SOCIUS
Special Issue: Fragile Families Challenge

ASA AMERICAN SOCIOLOGICAL ASSOCIATION

Socius: Sociological Research for a Dynamic World
Volume 5: 1–7

DOI: 10.1177/2378023118818720
srd.sagepub.com
SAGE

A Data-driven Approach to the Fragile Families Challenge: Prediction through Principal-Components Analysis and Random Forests

Ryan Compton[1]

Abstract

Sociological research typically involves exploring theoretical relationships, but the emergence of "big data" enables alternative approaches. This work shows the promise of data-driven machine-learning techniques involving feature engineering and predictive model optimization to address a sociological data challenge. The author's group develops improved generalizable models to identify at-risk families. Principal-components analysis and decision tree modeling are used to predict six main dependent variables in the Fragile Families Challenge, successfully modeling one binary variable but no continuous dependent variables in the diagnostic data set. This indicates that some binary dependent variables are more predictable using a reduced set of uncorrelated independent variables, and continuous dependent variables demand more complexity.

Keywords

Fragile Families Challenge, diagnostics, machine learning, feature engineering, principal-components analysis, random forests, data-driven methods

The Fragile Families Challenge involves social and data scientists in identifying at-risk families. The challenge data set is derived from 4,242 families, with 12,942 independent and 6 dependent variables. To address the diagnostic challenge, I pursue a data-driven approach using machine learning (ML) for feature engineering and model optimization. The methods I use to identify diagnostic predictors of vulnerability are therefore evidence based rather than theoretically motivated. Given the 12,942 independent variables in the challenge data set, it is impossible to directly evaluate every possible known predictor. Instead, by focusing on specific dependent variables (grade point average [GPA], grit, material hardship, eviction, layoff, and job training), I uncover unexpected associations and generate more powerful diagnostic models. In doing so I develop novel methods to identify families at risk. My approach is also replicable by focusing on building models with greater generalizability and predictive performance (Breiman 2001b). Nevertheless, the approach has limitations: spurious correlations and data biases may occur, and results need further theoretical verification before broader conclusions can be drawn.

My method involves the following steps. First, data are preprocessed to identify missing data and remove "bad features" (independent variables adding noise or no additional predictive power). Next, feature engineering generates predictive combinations of remaining features. Exploration then optimizes the models and features to produce the best performing model (assessed using the mean standard error) in a final test. This work also applies additional methods to deal with data set biases for ML models, one being imbalanced observed events (e.g., there are far fewer families experiencing eviction than not). I follow this procedure for all dependent variables in the Fragile Families Challenge, although performance varies for each. Compared with models using all available independent variables, I find an improvement in predicting one binary (job training) dependent variable but not the remaining (GPA, grit, material hardship, eviction, and layoff).

[1]University of California, Santa Cruz, CA, USA

Corresponding Author:
Ryan Compton, University of California Santa Cruz, Department of Computer Science, 1156 High Street, Santa Cruz, CA 95064, USA
Email: rcompton@ucsc.edu

Terminology and Metrics

My approach involves ML, so I introduce relevant terminology; independent variables are referred to as features and dependent variables as targets. Each family measured is referred to as an instance in the data set. ML also distinguishes different types of target variables. Continuous variables are referred to as regression targets and categorical variables as classification targets. The Fragile Families Challenge involves both target types, and appropriate ML terminology will be used for each.

Data-driven approaches need an evaluation metric to determine the model's predictive power. I use the evaluation function set by the Fragile Families Challenge, mean squared error (MSE). It is defined as (Wang and Bovik 2009)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{Y}_i - Y_i\right)^2,$$

where $\hat{Y}$ is the model prediction made and Y the vector of n observed true values. This metric applies to both regression (continuous target variables) and classification (categorical variables, which in this context take binary values, 0 and 1), making MSE a suitable evaluation metric for all challenge targets.

I also use standard ML evaluation metrics for categorical targets: precision (the number of true positives divided by the sum of true and false positives), recall (the number of true positives divided by the number of actual positives), and an aggregated metric of the prior two called F1 score (an aggregated metric of precision and recall) (Powers 2011). These metrics evaluate model performance per category, allowing adjustments to be made to improve model performance across categories. Each Fragile Families Challenge target is binary, so there are two categories per target: 1 (the event occurred) and 0 (the event did not occur).

Data Analysis

Data analysis involved the following steps: data cleaning, data splitting (creation of training and test sets), feature engineering, and principal-components search.

Data Cleaning

The first cleaning procedure dealt with missing data, as 23.8 percent of features included missing data. Features were removed if they could not be adjusted for with a mean substitution (Schlomer, Bauman, and Card 2010), leading us to exclude missing categorical data or where there were no measures in the training set. I removed only 796 (~7 percent) of the original set of more than 12,000 features, leaving many remaining predictor features. I also removed 1,775 features with no standard deviation, as a feature with no variance cannot contribute to the overall model.

That left 10,371 usable features. For these remaining cases, following a standard procedure (Schlomer et al. 2010), a mean substitution replaced missing values, preserving internal data set validity. Mean substitution reduces the number of dropped instances, which is appealing because there are only 4,242 family instances. However, using mean substitution assumes that data are missing completely at random (Little and Rubin 2014). If this assumption is not met, it can bias models to the present sample, but because only 4 percent of features were missing more than 100 values, substitution was considered appropriate.

Many features are also not normally distributed. Min-max normalization was therefore used, as it is appropriate for the current Fragile Families Challenge data set (Jayalakshmi and Santhakumaran 2011). It was implemented using the following formula, which is a standard method with the benefit of preserving relationships between features in a data set:

$$x' = \left(x_{newmax} - x_{newmin}\right) \times \frac{\left(x_i - x_{min}\right)}{\left(x_{max} - x_{min}\right)} + x_{newmin}.$$

Typically, x_{newmax} is set to 1 and x_{newmin} is set to 0 to rescale a feature to be within the range 0 to 1, and this is the case here

Data Splitting

The initial Fragile Families Challenge data set consisted of 12,942 features and 4,242 instances. Of these 4,242 instances, the challenge set preemptively identified 2,121 for training, 530 to calculate a participant leaderboard, and 1,591 instances as a holdout test set. Cleaning reduced the number of features to 10,371. In my work, I split the training set (2,121 instances) into three subsets: model training, model validation, and model test sets. In identifying these subsets, 33 percent of the data were randomly assigned to the model test set, which meant that no model was trained on these data, which were used only to evaluate performance. The remaining 66 percent were used for the model training and model validation sets. Model validation data were assigned through sixfold cross-validation (Kohavi 1995). This process is shown in Figure 1. The data are split into six equal sized folds; for each of the six folds I train a model on the remaining (89 percent) data, producing six different models, and then evaluate each model's ability to predict the held-out 11 percent chunk. The highest performing model of the six is then used on the model test set, for an on-hand generalizability test. This best model was then submitted to the Fragile Families Challenge, in which it was evaluated with the leaderboard and holdout test sets.

Feature Engineering

Identifying ML features is complex. Models with more features can fit a target with a large amount of variance.

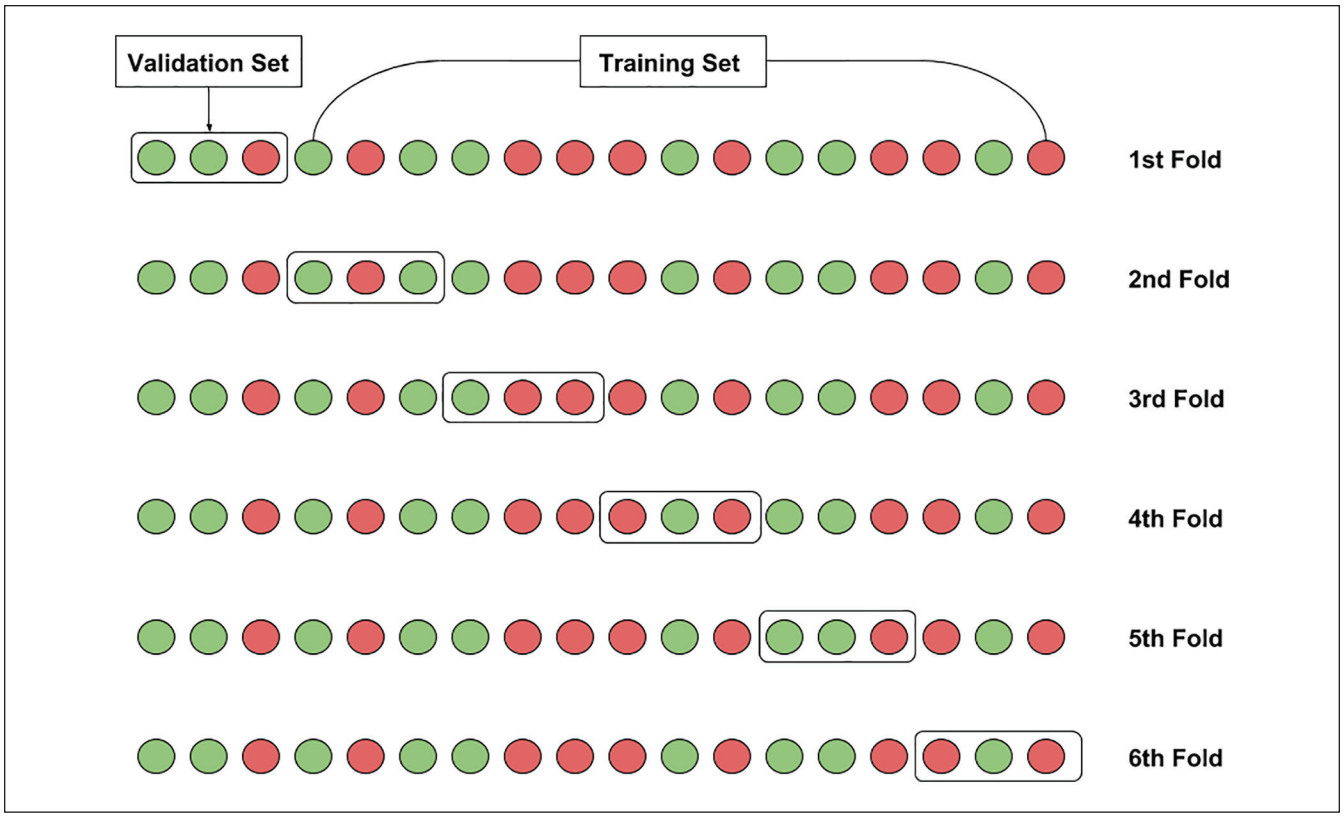

Figure 1. Visual representation of a sixfold cross-validation procedure.

However, having many features reduces a model's ability to generalize beyond the original data set. A complex model may overfit a training data set, fitting data noise rather than genuine relationships between features and target, making it ungeneralizable (Dietterich 1995). A model must therefore be simple enough to generalize to new data but complex enough to handle observed target variance in the data set. This heavily studied ML problem is called the bias-variance trade-off (Breiman 1996).

The set of 10,371 features remaining after data cleaning was large, requiring reduction to enhance generalizability (Breiman 2001b; Keogh and Mueen 2011). However, I did not want to remove features containing key information, so a feature engineering approach, principal-components analysis (PCA), was chosen as opposed to filtering (i.e., removing features that do not provide much predictive power). PCA reduces the number of features and potentially removes redundant information by converting the observed, possibly correlated features, into an aggregated set of uncorrelated variables called principal components (Wold, Esbensen, and Geladi 1987).

However, PCA has limitations. It is an unsupervised approach, meaning that it does not systematically determine the optimal number of components but instead requires this number to be set manually. Setting the number is achieved by exploring model performance across a range of component values (100 to 2,000 principal components), evaluating how much variance is explained for each target. The number of principal components producing the best model is typically selected as the number of components. Another issue is that PCA can potentially reduce the amount of information in the feature set (i.e., low complexity and variance), as target variables sometimes call for complex models (Breiman 2001b). This can be addressed by comparing the performance of different models; here a model using the optimal number of principal components was compared with a baseline model that does not use these principal components. Another issue with PCA is that models are hard to interpret, as features developed are combinations of perhaps tens or hundreds of original features, making it unclear which original features are important. However, that disadvantage is offset by an improved ability to predict new instances of each target and scalability. Finally, PCA is susceptible to outliers and nonnormal data, which I already addressed in preprocessing through min-max normalization. The PCA implementation from the Python module scikit-learn (Pedregosa et al. 2011) was used, which implements the standard approach of Tipping and Bishop (1999).

Principal-components Search

A parameter search approach was used to find the number of components from a space of 100 to 2,000 components iterating at 100 intervals; these bounds were found from initial

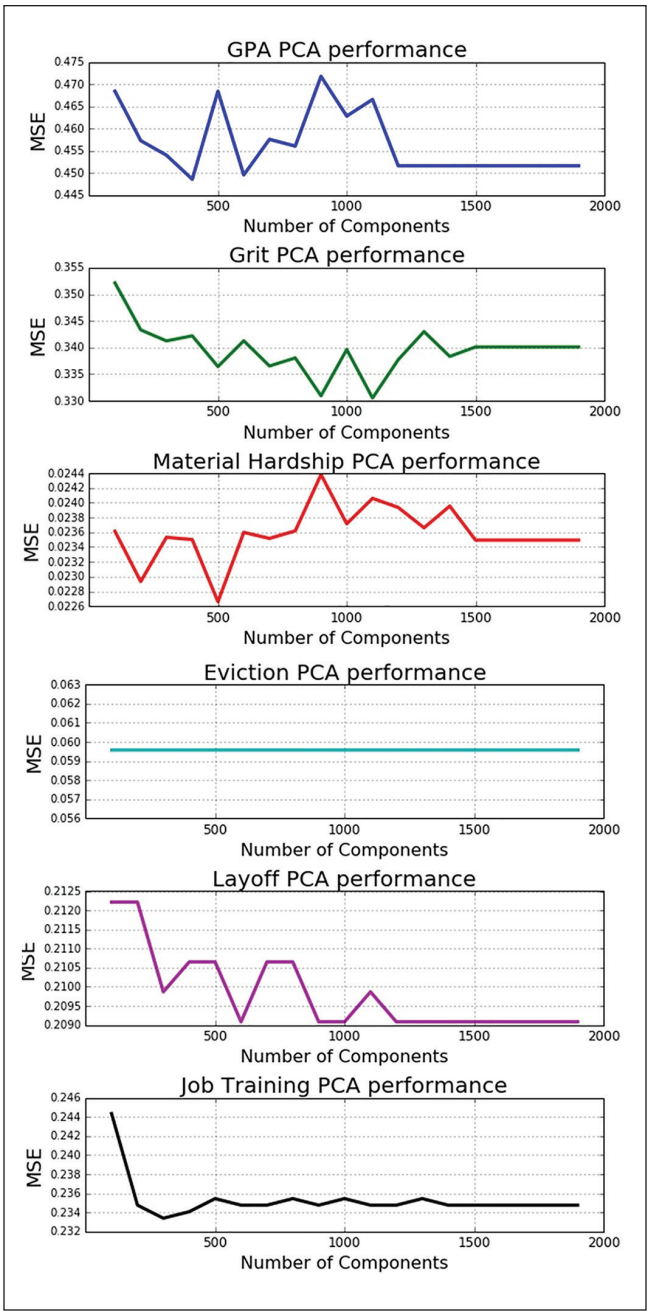

Figure 2. Model performance over number of principal components for each target.
Note: GPA = grade point average; MSE = mean squared error; PCA = principal-components analysis.

exploration in which fewer than 100 was too much information lost (the model was a poor predictor) and more than 2,000 was not enough dimension reduction (representing more features than instances). For each iteration, I produced a PCA transformation, and then a random forest (procedure described below) was fit to the transformed features. This was conducted for each target measure. Each target's MSE plotted against the number of components is shown in Figure 2, demonstrating the bias-variance balance needed to find the best model. However, it is not guaranteed that model predictions will change as components increase; eviction predictions are constant for all components. The component number producing the lowest MSE was used for further modeling.

The number of components producing the best model is not identical for each target, which needed to be accounted for in each final model. Although the number of features remained high, it was reduced from about 10,000 features, making modeling much simpler. Similar analyses were conducted for all targets, and the best components found for each target are listed in Table 1.

Modeling

I built models in Python 3.5 using the scikit-learn library (Pedregosa et al. 2011). For modeling, the ensemble technique random forests was chosen. Random forests is a technique that takes a large number of decision trees assembled on random subsets of features and instances and aggregates all predictions into a single prediction. Decision trees typically perform well with data sets with properties like those of the Fragile Families Challenge (categorical and continuous features, linear and nonlinear relationships, a small number of instances) and can produce both continuous and categorical outputs (Breiman 2001a).

The package XGBoost (Chen and Guestrin 2016) was used to build this model. XGBoost conducts an efficient tree boosting algorithm to increase model performance. XGBoost also has settable parameters: number of trees, learning rate, and a maximum depth for trees. The number of trees determines how many decision trees are used in the forest, learning rate is a boosting parameter to improve model learning, and maximum depth restricts the depth of each individual decision tree. The parameter values 250 trees, a learning rate of 0.05, and a maximum depth of 6 were used because they are the default parameters for such a model. More specific parameters can optimize performance on an individual target through parameter search (Bergstra and Bengio 2012), but this approach was not adopted here, because PCA feature engineering is my main focus. Therefore, parameters were set to be identical across all targets, and I can observe how engineered features affect performance. MSE is generally the set evaluation function for these models, but for classification targets (binary variables: eviction, layoff, and job training), additional ML metrics were used, namely, precision, recall, and F1 score (Powers 2011), as defined above in the section on terminology and metrics.

Classification Balancing

Results for an initial modeling test of the binary classification targets over the model training and model validation sets are shown in Table 2. On the left side of Table 2, the

Table 1. Number of Principal Components Found for Each Target and Their Explained Variance Found through PCA Modeling within scikit-learn.

Target Variable	GPA	Grit	Material Hardship	Eviction	Layoff	Job Training
Number of components	400	1,100	500	100	600	300
Percentage variance explained	~92	~99	~95	~80	~95	~90

Note: GPA = grade point average; PCA = principal-components analysis.

Table 2. Evaluation Scores for Binary Classification Targets.

	Classification Support for Binary Targets					
	Untouched Training Data			Resampled Training Data		
	Precision	Recall	F1 Score	Precision	Recall	F1 Score
Eviction						
No eviction	0.94	1.00	0.97	0.94	0.99	0.96
Eviction	0.00	0.00	0.00	0.00	0.00	0.00
Average/total	0.89	0.94	0.92	0.89	0.93	0.91
Layoff						
No layoff	0.78	0.99	0.87	0.78	0.98	0.87
Layoff	0.25	0.01	0.02	0.38	0.05	0.09
Average/total	0.66	0.77	0.68	0.69	0.77	0.70
Job training						
No job training	0.78	1.00	0.87	0.78	0.99	0.87
Job training	0.00	0.00	0.00	0.38	0.03	0.05
Average/total	0.60	0.78	0.68	0.69	0.77	0.69

Note: The results of training on unchanged target distributions are on the left, and the results of training with oversampling positive targets are on the right.

classification targets show low F1 scores for positive instances (instances in which eviction, layoffs, or job training took place). This is most likely due to the small number of positive instances for each target (87 positive instances for eviction out of 1,459, 267 positive instances for layoff out of 1,277, and 343 positive instances for training out of 1,461). Small numbers of positive instances create a modeling issue because there are not enough to properly train the model, leading it to fit to only the negative instances. For example, for eviction, if the model predicts a negative instance every time, it is still correct 94 percent of the time.

To counter this, I oversampled positive instances to balance the ratio of positive to negative instances (Chawla et al. 2002). Oversampling increases instances of a low-frequency class by sampling from the available instances of that class and duplicating them within the data set. The oversampling procedure I used involved random sampling with replacement, and the modeling results are shown on the right side of Table 2. Performance increased for job training and layoff, with no change for eviction. This may be because the positive instances in the training set do not accurately represent those in the validation set. However, the small number of positive instances may make it impossible to identify representative training samples for positive instances.

Cross-validation

Next, cross-validation was conducted to find the best performance on the validation set within the sixfold cross-validation procedure. The results in Figure 3 show model performance variance for each validation fold. MSE from the folds is similar, suggesting that results are not driven by outliers.

The best model from cross-validation for each target was evaluated on the model test set, and the results are shown in Table 3. To quantify the performance impact of using principal components, I constructed baseline random forest models using all features (the original set of 10,371 features) following the same procedures as outlined above (filtering zero standard deviation features, mean substitution, and oversampling), minus the feature engineering stage; results are also shown in Table 3.

Results

The models showing higher improvement using PCA features surprisingly involved only one binary classification target (job training). Baseline models performed better for regression targets (GPA, grit, and material hardship) on average, with baseline grit improved most, by about 28 percent.

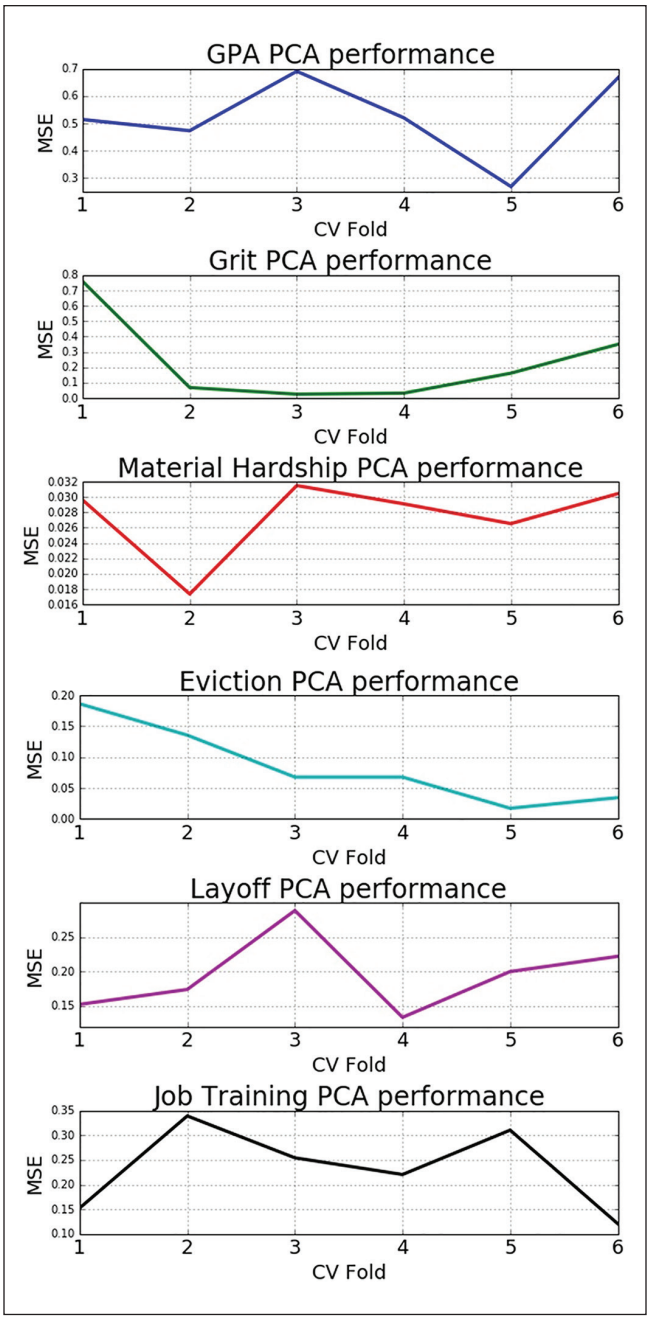

Figure 3. Cross-validation performance for all targets.
Note: CV = cross-validation; GPA = grade point average; MSE = mean squared error; PCA = principal-components analysis.

Baseline models for eviction and layoff did perform better than PCA but not to the extent of regression targets. This suggests that more features provide better predictive power for these targets and that the only target to benefit from the reduction in noise in the reduced principal components was job training.

The better performing model when comparing baseline with PCA was submitted to the Fragile Families Challenge. Examining the righthand column, most models performed at a level expected on the challenge holdout test set,[1] with a slightly higher error than in the model test set, indicating a good generalizable model. Interestingly, the layoff and GPA models performed better for the challenge holdout set than the modeling test set. It may be that instances harder to predict were present in the challenge training set compared with the challenge holdout set.

Eviction seems a harder classification problem than anticipated. Eviction initially showed high performance, but this was due to a bias in the number of nonevictions to evictions (1372 nonevictions vs. 87 evictions in the training set); even oversampling of evictions could not correct the model. Bias may also explain the lack of a relationship between number of components and performance. This may represent differences in train-test split created in the modeling stage, or a larger range of components (100–10,000) might reveal a relationship. Further work is needed to test such hypotheses, and further examination of the differences in positive instances between the Fragile Families Challenge train and holdout sets could help produce stronger generalizable results.

Overall, the job training target in the Fragile Families Challenge seems to benefit from PCA feature engineering, while the remaining targets were best fitted using original (but cleaned) features and balanced classes. Classification involved binary (0 or 1) targets, whereas regression involved continuous targets, allowing more possible noise, giving one possible reason why more features present in the model provided on average better performance. Regression targets may therefore demand more complex models, and this may be the case for the classification targets eviction and layoff as well. As seen in Figure 2, eviction showed no change with component exploration for PCA, and the higher numbered components (>1,200) for layoff showed just as good performance as the minimum chosen.

Limitations

My work has various potential limitations. First, it is heavily data driven and so does not reflect known theoretical relationships between features and targets. Future work could include theoretical feature filters or combinations to search for statistically significant predictors. Second, bias involving negative instances for specific classification targets might be better handled to increase performance using other methods than sampling with replacement. Third, the mean substitution method for missing data has known biases in causing overfitting; less biased methods should be explored (e.g., multiple imputation).

[1]Performance reported varies from challenge scores. The results here represent reworking for reliable models to be attached with this article.

Table 3. Evaluation Scores for Targets in the Baseline Models, PCA Feature Models on the Modeling Test Set, and Final Performance on the Fragile Families Held-out Test set.

	Baseline Performance (MSE)	PCA Feature Performance (Number of Components) (MSE)	Final Performance on Fragile Families Test Set (MSE)
GPA	**0.4197**	0.4581 (400)	0.3708
Grit	**0.2293**	0.2955 (1100)	0.2601
Material hardship	**0.0192**	0.0225 (500)	0.0203
Eviction	**0.0539**	0.0684 (100)	0.0616
Layoff	**0.2322**	0.2417 (600)	0.2233
Job training	0.2443	**0.2318 (300)**	0.2617

Note: Best models comparing baseline and PCA are in boldface type. GPA = grade point average; MSE = mean squared error; PCA = principal-components analysis.

Conclusions

My work used one type of ML feature engineering procedure to find generalizable models for a complex social science data set. My data-driven approach took advantage of relationships with the data, ignoring potential theoretical approaches to feature selection. Such models might be used to better identify certain aspects of families at risk. Furthermore, I highlight differences in predictive power reflecting the data set imbalance across classification instances, demonstrating a need for more observed cases of classification targets.

References

Bergstra, James, and Yoshua Bengio. 2012. "Random Search for Hyper-parameter Optimization." *Journal of Machine Learning Research* 13:281–305.

Breiman, Leo. 1996. "Bias, Variance, and Arcing Classifiers." Technical Report No. 460. Berkeley: University of California.

Breiman, Leo. 2001a. "Random Forests." *Machine Learning* 45(1):5–32.

Breiman, Leo. 2001b. "Statistical Modeling: The Two Cultures (with Comments and a Rejoinder by the Author)." *Statistical Science* 16(3):199–231.

Chawla, Nitesh V., Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. 2002. "SMOTE: Synthetic Minority Over-sampling Technique." *Journal of Artificial Intelligence Research* 16:321–57.

Chen, Tianqi, and Carlos Guestrin. 2016. "Xgboost: A Scalable Tree Boosting System." Pp. 785–94 in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York: Association for Computing Machinery.

Dietterich, Tom. 1995. "Overfitting and Undercomputing in Machine Learning." *ACM Computing Surveys* 27(3):326–27.

Jayalakshmi, T., and A. Santhakumaran. 2011. "Statistical Normalization and Back Propagation for Classification." *International Journal of Computer Theory and Engineering* 3(1):89–93.

Keogh, Eamonn, and Abdullah Mueen. 2011. "Curse of Dimensionality." Pp. 257–58 in *Encyclopedia of Machine Learning*. New York: Springer.

Kohavi, Ron. 1995. "A Study of Cross-validation and Bootstrap for Accuracy Estimation and Model Selection." Pp. 1137–45 in IJCAI '95: *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, Vol. 2. San Francisco, CA: Morgan Kaufmann.

Little, Roderick J. A., and Donald B. Rubin. 2014. *Statistical Analysis with Missing Data*. Hoboken, NJ: John Wiley.

Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. "scikit-learn: Machine Learning in Python." *Journal of Machine Learning Research* 12:2825–30.

Powers, David Martin. 2011. "Evaluation: From Precision, Recall and F-measure to ROC, Informedness, Markedness and Correlation." *International Journal of Machine Learning Technologies* 2(1):37–63.

Schlomer, Gabriel L., Sheri Bauman, and Noel A. Card. 2010. "Best Practices for Missing Data Management in Counseling Psychology." *Journal of Counseling Psychology* 57(1): 1–10.

Tipping, Michael E., and Christopher M. Bishop. 1999. "Probabilistic Principal Component Analysis." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 61(3):611–22.

Wang, Zhou, and Alan C. Bovik. 2009. "Mean Squared Error: Love It or Leave It? A New Look at Signal Fidelity Measures." *IEEE Signal Processing Magazine* 26(1):98–117.

Wold, Svante, Kim Esbensen, and Paul Geladi. 1987. "Principal Component Analysis." *Chemometrics and Intelligent Laboratory Systems* 2(1–3):37–52.

Author Biography

Ryan Compton is a PhD candidate at UC Santa Cruz and is interested in how individual and social behavior can be explained by large scale data analysis. His work primarily focuses on how and why individuals' social roles change over time in online communities. His research creates design implications that improve the success for people involved in online communities. He aims to build systems that improve the well-being and success of groups through research and data-driven design.