Predicting Employee Attrition Using Machine Learning Techniques

Anindita Saha
saha@sfu.ca

Brian Gerspacher
bgerspac@sfu.ca

Lakshayy Dua
ldua@sfu.ca

Prabhjot Singh
psa83@sfu.ca

Shariful Islam
msislam@sfu.ca

Abstract

In this project we analyze the IBM Watson employee attrition dataset using different machine learning algorithms. This is a binary classification problem. We applied different machine learning algorithms to classify employees as whether they are at a risk to leave their jobs. We addressed imbalance in the dataset and considered the difference in cost between false positives and false negatives.

1 Introduction

Retaining employees is valuable for any organization and predicting attrition is an important aspect for that task. Because of the nature of the problem we study not only the accuracy of predictions but also which features have the greatest effect on results. Organizations may not be willing to make personnel decisions without knowing why an employee has been identified as having a high probability of attrition.

Our objective for this project is to make a comparative analysis among different classifier algorithms. How different algorithms perform in terms of different performance metrics, and how to improve the performance, is our main area of interest. We explore upsampling and cost sensitive analysis for improving the performance and observe that, these methods can be very useful for imbalanced datasets.

To study the problem, we have used a synthetic data set from IBM Watson Analytics [1]. Because of privacy concerns, real-world data sets involving employee data are not widely available. Where they are available, they have been manipulated through normalization and feature extraction making it impossible to draw insights about the original features.

This dataset has 1470 rows and 35 columns and for this problem the `Attrition` column is our target feature, which is a binary feature. The task is to predict this feature based on the other attributes using machine learning algorithms. Only 237 of the 1470 rows relate to cases where the employee did leave the company. This means that we have more than five negative cases for every positive case. Because we have an imbalanced dataset we need to avoid overfitting the negative cases.

The report is organized in a few sections. Section 2 talks about the general methodologies that we followed. Section 3 discusses the algorithms we used for our experiments and different techniques to enhance classifier performance. Section 4 presents the results and discusses the findings.

2 Methodology

We applied several learning algorithms to compare their performances for this classification. Although the internals of the algorithms are different from one another, we followed a standard workflow to evaluate them.

Data Preprocessing

At the very beginning, we eliminated unnecessary columns. The Watson data set has 1470 rows over 35 columns. The `EmployeeNumber` column is a unique identifier that we have assumed does not contain any predictive value. We eliminated three other columns (`EmployeeCount`, `StandardHours`, and `Over18`) that contain the same value for every row.

The remaining 30 columns contain input features covering demographics (2 discrete columns and 1 numeric column), work/education history (1 discrete, 7 numeric), job details (4 discrete, 4 numeric), compensation (6 numeric), performance (1 numeric), and satisfaction (4 numeric). We encoded the discrete categorical variables and normalized the numerical variables.

We also added interaction terms, but they did not seem to improve any performance for this particular case, and thus we decided to drop any analysis using interaction term in this report.

Finally, the `Attrition` column contains the target binary feature.

Training and Testing

For the sake of consistency and reasonable comparison, we separated the dataset into a training set (75%) and a test set (25%) by a random split. We used the same test set for all the experiments so that the results remain consistent and the comparison is fair.

Performance Metrics

For any classifier accuracy is an important metric. But we should also conside true positives (TP), false positives (FP), true negatives (TN), false negatives (FN), recall and precision. Since the cost associated with false positives and false negatives is different, these metrics becomes necessary. We use the receiver operating characteristic (ROC) curve, which summarizes all of the necessary metrics and is also visually easy to understand. For this project, we consider the area under curve (AUC) for ROC as our determinant performance metric.

Class Imbalance

Since the dataset is highly imbalanced with more positive cases than negative cases, we studied how upsampling can improve classification. We have experimented with upsampled data. For most of the algorithms we trained on different cost ratios of false negatives to false positives and we report the best ratios in Table 1.

The tuning of hyperparameters of each algorithm is done during the experiments. The details of hyperparameters is beyond the scope for this report. We report the result with optimized parameters for each algorithm.

3 Classification Analysis

In this section, we discuss the different classification techniques that we used for experiments.

3.1 Different Classification Algorithms

For this project, we did not implement any machine learning algorithm from scratch. Rather, we used standard libraries and looked at different machine learning algorithms and their performances. We used the following algorithms for this project.

- Decision Tree Classifier
- Extra Tree Classifier
- Naive Bayes Classifier
- Logistic Regression Classifier
- Support Vector Machine Classifier
- Gradient Boosting Classifier
- Random Forest Classifier
- XGBoost Classifier
- Adaboost Classifier
- Neural Networks
- Ensemble Method

Other than `XGBoost` [5] and neural networks, we used the `scikit-learn` machine learning library for `Python` [2]. We used `Keras` [6] for machine learning using neural networks. For implementing ensemble method we take votes from Random Forest, Adaboost and Neural Net classifiers and consider the majority prediction as the prediction of Ensemble method. Figure 1 shows ROC curves for different learning algorithms using the original training and test set.

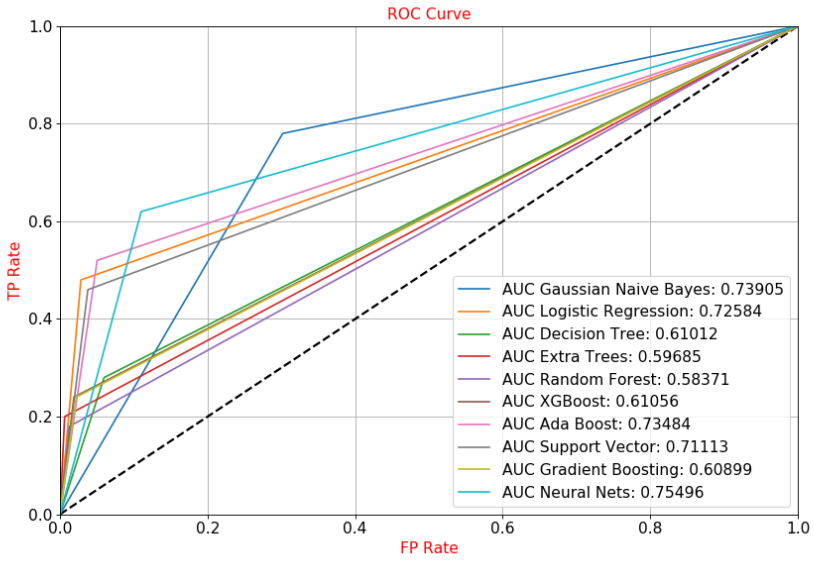

Figure 1: ROC curve for different learning algorithms for original dataset

3.2 Cost Sensitive Learning

The cost of a false negative (failing to identify that an employee may leave) is greater than the cost of a false positive (incorrectly identifying that an employee may leave). To relate false positives to false negatives we used the ROC curve. At any point along the x-axis, we can see the best algorithm at identifying positive cases given a tolerance for false positive cases. Where the cost ratio of positive cases to negative cases is high, we prefer the algorithm at the highest curve on the right side of the graph. The area under the curve (AUC) measures the quality of an algorithm in general.

The best results are reported in Table 1 and discussed in Section 4.

3.3 Balancing the Dataset

The original dataset contains class imbalance. The ratio of the majority class and the minority class is 5.2:1. This means more than 80% of the samples belong to a single class. Predicting all the classes as the majority class will lead to an apparently high accuracy. Another aspect of imbalanced training data is that the models will be trained primarily on the majority class, that is, the negative cases. This may result in a model that in does not efficiently identify positive cases.

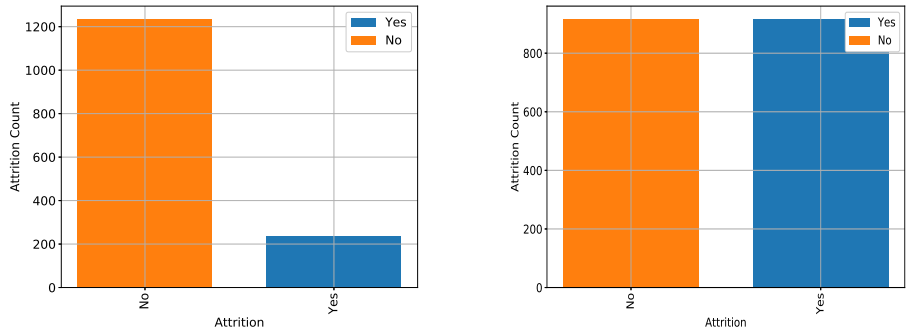

Figure 2: The ratio of positive and negative classes before (a) and after (b) upsampling the train data using SMOTE

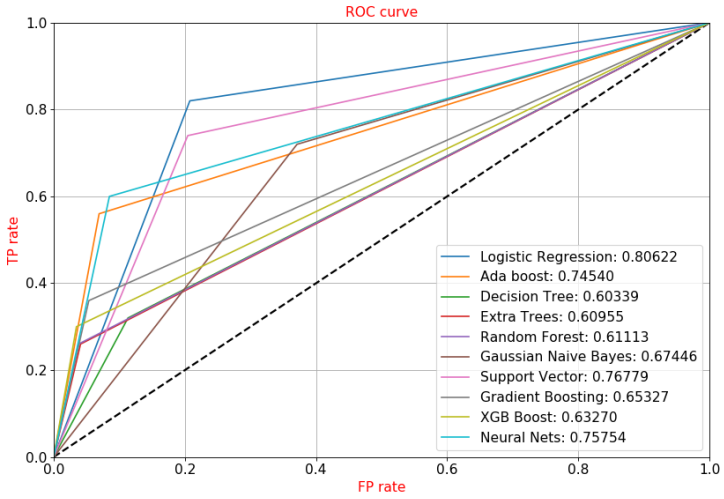

Figure 3: ROC curve for different learning algorithms for upsampled dataset using SMOTE

We can improve this performance in different ways. One way of doing this, is to upsample the available data in such a way that the majority and minority classes are balanced. One method named SMOTE for balancing the dataset is described in [3].

For this project, we are using a library named `imbalanced-learn` which can balance the data according to SMOTE algorithm [4]. To simulate real world scenario, we consider that, we only have the training data (that was used before) available, and the test data will appear at some later time. Therefore, we do not use the test data for the SMOTE algorithm. We keep the same test data (which is used for all the testing) separated for testing purpose only. This ensures that, the comparisons among different learning techniques are consistence and reasonable. Figure 2 depicts the ratio of majority and minority classes before and after balancing.

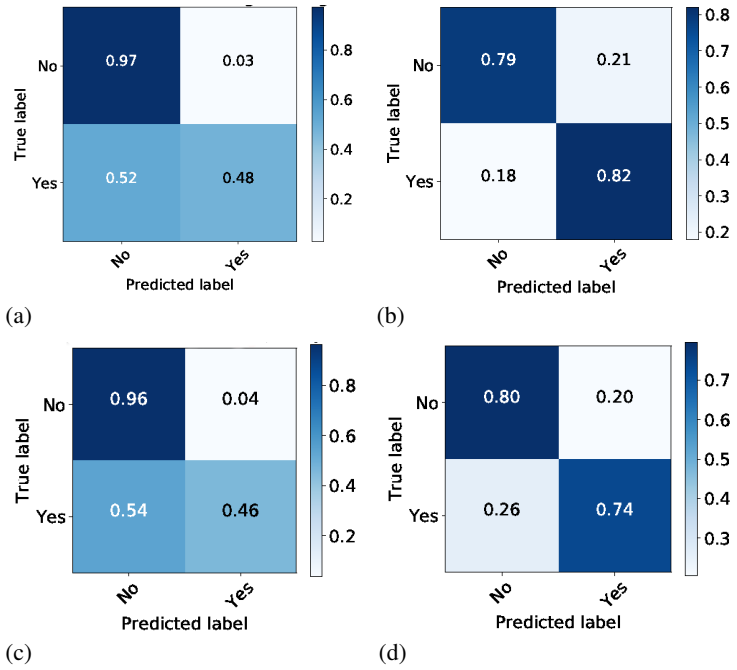

Figure 4: Confusion matrix for (a) logistic regression without SMOTE (b) logistic regression with SMOTE (a) SVC without SMOTE (d) SVC with SMOTE

Figure 3 shows the ROC curves for different learning algorithms. If we look at the AUCs for these ROC curves and compare it with Figure 1, we notice improvement. Figure 4 shows normalized confusion matrix for LR and SVC before and after class balancing. It is clear from the figure that, with a sacrificed rate for TP, we have significant improvement in terms of FP. Thus, balancing the classes in the training set by upsampling, can improve the performance of some classifiers.

3.4 Feature Importance

Some models allow us to tell which features have a greater effect on the classification than others.

Because neural nets have so many interactions between nodes there are a huge number of weights trained it is difficult to draw any conclusion about feature importance.

Decision trees can be inspected to identify the most important features. The split in the first one or two levels of the tree are likely to affect more classifications so we can assume that the features used in these splits are the most important. This is subjective because other features may appear in many splits further down the tree, but the first splits are the best candidates for important features.

Logistic regression trains weights for each feature. This makes it easy to quantify which features are most important based on the magnitude of the feature's weight.

Below are the top five apparent most important features according to the different algorithms.

Decision Tree	Gradient Boost	Logistic Regression
OverTime	OverTime	OverTime
YearsAtCompany	EnvironmentSatisfaction	Non-Travel
MonthlyIncome	JobInvolvement	Education_Other
EnvironmentSatisfaction	MonthlyIncome	Divorced
Age	RelationshipSatisfaction	Gender

`OverTime` is the most important feature for all three algorithms. `EnvironmentSatisfaction` appears in the top five of two of the lists. Otherwise, the features that appear are a mix of demographics, job details, and work history features.

4 Conclusions

Table 1: Results of classification analysis using different classification techniques

Learning Algorithm	Acc(%)	TP	TN	FP	FN	Prec	Recall	AUC
Naive Bayes	70.92	39	222	96	11	.29	.78	0.73
Naive Bayes	64.13	36	200	118	14	.23	.72	0.73
Dec Tree	85.05	14	299	19	36	.42	.28	0.61
Dec Tree (Balanced)	82.61	21	283	35	29	.38	.42	0.61
Extra Tree	86.95	12	308	10	38	.55	.24	0.58
Extra Tree (Balanced)	86.41	9	302	16	41	.36	.18	0.61
Log Reg	90.48	24	309	9	26	.71	.48	0.73
Log Reg (Cost 1:2)	88.85	27	300	18	23	.60	.54	0.74
Log Reg (Cost 1:3)	89.13	30	298	20	20	.60	.60	0.77
Log Reg (Balanced)	79.61	41	252	66	9	.38	.82	0.81
Random For	87.50	6	316	2	44	.75	.12	0.56
Random For (Balanced)	86.15	12	307	11	38	.52	.24	0.58
SVC	89.40	23	306	12	27	.68	.46	0.71
SVC (Cost 1:2)	87.50	30	292	26	20	.65	.60	0.75
SVC (Cost 1:2.5)	86.14	32	285	33	18	.49	.64	0.76
SVC (Balanced)	78.80	37	253	65	13	.36	.74	0.71
Grad Boost	87.77	12	311	7	38	.63	.24	0.61
Grad Boost (Balanced)	86.68	18	301	17	32	.51	.36	0.61
XGBoost	88.04	12	312	6	38	.60	.24	0.73
XGBoost (Balanced)	87.50	15	307	11	35	.58	.30	0.73
Neural Net	85.33	31	283	35	19	.47	.62	0.83
Neural Net (Balanced)	87.23	30	291	27	20	.52	.60	0.84
Ensemble of Rand Forest, Ada Boost, Neural Net	90.48	24	309	9	26	.72	.48	-
Ensemble of Rand Forest, Ada Boost, Neural Net (Balanced)	89.40	26	303	15	24	.63	.52	-

While logistic regression had the highest accuracy when using imbalanced training data, naive Bayes had the highest AUC score. In a scenario where false negatives have the same weight as false positives, logistic regression is best. But in the more plausible scenario that employee attrition has a high cost and false negatives are more important than false negatives, naive Bayes is the best choice simply because it makes so many positive predictions. Ensemble method, which we implement by taking votes from three different classifiers (Adaboost, Random Forest and Neural Net) also works very well with around 90% accuracy and a low count of FP.

When upsampling the training data, the accuracy of many algorithms decreases. In particular, the accuracy of logistic regression goes from 90% to 80%. However, its number of false negatives decreases significantly. Because of its high AUC, we can conclude that it effectively identifies positive cases without making too many positive predictions.

In terms of feature importance, whether an employee works overtime is the most important feature in determining whether the employee will leave his/her position.

Contributions

1. **Anindita Saha** - Decision Tree Classifier, Extra Trees Classifier, feature importance. Reporting observations before and after using interaction terms in the classifiers. Plotting of decision tree.

2. **Brian Gerspacher** - Data set pre-processing, Neural Networks, algorithm comparison.

3. **Lakshayy Dua** - Data set pre-processing, Gaussian Naive Bayess Classifier, ADA Boost Classifier, Ensemble learning.

4. **Shariful Islam** - SVM Classifier, Gradient Boosting Classifier, ROC curves, confusion matrix. Identifying the problem of dataset imbalance.

5. **Prabhjot Singh** - Logistic Regression Classifier, XG Boost Classifier. Resampling data using SMOTE, ROC curves.

References

[1] SAMPLE DATA: HR Employee Attrition and Performance - https://www.ibm.com/communities /analytics/watson-analytics-blog/hr-employee-attrition/

[2] scikit-learn: Machine Learning in Python - http://scikit-learn.org/stable/

[3] Nitesh V. Chawla , Kevin W. Bowyer , Lawrence O. Hall , W. Philip Kegelmeyer (2002) 'SMOTE: Synthetic Minority Over-sampling Technique', *Journal of Artificial Intelligence Research*, volumne 16, pp. 321–357.

[4] imbalanced-learn API - http://contrib.scikit-learn.org/imbalanced-learn/stable/generated/imblearn.over_sampling.SMOTE.html.

[5] XGBoost - http://xgboost.readthedocs.io/en/latest/model.html

[6] Keras: The Python Deep Learning library - https://keras.io/

[7] Charles X. Ling, Victor S. Sheng 'Cost-Sensitive Learning and the Class Imbalance Problem' *Encyclopedia of Machine Learning*, Springer, 2008.