CS36110: Employee Absenteeism

Ben Jaeger

November 24, 2019

Contents

1 Initial Building of Machine Learning Models

1.1 Initial Classifier Creation and Explanation

For this assignment I used a J48 and NaiveBayes classifier as prescribed in the assignment, however I also used a RandomForest classifier as I had some knowledge of it from previous work.

J48 is a decision tree-based classifier. It works by arranging the features into a tree where at each level the feature chosen splits the dataset with the most homogeneity in each branch. One advantage of J48 is that it is a very simple model to both produce and, arguably more importantly in some situations, a explain to someone who has little or no knowledge of machine learning.

NaiveBayes is a probabilistic classifier, meaning it uses the given dataset to find the probability of an instance belonging to a particular decision group. This is done by using each feature to update the likelihood that an instance belongs to a given class. NaiveBayes has the double edged sword of disregarding the interactions between features (hence why it's naive) this reduces the calculations needed greatly but is a very simplistic model. It also easily adapts to more data as there is nothing to restructure such as a decision tree.

A RandomForest is an example of an ensemble learning model, meaning that it is a model made up of smaller models, attempting to negate its/their negatives. A RandomForest is made up of lots of decision trees, the difference being, each is trained on a subset of the data and at each level it picks from a subset of the features, then, when classifying, a majority vote is taken between them. This greatly reduces the over-fitting decision trees are liable to have happen [Liaw et al., 2002].

1.2 Comparison Between Classifiers

When comparing machine learning models there are many ways to measure and compare them. We will be looking at the F1 Measure and the ROC Score.

A useful way to compare classifiers is to look at their F1 scores, this is the harmonic mean between the precision (the number of true positives divided by all predicted positives, how often the model is correct when it thinks it is) and the recall (the number of true positives divided by actual positives, how many positives the model misses) [Sasaki et al., 2007]. Due to the nature of the precision and recall there is often a trade-off between them, therefore by considering both we can very effectively evaluate a model's accuracy.

The ROC Score compares the model to a random guess by graphing the True Positive Rate (aka the recall as mentioned before) and False Positive Rate (FPR is false positives divided by all actual negatives) against each other at various levels of threshold, the ROC area being the area beneath the curve as plotted, 0.5 indicating a classifier being no better than a random guess and 1 being perfect classification.

Below are the score for each classifier, the best in bold, as we can see RandomForest outperformed both NaiveBayes and J48 in both metrics.

Classifier	F1 Measure	ROC Score
NaiveBayes	0.549	0.782
J48	0.718	0.854
RandomForest	**0.788**	**0.934**

1.3 A Baseline for Comparison

A very basic baseline would be a "blanket" assignment of a particular decision class, under this approach we would give a result of, for example, "A" to any data given, in this case resulting in approximately a 25% accuracy rate, given all the classifiers achieved higher than this we can say they are all at least better than this "blanket" approach, this is a fairly unsurprising result as a blanket baseline is particularly useful in unbalanced datasets however this is not (being an approximately equal split between each 4 decision classes).

Another good baseline would be a random approach, such as that used when comparing ROC scores, given that a 0.5 ROC score is equivalent with a model being no better than a "coin flip" approach and all of the three models discussed beat this by a considerable margin we can say they surpass this baseline too.

A third baseline and a well-accepted industry standard for comparison is the NaiveBayes classifier in itself, of course NaiveBayes is comparable with itself so that classifier is good enough and referring back to the comparison made in section 1.2 we can say that RandomForest and J48 are both good enough too given they did better in both F1 measure and ROC score than NaiveBayes.

2 Dataset Imputation and Reclassification

2.1 Dataset Imputation and Justification

The dataset given with this assignment has several missing values in it, namely 46 (7%) in *Reason_for_absence* and 3 (<0.5%) in *Month_of_absence*. There are a number of ways to classify missing values, namely:

- MCAR or Missing Completely At Random (there is no relationship between the missingness and the data)
- MAR or Missing at Random (there is a relationship between the missingness and the data but not the missing value)
- MNAR or Missing Not At Random (the missing data and the rest have a strong relationship)

I imported the dataset into R [R Core Team, 2019] so I could analyse the data (using the package farff [Bischl and Bossek, 2019]). If the *Reason_of_absence* was MCAR the distribution of each feature for both missing and non-missing reasons would be the same, however we see this is not the case (plots were made with ggplot2 [Wickham, 2016]). As can be observed there is a vast difference between the distributions of each feature depending on if there's an associated reason for the absence.

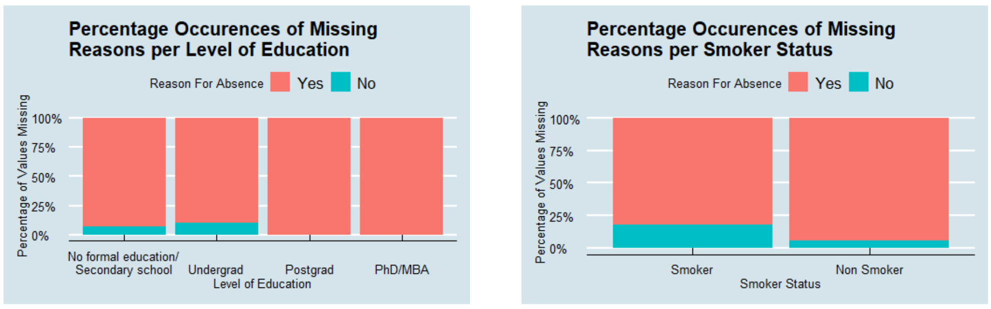
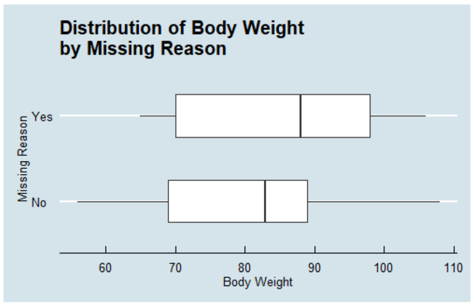


Figure 1: Some plots to show the percentage missing depends on a feature's value and is not uniform

The fact that the missing data is not MCAR, it represents a significant proportion of our dataset and removing it would change the distributions of our features, we must therefore impute the data [Cheema, 2014]. Given the data was already R, I used a package called mice (Multivariate Imputation by Chained Equations) [van Buuren and Groothuis-Oudshoorn, 2011]. mice works by replacing each variable with a starting replacement (usually the mean or mode) it then using all of the other variables to impute each



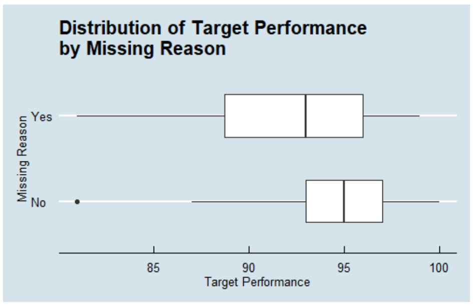


Figure 2: Some plots to show the distribution of a feature depends on the missingness of the reason and is not identical

column one by one, replacing those values that were originally missing, the idea being that after many rounds of this the values will converge together [Azur et al., 2011].

There were several reasons for using a MICE approach, namely, it works with MAR data as we proved was the case, it respects the existing variance of the features unlike a mean or mode replacement (such as what WEKA's filters use) and it works with categorical data unlike most imputation methods. However it should be acknowledged that MICE has flaws, for example, research has yet to be done on the best number of iterations to perform and there is no mathematical basis to this method of imputation unlike other methods.

2.2 Reclassification and Re-Evaluation

After imputation next was to reclassify on the new data and compare against the old classifiers. I loaded the .arff file into WEKA and retrained the same three classifiers, NaiveBayes, J48 and RandomForest. Again we will compare their F1 measures and ROC scores, the better of the two being marked in bold.

Classifier	F1 Measure		ROC Score	
	Old	New	Old	New
NaiveBayes	0.549	**0.554**	0.782	**0.784**
J48	**0.718**	0.698	**0.854**	0.847
RandomForest	**0.788**	0.751	**0.934**	0.858

Interestingly, and personally unexpectedly, we can see that NaiveBayes did slightly better (though really negligibly so) whereas J48 did slightly worse and RandomForest even more so. Most likely this is due to faults in the imputation method, maybe some features should have been excluded from contributing towards the imputation and the method of making the models could have been wrong too, here mice used multinomial logistic regression but maybe another method would be more appropriate. Of course the possibility can't be ignored that this method of imputation was inappropriate too, maybe training a classifier on complete case data and using it to find the missing values would have worked better.

3 Decision Trees Comparison

As per the assignment I used WEKA to remove the *Month_of_absence*, *Day_of_the_week*, *Season*, *CommuteDistance*, *YearsService*, *WorkLoad*, *Education* and *Smoker* features before training a new J48 classifier to see how this would affect it's performance. As before we will compare their F1 and ROC metrics to evaluate performance at a glance.

J48	F1 Measure	ROC Score
Original	**0.718**	0.854
Reduced	0.700	**0.862**

As we can see despite the huge reduction in data given the classifier was still trained to a relatively good degree, in fact neither the original or reduced dataset models won outright.

This lack of a big difference between them implies that the data excluded was unnecessary for a good quality of classification.

Next I looked into the actual structure of the trees. The first observation I made was that the original tree would often split the data on *Month_of_absence* and *Day_of_the_week* making for a very wide tree with a lot of leaves, according to WEKA the original had 157 leaves but the reduced only had 87. While the original often split the data with *Month_of_absence* or *Day_of_the_week* under the reduced dataset it makes frequent use of *Target*.

My next observation was that the reduced dataset tree was not only less wide but also less deep, it only had seven levels at it's deepest point and normally only went 3 or 4 deep, whereas the original went 9 deep twice and usually was about 5 levels deep. This means that the reduced tree was much more efficient with its classifying.

I think this is a different classification problem as the original because the trees were built from different datasets in that they had a different selection of features, the results were massively different because J48 is a greedy algorithm and it makes some sense that the features that split the dataset best would be those with the most values, however this made the original actually very inefficient. The difference in problem is plain to see in that the trees had very different depths, leaf counts and parameters at each level after the first.

Final word count: 1726

References

[Azur et al., 2011] Azur, M. J., Stuart, E. A., Frangakis, C., and Leaf, P. J. (2011).
Multiple imputation by chained equations: what is it and how does it work?
International journal of methods in psychiatric research, 20(1):40–49.

[Bischl and Bossek, 2019] Bischl, B. and Bossek, J. (2019).
farff: A Faster 'ARFF' File Reader and Writer.
R package version 1.1.

[Cheema, 2014] Cheema, J. R. (2014).
Some general guidelines for choosing missing data handling methods in educational research.
Journal of Modern Applied Statistical Methods, 13(2):3.

[Liaw et al., 2002] Liaw, A., Wiener, M., et al. (2002).
Classification and regression by randomforest.
R news, 2(3):18–22.

[R Core Team, 2019] R Core Team (2019).
R: A Language and Environment for Statistical Computing.
R Foundation for Statistical Computing, Vienna, Austria.

[Sasaki et al., 2007] Sasaki, Y. et al. (2007).
The truth of the f-measure.
Teach Tutor mater, 1(5):1–5.

[van Buuren and Groothuis-Oudshoorn, 2011] van Buuren, S. and Groothuis-Oudshoorn, K. (2011).
mice: Multivariate imputation by chained equations in r.
Journal of Statistical Software, 45(3):1–67.

[Wickham, 2016] Wickham, H. (2016).
ggplot2: Elegant Graphics for Data Analysis.
Springer-Verlag New York.