Fragile Families Challenge

Andrew Or
Department of Computer Science
Princeton University
andrewor@cs.princeton.edu

Thomas Shaffner
Department of Computer Science
Princeton University
tfs3@cs.princeton.edu

Abstract

This paper describes our submission to the Fragile Families Challenge [2], an effort that applies techniques in predictive modeling and causal inference to surveys conducted with disadvantaged families. The goal of the project is to improve the lives of children in these families by identifying underlying patterns in the answers given by the families. In this work, we use regression and classification methods to predict the values of six key attributes: GPA, grit, material hardship, eviction, layoff, and job training. Because of the abundance of invalid entries in the data, feature selection plays a crucial role in our effort.

1 Introduction

The Fragile Families and Child Wellbeing Study (FFCWS) [3] is an academic collaboration that followed around 5,000 urban families in the United States and collected information about how their social environments changed over time. The data collected in this study has been used by academic papers published in many journals since then [4]. Insights derived from this data are shared with policy makers through the Future of Children [5], a collaboration between the Woodrow Wilson School of Public and International Affairs at Princeton University and the Brookings Institution. The Fragile Families Challenge is an open challenge for data scientists to apply machine learning and statistical methods to the data collected in the study.

In this paper, we discuss our submission to the Fragile Families Challenge. We modeled continuous labels and binary labels separately, using regression methods for the former and classification methods for the latter. The regression methods we evaluated are:

1. Linear Regression with Lasso Regularization (LRL)
2. Support Vector Regression (SVR)
3. Random Forest Regression (RFR)

and the classifiers we evaluated are:

1. Bernoulli Naive Bayes Classifier (BNB)
2. Multinomial Naive Bayes Classifier (MNB)
3. K-Nearest Neighbors Classifier (KNN)
4. Random Forest Classifier (RFC)
5. Gradient Boosting Classifier (GBC)

We used the `scikit-learn` [7] implementation of all of the above regression and classification methods. The main metrics we used in our evaluation are the coefficient of determination (also known as the R^2 score) for continuous labels and accuracy for binary labels. In both cases, we found that random forest performs the best. We used k-fold cross validation to tune the hyperparameters of these models to yield better generalization error. To account for the abundance of missing or invalid

values in the dataset, we experimented with various feature selection and imputation techniques, such as χ^2 feature reduction and filling null values with the mean and mode of the column.

All source code is available on Github at [1]. We have submitted code and predictions to the Fragile Families online service under the usernames "andrewor" and "t.f.schaffner".

2 Dataset

The background dataset primarily consists of survey answers from the FFCWS. Each row uniquely identifies a child and each column is either an ID of some form or an answer to a survey question. In this dataset, there are 4242 rows and 12945 columns.

The label data consists of six labels we wish to predict based on the features in the background data. These labels are derived from a final interview with the child when he or she is age 15. The six labels are: GPA, grit, material hardship, eviction, layoff and job training. The first three are continuous labels while the latter three are binary labels. There are 2121 rows in the label data, exactly half the number of rows in the background data. The goal is to predict the six labels for the 2121 missing rows using the background data.

Unfortunately, both sets of data were filled with invalid entries, such as N/A and specific negative values in the background case. The background data has 12945 columns, of which 3090 columns have null values (23.9%), 147 columns have string values (1.14%) which are not trivially parsable by our models, and 10459 have negative values (80.8%). Only 42 columns have neither null nor negative values, but up to 14 of these refer to some sort of ID and thus are not appropriate for use as features. Therefore, aggressive feature selection and imputation strategies must be used to yield a good fit, as we discuss in the next section.

Similarly, the label data is filled with null values. In particular, out of the 2121 rows, 956 GPA values (45.1%), 703 grit values (33.1%), 662 material hardship values (31.2%), 662 eviction values (31.2%), 844 layoff values (39.8%), and 660 job training values (31.1%) were null. Across all labels, as many as 655 rows were all nulls (30.9%). The null label values were not considered in our evaluation. Additionally, for the binary labels, a significant fraction of the values were `False` (64.7%, 47.6%, and 52.7% for eviction, layoff, and job training respectively). This heavy skew is an important factor for choosing an appropriate model for classification.

3 Data processing

We used several imputation and feature selection methods to prepare the background data before training classification and regression models. Here we describe the individual strategies used for imputation and feature selection, and the schemes produced by combinations of these techniques.

3.1 Imputation

Because such a large fraction of the background dataset consists of invalid entries, we tested strategies to impute the underlying true values for these entries. In particular, we experimented with the following:

- **Simple imputation:** Replace N/A with mode, negative values with 1.
- **Averaging imputation:** Replace N/A with mode, negative values with mode or mean of valid values of the feature. Mode is used if the feature type is integer and mean is used if the feature type is float.

Imputation does not make sense for string columns so we simply discarded these columns instead, as we discuss next.

3.2 Feature selection

We used several methods of varying complexity to remove features from the background dataset.

- **String removal:** Remove features that contain at least one string
- **Null removal:** Remove features that contain at least one N/A value
- **Negative removal (r):** Remove features whose fraction of negative values is $\geq$ `r`
- **Chi-squared (k):** Select k features with lowest χ^2 score

3.3 Evaluated schemes

We evaluated five schemes composed of combinations of the imputation and feature selection strategies listed above:

- **Baseline:** simple imputation + string removal = 12798 features
- **No null:** simple imputation + string removal + null removal = 9844 features
- **Data type dependent imputation:** average imputation + string removal + null removal = 9844 features
- **Negative proportions (r):** simple imputation + string removal + null removal + negative removal (r)
  - 913 features for $r = 0.25$
  - 2718 features for $r = 0.5$
  - 3807 features for $r = 0.75$
- **Chi-squared feature reduction:** simple imputation + string removal + null removal + chi-squared (k) = k features

In our evaluation, we found that the *no null* scheme yielded the best results for both continuous and binary labels. We decided to experiment with *data type dependent imputation* because we found simply replacing all negative values with 1 arbitrary. We also believed (incorrectly) that removing features with negative values in the *negative proportions* scheme would help because negative values indicate invalid or missing answers.

Similarly, *chi-squared feature reduction* consistently led to worse results. In an effort to understand why this is the case, we confirmed that non-indicative features such as `challengeID` and `mothid1` (mother ID) had high χ^2 scores, while hundreds of features had a χ^2 score of 0. Oddly, even setting $k = 99.5\%$ of the total number of features led to a significant drop in accuracy for the binary case. As such, we did not consider chi-squared feature reduction further in our project.

4 Classification and Regression

We tested several classification and regression methods (detailed in Section 1), but ultimately used random forest methods for both classification and regression tasks. Here we discuss in more detail the specifics of random forest regression.

Random forest [6] regression methods are used to combine multiple regression trees. The random forest regression algorithm draws uniformly at random m subsets of the training data. The algorithm then trains a regression tree on each of the subsets.

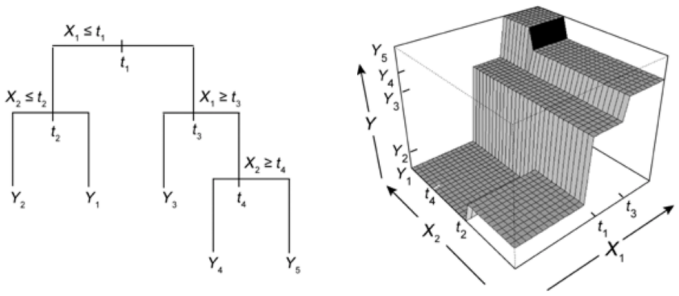

Figure 1: An example regression tree, borrowed from [8]

In the same manner as decision trees, regression trees split samples at branch points based on feature values (see Figure 1). Given training samples x_i and (continuous) training response variables y_i, regression trees are built in the following manner.

1. Select the optimal feature f and value v_f for branching by minimizing RSS
2. Partition samples into $L = \{i|x_{if} \leq v_f\}$ and $R = \{j|x_{jf} > v_f\}$
3. Recurse on the subsets L and R of input data

with RSS defined as

$$RSS = \sum_{i \in L}(y_i - y_L^*)^2 + \sum_{j \in R}(y_j - y_R^*)^2$$

and where $y_L^* = \frac{1}{|L|}\sum_{i \in L} y_i$ is the mean of all response variables corresponding to samples in the left subtree of the branch point, and $y_R^* = \frac{1}{|R|}\sum_{j \in R} y_j$ is the mean of all response variables corresponding to samples in the righthand subtree.

Given a new sample x, a single regression tree determines which leaf represents x and assigns to x the predicted y value of the mean of all response values represented by that leaf. Random forest regressors, given a new sample x, return as a prediction the average of the responses of each regression tree.

5 Evaluation

5.1 Results

We evaluated continuous variable predictions using the R^2 coefficient of determination defined as follows. Given n true continuous labels y_i, let $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ be the mean of all labels. Let the predicted labels be denoted $\hat{y}_i$. The R^2 coefficient of determination is then defined as

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

As for binary classification, we evaluated precision, recall, and f1 score as well as accuracy. We chose these additional metrics because accuracy alone does not take into account false positives and false negatives. Since both precision and recall are desirable properties, we also considered the f1 score, which is the harmonic mean of the two. Tables 1 and 2 summarize how each model performs with respect to these metrics.

	LRL	SVR	RFR
R^2	0.063	0.123	0.157

Table 1: Evaluation of different models for continuous regression of `gpa`

	BNB	MNB	KNN	RFC	GBC
Accuracy	0.761	0.584	0.725	0.765	0.679
Precision	0.040	0.244	0.253	0.000	0.204
Recall	0.003	0.362	0.085	0.000	0.130
F1	0.005	0.285	0.126	0.000	0.157

Table 2: Evaluation of different models for binary classification of `jobTraining`

In terms of both R^2 and accuracy, random forest (RFR and RFC) outperformed all other models. Because this dataset is likely not linearly separable under reasonable imputation and feature selection schemes, we expect random forest regression to yield a better fit than support vector regression and linear regression.

In the binary case, because there are significantly more `False` values than `True` values across the boolean variables, all predictors are heavily skewed towards predicting `False`. This leads the random forest classifier to predict all `False` scores, which is why we observe precision, recall, and F1 scores of 0 for `RFC`.

5.2 Hyperparameter tuning

We used 3-fold cross validation to tune hyperparameters for our random forest models. We used the `GridSearchCV` and `RandomSearchCV` methods from the `scikit-learn` library to perform cross validation. The parameter space we explored is listed as follows. The optimal hyperparameters by label are summarized in Table 3.

Parameters for regression	Parameters for classification
`n_estimators` : $50, 75, 100$	`n_estimators` : $50, 100$
`max_features` : $0.75, 1.0$	`max_features` : $0.75, 1.0$
`min_samples_split` : $10, 25, 50$	`min_samples_split` : $10, 25$

	gpa	grit	materialHardship	eviction	layoff	jobTraining
n_estimators	100	75	100	50	50	50
max_features	1.0	0.75	0.75	0.75	1.0	0.75
min_samples_split	50	50	10	25	25	25

Table 3: Optimal hyperparameters for each label

The parameters we tuned are `n_estimators`, which represents the number of regression trees trained, `max_features`, which represents the portion of all features to examine while generating splits, and `min_samples_split`, which represents the minimum number of samples in a node before it is allowed to split. We chose these hyperparameters to tune because they all reduce overfitting, which we conjecture to be an important reason why random forest outperformed the other models we evaluated.

5.3 Beating the odds

We were interested in determining which children ended up with better metrics than we would predict with our model. Higher values of gpa and grit are considered better, while a lower value of material hardship is considered better. Therefore, we consider samples with higher gpa, higher grit, or lower material hardship than our predictions to have "beaten the odds". Similarly, we consider any sample that disagrees with our prediction and is `False` for eviction, `False` for layoff, or `True` for job training to have beaten the odds.

For each continuous variable, we ranked the samples by the margin to which they beat their prediction. We hoped to see high overlap in the tops of each of these ranked list. However, we found that there were no samples common to the top 100 positions in all lists, and there were only ten samples common to the top 250 samples in all lists.

Because our best-performing model predicted `False` for all boolean variables for all samples, we observe no samples beating the odds for eviction or layoff, while every subject that has received job training has beaten the odds according to our model and definition.

We hypothesize that we do not observe subjects consistently beating the odds across the labels because we train models for each variable independently using different hyperparameters. For future work, it would be interesting to train multiple labels together to see if larger overlaps occur between the subjects who beat the odds by the highest margins for each label.

6 Conclusion

Due to the nature of the background data provided, we find that random forest methods perform best for both classification and regression problems. However, the fact that boolean training data is heavily skewed towards responses of `False` leads our random forest classification model to predict `False` for every test sample. Furthermore, there are many invalid responses throughout the background dataset. We hypothesize that this data sparsity causes standard feature selection techniques like chi-squared analysis to produce worse results.

We also identified rankings subjects who beat the odds by the highest margin for continuous variables. Though there is little overlap between the tops of these rankings, we propose training a single predictive model on a combination of labels in order to better determine whether any subjects consistently beat the odds by high margins.

References

[1] https://github.com/andrewor14/ffc

[2] Fragile Families Challenge: http://www.fragilefamilieschallenge.org/

[3] Fragile Families and Child Wellbeing Study: http://www.fragilefamilies.princeton.edu/

[4] Fragile Families publications: http://crcw.princeton.edu/publications/publications.asp

[5] Future of Children: http://www.futureofchildren.org/

[6] Breiman, Leo. "Random forests." Machine learning 45.1 (2001): 5-32.

[7] Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." Journal of Machine Learning Research 12.Oct (2011): 2825-2830.

[8] Elith, Jane, John R. Leathwick, and Trevor Hastie. "A working guide to boosted regression trees." Journal of Animal Ecology 77.4 (2008): 802-813.