Predicting Attrition of Employees using Classification Models

Vini Kalra, Husain Kanchwala, Kaliraj Kalimuthu, Sravani Kandarpa
San Diego State University
vkalra@sdsu.edu, hkanchwala@sdsu.edu, kkalimuthu@sdsu.edu, skandarpa@sdsu.edu

Abstract

In this project, we have tackled a binary classification problem of predicting whether an employee is leaving a company or not, in other words we are predicting employee's attrition. We have evaluated various machine learning models: Support Vector Machine, Logistic Regression, k-Nearest Neighbor and the Random Forest algorithms for predicting attrition of Employee. We have made use of human resource data from the Kaggle repository for the learning models. Finally, we compared the performance of each of these learning models in terms accuracy and execution time of each classification model to see which model perform well in predicting the attrition of an employee.

1. Introduction

Employee churn is a major problem for many firms these days. Great talent is scarce, hard to keep and in high demand. Given the well-known direct relationship between happy employees and happy customers, it becomes of utmost importance to understand the drivers of employee dissatisfaction. In doing so, predictive analytics can be a core strategic tool to help facilitate employee engagement and set up well targeted employee retention campaigns. In this project, we will work on finding best approach to solve this issue and to provide deep inside into the reason behind a employees leaving the company and will zoom in on some of the challenges that accompany this.

Our focus in this project was to gain experience in taking up and solving a classification problem. After obtaining a dataset, as a first step, we tried throwing the classic machine learning algorithms (Logistic Regression, K-Nearest Neighbor and Random Forest) at our problem. Finally, we have compared each of these models and derived good numerical analysis of each of the models used to evaluate the best way to predict employee attrition.

2. Dataset Information

We have made use of human resource data from the Kaggle repository [2]. The problem we address is a binary classification problem to predict if a person will leave the company or not based on few factors.

The main features that are utilized in our model are: employee satisfaction level, last Evaluation, number of projects worked on, average monthly hours, time spend in company, any work-related accident, any promotion last 5 years, sales salary. The description of the data set is as follows. • Number of attributes: 10, after completing data preprocessing we worked with relevant 8 features, Total number of records: 15000, Number of training instances • Number of test instances: 12000 • Number of testing sample 3000. Our Dataset consist of both numerical and categorical data. We have used Support Vector Machine, Logistic Regression, k-Nearest Neighbor and the Random Forest algorithms for the classification task making use of the scikit-learn python package.

3. Data Pre-processing

In machine learning, feature selection is the process of selecting a subset of relevant features to be used in model construction. Feature selection techniques are applied to dataset to eliminate any redundant and irrelevant features. When constructing predictive models, feature selection techniques improve model interpretability, shorten training times and enhance generalization by reducing overfitting. Feature selection is useful part of the data analysis process showing which features are important for prediction, and how these features are related. We have achieved this in two steps: First we have normalized our data so all features are in same range. Second we have employed Correlation Matrix for finding relevant features. Finally, we have performed feature scaling or Standardization. We have normalized our data by converting categorical columns: Salary and Department

into numerical values, so we can further work with only numerical type of feature. Fig (1) is the sample of our data normalizing our data.

3.1. Correlation Matrix

A feature is useful if it is correlated with or predictive of the class; otherwise it is irrelevant. Kohavi and John [KJ96] formalize this definition as

satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_last_5years	department	salary
0.38	0.53	2	157	3	0	1	0	0	0
0.8	0.86	5	262	6	0	1	0	0	1
0.11	0.88	7	272	4	0	1	0	0	1
0.72	0.87	5	223	5	0	1	0	0	0
0.37	0.52	2	159	3	0	1	0	0	0

Figure 1. Converting Categorical into Numerical data

Definition 1: A feature Vi is said to be relevant iff there exists some vi and c for which $P(V_i = v_i) > 0$ such that

$$P(C = c \mid V_i = v_i) \neq P(C = c).$$

Empirical evidence from the feature selection literature shows that, along with irrelevant features, redundant information should be eliminated as well. A feature is said to be redundant if one or more of the other features are highly correlated with it. [1] We have used correlation matrix for finding the correlation and for plotting we have used heatmap. The diagonal elements represent the number of points for which the predicted label is equal to the true label, while off-diagonal elements are those that are mislabeled by the classifier. The higher the diagonal values of the confusion matrix the better, indicating many correct predictions. Fig (2) describes the correlation matrix of our dataset.

We have standardized the features by removing the mean and scaling to unit variance. Centering and scaling happen independently on each feature by computing the relevant statistics on the samples in the training set. Mean and standard deviation are then stored to be used on later data using the transform method. Standardization of a dataset is a common requirement for many machine learning estimators: they might behave badly if the individual feature do not more or less look like standard normally distributed data.[1] **fit_transform()** is employed.

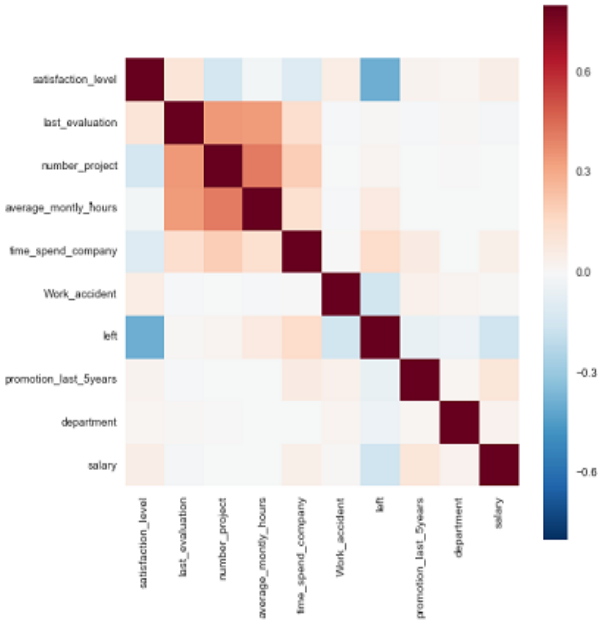

Figure 2. Correlation Matrix

4. Machine Learning Models

In this section, we evaluate the performance of both conventional model and DCRep model using a synthetic, yet realistic workload. Initially, we will present data that compares the latency time improvement of DCRep with conventional model by keeping fixed number of nodes in system and varying file size to be replicated. Later, we will keep fixed size with fluctuating nodes in system to illustrate the benefits of using DCRep under heavy workloads.

4.1. Logistic Regression

Logistic regression is named for the function used at the core of the method, the logistic function. The logistic function, also called the sigmoid function was developed by statisticians to describe properties of population growth in ecology, rising quickly and maxing out at the carrying capacity of the environment. It's an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits.

1 / (1 + e^-value)

Where e is the base of the natural logarithms (Euler's number or the EXP() function in your spreadsheet) and value is the actual numerical value that you want to transform.

Logistic regression uses an equation as the representation, very much like linear regression. Input values (x) are combined linearly using weights or coefficient values (referred to as the Greek capital letter Beta) to predict an output value (y). A key difference from linear regression is that the output value being modeled is a binary value (0 or 1) rather than a numeric value. Below is an example logistic regression equation:

y = e^(b0 + b1*x) / (1 + e^(b0 + b1*x))

Where y is the predicted output, b0 is the bias or intercept term and b1 is the coefficient for the single input value (x). Each column in your input data has an associated b coefficient (a constant real value) that must be learned from your training data. The actual representation of the model that you would store in memory or in a file are the coefficients in the equation (the beta value or b's).

4.2. Random Forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

The first algorithm for random decision forests was created by Tin Kam Ho using the random subspace method, which, in Ho's formulation, is a way to implement the "stochastic discrimination" approach to classification proposed by Eugene Kleinberg.

An extension of the algorithm was developed by Leo Biermann and Adele Cutler, and "Random Forests" is their trademark. The extension combines Biermann's "bagging" idea and random selection of features, introduced first by Ho and later independently by Amit and German inorder to construct a collection of decision trees with controlled variance

The number of features that can be searched at each split point (m) must be specified as a parameter to the algorithm. You can try different values and tune it using cross validation.

For classification, a good default is: m = sqrt(p)

For regression, a good default is: m = p

Where m is the number of randomly selected features that can be searched at a split point and p is the number of input variables.

4.3. K-N Neighbor

The k-nearest neighbor's algorithm (k-NN) is a non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression.

In k-NN classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor. In k-NN regression, the output is the property value for the object. This value is the average of the values of its k nearest neighbors. k-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. The k-NN algorithm is among the simplest of all machine learning algorithms. Both for classification and regression, it can be useful to assign weight to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. For example, a common weighting scheme consists in giving each neighbor a weight of 1/d, where d is the distance to the neighbor.

The neighbors are taken from a set of objects for which the class (for k-NN classification) or the object property value (for k-NN regression) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required. A shortcoming of the k-NN algorithm is that it is sensitive to the local structure of the data. citation needed] The algorithm is not to be confused with k-means, another popular machine learning technique. To determine which of the K instances in the training dataset are most similar to a new input a distance measure is used. For real-valued input variables, the most popular distance measure is Euclidean distance. Euclidean distance is calculated as the square root of the sum of the squared differences between a new point (x) and an existing point (xi) across all input attributes j.

Euclidean Distance: ***(x, xi) = sqrt(sum((xj – xij)^2))***

4.4. Support Vector

In machine learning, support vector machines (SVMs, also support vector networks) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces. When data are not labeled, supervised learning is not possible, and an unsupervised learning approach is required, which attempts to find natural clustering of the data to groups, and then map new data to these formed groups. The clustering algorithm which provides an improvement to the support vector machines is called support vector clustering and is often [citation needed] used in industrial applications either when data are not labeled or when only some data are labeled as a preprocessing for a classification pass. The SVM algorithm is implemented in practice using a kernel.

4.4.1. Linear Kernel SVM

The dot-product is called the kernel and can be re-written as:

K(x, xi) = sum(x * xi)

The kernel defines the similarity or a distance measure between new data and the support vectors. The dot product is the similarity measure used for linear SVM or a linear kernel because the distance is a linear combination of the inputs.

4.4.2. Polynomial Kernel SVM

Instead of the dot-product, we can use a polynomial kernel, for example:

K(x,xi) = 1 + sum(x * xi)^d

where the degree of the polynomial must be specified by hand to the learning algorithm. When d=1 this is the same as the linear kernel. The polynomial kernel allows for curved lines in the input space.

4.4.3. Radial Kernel SVM

Finally, we can also have a more complex radial kernel. For example:

K(x,xi) = exp(-gamma * sum((x – xi^2))

Where gamma is a parameter that must be specified to the learning algorithm. A good default value for gamma is 0.1, where gamma is often $0 < \text{gamma} < 1$. The radial kernel is very local and can create complex regions within the feature space, like closed polygons in two-dimensional space.

4.5. Gradient boosting Classifier

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

The idea of gradient boosting originated in the observation by Leo Breiman that boosting can be interpreted as an optimization algorithm on a suitable cost function. Explicit regression gradient boosting algorithms were subsequently developed by Jerome H. Friedman simultaneously with the more general functional gradient boosting perspective of Llew Mason, Jonathan Baxter, Peter Bartlett and Marcus Freon. The latter two papers introduced the abstract view of boosting algorithms as iterative functional gradient descent algorithms. That is, algorithms that optimize a cost function over function space by iteratively choosing a function (weak hypothesis) that points in the negative gradient direction. This functional gradient view of boosting has led to the development of boosting algorithms in many areas of machine learning and statistics beyond regression and classification.

4.6. Neural Network

Artificial neural networks. An artificial neural network (ANN) learning algorithm, usually called "neural network" (NN), is a learning algorithm that is

inspired by the structure and functional aspects of biological neural networks. Information that flows through the network affects the structure of the ANN because a neural network changes - or learns, in a sense - based on that input and output. ANNs are considered nonlinear statistical data modeling tools where the complex relationships between inputs and outputs are modeled or patterns are found.

An ANN has several advantages but one of the most recognized of these is the fact that it can learn from observing data sets. In this way, ANN is used as a random function approximation tool. These types of tools help estimate the most cost-effective and ideal methods for arriving at solutions while defining computing functions or distributions. ANN takes data samples rather than entire data sets to arrive at solutions, which saves both time and money. ANNs are considered simple mathematical models to enhance existing data analysis technologies. ANNs have three layers that are interconnected. The first layer consists of input neurons. Those neurons send data on to the second layer, which in turn sends the output neurons to the third layer. Training an artificial neural network involves choosing from allowed models for which there are several associated algorithms.

5. Performance Evaluation

In this section, we will evaluate the results of different classification models in terms of accuracy and execution time to see how well that they predict an employee will leave the company or not. We have applied six different classification models such as Linear regression, Support Vector machine, Neural Network, Random Forest, Gradient Boosting respectively. We have also plotted their confusion matrix and tabulated the accuracy and execution time for each model.

5.1. Environmental Setup

Jupyter ipython Notebook and Python language have been used to implement the preprocessing of data and to apply different classification models. We have used few python libraries for achieving different tasks as part of our implementation. Pandas library have been used for loading and handling the data from input csv data file. Sickit Learner library have been used for preprocessing modules and implementing classification model. Numpy and Matpyplot have been used for plotting the heat map and confusion matrix for each learning models. Human resource input dataset is used as input file for all our classification model. It contains 10 features and total number of records 15000. We have applied 6 different classification models which uses only 8 features from the input file for prediction model.

Parameter	Values
Input data file	hr_employee_dataset.csv
Number of features	10
Number of Records	15000
Classification Models	Linear Regression, Nearest Neighbors, Gradient Boosting, Neural Network, Random Forest, Support Vector Machine

Table 1. Systems Parameters

5.2. Classification Models with Accuracy

Figure (3) shows the confusion matrix generated by each classification models. Each model take some unique parameters along the with the input data. The input data is preprocessed and contains only the relevance features for the learning models. It uses K-fold cross validation which leads to more accurate estimate of model prediction performance.

Classification Model	Accuracy(Percent)
Logistic Regression	79.20%
Neural Network	95.60%
Nearest Neighbors	95.80%
Random Forest	99%
Support Vector Machine	78.10%
Gradient Boosting	97.60%

Table 2. Models and their Accuracy

Table (2) shows the accuracy of each classification models for the prediction model. Though most of the models gives accuracy more than 90%, Random forest classification model tops the list with highest accuracy of 99%.Gradient boosting model also performs well and gives accuracy close to random forest model.

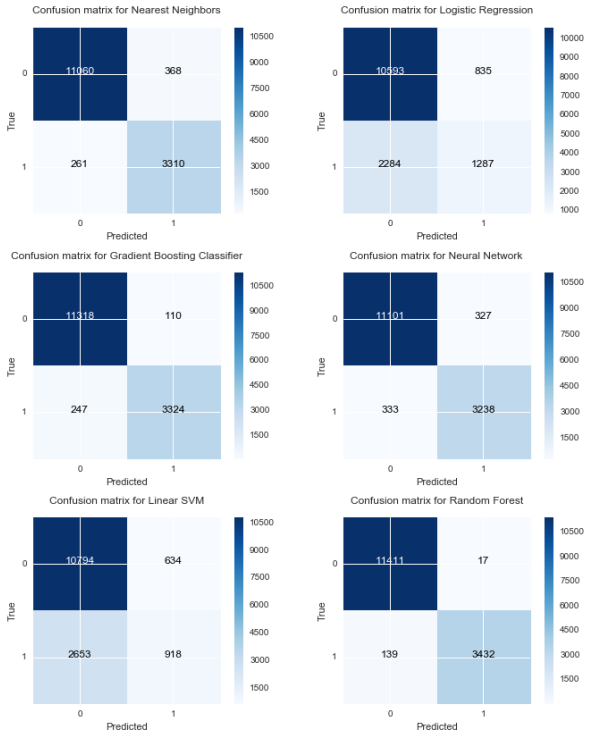

Figure 3. Confusion Matrix for each model

5.2. Classification Models with Execution Time

Table (3) shows the execution time for each classification model applied on human resource input dataset. Though Logistic regression takes less time to process the file, its accuracy is very less when compared with other classification models. Random forest model also performs better in terms of execution time. All other classification models take considerable time to execute.

Classification Model	Execution Time (seconds)
Logistic Regression	0.040s
Neural Network	3.457s
Nearest Neighbors	1.066s
Random Forest	0.385s
Support Vector Machine	5.77s
Gradient Boosting	1.389s

Table 3. Models and their execution time

6. Conclusion

In this project, we experimented different classification models on human resource dataset and evaluated their results to see which model performs well to predict whether an employee will leave the company or not. Input dataset was preprocessed and relevant features were only considered for applying learning models using correlation matrix. We employed six different classification models for the prediction model. After comparing all the algorithms, we see that random decision Forest algorithm has outperformed by achieving 99% accuracy. Although we see logistic regression took least execution time 0.040s, but it has just managed to get only 79.20% accuracy, which is 20% lesser than Random Decision Forest. Hence we conclude Random Decision Forest performed best with human resource data to predict the attrition of an employee.

7. References

[1] Scikit-Learn Package Python - http://scikit-learn.org/stable/

[2] https://www.kaggle.com/datasets

[3] http://rupeshkhare.com/wp content/uploads/2013/12/Employee-Attrition-Risk-Assessment-using-Logistic-Regression-Analysis.pdf

[4]https://thesai.org/Downloads/IJARAI/Volume5No9/Paper_4-Prediction_of_Employee_Turnover_in_Organizations.pdf

[5]https://thesai.org/Downloads/IJARAI/Volume5No9/Paper_4-Prediction_of_Employee_Turnover_in_Organizations.pdf

[6] http://dl.acm.org/citation.cfm?id=1553650

[7] http://ieeexplore.ieee.org/abstract/document/5460732/

[8] S. V. N. Vishwanathan, and M. N. Murty. "SSVM: a simple SVM algorithm". In Neural Networks, 2002. IJCNN '02.Proceedings of the 2002 International Joint Conference on, Vol.3, 2393-2398, Honolulu,HI, USA, 2002