Predicting Health Behaviors and Outcomes for U.S. Counties

Zhengyuan Xu
University of Pennsylvania
CIS
zhengyx@seas.upenn.edu

Jiacheng Xu
University of Pennsylvania
CIS
jiacxu@seas.upenn.edu

Abstract

In order to investigate and predict the health situations of the people in the United States, relevant data are collected throughout the country and are divided into counties. The task of this project is to predict health behaviors and outcomes for U.S. counties as a function of: demographics; Socio-Economic Status (SES); urban/rural environment and words in tweets from that county. Multiple models are trained, but they have different performances. The performance of the best model has a training error 0.0596, and a test error of 0.0782.

1. Introduction

There are about 3,000 counties in the U.S. However, only 2000 of them are valid data because the rest 1,000 have too few valid samples. The data of the 2,000 counties contains two sets of features:

- Word frequencies from tweets
- SES and demographics

Data from about 1,000 counties are selected to be the training data and the rest are treated as test data.

1.1. Features and Labels

Tweets feature are mapped to source county and tokenized. Tokens are then counted and mapped to 2,000 LDA topics, which are from LDA on a large set of Facebook posts. The topic frequencies in each county are provided in the features. Also, the top words in each LDA topic are available.

Features other than tweets and their description are given in Table 3.

2. Error Metric

The evaluation metric we use is a modified L2 prediction accuracy. Suppose our data set contains K labels $\{Y_1, Y_2, \ldots, Y_m, \ldots, Y_K\}$ that we are predicting. Let us denote the training data set labels as $\left\{Y_1^{\text{train}}, Y_2^{\text{train}}, \ldots, Y_m^{\text{train}}, \ldots, Y_K^{\text{train}}\right\}$. So each Y_m^{train} is a column vector containing entries $Y_{m,i}^{train}$ for each instance X_i. For each label, determine the following quantity:

$$\begin{aligned} MaxMin_m &:= \max\left(Y_m^{train}\right) - \min\left(Y_m^{train}\right) \\ &= \max_i\left(Y_{m,i}^{train}\right) - \min\left(Y_{m,i}^{train}\right) \end{aligned}$$

The purpose of this quantity will be to normalize errors for labels that are on different scales.

Now, suppose we have a test data set of N instances, with given labels $\{Y_1, Y_2, \ldots, Y_m, \ldots, Y_N\}$ and we predict labels $\left\{\hat{Y}_1, \hat{Y}_2, \ldots, \hat{Y}_m, \ldots, \hat{Y}_N\right\}$. The error metric is defined as:

$$\begin{aligned} error &= \frac{1}{N}\left(\Sigma_{m=1}^{N}\left(\frac{\text{RMSE of } Y_m}{MaxMin_m}\right)\right) \\ &= \frac{1}{N}\left(\Sigma_{m=1}^{N}\left(\frac{\sqrt{\left\|Y_m - \hat{Y}_m\right\|_2^2}}{MaxMin_m}\right)\right) \\ &= \frac{1}{N}\left(\Sigma_{m=1}^{N}\left(\frac{\sqrt{\frac{\Sigma_{i=1}^{N}\left(\left|Y_{m,i}-\hat{Y}_{m,i}\right|^2\right)}{N}}}{\text{MaxMin}_m}\right)\right) \end{aligned}$$

3. Best Model

Our final best model is an ensemble model, along with Principal Component Analysis (PCA). The final training error is 0.0596, and the final test error is 0.0782. Hyperparameters of this model is given in table 1.

3.1. Pre-processing and PCA

The features in the training data are separated into 3 blocks according to their values. First all features are divided into 2 blocks: real-numbered tweets topic frequencies, and other features. Then the other features are further divided into 2 new blocks: real numbers and categoricals.

Number of PCs	125
Weight of Ridge Regression	0.78
Ridge Regression Penalty	0.07
Weight of Bagging Ensemble	0.22
Number of weak Bagging models	50
Base Learners for Bagging	DecisionTree

Table 1. Hyper-parameters of the final model

We mean-centered the real numbers and performed PCA on the tweets features. The number of principal components the final model uses is 125, which is selected using 5-fold cross-validation on the training dataset.

After the PCA on tweets frequencies, we concatenate the PCA scores of the training data with the normaliized real-numbered features and categorical features to form the final training input for the model. The final number of features for any training example $x \in X_{train}$, is $125+21+1 = 147$. The extra 1 column refers to the "1s" appended to the feature vector to represent the bias in the model. This method drastically reduced the number of dimensions in the training features, from 2021 to 147, thus decreasing the training time and also the cross-validation error. Overfitting is also reduced as a result. A plot of how the training and validation error changes with the number of principal components included in the features is given in figure 1.

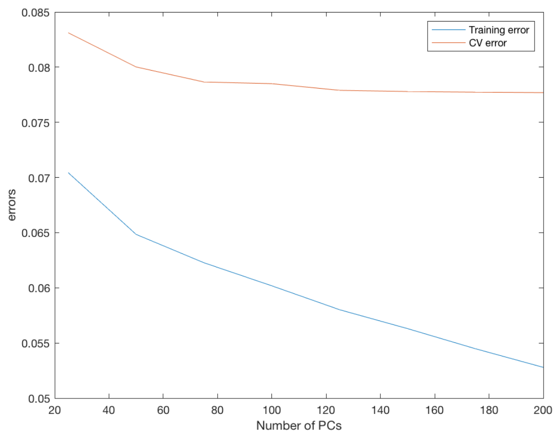

Figure 1. Training Error and Validation Error vs. number of principal components.

For validation set and test set, the PCA coefficients calculated from the training set is used to map the features in the two sets onto the principal components.

3.2. The Model

The final model used ensemble learning, by adding weighted predictions of two models. The first model is a linear regression model with L2 (ridge) penalty. We implemented this regression by explicitly solve the model weights using its closed form formula:

$$W = (X^T \cdot X + k \cdot I)^{-1} \cdot X^T \cdot y \tag{1}$$

where W is the regression model weights, X is the training features, k is the L2 penalty coefficient, I is the identity matrix, and y is the training labels. At the very beginning, we achieved a test error of 0.079 by only using the linear regression model with ridge penalty coefficient = 0.01. Then we discovered that the gap between the validation error and training error is small, which could be indicating an underfitting model. The training error for the linear model is 0.0653 and the validation error is 0.0793. To increase the model complexity, we used matlab's fitrensemble to train a bagging ensemble model. We used bootstrap aggregating (bagging) for the ensemble-aggregation method. Then the number of weak learners is 50. Each learner is a deep regression tree. The final prediction is given by:

$$w_1 * Prediction_{linear} + w_2 * Prediction_{bagging}$$

where w_1 is the weight fraction of linear model and w_2 is the weight fraction of the bagging ensemble. Note that $w_1 + w_2 = 1$.

This ensemble method successfully reduced the training error to 0.0586, and the validation error to 0.0771. This indicates that the overall model complexity has increased, as the ensemble method is able to overfit the data distribution very well. Also, the validation error did not suffer from increased model complexity. Therefore, the new ensemble model is a better fit for the training dataset. A plot of how the errors vary when the ratio of the weight of the linear model predictions to the weight of the ensemble model predictions is given in figure 2.

As can be seen from figure 2, the curve for training error keeps going up as more weight is allocated toward linear regression model, because the linear model is under-fitting. The best balance is found at around 0.8, where the validation error reaches at a minimum. The training error at that point is also reasonably low. This value corresponds to the value of the final weight of linear model in our final model.

4. Other Methods

This section discusses other models we trained. These models(except the final model) were not selected as the final submission model because they had higher errors. Cross validation errors and training errors of these models are given in table 2. All the methods used PCA on the 2000 tweets topics frequencies, with number of components = 125. The rest of the features are mean-centered if they are real numbers. 5-fold validation is used to tune the hyper-parameters.

Model	Training Error	Cross Validation Error	Hyper-parameters
Linear Regression (Discriminative)	0.0654	0.0795	L2 penalty: 0.01
K-means (Generative)	0.1243	0.2005	Number of clusters: 35
K-nearest-neighbors (Instance-based)	0.1349	0.1379	Number of neighbors: 40
Weighted Bagging and Ridge Regression (Final Model)	0.0585	0.0785	Too long, see caption

Table 2. Training and cross validation errors on 4 models trained. Hyper-parameters of the final model: L2 penalty: 0.01, Weight of Linear: 0.8, Weight of Bagging: 0.2, Bagging number of weak learners: 50.

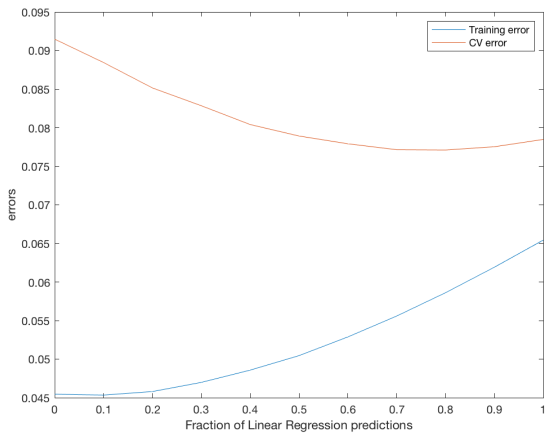

Figure 2. Weight fraction of linear model. The final prediction is calculated as $w_1 * model_1 + w_2 * model_2$

4.1. Discriminative Method

We used closed-form formula to find the weights of the linear ridge regression, as in equation (1). Training is straight-forward as the weight matrix is calculated once and used to produce predictions by finding the dot product of features and weights. An error-vs-hyperparameters plot is given in figure 3. The hyper-parameter tuned here is the L2 penalty coefficient. We used cross-validation to test a number of coefficients to use, and found out that the optimal coefficient is around 0.8.

4.2. Generative Method

For this method, we first run K-means on the training dataset and assign a cluster index to each training example. Then we transform the indices to a one-hot vector. This vector is then used as the new feature for a training example. We then run a linear regression with ridge penalty on the newly obtained one-hot features. An error plot is given in figure 4. The hyper-parameter tuned here is the number of clusters. We did not know how many clusters there might be in the examples, so we tried a range of values for k. The optimal k we found is k = 25, the corresponding training and validation errors are 0.1288 and 0.1855. We also found that as k increases, the training error decreases monotonically, as the model complexity has been increasing with k. However, the validation error stays high, meaning that the model is only doing overfitting on training examples.

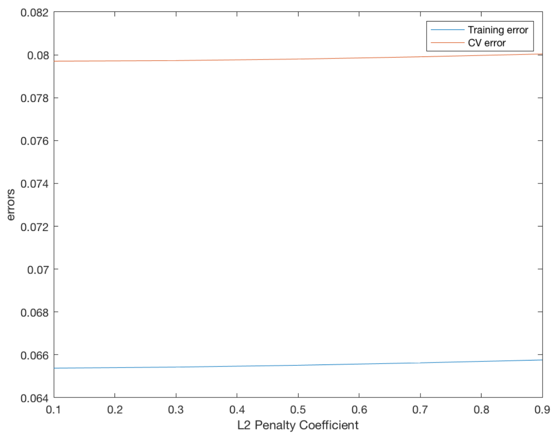

Figure 3. Training and CV errors as a function of L2 penalty.

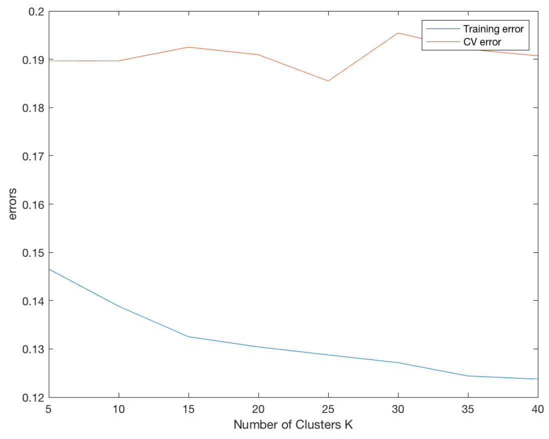

Figure 4. Training and CV errors as a function of number of clusters.

4.3. Instance-based Method

We used K-nearest-neighbors for this method. The distance between any two example points used is the L2 norm distance. An error plot is given in figure 5. The hyper-parameter tuned here is the number of nearest neighbors used when predicting the value of a test data point. As we

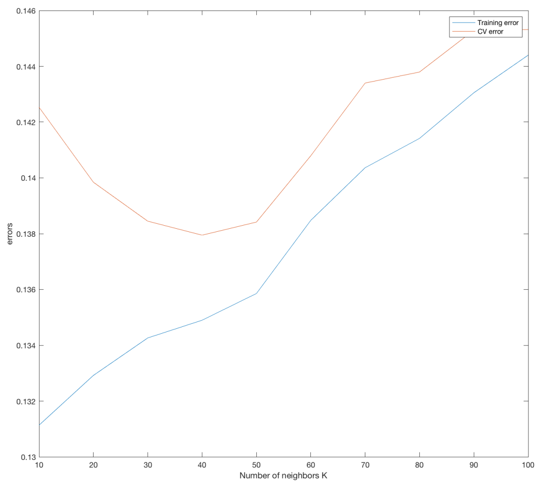

Figure 5. Training and CV errors as a function of number of nearest neighbors.

can see, the training error increases as k increases. On the other hand, a minimum is found at k = 40 for the cross validation error. We found that the training error and cross validation error when k = 40 are 0.1349 and 0.1379.

5. Interpretation

The model tells us that public health in the US is highly different cross different counties. We can see that counties that share similar situations (i.e similar values for features) also share similar health conditions. Particularly, some features are very good indicators of public health by counties. There are correlations among y-variables. The correlation plot of y-variables are following: It is clear that

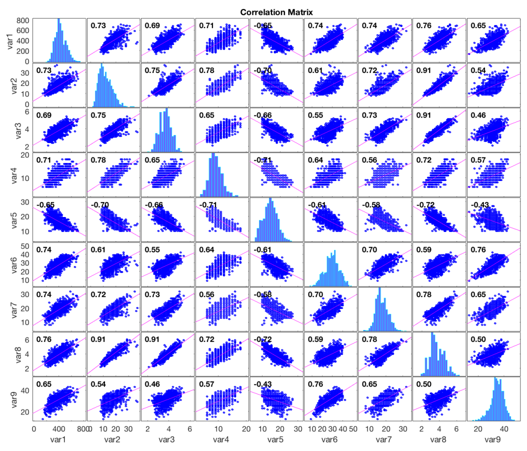

Figure 6. Correlation map of 9 y-variables in training data-set

most variables have correlation value bigger than 0.5 or smaller than -0.5, implying they are either positively related or negatively related. The highest correlation value is 0.78, which occurred between first variable and fourth variable. From our experiment, we found out that the most predictive features are our top principle components of twitter data-set. This makes sense because top principal components of twitter data-set captures most variance of the distribution of dataset and thus gives more information.

For counties demographic data, the most predictive features are logged median house income and measure of access to a healthy food environment. In fact, these two features themselves are highly correlated too. Based on our model, it is hard to tell the impact of a single twitter topic since we normalized the twitter data-set and did PCA before any model. However, the prediction accuracy of top principal components showed the predictive power of twitter topics.

6. Conclusion

From the training/CV error plots showed above, we can see that all models have such trend: as the model complexity increases, the training error decreases but the CV error decrease at beginning but increase eventually. The reason for such trend is because the model is trying to overfit the data as the complexity grows. The model that did best both in training and test data-set is the final model which is a weighed bagging of ridge regression. The method that did worst is the K-means algorithm.

In order to improve the accuracy, we did some feature engineering and picked several methods. For example, we firstly tried to do PCA on all training data but then later we found out doing PCA only on twitter data-set was more useful. Before the PCA, we mean and normalize the twitter data set, which also helped since the original data-set contains value small values, which might lead to floating errors in the matrix manipulation. The choice of hyper-parameters is definitely a key step. We basically use exhaustive search method to choose best hyper-parameters. Last but not least, we tried different cross validation folds to see which balance the best between too few training data and too few cross validation data.

We did many things that turned out to be not useful or even wrong. We used to do PCA in the wrong way. We projected the test data-set onto its own principal components instead of the PCA space of training data-set. Thanks to one of the teaching assistant who pointed out our error. And same type of error for mean and normalize of test data-set.

The explanations of the trends above is a fundamental concept in machine learning, which is model complexity. In the ridge regression one, model complexity grows as $lambda$ decreases. In K-means one, model complexity grows as number of cluster increases. In KNN one, model complexity grows as number of neighbors k decreases. In the final model, model complexity increases as the penalty coefficients decreases, depth of tree bagging increases. The trends showed that when model complexity gets too large, the model would start to fit the noises of the data-set, which made models dis bad job in testing data-set.

Potential future researches: There are several important issues that can be addressed in the future. One is to handle data-set that has much less observations compare to number of features. We know that many models tend to over-fit the data-set, especially when number of observations this small. Another direction of research could be how to best ensemble different models. In our case we only tried ensemble tree method and ridge regression. There should be more systematic way in the future of selecting ensemble models.

	Demographic and socioeconomic factors
demo_pcblack	Percent African American population
demo_pcfemale	Percent female population
demo_pchisp	Percent Hispanic population
demo_pcwht	Percent White population
demo_under18	Percent population under 18 years
demo_over65	Percent population over 65 years
ses_edu_coll	Percent of population with at least a college degree
ses_pcunemp	Percentage unemployed
ses_incomeratio	Ratio of household income at the 80th percentile to that at the 20th percentile
ses_log_hhinc	Median household income (logged)
	Infrastructural factors
ses_foodenvt	A measure of access to a healthy food environment
ses_pcaccess	Percent of population with limited access to healthy foods
ses_pcexerciss	Percent of the population with access to places for physical activity
ses_pchousing	Percent of households with at least 1 of 4 housing problems
nchs_2013	Counties classified as large urban, suburban and so on, based on population statistics
largemetro	Central and fringe metro counties with 1 million or more residents
mediummetro	Counties with 250,000 - 999,999 residents
smallmetro	Counties with 50,000 - 250,000 residents
micro	Central and noncore counties within micropolitan areas
metrononmetro	coded 0 for rural counties; NA for everything else
ses_pcrural	Percentage of the county which is considered rural

Table 3. Other features