Data Mining for Detection of Acetabular Cartilage Delamination in Femoroacetabular Impingement Patients

Amir Farrag[1,*]

[1]Dalhousie University Department of Mathematics and Statistics, Halifax, B3H 4R2, Canada
[*]farragamir@gmail.com

ABSTRACT

Data mining methods can be used to predict the presence of cartilage delamination in patients with femoroacetabular impingement. Logistic regression, LASSO logistic regression and random forest statistical learning prediction models were built on a dataset of 229 FAI patients. When training a random forest model on an imputed dataset, a delamination prediction rate of 77.86% was achieved. It is unlikely that BMI, alpha angle in frog view, grade of kellgren lawrence system on hip ap audiographs, the presence of two anchors, the presence of four anchors, or sagittal length are able to contribute to the prediction of delamination, upon inspection of the LASSO models.

1 Introduction

Patients suffering from femoroacetabular impingement (FAI) are forced to sustain hip pain and early development of osteoarthritis. In order to establish appropriate treatment of FAI patients, it is essential to be able to identify which of them are at risk of developing significant hip cartilage damage.[1] Hip cartilage delamination has a direct relationship to FAI surgical outcomes, and it is very difficult to detect, mainly because of limited joint distensibility.[1] This paper attempts to construct a method of predicting hip cartilage delamination in FAI patients, and identify the predictors that are most strongly linked to the forecasting of cartilage delamination.

Data from the magnetic resonance arthrography images and anteroposterior radiographs of 229 FAI patients has been converted into an easily computable format, and subsequently analyzed. This data along with the medical records of the patients forms the dataset that has been examined. Included in the medical records for every patient is a post-operative surgical observation of whether or not they have some form of cartilage delamination. This delamination variable, or response variable is encoded in a categorical format, where a 1 signifies that the patient does have delamination, and a 0 signifies that they do not have delamination. All of the medical data factors are referred to as regressor variables, which can be used to try and help prediction of delamination. They are encoded as numerical data, or as discrete variables that can assume 2 or more possible values, similar to the delamination response variable. The first goal of the analysis is to construct a function of the regressor variables which can accurately predict the output response variable. The secondary goal is to identify which of the regressors are most strongly linked to the prediction of the output.

2 Methods

Statistical learning methods were used to build a function of the regressor variables that can accurately predict the output of the delamination response variable. The observations contained in the dataset correspond to input-output pairs for the prediction function. To construct, or in other words train a prediction function, a statistical learning algorithm needs to be run on the dataset to fit the function to the data. Statistical learning methods we will use to create a model include the logistic regression, LASSO (least absolute shrinkage and selection operator) logistic regression, and random forest algorithms.

Due to the large amount of incomplete observations in the training dataset, data imputation methods were used to estimate missing regressor variable values. Data imputation algorithms that were used include multiple imputation by chained equations (MICE), k-nearest neighbors imputation and random forest imputation. To infer which of the regressor variables are most likely able to be used for prediction of the response, an inspection of the LASSO feature selection algorithm output for each model trained using that algorithm was completed. All data processing was done in R, and all algorithms, plots, and models were implemented in R, making use of the packages mice[2], caret[3], MissForest[4], glmnet[5 6], and DMwR[7].

2.1 Logistic Regression

A logistic regression model can be trained to predict the output of discrete response variables that take on a maximum of two values. The model makes use of the logistic function, which is defined as follows:

Definition 1: Let $x = (x_1, x_2, \ldots, x_k)$ be a vector of k real numbers, and let $\beta_0, \beta_1, \ldots, \beta_k \in \mathbb{R}$. The **logistic function** is then:

$$\pi(x) = \frac{e^{\beta_0+\beta_1 x_1+\ldots+\beta_k x_k}}{1+e^{\beta_0+\beta_1 x_1+\ldots+\beta_k x_k}} = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\ldots+\beta_k x_k)}}$$

We assume that for any given input vector of regressor variables X, the output response variable y will follow a Bernoulli distribution with probability parameter $\pi(X)$. As a result, at any given point X, y will have an expected value of $\pi(X)$ and variance of $\pi(X) \cdot (1-\pi(X))$. The model parameters that are found by inspecting the data are the coefficients $\beta_0, \beta_1, \ldots, \beta_k \in \mathbb{R}$.

One thing that must be considered when using a logistic regression model is how the logistic function only accepts real number input values, while our true set of regressors includes discrete variables. To combat this, every discrete regressor $x_{discrete}$ that could take on N possible values was encoded using $N-1$ real valued replacement regressor variables $x_1, x_2, \ldots, x_{N-1}$ in the following way: In observations that $x_{discrete}$ took on its m^{th} possible value where $1 \leq m \leq N-1$, we would set all of the real valued replacement regressors to be equal to zero, and set $x_m = 1$. Alternatively, if $x_{discrete}$ took on it's N^{th} possible value, we would set every one of the real valued replacement regressors to be equal to zero. This way we created an encoding of all the possible categorical values that the discrete regressor can assume such that each of the values have a real number coefficient contributing to the model.

Our goal when constructing a logistic regression model is to choose coefficient parameters that will minimize the probability of making a prediction error for any future observation. One method of coefficient estimation is maximum likelihood estimation, which chooses parameter values that maximize the probability of observing what constituted our collected random sample. We assume that our sampled data points are independent of one another and identically distributed according to the density described earlier in order to find the following likelihood function that describes the probability of finding our set of observations:

$$L(\beta_0, \beta_1, \ldots, \beta_k) = \prod_{i=1}^{n} \pi(x_i)^{y_i}(1-\pi(x_i))^{1-y_i}$$

The set of coefficients which maximize the easier to work with logarithm of the likelihood function, or log-likelihood will equivalently maximize the original likelihood function. We have the following expression for the log-likelihood:

$$l(\beta_0, \beta_1, \ldots, \beta_k) = \sum_{i=0}^{n} [y_i log(\pi(x_i)) + (1-y_i) log(1-\pi(x_i))]$$

To maximize the log-likelihood function, we differentiate it with respect to each model parameter, and set each derivative equal to zero. The coefficient values that solve that system of nonlinear equations will maximize the likelihood. The set of derivative equations that need to solved are:

$$\sum_{i=1}^{n} [y_i - \pi(x_i)] = 0$$

$$\sum_{i=1}^{n} [x_{ij}(y_i - \pi(x_i))] = 0, \ j = 1, 2, \ldots, k$$

There isn't an explicit solution for the coefficient values satisfying this set of nonlinear equations, so we approximate the solution using a numerical method. As we let a beta vector β_t represent a vector of all the model parameters at time t, we use the Newton-Raphson method to iteratively solve the equations. It takes an initial guess of the beta vector, β_0 and then uses the gradient of the log-likelihood function with respect to the coefficient variables to iteratively construct beta vectors close to the true solution until getting adequately close, using the following scheme:

$$\beta_t = \beta_{t-1} - (l''(\beta_{t-1}))^{-1}(l'(\beta_{t-1}))$$

2.2 LASSO Logistic Regression

A logistic regression model trained using LASSO will have the same model structure as one constructed without using LASSO, however the method of coefficient estimation is different. The LASSO method first standardizes the values of all regressor variables in order to constrain the absolute value sum of model coefficients. It inserts a penalization of large coefficient values into the logistic regression training process, with the penalization strength controllable using a tuning parameter λ. Instead of

trying to maximize the log-likelihood function, LASSO equivalently aims to choose coefficients that minimize the negative log-likelihood. We can refer to this negative log-likelihood as a loss function L:

$$L = -l(\beta_0, \beta_1, \ldots, \beta_k) = -\sum_{i=0}^{n} [y_i log(\pi(x_i)) + (1 - y_i) log(1 - \pi(x_i))]$$

$$L = \sum_{i=0}^{n} \left[-y_i log\left(\frac{\pi(x_i)}{1 - \pi(x_i)}\right) - log(1 - \pi(x_i)) \right]$$

$$L = \sum_{i=0}^{n} \left[-y_i \sum_{j=1}^{k} (\beta_j x_{ij}) - log(1 - \pi(x_i)) \right]$$

In addition to this loss function from the unchanged logistic regression method of training, LASSO adds the absolute value coefficient sum term to form its own loss function L_{LASSO}. The LASSO operator seeks to minimize the LASSO loss function, and by following that policy, we are left with an expression for the model parameters:

$$L_{LASSO} = L + \lambda \sum_{j=1}^{p} |\beta_j|$$

$$\hat{\beta}_\lambda = \arg\min_{\beta} (L_{LASSO})$$

To find the regression coefficient values which minimize the LASSO loss function, the glmnet package uses a numerical optimization method called cyclical coordinate descent.[6] The λ tuning parameter is chosen by iteratively building LASSO models using many lambda values, and choosing the one that minimizes prediction error. Depending on the size of the tuning parameter choice, LASSO will shrink some model coefficients to 0, removing certain regressors from being considered if they don't have a significant enough correlation with the response. This process of variable selection gives a view of which regressors most likely contribute to the true underlying prediction model. By getting rid of the use of non-contributing variables and preventing overfitting to the data, LASSO can also increase the model prediction accuracy.

2.3 Random Forest

Random forest is another method of supervised learning that makes use of multiple classification and regression trees (CART) to partition the space of input data into discrete areas. Each area is represented using a leaf in a binary tree and is assigned a label corresponding to the predicted output response for any input vector that happens to be there. The response labels are constructed by generalizing the output behaviour from observations that fall in each area.

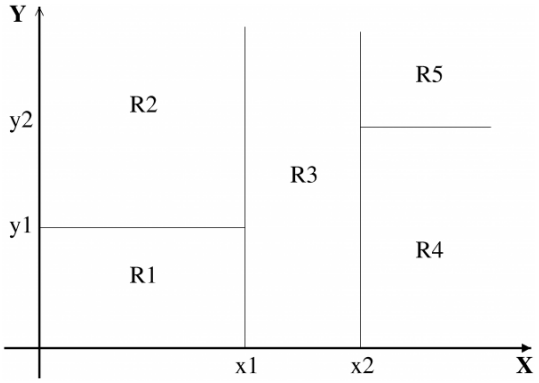

Figure 1. Example partition of a two-dimensional input space[8]

To help ensure a high prediction accuracy of a CART, it is important for it to choose a partition of the input space such that each disjoint area will have a consistent output labeling. The Gini impurity can be defined to establish a measure of how consistent, or pure any given labeling is for a partition:

Definition 2: Let P be a partition of sampled observations, and $p_1, p_2, \ldots, p_k$ be the fraction of items in that partition labelled with class $1, 2, \ldots, k$ in the set. The **gini impurity** is defined to be:

$$I_G(P) = 1 - \sum_{i=1}^{k} p_i^2$$

To build the partition, we can iteratively split up the input space in two by choosing a variable and splitting point such that the resulting partition will have a minimized Gini impurity. The partition process will iteratively run until we reach some sort of stopping criteria that we define. An example of a stopping point would be to finish training once each disjoint area has no more than 5 observations. Once a partition has been established, it can be generalized to cover the entire input space and encoded into a binary tree with each leaf representing a specific set of inputs. A probability density for the response variable is established at each leaf, with the probability of any classification label being the proportion of observations in that area which have the label.

Predictions from a random forest model will come from merging the output predictions of many CART models as illustrated in Figure 2. Samples of the set of observations are repeatedly made (with replacement) to train many classification trees. Every tree only regresses on a subset of the k available regressors. It is commont to force each tree to randomly choose $\lfloor \sqrt{k} \rfloor$ of the available features to consider. To make a new response prediction given an input vector, a probability density for the input is constructed by averaging the proportions of each possible response from each of the classification trees. The predicted response will be the value that has the highest probability of occurring in the combined probability density. Random forest models reduce the variance of prediction in comparison to classification trees, while retaining low bias.[9] Consequently, random forest models have a lower mean squared error than classification trees, and they are more effective in prediction of the response.

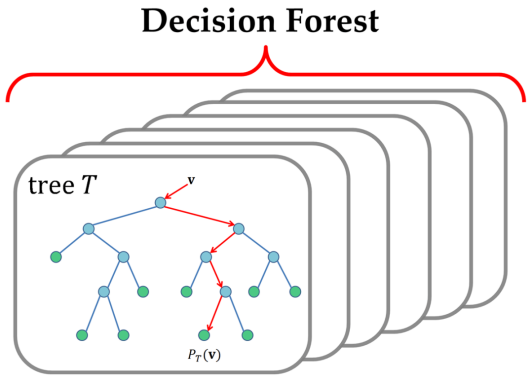

Figure 2. Random forest makes use of many decision trees to build a prediction policy.[10]

2.4 Data Imputation

Data imputation algorithms estimate the values of missing regressor data in incomplete observations. There were many missing data points in the delamination training set, which left the dataset having nearly only half of its original size in complete observations. Having a large number of contributing observations is important for building an accurate model, and estimation of missing data points can allow for many more valid observations to be used in the training set. Data imputation amounts to modeling the variables with missing values as response variables, and training a model on the remaining regressors in order to make an accurate estimation of missing data points. The three data imputation methods that were used were multiple imputation by chained equations (MICE)[2], k-nearest neighbor imputation[7], and random forest imputation[4].

MICE Imputation: MICE is a technique that operates by imputing missing data many times over, in an attempt to get iteratively better estimations for the missing values. Firstly, a simple imputation technique such as imputing the mean (for numerical regressors) is used to provide placeholder estimates for the value of every missing data point for every variable except for one. The missing values for the excluded variable are then estimated by using a supervised learning model dependent on all the other regressors. The type of supervised learning model that is chosen will depend on the form of the regressor being estimated. For instance, if the variable containing the missing values is categorical, then logistic regression (if it takes on a maximum of two values) or polytomous regression can be used to impute it. Conversely, linear regression can be used on numerical regressors.[11]

Once the first set of missing values has been imputed, the placeholder estimates are deleted for the second variable that had missing values, and the regression process is repeated to get a new estimation for each of the missing values of the second

variable. This is done for all variables that had missing data points so that each of them benefit from the use of a regression model. After that is complete, the entire process is repeated a number of times (5-10 iterations is common) using the imputed dataset as the new placeholder estimation values.[11]

K-Nearest Neighbor and Random Forest Imputation: K-nearest neighbor imputation is an imputation method that makes use of local input space to infer missing regressor values. For each regressor, only a certain number of points, ($k \in \mathbb{N}$) that are closest to the missing observation according to a user defined distance metric will be considered for the regression. A common choice for k is 10, and an example of an acceptable distance metric for use is the Euclidean distance between observations. When predicting a categorical variable, the category that appears most often in the response from the k nearest observations will assume the value of the missing response. Missing numerical values are estimated by taking the arithmetic mean of the response from the k nearest observations. Random forest imputation, as suggested by the title will construct random forest models for each of the variables with missing values to estimate their missing data points. A proximity matrix defining the distance between two observations as the fraction of trees in which both observations fall in the same terminal node is constructed to provide a weighted average for predicting missing values. Random forest imputation methods have the option of operating iteratively to constitute a multiple imputation algorithm, similar in operation to the MICE algorithm.

2.5 Model Assessment

To assess the accuracy of each built model, 10-fold cross validation was used. 10-fold cross validation first partitions the training data into ten subsets. It then trains a model and evaluates the prediction error rate 10 different times, each time using one partition as a validation set to test the predictions, and the remaining partitions as the training set. After completing all the evaluations, we inspect the average of the resulting errors from all ten models to get an estimation of the true prediction error.

Cross validation guarantees that any influential observation will be left out of exactly one training set from our 10 different prediction models. This increases the prediction error of that particular model, but it will not strongly affect the overall prediction error, since it is constructed as an average using 9 other models which do include use of that influential observation. Cross validation gives a good overall look at how our model performs, and it is resistant to potential mistakes that could be made in manually choosing a training and validation set. In order to alleviate some of the inconsistency from the assessment of models that trained on imputed data, the data imputation process was repeated 10 times to get 10 complete imputed datasets. An average of the cross validation error was then recorded from the models that trained on each set.

3 Results

On the reduced dataset that did not contain any imputed data, but only had half of its size worth of complete observations, all the regression models discussed earlier were trained on it and assessed. LASSO logistic regression and random forest models were trained and assessed using the imputed datasets that resulted from the three imputation methods mentioned before. A summary of the performance of each model is visible in the tables below. The models that were trained on imputed datasets performed better in comparison to those that trained on the dataset with incomplete observations. A random forest model trained on an imputed dataset using the random forest imputation method gave the largest estimated prediction accuracy of about 77.86%.

No Imputation	
Model type	Estimated prediction accuracy
Logistic Regression	0.6410839
LASSO	0.7179487
Random Forest	0.7595685

MICE Imputation		
Model type	Estimated prediction accuracy	Standard deviation
LASSO	0.70131	0.01724753
Random Forest	0.7755731	0.01440532

KNN Imputation		
Model type	Estimated prediction accuracy	Standard deviation
LASSO	0.710917	0.01591875
Random Forest	0.7764756	0.01280231

RF Imputation		
Model type	Estimated prediction accuracy	Standard deviation
LASSO	0.7213974	0.01196786
Random Forest	0.7785804	0.01224433

A roundup of which variables were selected from each of the LASSO models was collected, and it followed that none of the following regressors ended up being chosen in any of the LASSO models: BMI, alpha angle in frog view, grade of kellgren lawrence system on hip ap audiographs, the presence of two anchors, the presence of four anchors, and sagittal length.

4 Discussion

The results of delamination prediction accuracy for the models is acceptably good, with the best model having a success rate of 77.86%. If there was a larger dataset available with more complete observations, a model with even higher prediction accuracy could potentially be built. The data imputation for estimation of missing data points did not give a hugely significant improvement to the models, only giving an estimated increase in prediction rate of approximately 1.9% when comparing the best model that used imputation to the best model that did not. It is not likely that the regressors that were not chosen in any LASSO model can contribute to the prediction of cartilage delamination very well.

Computer vision algorithms that use deep learning could be useful for this and other diagnoses prediction problems, given the raw medical images. It would get rid of potential human errors that could have come up while researchers were inspecting the images. Computer vision also alleviates the pressure that was put on the researchers of this study to sample such a wide set of variables, since the only resource that would be required for analysis would be the images and the post-operative examination of delamination presence. This could help get rid of the problem of missing data points from the provided dataset. Computer vision algorithms often require a large sample of data to be useful, so a large amount of images could potentially be needed for the method to be more effective than what was done in this study. It would also be more difficult to implement, and the image data could carry a large amount of distortion and noise, which would make it difficult to get accurate prediction results.

References

1. Wong, I., Alqarni, A., Alfaraidy, M. & Lewington, M. Acetabular cartilage delamination in femoroacetabular impingement. correlation between magnetic resonance imaging diagnosis and arthroscopic finding. Capital District Health Authority (CDHA) Research Ethics Board.

2. van Buuren, S. & Groothuis-Oudshoorn, K. mice: Multivariate imputation by chained equations in r. *J. Stat. Softw.* **45**, 1–67 (2011). URL http://www.jstatsoft.org/v45/i03/.

3. from Jed Wing, M. K. C. *et al.* *caret: Classification and Regression Training* (2017). URL https://CRAN.R-project.org/package=caret. R package version 6.0-78.

4. Stekhoven, D. J. & Buehlmann, P. Missforest - non-parametric missing value imputation for mixed-type data. *Bioinforma.* **28**, 112–118 (2012).

5. Simon, N., Friedman, J., Hastie, T. & Tibshirani, R. Regularization paths for cox's proportional hazards model via coordinate descent. *J. Stat. Softw.* **39**, 1–13 (2011). URL http://www.jstatsoft.org/v39/i05/.

6. Friedman, J., Hastie, T. & Tibshirani, R. Regularization paths for generalized linear models via coordinate descent. *J. Stat. Softw.* **33**, 1–22 (2010). URL http://www.jstatsoft.org/v33/i01/.

7. Torgo, L. *Data Mining with R, learning with case studies* (Chapman and Hall/CRC, 2010). URL http://www.dcc.fc.up.pt/~ltorgo/DataMiningWithR.

8. Bontempi, G. & Souhaib, B. T. Classification and regression trees (2016). URL https://www.otexts.org/1512. [Online; accessed 22-April-2018].

9. Svetnik, V. *et al.* Random forest: A classification and regression tool for compound classification and qsar modeling. *J. Chem. Inf. Comput. Sci.* **43**, 1947–1958 (2003). URL https://doi.org/10.1021/ci034160g. DOI 10.1021/ci034160g.

10. Wang, Q. Decision tree and decision forest (2014). URL https://www.mathworks.com/matlabcentral/fileexchange/39110-decision-tree-and-decision-forest?focused=3812787&tab=function. [Online; accessed 22-April-2018].

11. Azur, M. J., Stuart, E. A., Frangakis, C. & Leaf, P. J. Multiple imputation by chained equations: What is it and how does it work? *Int. journal methods psychiatric research* **20**, 40–49 (2011). URL http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3074241/. DOI 10.1002/mpr.329.