to build processes and policies around retaining employees instead of watching them leave. Unfortunately, where this data set is simulated, I do not see it as adding much value to this article.