Machine Learning Engineer Nanodegree

Employee Turnover Prediction

Capstone Project

Kim Alderman
October 11, 2017

Definition

Project Overview

Machine learning (ML) is being applied to many facets of business, from supply chain management to customized/targeted marketing plans. One business area that is gaining traction in the use of data science more broadly is human resource analytics (HR analytics). Data mining was successfully used to evaluate applicants (Chien & Chen) and Harris showed a creative use for machine learning in HR by using incentivized crowdsourcing to review job applicant resumes. There has also been research into using decision trees to assist HR professionals in classifying talent among existing employees and identifying potential targets for promotions.

Even with the current research, there is significant work to be done to fully realize the benefit of HR analytics, both because of the of the potential benefits and the complexity of the problem. It is costly to attract and hire new employees, and employee performance can help drive some key performance indicators (KPI), so HR professionals would benefit from models to improve hiring and retention. However, unlike much of the numerical data streams utilized in other departments like finance and purchasing, HR often relies on more subjective data sources. It is probable that machine learning could be used to uncover hidden factors that improve these prediction models.

I personally became interested in this machine learning application through a rather serendipitous circumstance. A colleague in the quick-serve restaurant (QSR) industry recently hired me to perform an analysis of an employee engagement survey his company had completed. In researching relevant studies for these types of surveys, I found that data analysis (and machine learning) were becoming recognized as important tools for HR. Further, I was fortunate to be granted a scholarship to attend the inaugural O'Reilly Artificial Intelligence Conference (AICon) in New York in September 2016. While there were many discussions around using ML and AI for applications like autonomous vehicles, object recognition, and chat bots, there was virtually no attention to HR issues. Thus, I feel this is an area ripe for further research.

Attracting and training a new employee is costly for companies. Initially, the new employees contribute little to overall productivity or profitability. Over time though, the hope is that there will be a breakeven point, and eventually those upfront costs will be worth it as the employee starts to create value for the company. Therefore, it is important to reduce attrition, particularly of high-performing employees. In researching possible datasets for this project, I found an open dataset on

Kaggle that includes numerous employee features that could be useful for creating a retention model.

Problem Statement

The problem is identifying the high-performing employees that are most likely to leave so that HR can intervene to possibly retain them, thereby reducing turnover of these high-value employees. The machine learning algorithm will be used to predict classification probability for leaving the company. The inputs will be employee data features such as satisfaction and time spent at the company (see next section for detailed discussion of inputs). This classification model can them be used on existing employees to identify the highest risk, high performers, allowing HR to focus retention efforts.

Companies routinely collect data on employees, from the hiring phase all the way through when an employee leaves the company. This project will attempt to build a model to predict employee retention given some of that data as input features. This will be a supervised learning project as the observations (i.e. employees) are labeled with their employment status. The metric will be the accuracy of the model in predicting if the employee is still with the company or has left.

In addition to providing a predictive model of which employees are likely to leave, I hope to gain some insight to provide prescriptive analytics. By observing correlations between high-performing employees (as measured by the last evaluation feature) and the likelihood of leaving, we may be able to identify high-value targets for the company to work with for retention. If a company could know in advance the signs that a high-performer is at risk for flight, concrete actions might be used to increase chances for the employee to remain. Also, because the features used for this project are collected routinely by human resources, it should be possible to use new and updated data to increase the accuracy of the model.

Metrics

While I had originally proposed that the evaluation metric should be classification accuracy, a Udacity reviewed pointed out that because the data is imbalanced (more observations of employees staying than of leaving), simple classification accuracy would not be appropriate. Instead, the metric will be F1-score which is a weighted average of precision and recall (https://en.wikipedia.org/wiki/F1_score). Precision is the number of correct positive predictions (true positive, TP) divided by the number of all positives (TP+FP, where FP is false positive). Recall is the number of correct positive predictions (TP) divided by the number of actual positive observations (TP + FN, where FN is false negative). The F1-score is then calculated by:

$$F_1 = 2 \cdot \frac{1}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Analysis

Data Exploration

The data was obtained from a Kaggle dataset (https://www.kaggle.com/ludobenistant/hr-analytics), which was release under a CC BY-SA 4.0 License. Fortunately this data set is very clean and has no missing values which would require imputation. Input features of the dataset include: employee satisfaction, last evaluation, number of projects, average monthly hours, time spent at the company, work accident (yes/no), promotion in last 5 years, sales, and salary. The dependent variable is the data field indicating whether the employee has left the company.

Employee satisfaction is expressed on a scale of 0-1, with 1 representing highest satisfaction. It seems logical that this may be correlated with retention – unsatisfied employees may be more likely to leave the company. Last evaluation is the overall "score" given to the employee at their last annual evaluation on a scale of 0-1, with 1 representing the highest performers.

The number of projects, average monthly hours, and time spent at company are all expressed in integer form, although average monthly hours will need to be normalized since they are on a different scale than the rest of the features. It will be interesting to see if these features correlate positively or negatively with attrition. Work accident and promotion within last 5 years are binary features (1 yes/0 no). Sales and salary are categorical features denoting the department (i.e. Sales, Accounting, etc.) and salary tier (low, medium, or high). These will need to be encoded into numerical format for several of the chosen algorithms.

There are 14999 employees in the data set. The independent variable is labeled "left" and indicated whether the employee has left the company (left = 1) or is still employed (left = 0). 23.81% of the employees in the data set have left the company, while 76.19% are still employed by the company. It will be important to maintain this ratio when splitting the data for training and testing. There are nine features in the data set. This table provides some summary statistics for the seven numerical features:

	Satisfaction Level	**Last Evaluation**	**Number of Projects**	**Average Monthly Hours**	**Time Spent w/Company**
Mean	0.612834	0.716102	3.803054	201.050337	3.498233
Stan. Dev.	0.248631	0.171169	1.232592	49.943099	1.460136
Minimum	0.090	0.360	2.00	96.00	2.00
25%	0.440	0.560	3.00	156.00	3.00
50%	0.640	0.720	4.00	200.00	3.00
75%	0.820	0.870	5.00	245.00	4.00
Maximum	1.00	1.00	7.00	310.00	10.00

(cont'd)

	Work Accident	**Promotion in last 5 years**
Mean	0.144610	0.021268
Stan. Dev.	0.351719	0.144281
Minimum	0.00	0.00

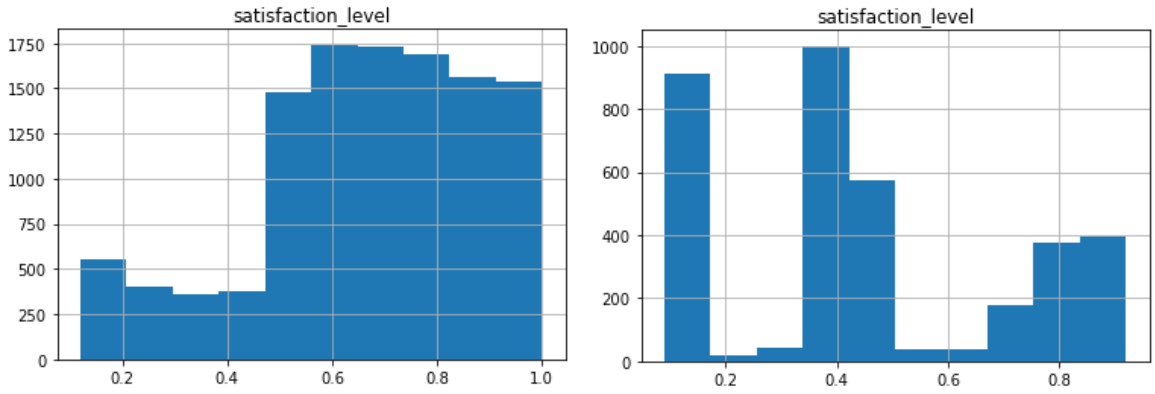
25%	0.00	0.00
50%	0.00	0.00
75%	0.00	0.00
Maximum	1.00	1.00

The departments are listed under the feature “sales” and are as follows: IT, RandD, accounting, HR, m anagement, marketing, product management, sales, support, and technical. The “salary” feature is an ordinal categorical variable with values of low, medium, and high.

Exploratory Visualization

Even with the growth of artificial intelligence and machine learning, an important step in data analysis is visualization. Sometime a simple graph can yield insight and is often a great tool to back up a narrative and deeper analysis. A fundamental assumption underlying this project is that there is a difference between the employees that have left the company and those still remaining. Therefore, I split the data on this target variable and created some histograms to look for differences.

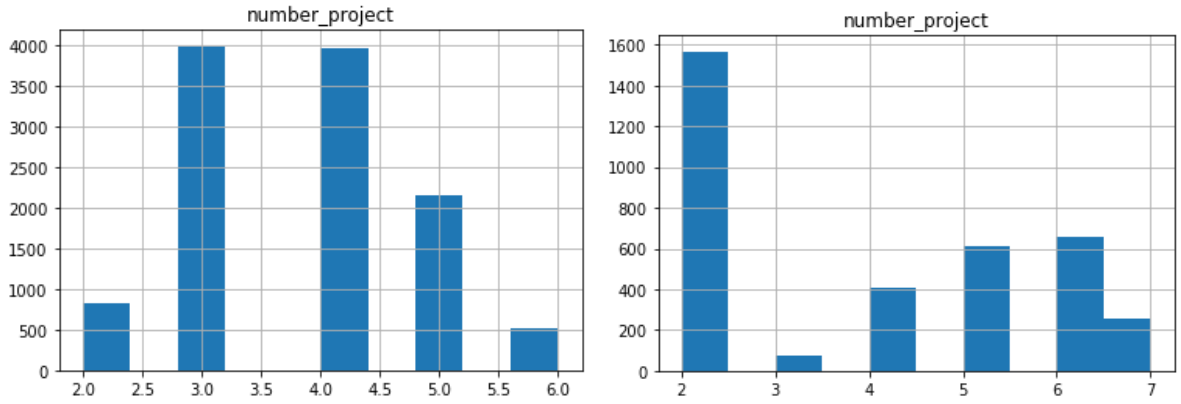
First up was satisfaction level. It seemed logical to expect that this feature would correlate with the independent variable. Satisfied employees would be more likely to stay, while less satisfied employees may be the ones more likely to leave.



(remaining employees on the left; former employees on the right)

As anticipated, the current employees appear to be more satisfied than the ones who have left. A quick heuristic might be for human resources to red-flag any high-performing employee with low satisfaction levels for immediate intervention. They might be at high risk of leaving.

Based on a comparison of the summary statistics of each subset of employees, another possible factor to predict attrition if the number of projects each employee was working on.



(remaining employees on the left; former employees on the right)

This indicates the either too few project (ex. 2) or too many (6 or 7) could be a signal that employee is likely to leave the company. It is not possible to see if the number of projects actually causes an employee to leave, for instance by making them feel underutilized or overworked. It could be that supervisors are assigning less projects to employees they view as high flight risk. Either way, it does appear that the number of projects may correlate with the independent variable.

Algorithms & Techniques

This project is a classification model: can a model be built to correctly predict which employees will leave and which will stay? The following algorithms will be used on this project:

- Logistic Regression
- Support Vector Machines (SVM)
- *K*-nearest Neighbors
- Random Forest

These models all provide predictions of the employee leaving or staying, but there are differences between the methods, including strengths and weaknesses. Logistic regression gets its name from the core of the method, the logistic function, also known as the sigmoid function. Logistic regression's task is to calculate the log odds of an event (e.g. employee leaving). If the likelihood is >0.5, the predicted dependent variable is true, or a 1. This algorithm performs well if the classes are linearly separable, but may not capture more complex relationships. The results are interpretable and it is easy to update a logistic regression model with new data.

Support Vector Machines (SVM) use a maximum-margin loss function to train. The flexibility of SVM lies in its ability to use different kernels. With a linear kernel, SVM performs similarly to logistic regression, but if the data is not linearly separable you can use other kernels (polynomial, radial basis, etc) which allow you to project data onto other dimensions. This allows us to model more complex relationships, but the tradeoff is that they are more memory intensive.

K-nearest neighbors doesn't require training like many other algorithms because it performs computations only upon classification. This is an example of an instance-based learner. The *k* is the number of neighbors that will be used to predict classification. This simple model is easy to

understand and responds quickly to changes in data. However, it is sensitive to local data anomalies, requires most of its computation during testing, and has difficulty with highly imbalanced classes.

Random forest is an ensemble method that uses numerous decision trees. Each tree takes a random subset of the data with a random subset of the independent variables (or features). It then combines the prediction of each tree to make a prediction on a new observation. This model does not presume any linearity nor do the features need to be normalized. Although individual decision trees are prone to overfitting, this is usually mitigated by having a sufficiently large number of trees in the forest.

Benchmark

Based on initial exploratory data analysis, 23.81% of the observations (employees) have left the company's employment, with the remaining 76.19% remaining with the company. The default prediction would therefore be to essentially guess that the employee had not left, yielding an accuracy of classification of 76.19%. My model needs to surpass this accuracy to be of any practical use.

Methodology

Data Preprocessing

The first step was to transform the categorical features ("salary" and "sales", or department) into numerical features. Fortunately, this is easy to implement using the pandas get_dummies function. In could have opted to just replace categorical features with integers but that would imply a ordinal relation. If HR was assigned 3, and IT was assigned 1, does that mean HR is "better" than IT? By using the one hot encoding of get_dummies, non-existent relations were not introduced. This transform does create extra columns. For instance, a feature with N categories will have N-1 additional columns after encoding. However, with a data set of this size, training time was not negatively impacted.

Next, the independent variable "left" was pulled out of the dataframe to create the labels dataframe, Y. Then, the data was split into a training set (80% of the data) and test set (20% of the data) using the test_train_split within sklearn's cross_validation. Stratify was set to yes so that the ratio of leave-to-stay would be the same in the training and test data sets.

Finally, because not all the data was on the same scale, I then performed standardization using sklearn's StandardScaler function. This means each feature had a mean of 0 and standard deviation of 1, ensuring that no features is unduly weighted based on initial scale. Although this was not as important for the random forest model, most machine learning algorithms perform better if features are on the same scale.

Implementation

The appropriate sklearn models were then used train and test the data using the four algorithms discussed above. For each model, a training score, test score, and F1 score was reported. As

discussed in the Metrics portion of this report, F1 was used for evaluating models because of label imbalance – there were far fewer people who had left than people who had stayed.

- Logistic Regression – used LogisticRegression()
- Random Forest – used RandomForestClassifier()
- Support Vector Machines – used SVC()
- *K*-Nearest Neighbors – used KNeighborsClassifier()

An advantage to using Python with libraries such as sklearn is that there are many machine learning algorithms available. Included in this project file is a Jupyter notebook with the Python code required to reproduce this project. However, to demonstrate the simplicity of using sklearn, here is the code for implementing the logistic regression model:

```
logit = linear_model.LogisticRegression()
logit.fit(X_train_std, Y_train)
Y_pred = logit.predict(X_test_std)
logit_score_train = logit.score(X_train_std, Y_train)
print("Training score: ",logit_score_train)
logit_score_test = logit.score(X_test_std, Y_test)
print("Testing score: ",logit_score_test)
print("F1: %2f" % metrics.f1_score(Y_test, Y_pred, average="macro"))
```

The other three algorithms simply required a different classifier, as specified above. With the advent of these easy-to-use models, it is incumbent upon the professional to understand the importance of proper inputs (training/test splits, normalization, etc.) and model parameters for hypertuning and further refining the model.

Refinement

Since random forest appears to have the best performance on this data set, I will work on tuning hyperparameters to improved performance. Although, admittedly, the F1 score is very good for an out-of-the box first attempt. There are basically three hyperparameters we can tweak to improve predictive metrics of a random forest:

1. the maximum number of features (max_features) to be used in the model, particularly useful when there are a large number of features with very small contributions,
2. the number of trees to be used in the random forest model, useful when a very large data set has long training times, and
3. minimum sample leaf size (min_samples_leaf), which limits the minimum size of an end node.

Criterion (the function to measure the quality of the split) is also sometimes tuned, but Gini impurity and entropy typically yield similar performance. Since entropy uses logarithms and can be computationally expensive on large data sets, I usually default to Gini.

Because this data set is not large, attempting to optimize the number of trees will probably not increase performance. With this data set, the more attempts, the better. Attempting to find the best combination of the number of features and minimum sample leaf size will be done through a randomized search.

When conducting a randomized or grid search, it is best practice to hold out a portion of the data from the hyperparameter tuning to estimate the generalization error on a separate validation set. However, because the random forest classifier is a bagged ensemble with a portion of the data already held out in each iteration, we already have a built-in, no-cost set of data for evaluating performance of each combination of hyperparameters. To perform the search, I used sklearn's RandomizedSearchCV function. This is the result for optimal value after 20 iterations:

Hyperparameter	**Values Tested**	**Optimal Values**
Max_features	Random integers 1-9	4
Min_samples_leaf	Random integers 1-9	1

Results

Model Evaluation and Validity

The metrics for each of the four models are presented here.

Model	**F1**	**Training Score**	**Testing Score**
Logistic Regression	0.687201	0.7936	0.8067
K-nearest Neighbors	0.920710	0.9603	0.9413
Support Vector Machines	0.932511	0.9532	0.9507
Random Forest	0.984711	0.9985	0.9890

The random forest model yielded the best F1 score, 0.98. This was much better performance than the base model of assuming all employees would stay (76.2% accuracy). Using the random forest classifier, feature importance was examined to determine which features had the highest contribution to the predictive value of the model. Employee satisfaction level accounted for 35.33%, while time spent at company (18.07%) and average monthly hours (15.65%) were next. This supported what we saw earlier in the exploratory visualization. Satisfaction level is the most important feature in predicting if an employee will leave or stay.

The randomized search for hyperparamaters yielded the following results:

- Max_features = 4
- Min_samples_leaf = 1

Training a new random forest model with these hyperparameters did not result in a performance increase. This is not surprising given the relatively small size of the data set.

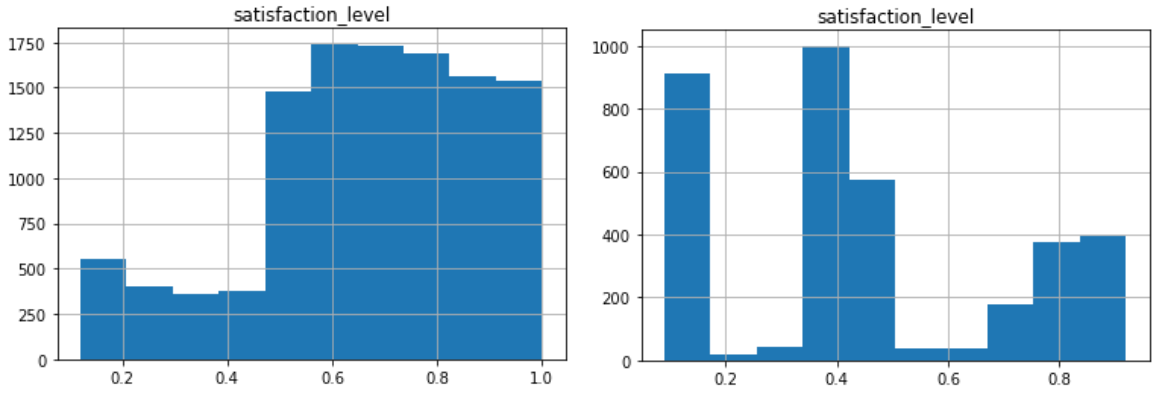
For the random forest model that was chosen as the solution for this problem, the training score was .9985 while the testing score was .9890. This indicates that the model is robust to unseen data. Another way to ensure robustness is data thresholding (limiting numerical features to be with 2nd and 98th percentiles of training data), but this method is not useful for tree based classifiers. Although setting maximum features to 4 did not improve model performance on the test set, I would still use it for new data to minimize chance that model was overfitting training data. Less complex models (e.g. smaller feature set) generally are more robust than very complex models.

The optimized random forest model provides a good solution to the adequate problem. Given an set of input features, it is much better at accurately predicting whether an employee has left or is still employed with the company, compared to the base model (98.9% vs. 76.2%).

Conclusion

Free-Form Visualization

The most important feature in the prediction of turnover was employee satisfaction. As shown in the exploratory analysis section here are the histograms for employee satisfaction scores, comparing current employees on the left, and former employees on the right. The difference is quite stark.



Reflection

The problem was could we identify employees that are likely to leave a company's employment? For this data set, it appears that a random forest classifier was quite accurate at predicting which employees are still with the company and which have left. Based on the importance of employee satisfaction level in making this prediction, these satisfaction scores could be highly useful in spotting employees that might be at risk for turnover.

A caveat here is that this Kaggle data set is simulated. It is unknown if a real company would have data that so clearly delineated the two groups.

Improvement

There are other algorithms that could be tried to achieve better performance or increased interpretability:

- Naïve Bayes – this is typically faster than random forest, so if the data set was very large, this might be viable. However, it usually has a lower predictive value than random forests.
- Gradient Boosting – examples of this include XGBoost and AdaBoost.
- Decision Trees – although this might have a lower predictive accuracy, if I was presenting to a group with less knowledge of machine learning algorithms, the high interpretability might be worth the sacrifice in accuracy.

Because the random forest model performed so well, further experimentation did not seem to be warranted.