A comparison of three algorithms for missing data Imputation

Szu-Ling Chen

Li-Tse Hsieh

Abstract

In the reality, the dataset for machine learning, especially in the medical data, is rarely clean and often has missing values or corrupt data which would cause significant influence and errors with machine learning algorithms on improving the quality of therapy. In this report, we present a method to impute missing data based on the relation results by the following steps: Exploratory Data Analysis (EDA), Missing Data Imputation, and Model Selection and Training. We also evaluate the accuracy by cross-validation in order to avoid overfitting. The results show that Neural Network have better performance in predicting these attributes in general compared to Random Forest and others.

I. Introduction

In the past decade, machine learning has been widely used for medical research, such as disease prediction and classification. However, the best test of any algorithm is how well it works upon our own data set. In this report, we apply many machine learning algorithms to predict the missing value of each of the five target features. Our goal is to predict those missing values as accurately as possible. We hope to bring more insights in this dataset, which comes from The Healthcare Cost and Utilization Project (HCUP), to improving both healthcare environment and equality of the therapy in healthcare industry.

II. Background

As there are many machine learning models out there to be used, we inspect our features carefully and choose the following models to train our data.

- **Random Forest**

Since most of our features are category numbers, we go for Random Forest as one of our classifiers. The model Random Forests is one of the most popular classifiers. It comes from the method of training each tree by given a set of simple trees and a set of random predictor variables, and defines a margin function that measures the extent to which the average number of votes for the correct class exceeds the average vote for any other class present in the dependent variable. [1] That is, a random forest takes a random subset of features from the data, and creates n random trees from each subset. Trees are aggregated together at end, which can be seen from the figure 1.

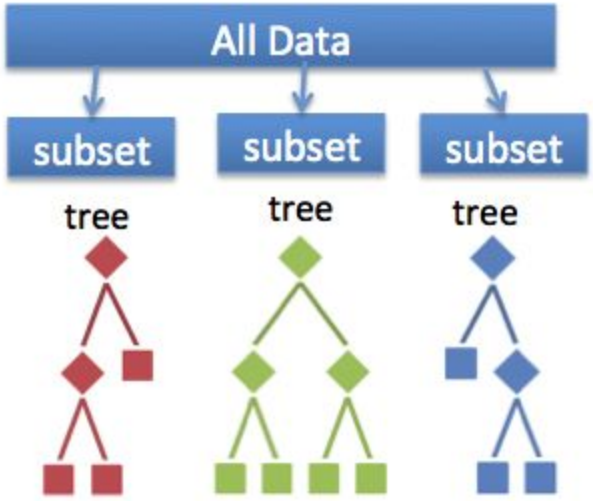

Figure 1

There are many parameters can be adjusted, the most important settings we applied to are the following:

1. **n_estimators: number of trees. The larger the better, although improvements become marginal eventually.**

2. **max_features: number of random features per subset. Lower numbers decrease variance and increase bias. Rule of thumb for classification: max_features = sqrt(all_features).**

- **Light GBM**

We also apply Light GBM to train our data. For all we know is that Light GBM has been viewed as a fast, distributed, high-performance gradient boosting framework based on decision tree algorithm, used for ranking, classification and many other machine learning tasks. Since it is based on decision tree algorithms, it splits the tree leaf wise with the best fit whereas other boosting algorithms split the tree depth wise or level wise rather than leaf-wise , which can be seen from figure 2. [2] The advantages of the algorithms are that it is faster in training speed and has higher efficiency as well as lower memory usage.

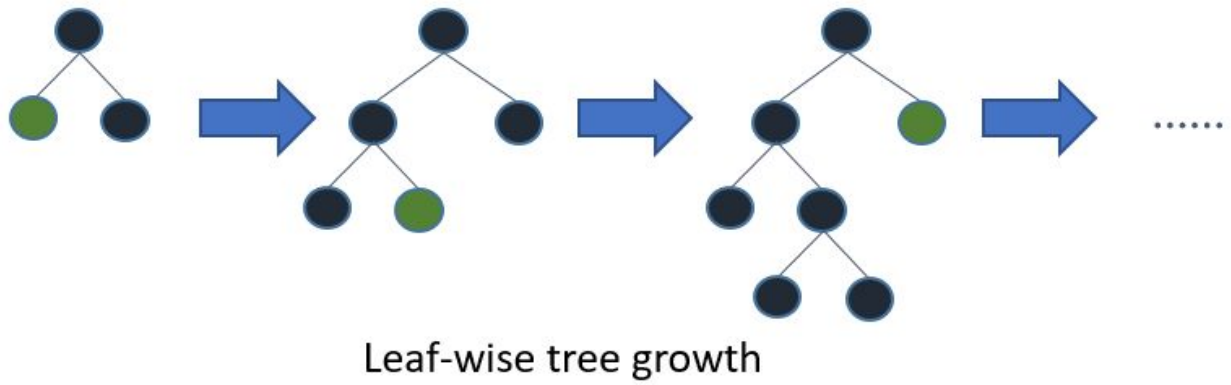

Figure 2

- **Neural Networks**

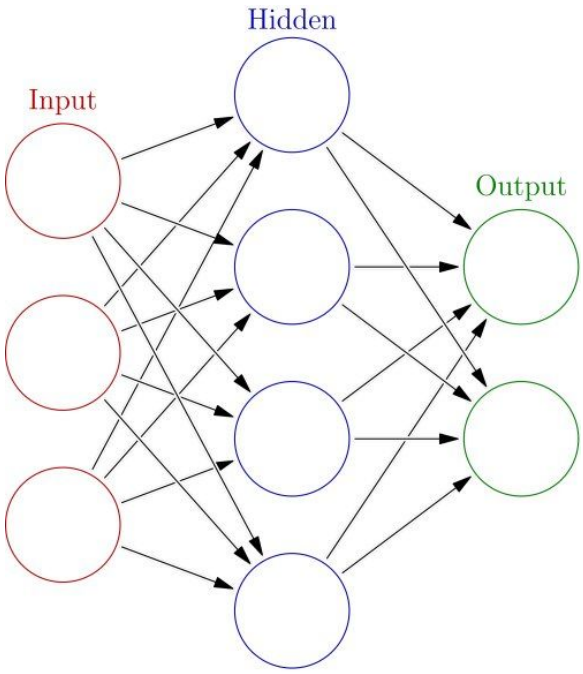

Figure 3

Neural Network model is one of the most common supervised machine learning models. The original idea was inspired from biology which is composed by a set of connected nodes called neurons. In this report, we applied one of Neural Network implementations called Multi-Layer Perceptron (MLP) in Keras [3]. Figure 3 shows the structure of MLP Neural Network. Neural Network is constructed by three kinds of layers, Input, Output and Hidden layers. Intuitively, the data comes in from Input layer and comes out from Output layer. Input dimension decides how many input variables (features), and we also need to specify the number of neurons, network weights and activation function for each layer [4]. In this report, we used three layers (one hidden layer) with 128, 64 and 1 neurons for each layer.

III. Exploratory Data Analysis

- **Initial Exploration**

At first, we look at our data in as many different ways as possible, including checking data types, understanding the meaning of each feature, visualizing data in various

types of graphics. We ask several questions about the data such as how many numerical and categorical data we have and what their typical shape and content are. These questions help us get the insights to the data more precisely. During the process, some properties and connections are shown obviously. Generally speaking: this is the part where the detective finds the clues.

- **Missing Data Imputation**

After exploring the data, we move on to the next step: Missing Data Imputation. Knowing about missing values is important because they indicate how much we don't know about our data and how many data we can fix. At a broad overview, we notice we have 56 columns with one or more missing values. We have the following insights for these features:

- **DXCCS1-DXCCS25 : As can be seen in the figure 4, the number of missing data has an increase in this category. Therefore, we assume that patients stop receiving the therapy for somehow reasons. Therefore, we fill in the missing value with 0 as a new classification showing patients are no longer proving the information.**

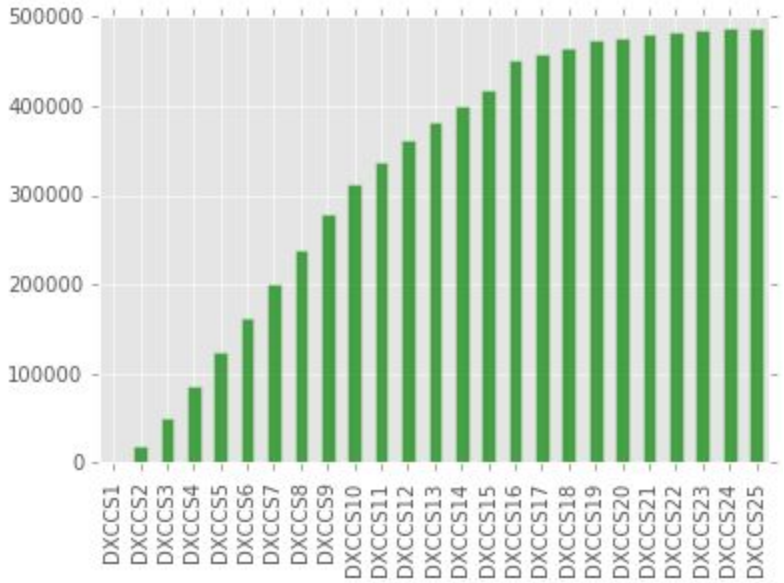

Figure 4

- **PRCCS1 - PRDAY15: As can be seen in the figure 5, the number of missing data has an increase in this category. We make the same assumption like we have in DXCCS features. Likewise, we replace missing value NAN with 0.**

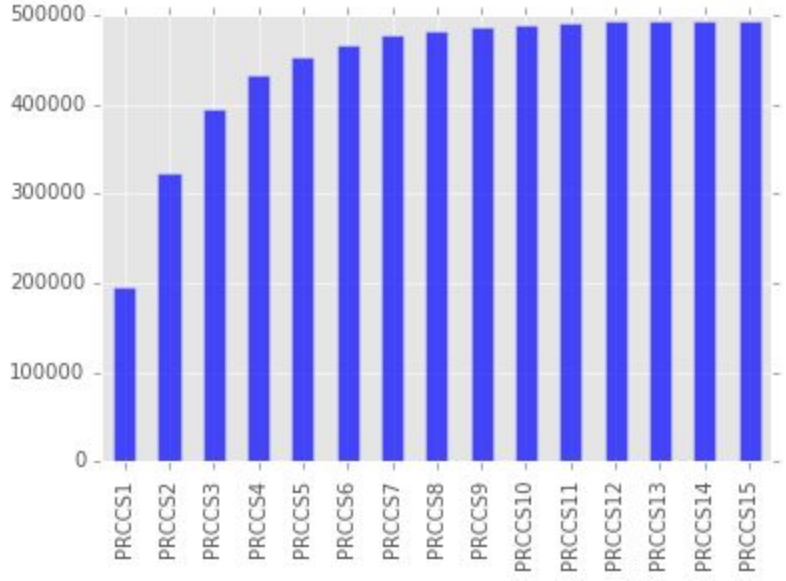

Figure 5

After dealing with the features above, we have other 16 columns left with their missing value to cope with. However, since the number of the missing value in each rest of the features are lower than 1000, which compared to the total records are relatively small, we decide to delete all the raws with missing data. We end up having 493,420 records in our new dataset.

IV. Experiment

The whole experiment process is designed as two parts: model training and testing. Before moving on to the modeling stage, we split the training sample into two sub-samples: training and testing with the portion of 70% of the dataset for testing and 30% of the dataset for testing. In our points of view, it should not be tested on the same data they are trained on. This avoids overfitting. Next, we start applying the different models that we mentioned in the previous paragraph to train

and test our data. Our goal is to predict these five features, including RACE, ZIPINC, ASOURCE, ATYPE, and TOTCH. We try our best to bring out the best accuracy rate of each of these five attributes in our model with carefully evaluating the performance of our models. For example, we used cross-validation to compute the scores for our random forest classifiers. Each model has its own feature and for each attributes, we iteratively test them among these model in order to reach the highest accuracy rates. We will be presenting our results in the following sections.

V. Results

We decide to choose the best performance in each model for presenting our final results. Also, We came to realize that the TOTCH attribute can be turned into category number for the prediction. However, it is not as precise to predict it using classifier for continuous number as using Neural Networks. Therefore, in our case we apply Neural Networks and LightGBM to training this feature instead of using Random Forest. Even though we can't really compare all of the models together, we do have an acceptable accuracy rate to present for each of the target attributes. The final results has shown in table 1. The results show that Neural Network and Random Forest have better performance in predicting these attributes in general, while Light GBM has only been used in the prediction on the continuous number in our case.

Table 1: Accuracy Rates

Attributes	Model	Accuracy
RACE	Random Forest	74.4 %
	Neural network	99 %
ASOURCE	Random Forest	20.4 %
	Neural network	100 %
ATYPE	Random Forest	80 %
	Neural network	88 %
TOTCH	LGBM	52.4 %
	Neural network	72 %
ZIPINC	Random Forest	56.5 %
	Neural network	73 %

VI. Conclusion

Based on the results that we have, Neural Networks and Xgboost have better performance in predicting these attributes by far. At the end, we have came accross to the conclusion that each of the individual classifiers we have used above has its strengths and weaknesses, and we should always choose the classifier or models that's best equipped to handle a certain problem and/or has been found to perform with the highest accuracy.

We also suggest some additional steps that may be taken to improve one's score :

1. **implement a good cross-validation strategy in training the models to find optimal parameter values.**
2. **Implement a greater variety of models for learning since the more uncorrelated the results, the better the final score.**

After the presentation, We also append results of other algorithms to display more complete comparison to our original models in this report. The results can be

seen on Table 2. It seems that we have almost incredible high accuracy in our Neural Network model compared to other groups. The reason we inferred might be that we contained as much data as possible, while dealing with missing data imputation. Therefore, if our inference is correct then it shows the importance of the process of dealing with missing data.

Table 2

Attributes	Model	Accuracy
RACE	Random Forest	74.4 %
	SVM	68.28%
	Neural network	99 %
	Xgboost	100%
ASOURCE	Random Forest	20.4 %
	SVM	23.01%
	Neural network	100 %
	Xgboost	100%
ATYPE	Random Forest	80 %
	SVM	93.95%
	Neural network	88 %
	Xgboost	99.98%
TOTCH	LGBM	52.4 %
	SVM	5.91%
	Neural network	72 %
	Xgboost	99.98
ZIPINC	Random Forest	56.5 %
	SVM	75.49%
	Neural network	73 %
	Xgboost	82.04%

References

[1] Scikit-learn.org. (2017). *1.17. Neural network models (supervised) — scikit-learn 0.19.1 documentation*. [online] Available at: http://scikit-learn.org/stable/modules/neural_networks_supervised.html [Accessed 31 Oct. 2017].

[2] "Which algorithm takes the crown: Light GBM vs XGBOOST?", *Analytics Vidhya*, 2017. [Online]. Available: https://www.analyticsvidhya.com/blog/2017/06/which-algorithm-takes-the-crown-light-gbm-vs-xgboost/. [Accessed: 31- Oct- 2017].

[3] "1.17. Neural network models (supervised) — scikit-learn 0.19.1 documentation", *Scikit-learn.org*, 2017. [Online]. Available: http://scikit-learn.org/stable/modules/neural_networks_supervised.html. [Accessed: 31- Oct- 2017].

[4] J. Brownlee, "Develop Your First Neural Network in Python With Keras Step-By-Step - Machine Learning Mastery", *Machine Learning Mastery*, 2017. [Online]. Available: https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/. [Accessed: 31- Oct- 2017].