Predicting Outcomes of Children from Fragile Families Using Regression Modeling

Crystal Qian
cqian@princeton.edu

Jonathan Tang
jontang@princeton.edu

Abstract

The Fragile Families Challenge seeks to predict key outcomes of disadvantaged American children using machine learning. These predictions provide important insights that can help improve the childrens' lives. The Fragile Families dataset measures six key outcomes, collected over the span of a child's early life, with the baseline household data being collected in 1998 and further "waves" of data collected in roughly 3-year increments on the same households. Given this baseline household data, we present our approach to achieving reasonable predictions of the six continuous and binary outcomes using a combination of techniques such as encoding, imputation, hyperparameterization, regressor selection, and cross validation.

1 Introduction

Children from disadvantaged households in America face environmental challenges that have profound impacts not only on their lives but on their society. To address these challenges, it is important to analyze the background conditions and potential prospects of these children. Predicting the outcomes variables of these children will allow us to generate and implement effective solutions to the highly-compelling social issues behind the Fragile Families Challenge.[4]

Regression and binary classification methods are needed to predict the six continuous and binary outcome variables. We approach these predictions in three key steps:

1. **Preprocessing** the data (encoding, imputation, etc.)
2. Performing **regression** (regressor selection, hyperparameterization)
3. Conducting **analysis** of predicted results (cross validation)

2 Dataset

The Fragile Families Challenge contains data pertaining to 4,242 children with 12,943 features of continuous, binary, string, or inconsistent types.[1] These features can be binary, continuous, or contain string values such as "4-Jul", "milk cereal eggs cheese", and "sponge bob square pants". Additionally, there is a significant amount of missing data in the form of "NA" responses, empty quotation marks, etc. Half of the children (2,121) are *labelled*, meaning we have additional data about their six key outcomes. These outcomes are grit, GPA, and material hardship (continuous variables) and eviction, layoff, and job training (binary variables). are the continuous variables of: grit, GPA, material hardship; as well as the binary variables of: eviction, layoff, and job training.

As an extension, we found that 17.51% percent of all cells in the dataset consist of missing data. 811 of the 12,944 features in the dataset are completely missing, with no non-"NA" responses, and each of the childrens' inputs are missing at least one response. For these reasons, we're highly motivated to compensate or even take advantage of this missing data.

3 Approach

We heavily preprocessed the missing data and predicted the six outcome variables using different regression methods, using Python scripting and the scikit-learn machine learning library.

We initially explored classification in addition to regression methods for binary variables in the dataset; however, the Fragile Families Challenge incentives participants to provide continuous estimates of binary variables. In particular, the performance of the binary models is based on Brier loss, which is essentially a mean-squared error based on continuous probabilities of binary variables.[1] Because our intended classifiers (Decision Trees, etc.) do not provide continuous probability estimates and because we were interested in exploring different approaches to regression, we have pursued a regression-based solution for all six outcome variables.

3.1 Preprocessing

Not all of the Fragile Families data is necessarily useful for our regression (mentioned before, many variables have some form of redundancy); we also address the berth of missing data values.

3.1.1 Feature Pruning

This dataset has 12,943 columns, some with little or no responses. (For example, see $f3natwtx\_rep15$, $m3natwtx\_rep8$, and $c3natwt\_rep25$.) We assign each column an **importance** score, which is the ratio of non-"NA" responses to total responses. The features are then filtered by varying importance score cutoffs of 0.5 (10693 features remaining), 0.75 (10050 features), 0.95 (9975 features), and 0.99 (9960 features). By pruning columns with low response rates, we weight columns like $k5f1$, $f3j8$, and $f3j17$, which have a 100% response rate. Results discussed in this paper are made with the dataset pruned with importance threshold $= .75$.

3.1.2 Encoding

We use a *LabelEncoder* to encode all n distinct values of any given feature to integers $[0, n-1]$, which cleans up all the cases containing string values through a one-to-one mapping. Looking at the few unique string responses (which is viewable privately at https://codeshare.io/ar4xkY) across all variables, we found that label encoding the string values is a sufficient representation due to the general lack of semantic patterns in this data. Note that the "NA" value is specially marked for imputation.[3]

3.1.3 Approach to Imputation

1. **Mean:** First, we used an *sklearn.preprocessing.Imputer* to impute the values of the priors by setting "NA" values to the mean of all encoded non-"NA" values in its corresponding feature.[3] This is necessary to prevent the "NA" value being treated as a separate label by the label encoding step above; if the "NA" label receives a unique encoding value, future priors can be classified as "NA", which is not a valid response.
2. **Regression:** Because we're analyzing thousands of features, however, mean imputation is not appropriate for our multivariate analysis. Since mean imputation sets each "NA" value to the mean of a feature, we lose the relationships between answers for any given child. So, we implemented regression imputation as follows, populating predictions given the set of training priors (labeled data with corresponding training classes) and test priors (no corresponding classes):
   (a) First, we use all the non-"NA" class values and corresponding priors to train a regressor for each of the six classes.
   (b) We use these regressors to predict the "NA" values missing for each of the six training classes.
   (c) These regressors then predict the entirety of the test classes, since that data is unlabeled.
   (d) We concatenate the now-filled training classes and test classes by their corresponding challenge IDs, making a final prediction array.

3.2 Regressors and Hyperparameter Tuning

We tested the following regressors on each of the six features, using cross validation and grid search to find optimal parameter values based on the resultant predictions:

1. Linear, a baseline regression approach
2. Lasso, a linear model that would reduce our giant number of variables through spare coefficients. We tested on each feature with α from .1 to 1000.
3. Lasso Lars, Lasso implemented with Least Angle Regression. This was chosen due to its efficiency on large priors and usefulness in cross validation. This was also hyperparameter tuned on its α parameter.
4. Elastic net, useful with correlated features. Also tuned on its α parameter.
5. Ridge regression, also tuned on α.

Note that we also attempted tests with *scikit's* LassoCV, RidgeCV, and ArdAggression, but these tests took an extremely long time and became computationally infeasible.

3.2.1 Testing of Regression Methods: Cross Validation

We're unable to evaluate our results on the class values predicted from our regression imputation, since this data was unlabeled. So, we use the priors and classes used to fit the regressors in the regression imputation as a complete set of data. For all children i had a non-"NA" value for class j, its class response (the non-"NA" value) and corresponding feature vector (all values in row i) would be a complete entry for the j^{th} regressor.

We use *train_test_split* to do k-folds (in our case, $k = 5$) cross validation on this regressor data, calculating MSE and other metrics for predictions on each fold using the remainder of the folds to train. The results of this local testing are roughly equivalent to the results found on the leaderboard, which gave us an approximation of the standing of our results.

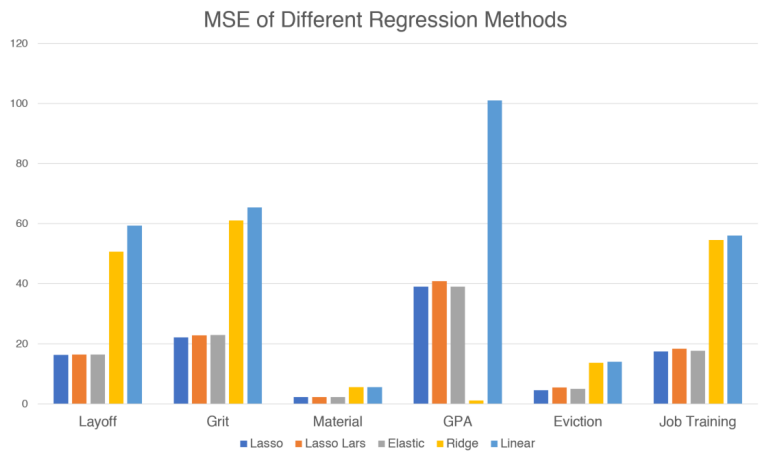

Figure 1: MSE of Different Regression Methods

This figure shows our observations on different regressors for the six outcome variables. Notice that Lasso, Lasso Lars and Elastic Net all performed approximately equally well, while ridge and linear regression had higher error. The questions asked in this study could be highly correlated, explaining the success of elastic net and lasso regression. Both regressors allow similar to be eliminated or penalized easier than in ridge or linear regression.

3.3 Spotlight Method: Lasso Regression

Lasso regression (Tibshirani, 1996) is a regression method that performs well with sparse data and can behave as a feature selector for continuous variables.[5] Given a continuously-valued dataset with

n labels $y_i \in R$ and n samples $x_{i,j} \in R^m$, with $i \in 0..n$ and $j \in 0..m$, Lasso regression utilizes the solution to the following optimization problem to generate a statistical model:

$$\hat{\beta} = \underset{\beta \in B}{\arg\min} \left\{ \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{p} \beta_j x_{i,j} \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}$$

The easiest motivation for using Lasso regression is for the sake of creating an optimal solution with fewer variables while still satisfactorily reducing the prediction error. Lasso regression assigns β_j weights to different features of the data while penalizing the values of β_j that are far away from zero. Variables that tend to have stronger statistical power will have larger weights associated with them, while extremely weak variables will have their coefficients set to zero with $\beta_j = 0$, effectively causing those weaker variables to be ignored. This zeroing out is a highly desirable and very distinct property of Lasso regression, as it allows Lasso regression to act as a form of feature selection for continuous variables.

The Lasso regression provides additional utility by granting insight about the relative statistical significance of different features. By examining and comparing their coefficients, we can see which features contribute more to a successful model: large coefficients imply higher statistical value, while small coefficients imply the opposite.

As the parameter λ becomes larger, the regression will tend to become tighter and force more variables to be zero, causing those variables to be ignored and for the solution to be more sparse. This is because a larger λ causes the β_j coefficients to contribute more, but because our objective is to minimize $\hat{\beta}$, we wish to minimize the impact of the β_j weights, and it becomes more desirable to ignore variables by setting their weights to zero.

Because the Fragile Families dataset is fundamentally based upon a large number of continuous values, with many variables being sparse or missing from the data, our intuition is quickly motivated to use Lasso regression in order to solve this well-suited problem. Ultimately, we found that Lasso regression performed the best on this dataset (although as seen in the graph above and subsequent submissions, Lasso Lars and Elastic Net yielded nearly comparable results). Our final methodology is explained in Table 1 below.

Table 1: Personal Best Codalab Submission Results

	GPA	**Grit**	**Material Hardship**	**Eviction**	**Layoff**	**Job Training**
Mean-Squared Error	.401	.219	.026	.053	.172	.202
Leaderboard Rank	23	13	8	6	4	5
Regressor	Lasso	Lasso	Lasso	Lasso	Lasso	Lasso
Parameters	$\alpha = 15$	$\alpha = 125$	$\alpha = 1.03$	$\alpha = 10$	$\alpha = 10$	$\alpha = 300$

4 Results

Note: our online Challenge submission is under the username *cjqian* on Codalab.

4.1 Evaluation Methods

The Fragile Families Challenge online evaluator uses mean-squared error (MSE) for continuous variables and Brier loss for binary variables. The MSE is simple, helpful, and fairly straightforward; ideally, we aim for getting zero mean-squared error, but we are generally acceptable of small deviations that will almost inevitably occur. because these give us a fairly objective evaluation metric. In addition to the MSE, our second main evaluation metric is the empirical ranking of our implementation on the Fragile Families Challenge student leaderboard.

Perhaps most interestingly, our implementation enjoyed very large accuracy values and small error. These high values were due to the peculiar properties of the dataset, the Fragile Families error calculation method, and the Lasso regression method in general.

Table 2: Binary Variables in Fragile Families Dataset

	Eviction	Job Training	Layoff
P(y=1)	0.0630	0.2486	0.175
Accuracy	0.937	0.751	0.825
Loss	0.0651	0.1877	0.1491
F1 score	0.0	0.0	0.0

Table 3: Significant and Insignificant Variables

	Most Significant	**Value**	**Least Significant**	**Value**
1	*f5k10f*	3.23e+38	*f2j18c*	0.0
2	*f4h6*	2.88e+38	*m3d9p1*	0.0
3	*m5a6g02_11*	2.56e+38	*m3d9q*	0.0
4	*p5h1cb*	2.46e+38	*m3d10*	0.0
5	*m5c2d*	2.33e+38	*m3d10a*	0.0

Table 2 illustrates these peculiarities. Note that P(y=1) corresponds to the proportion of "1" labels in the true training set. All three of the binary variables, eviction, job training, and layoff, are extremely biased towards one answer, with 0 being far more common in the testing labels than 1. Any regressor that outputs values close to 0 will enjoy very high accuracy. Because we are using a Lasso regressor, we will often have coefficients that are very small due to the regressor trying to throw away as many variables as possible while still having a low error rate. Because Brier loss, the Fragile Families Challenge's error calculation for binary values, is fairly forgiving, the Lasso regressor throws away many variables so that its outputs are very small and close to zero. Meanwhile, false negatives and systematic errors are not punished by the Brier loss calculation, which only cares about the squared differences: $Loss = 1/n \sum_{i=1}^{n} \left(Y_i - \hat{P}(Y_i = 1)\right)^2$.

A regressor that guesses zero every time will suffer a loss of only $sum(Y^2)/n$, which is a small number due to the abundance of zeros in this data. In contrast, in Table 2, we can see that the F1 scores of the variables are exactly equal to zero, far from the perfect F1 score of 1, because the regressor that always guesses zero never can detect true positives and false negatives. The major lesson learned from this is that accuracy, and even certain forms of loss calculation, can be highly deceptive when it comes to biased data.

5 Discussion and Conclusion

Using the sorted coefficients of the Lasso regression, we examined the top five most and least valuable variables during a particular run of our implementation, with the results listed above in Table 3.

Using the Codebook available publicly online at the Fragile Families website, the top three most significant variables, in terms of contributing to predictive accuracy: "In past year, you shouted, yelled, or screamed at child", "Is there someone you could count on to co-sign bank loan for $5000?", and "There were problems with neighborhood safety". Unfortunately, it makes sense that these questions provide a huge amount of information regarding a child's future. In contrast, the least significant question was: "Did/do you worry about one particular thing?", an extremely vague and unrevealing question.[2]

Throughout this implementation, we have placed a lot of emphasis on regression and data analysis. In terms of future extensions, it would be very interesting to develop the imputation methods even further, possibly extending with fuzzy k-means imputation, or with multiple imputation using a hot deck or carry-forward method. In addition, it would also be interesting to explore more opportunities with the binary classification variables, since the highly peculiar bias likely leaves the way open for alternative methodologies.

References

[1] Fragile Families Challenge blog. http://www.fragilefamilieschallenge.org/blog-posts/. Accessed: 2017-03-28.

[2] Fragile Families documentation. http://www.fragilefamilies.princeton.edu/documentation. Accessed: 2017-03-28.

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[4] N. Reichman, J. Teitler, I. Garfinkel, and S. McLanahan. Fragile families: Sample and design. *Children and Youth Services Review*, 23:303–326, 2001.

[5] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, 58:267–288, 1996.