Wellbeing Prediction of Fragile Family & Children

Yue Xing
Department of Electrical Engineering
yuex@princeton.edu

Guangyuan Hu
Department of Electrical Engineering
gh9@princeton.edu

Abstract

In this project, we participate in Fragile Family Challenge, which is a data analysis task to predict the levels of wellbeing for children in a variety of fragile families. We are provided with the background data collected when the children were nine years old to infer six key outcomes when they are fifteen.

As the basic analysis methods, different machine learning algorithms like linear regression and lasso are adopted to model the correlation between the background data and the key results. Then we also utilize Principal Component Analysis to choose the critical features for prediction. Moreover, we make three extensions, namely the feature selection to remove extremely sparse features, the feature space extension to make a more reasonable missing data imputation and the bootstrapping method to produce an error estimation.

The results show that our proposed feature removal and feature extension methods improve the prediction performance by more than 20% comparing with just using mean to complement hollow feature. The rank and precision in FFC and the bootstrap extension work also makes our conclusion more solid.

1 Introduction

The Fragile Family & Children Wellbeing Study (FFCWS) is a project to help researchers and policy makers understand the status of disadvantaged children in the United States. Since the late 1990s, the FFCWS staff has been keeping track of thousands of fragile families which refer to the unmarried parents and their children. The Fragile Families Challenge (FFC) incorporates one of FFCWS's subtasks to yield the insights about the lives of the children aged 9 to 15.

In this project, we mainly focus on six important outcomes to evaluate the status of the children when they are 15 years old. The outcomes include their GPA, grit, material hardship and whether they are evicted, laid off or trained for jobs. We use given training data to get a good model for inference and compare our prediction with the on-line database. The results are give in Section 4.

Our contributions to make better predictions include:

- Use Principal Component Analysis (PCA) to select more meaningful features
- Remove sparse features with so many missing entries
- Add to the dimension of feature space to indicate the missing status of data
- Do bootstrapping and give the reader/user an insight on prediction error
- Make predictions through various machine learning models after background data is preprocessed by preceding methods

2 Related Work

2.1 Dataset and Features

As is shown in the website [1], the dataset is organized like Figure 1. The researchers have the background data and the ground truth for thousands of fragile families. To facilitate supervised learning methods, some of the families are used as training data while the others are kept secret for testing. From each family, about twelve thousand features are collected. The features can be continuous or binary and there are lots of missing data due to the subjects' unwillingness to answer or the branch divergence in some questionnaires.

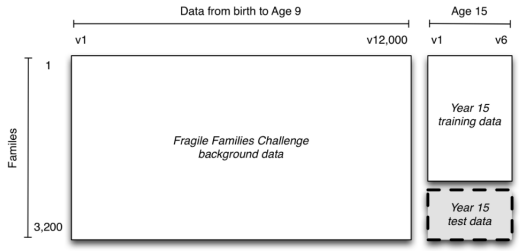

Figure 1: Dataset Organization for Fragile Family Challenge

2.2 Data Preprocessing

According to description of the dataset, the initial feature set is given by the FFC group so the preprocessing stage mainly involves feature selection based on the provided ones. In this project, it is easy to observe from Figure 1 that we have a exceedingly big dataset. To get the critical features for prediction, we compress the feature space using PCA algorithm as the primary selection method. This feature analysis is tuned to present us with 100 most useful features for prediction. We will compare the performance of using and not using PCA in Section 4.

2.3 Regression/Classification Methods

For this assignment, in total we tried 12 regression models to predict continuous variables and 7 classification models for binary prediction. We implemented those models by using the scikit-learn package in python[2]. With different data preprocessing and feature manipulation methods suggested in Section 3, we get the evaluation results to demonstrate whether these methods are able to improve the performance. All the regression models used in this project are listed below:

- Linear Regression
- Ridge Regression
- Lasso Model: Linear Model trained with L1 prior as regularizier
- ElasticNet: Elastic Net model with iterative fitting along a regularization path
- Lars: Least Angle Regression model
- LassoLars: Lasso model fit with Least Angle Regression
- Orthogonal Matching Pursuit (OMP)
- Bayesian Ridge Regression (BR)
- Bayesian ARD Regression (BA)
- Decision Tree Regressor with max_depth = 2 (DTR2)
- Decision Tree Regressor with max_depth = 5 (DTR5)
- Random Forest Regressor (RFR)

Moreover, we use a variety of classification models for the three binary outcomes:

- Support Vector Machine (SVM)
- K Nearest Neighbor (KNN)
- Gaussian Naive Bayesian (GaussianNB)
- Decision Tree Regressor with max_depth = 2 (DTR2)
- Decision Tree Regressor with max_depth = 5 (DTR5)
- Perceptron
- Random Forest (RF)

2.4 Evaluation

We evaluated the performance in two different ways: First we use the submission result from the FFC website. This will generate the evaluation of our 6 key output predictions. However, limited by its maximum daily submission times, we use a second method to evaluate our models. Since for each of the 6 outputs, we have more than 1000 train data (we call them true data). We divide that 1000 true data into 2 groups. As what we did for cross validation, we randomly select 80% as training data and the other 20% as test data. Then we use the training group to train the model and use the test group to test. As we can imagine, the second method could provide a lower bound of the performance because we just use 80% of the whole true data for training, which is less than our real training process which uses all the true data.
We use two measurments for evaluating the performance of continuous outputs: one is R Square Value and another one is Mean Square Error(which is also used in the FFC). For binary prediction, we just use prediction accuracy to evaluate the performance.

3 Extensions

3.1 Basic Feature Preprocessing

In order to fit the data into our regression and classification models, we need to regularize the input data, and properly map the features to a better space. In this subsection, we assume the training data has no holes, which means all the data are known and no NaN/-1/-2/etc (These complex situations will be processed in next subsections), an here we just introduce the basic preprocessing we have done in this assignment.
We standardlize the data by mapping each of all features to a zero-mean one-variance property. Then we use PCA(Principal Components Analysis) to decrease the number of features. As we know, we only have 1000 training data but each of them has more than 10000 features(if we don't use the preprocessing method in the next subsections), so it is necessary to do the dimensions decrement.

3.2 Sparse Feature Removal

By briefly going through the whole dataset, it is observed that there are many sparse features whose answers are almost blank (missing data). To be more specific, if 80%(or some other percentage) data is missing for one feature, it's very imprecise to use some statistic value to predict the missing data. It's also useless to try machine learning method to use other features to complement these missing data in this feature since if other features are unrelated to this almost-missing feature, the prediction is imprecise and if other features are related to this almost-missing feature, this missing feature doesn't contribute to the model. Also due to the large size of the feature space, it is hard to do a case-by-case analysis for why answers are always missing for a certain feature. Therefore, it is unfair and imprecise for most of the families to use the artificial data and then make prediction. As a solution, we choose to throw away features with more blanks than a certain threshold(we call these features are unusable) because they are not considered useful for a general model applied to all the families. We will compare the performance when using different thresholds in Section 4.

3.3 Feature Space Extension

In the previous Section 3.2, we remove some highly sparse features but this won't let us completely get rid of the missing data. Thanks to the idea suggested in [3], we propose a method to extend the

feature space to present missing data in a more reasonable way. From a matrix perspective, we are actually adding columns to the feature matrix. For each missing data like a NaN, -1, -2 or etc. in each feature vectors, we replace the inplace missing data with the mean of this column and add an extra column with 1s for the missing entries and 0s for valid and other kinds of missing entries(For example, when we see there's an -1 for one item in one of its features saying its f1. We will add one extra feature in the feature space, set the extra feature of all items whose f1 is not -1 to zero and whose f1 is -1 to 1). The additional column works like an indicator of missing status of this feature and more missing status columns are needed if we take the different types of missing into consideration.

3.4 Bootstrapping

After we evaluate the performance by both getting the performance information from FFC website and using ourselves evaluating methods. We have some sense of what a best-performance regressor/classifier should look like. However, then we are more interested in what is the real performance that a regressor/classifier can achieve. Then we tried the bootstrapping method to predict the prediction performance. To be more specific, we are more interested in the tight lower bound of the performance so that we can proudly claim the performance of our classifier. So we use least-one-out bootstrap to evaluate the model to get the prediction precision distribution.

4 Results

In this section, we show the experiment results both evaluated from FFC website and by our own evaluation method(including comparing basic Feature Processing methods, Sparse Feature Removal method and Feature Space Extension Method). Some result is denoted by NC, which means that method does not converge.

Model	w/o PCA	with PCA	Model	w/o PCA	with PCA	Model	w/o PCA	with PCA
Linear Regression	1.163	0.539	OMP	1.730	0.518	Ridge Regression	1.148	0.539
BR	1.149	0.510	Laaso	0.503	0.503	BA	0.503	0.503
ElasticNet	0.503	0.503	DTR2	0.533	0.508	Lars	NC	0.539
DTR5	0.604	0.548	LassoLars	0.503	0.503	RFR	0.545	0.565

Table 1: The mean square error for the prediction of GPA.

Model	w/o PCA	with PCA	Model	w/o PCA	with PCA	Model	w/o PCA	with PCA
Linear Regression	NC	0.266	OMP	2.519	0.248	Ridge Regression	0.653	0.266
BR	0.654	0.245	Laaso	0.242	0.242	BA	0.670	0.242
ElasticNet	0.242	0.243	DTR2	0.247	0.248	Lars	NC	0.266
DTR5	0.246	0.266	LassoLars	0.242	0.242	RFR	0.267	0.274

Table 2: The mean square error for the prediction of grit

Model	w/o PCA	with PCA	Model	w/o PCA	with PCA	Model	w/o PCA	with PCA
Linear Regression	0.050	0.021	OMP	0.533	0.021	Ridge Regression	0.050	0.021
BR	0.021	0.021	Laaso	0.021	0.021	BA	0.022	0.021
ElasticNet	0.021	0.021	DTR2	0.023	0.022	Lars	NC	0.021
DTR5	0.026	0.024	LassoLars	0.021	0.021	RFR	0.026	0.024

Table 3: The mean square error for the prediction of material hardship.

4.1 Basic Feature Preprocessing

In Table 1, Table 2 and Table 3 we compare the result of predicting all 6 values w/wo PCA. As we can see, in most cases, PCA improves the precision a lot since it decreases the unrelated features.

4.2 Sparse Feature Removal

Here we compare the precision(by using MSE) of GPA. Three cases are without feature removal, removing features with holes more than 40% data and more than 80%. As we can see, by removing those unusable features, it improves the precision(decreasing MSE) a lot.

Model	80 Removal	40 Removal	No Removal	Model	80 Removal	40 Removal	No Removal
Linear Regression	0.474	0.494	0.594	OMP	0.456	0.481	0.575
Ridge Regression	0.474	0.494	0.594	BR	0.445	0.452	0.553
Laaso	0.439	0.446	0.544	BA	0.434	0.446	0.544
ElasticNet	0.439	0.446	0.547	DTR2	0.45	0.467	0.564
Lars	0.474	0.494	0.594	DTR5	0.523	0.515	0.663
LassoLars	0.439	0.446	0.544	RFR	0.482	0.501	0.582

Table 4: Evaluation using mean square error for GPA. Comparing different threshold of removal

Model	w/o PCA	with PCA	Model	w/o PCA	with PCA	Model	w/o PCA	with PCA
Linear Regression	1.163	0.539	OMP	1.730	0.518	Ridge Regression	1.148	0.539
BR	1.149	0.510	Laaso	0.503	0.503	BA	0.503	0.503
ElasticNet	0.503	0.503	DTR2	0.533	0.508	Lars	NC	0.539
DTR5	0.604	0.548	LassoLars	0.503	0.503	RFR	0.545	0.565

Table 5: The mean square error for the prediction of GPA.

4.3 Feature Space Extension

Here we still compare the precision(by using MSE) of GPA. Two cases are wo/w feature space extension. The feature space extension doesn't help too much here because there are too many unusable features and it will improve when combining this with feature removal.

Model	w/o Extension	with Extension	Model	w/o Extension	with Extension	Model	w/o Extension	with Extension
Linear Regression	0.539	0.594	OMP	0.519	0.575	Ridge Regression	0.539	0.594
BR	0.51	0.553	Laaso	0.503	0.544	BA	0.503	0.544
ElasticNet	0.503	0.547	DTR2	0.509	0.564	Lars	0.539	0.594
DTR5	0.548	0.663	LassoLars	0.503	0.544	RFR	0.566	0.582

Table 6: Evaluation using mean square error for GPA. Comparing with/wo feature extension.

Model	w/o PCA	with PCA	Model	w/o PCA	with PCA	Model	w/o PCA	with PCA
Linear Regression	1.163	0.539	OMP	1.730	0.518	Ridge Regression	1.148	0.539
BR	1.149	0.510	Laaso	0.503	0.503	BA	0.503	0.503
ElasticNet	0.503	0.503	DTR2	0.533	0.508	Lars	NC	0.539
DTR5	0.604	0.548	LassoLars	0.503	0.503	RFR	0.545	0.565

Table 7: The mean square error for the prediction of GPA.

4.4 Bootstrap

We run a bootstrap as mentioned in Section 2.4 with k = 40, for the Bayesian ARD model. The distribution of prediction MSE is shown in Figure 2. As we can see, the 95% confidential-interval of the estimated gpa is about [0.405, 0.4575].

4.5 Evaluation from FFC Website

We checked the FFC submission page at Tue 28th 2017, 4:46pm. Our prediction precision and rank for each of the 6 key values is as below: ID: yuex. GPA:0.39273 (21). Grit:0.21997 (13). MaterialHardship:0.02978 (24). Eviction:0.05660 (13). Layoff:0.22453 (16). JobTraining: 0.27925 (16).

5 Discussion and Conclusion

In this work, we first tried to use basic method to complement holes in the dataset, and also used basic preprocessing methods to fit the data to the model. Then we dug deeper into methods to make more used of different types of hollow features by adding more features to describe the hollow situations in each existing features. Besides, we also found the situation that some features are really unusable, which means there are too many holes in that feature. No matter that feature is related to our model or unrelated to our model, using it won't help prediction, so we used features removal to solve the problem. The evaluation result shows that our method using feature removal improves

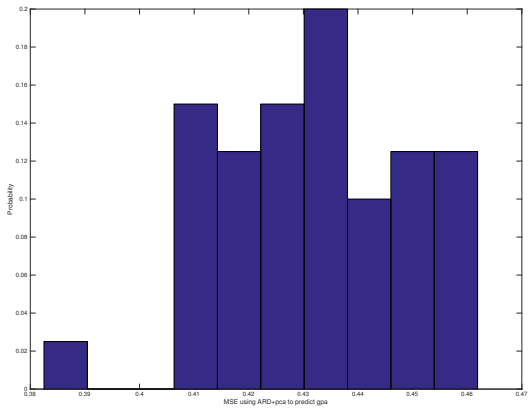

Figure 2: Bootstrap: The distribution of prediction

the performance of prediction but using feature extension alone doesn't improve the performance because that makes the feature space too large. However, the best performance appears actually when combining feature removal and extension together. Finally, we submitted our prediction result to the FFC and got quite good result as denoted in Section 4.5.

In order to make our conclusion more concrete, we used the least-one-out bootstrap to analyze the distribution of predicting precision and shows that the lower bound 95% confidential-interval of the best performance predictor is [0.405, 0.4575].

References

[1] Challenge FF. Fragile Family Challenge; 2017. `http://www.fragilefamilieschallenge.org/`.

[2] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. 2011;12:2825–2830.

[3] Subasi MM, Subasi E, Anthony M, Hammer PL. A new imputation method for incomplete binary data. Discrete applied mathematics. 2011;159(10):1040–1047.