CIS 520, Machine Learning, Fall 2018
Final Project
Final Project Report

Wentao He, Xinqi Li

December 11, 2018

1 Results

1. The table summarizing the accuracy for each of our methods is shown below:

Methods	Test Error	Training Error
Our own method	0.0810	0.0805
k-means	0.8969	0.7658
SVM	0.1657	0.1043
knn	0.1880	0.1895
Elastic Net	0.1770	0.1532

2 Leaderboard Submission - 0.0810

1. Overview

   This particular problem presented by this final project is a regression problem. We are asked to make our label prediction and compare our predicted labels against the real labels in order to achieve the highest accuracy possible. At our disposal we have the observed data and the external variable taht we are trying to predict. To summarize the features and labels that are contained in this project, column 1 contains the unique FIPS code of each county. Columns 2-22 are demographic and SES features, and columns 23-2022 are LDA topic frequencies from tweets. We are supposed to implement a method to fit the model and a prediction method that returns the predicted label given existing observations. We are also given an error metric to judge our algorithm's performance, namely, accuracy. Therefore, there are many algorithms such as linear regression, random forest, Support Vector Machine (SVM), et cetera, available, and our goal is to make a prediction that is as accurate as possible while experimenting with different models and parameters.

2. Training Procedure

   (a) 0.93

   This is our first approach, the simplest one. We just want to experiment with the data and labels first and make our first prediction, regardless the accuracy. For features, we kept the first 21 column, and we performed a theme clustering on the rest of the features to reduce dimensionality, which is done using Python. We first load the `topics.csv` file, and we gropued the features into arbitrary number of groups based on their meanings and similarities. The we used linear regression, `fitrlinear`, as our model to make label prediction.

(b) 0.1135

This approach is similar to what we did for our first run, instead, we only chose the first 21 columns of features. We simply ignored the rest of the columns as we are uncertain of the performance of our theme clustering algorithms. For our model, we sticked with linear regression, `fitrlinear`.

(c) 0.0810

0.0810 is by far the most accurate prediction result that we have ever achieved. What we did is that we picked the first 21 columns of features, and we performed principal component analysis (PCA), `pca`, on the rest of the data and we only picked the first 100. Then we performed normalization on the first 121 columns of features. Then we used linear regression, k-nearest-neighbors, neural net with 25 hidden layers, SVM, and random forest with 450 trees. Finally we ensembled all prediction from all the models listed above and used random forest with 70 trees to make our final label prediction.

3. Analysis of Methods

(a) Training Procedures

As for dealing with features,we didn't reduce the dimensionality of the first 21 columns of features and only scaled them using the standard deviation. As for the rest of the features starting from column 22, we need to reduce their dimensionality. To reduce the dimensionality of the rest of the 2000 plus columns of features, we used `pca`. Theoretically, after performing `pca`, we only need the first 35 columns to actually repeat it on the remaining 90% of the data. However, after training our model, we have discovered that the more columns of features we used, the better the performace. Therefore we used 100 columns in the end. Regarding feature selection, `pca` actually made the most difference as you can see from section 'What did not work.'

(b) What works

Every model that is mentioned above worked fined and we only need to keep adjusting the parameters to lower the cost function, namely, our error. To summarize, we used `fitrlinear`, `k_nearest_neighbors`, neural net with `net=feedforwardnet(25)`, `fitrsvm` and `TreeBagger` as our models. At first, we used matlab fuctions to optimize hyperparameters automatically, but later we found that cross validation sometimes worked better than automatical adjust. We implemented each of the methods separately and we tried different combinations of them to ensure we reach the highest accuracy. We first thought of using Lasso but the Lasso algorithm simply took too long to run so we gave it up. Since we only used one model that is `fitrlinear` for our first and second attempt, we naturally thought of combining linear regression with another model to lower the error to hit the baseline. As it turned out, adding just one more model was not enough, so we added all of the aforementioned model to ensure our error hit the baseline. Our most innovative method was that, after we obtain the prediction from random forest, knn, neural net, linear regression and SVM, we performed random forest again on all of the predictions provided by those models at the very end, which is shown below:

```
%% All Labels
ALL = [RF_Y_est, KNN_Y_est, NN_Y_est, LR_Y_est, SVM_Y_est];
ALL_pred = [RF_Y_pred, KNN_Y_pred,NN_Y_pred, LR_Y_pred, SVM_Y_pred];
ALL_Y_est = zeros(size(XTrain, 1), 9);
ALL_Y_pred = zeros(size(XTest, 1), 9);

for n=1:9
    Mdl = TreeBagger(70, ALL, YTrain(:, n),'Method', 'regression');
    ALL_Y_est(:,n) = predict(Mdl,ALL);
    ALL_Y_pred(:,n) = predict(Mdl,ALL_pred);
end

pred_labels = ALL_Y_pred;
```

This has worked surprisingly for us as it successfully lowered the error.

(c) What did not work

i. Theme Clustering
The theme clustering algorithm based on the topics file, `topics.csv`, did not work very well for us. However, we are still convinced that our approach to can still be very accurate and effective in terms of lowering the overall error. Based on the 2000 topics given by the professor, what we are trying to achieve is that we want to cluster all those topics into different groups. We first count word occurrences from the topic list. Then we formed a frequency matrix. Words that occur frequently within a document receive a higher weighting as those words are assumed to contain more meaning in relation to the document. What we used is the `TfidfVectorizer` function from `sklearn.feature_extraction`. Then we performed K-means clustering with a pre-determined number of clusters. Each observation is assigned to a cluster (cluster assignment) so as to minimize the within cluster sum of squares. Next, the mean of the clustered observations is calculated and used as the new cluster centroid. Then, observations are reassigned to clusters and centroids recalculated in an iterative process until the algorithm reaches convergence.

The overall concept of this approach was very convincing to us and we were sure that this could help us achieve an accurate prediction. However, the change in our prediction error was somewhat insignificant. We have tried adjusting the number of clusters, different parameters in the `TfidfVectorizer` function, but we were not able to lower the error. This led us to think that in `topics.csv`, maybe the words within each category were not so differentiable that can let us cluster them into different groups. Other hypothesis is that each topic word is simply too similar for our algorithm to differentiate.

ii. `lasso`

In order to process the features, we also tried `lasso`, but the result of our implementation of `lasso` is never satisfying. What we did is that we load the entire lists of features into our `lasso` algorithm and generate a feature file, `'features.mat'`. Then we directly load the feature file into our training algorithm.

We think `lasso` suffers from the following three major problem: 1. For $n << p$ case (high dimensional case), `lasso` can at most select n features. This has to do with the nature of convex optimization problem `lasso` tries to minimize. 2. For usual case where we have correlated features which is usually the case for real word datasets, `lasso` will select only one feature from a group of correlated features. That selection also happens to be arbitrary in nature. Often one might not want this behavior. Like in gene expression the ideal gene selection method is: eliminate the trivial genes and automatically include whole groups into the model once one gene among them is selected (grouped selection). `lasso` doesn't help in grouped selection. Under many circumstances beyond the grouping issue of features, the feature selection is not consistent. For instance, compare the `lasso` result before and after you normalize(standardize) features of interest. You will likely to get different feature selection result. 3. Even for $n >> p$ case, it is seen that for correlated features , Ridge (Tikhonov Regularization) regression has better prediction power than `lasso`. Though Ridge won't help in feature selection and model interpretability is low. We think the biggest issue of `lasso` is that, as any dimensionality reduction algorithm, it might lose some relevant independent variables along the way. This mainly depends on how much penalized the system is (usually determined by the shrinkage factor lambda).

3 k-means - 0.8969

1. Training Procedure

   This is just a demo of using k-means. We used the Matlab function `[idx,C] = kmeans(X,k)` and set $k = 3$.

2. Analysis of Methods

   Since we hand tuned our parameters, training curves as well as the error change as a function of hyper-parameters are not applicable for our projects.

   For k-means, we found that it is difficult to predict the number of clusters (k), even if we have a static data set and previous domain knowledge about the data. One of the simplest methods is the so called elbow method. Using the elbow method we run k-means clustering for a range of values of k. (e.g. 1 to 150). The elbow method just gives an orientation where the optimal number of k might be, but it is a very subjective method and for some data sets it might not work. So it really was pretty time consuming to fine tune k's value. Also the k-means ten to have an uniform effect, meaning it often produce clusters with relatively uniform size even if the input data have different cluster size.

4 SVM - 0.1657

1. Training Procedure

   For SVM, we merely tuned the kernal function from the list `linear`, `gaussian`, `rbf`, and `polynomial`. We decided to use `gaussian` in the end.

2. Analysis of Methods

   Since we hand tuned our parameters, training curves as well as the error change as a function of hyper-parameters are not applicable for our projects.

   The complexity of kernel methods is therefore a function of the number of training instances, rather than the number of input dimensions. For low-dimensional problems with many training instances ($N >> p$), linear methods may yield poor predictive accuracy. Based on our tuning process, we have discovered that choosing a good kernel function is not easy and the training time can be fairly long. Also SVM makes it difficult to understand and interpret the final model, variable weights and individual impact. Since the final model is not so easy to see, we can not do small calibrations to the model hence its tough to incorporate our business logic. The nature of SVM means that we have to provide the true structure of the data as an input, while other algorithms, like neural networks or random-forests, try to automatically find the structure. Also we have to tune the parameters for the kernels and the C parameter which can be time consuming and decrease the performance. So we can either find some non-linear dataset and try to fit it without a kernel to show that it doesn't work or we can set some random values at the C parameter to show that it decreases the accuracy. Or we can find a very simple and big dataset and show that it takes very long for the SVMs to train while a simple logistic regression has the same results in less time.

5 knn - 0.1880

1. Training Procedure

   For knn, we originally used hyper-parameters provided by MATLAB to tune the variable K, the number of nearest neighbors to find in X for each point in Y, specified as the comma-separated pair consisting of 'K' and a positive integer. However, we realized that the hyper-parameter tuned by MATLAB was not significally better than hand tuning. Therefore we hand tuned that parameter and we settled it down at 3.

2. Analysis of Methods

   Since we hand tuned our parameters, training curves as well as the error change as a function of hyper-parameters are not applicable for our projects.

   We believe that reducing the value of k when running this algorithm actually increases the complexity as it has to run more smoothing. Therefore, a parameter of higher value that decreases complexity. It is quite obvious that the accuracy might increase when you increase k but the computation cost also increases. For a very low value of k (suppose $k = 1$), the model overfits on the training data, which leads to a high error rate on the validation set. On the other hand, for a high value of k, the model performs poorly on both train and validation set. Generally speaking, the optimal value for K will depend on the bias-variance tradeoff. A small value for K provides the most flexible fit, which will have low bias but high variance. This variance is due to the fact that the prediction in a given region is entirely dependent on just one observation. In contrast, larger values of K provide a smoother and less variable fit; the prediction in a region is an average of several points, and so changing one observation has a smaller effect.

6 Elastic Net - 0.1770

1. Training Procedure

   For our elastic net implementation, we tuned several parameters including λ, α, and K. We hand tuned them to the point where the error is reasonable, and the final value we picked are 1, 1, and 2.

2. Analysis of Methods

   Since we hand tuned our parameters, training curves as well as the error change as a function of hyper-parameters are not applicable for our projects.

   Generally speaking, `lasso` is preferred over ridge regression when the solution is believed to have sparse features because L1 regularization promotes sparsity while L2 regularization does not, and Elastic Net is preferred over `lasso` because it can deal with situations when the number of features is greater than the number of samples, and with correlated features, where LASSO behaves erratically. The additional L2 terms as a preconditioner or stabilizer by introducing strong convexity. When comparing elastic net to `lasso`, elastic net doesn't have the problem of selecting more than n predictors when $p >> n$, whereas `lasso` saturates when $p >> n$. When $n >> p$, and the predictors are correlated, the prediction performance of `lasso` is smaller than that of elastic net. Therefore what we did is that we used `lasso` for feature selection and used elastic net as our main process to predict the label.

7 Interpretation

Throughout this project and playing with both data and features, we did find some interesting correlations. First of all, data such as demographic, social, economic that is within the topic features, are all predictive based on health conditions. Some of the intepretations we achieve are very intuitive, based on our common sense, as we will talk about in the following paragraphs.

1. The median household income is highly correlated to the potential life lost. Factors such as food related issues, eduction level (a college degree) and mood, also have an impact on health lost. All these results made sense because people with lower incomes typically have less money to spend taking care of themselves, whether paying for visits to the doctor, medicine, or healthy food. Furthermore, stress is associated with a lower income, especially during childhood, increases risk for heart disease, stroke, cancer, and diabetes. People with higher incomes ten to live in areas with healthier resources available, like good grocery stores, safe housing, opportunities to exercise, clean air, and better schools.

2. Education also has an impact on the health lost. On a psychological perspective, people with more educationand thus higher incomesare often spared the health-harming stresses that accompany prolonged social and economic hardship. Those with less education often have fewer resources (e.g., social support, sense of control over life, and high self-esteem) to buffer the effects of stress. In a social perspective, educated people tend to have larger social networksand these connections bring access to financial, psychological, and emotional resources that may help reduce hardship and stress and improve health.

3. Mood also has a correlation to health, as can be seen from topic 72 and 66. It makes sense because a bad mood, or even worse, depression and other mental health issues can contribute to digestive disorders, trouble sleeping, lack of energy, heart disease, and other health issues. Also one's overall health can affect his or her mood. People who have good emotional health are aware of their thoughts, feelings, and behaviors. They have learned healthy ways to cope with the stress and problems that are a normal part of life. We believe that mood and health affect each other mutually. There is not a clear casuality of what causes what. This correlation can be further confirmed by the fact that average number of reported mentally unhealthy days per month is correlated to average number of reported physically unhealthy days per month. So again, we were not able to detect a clear causality, but we are certain that mental health and physically health are correlated for sure.

4. Obesity is correlated with homosexuality. At first we thought our `pca` is somewhat wrong as we cannot fathom a connection between obesity and homosexuality. After researching online, we did find a research paper stating that "lesbian women have a higher prevalence of overweight and obesity than all other female sexual orientation groups. This finding suggests that lesbians are at greater risk for morbidity and mortality linked to overweight and obesity. This finding also highlights the need for interventions within this population," which can be found at https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1874217/.

5. Obesity does not have a strong correlation with either excessive drinking or diabetes. This particular finding did suprise us due to the fact that based on common sense we think those factors should be correlated. After doing some research online, we found that it is true that obesity does not have a strong correlation with either excessive drinking or diabetes. It is said that there may be differences in alcohol metabolism between age groups; however, it is important to note that this study assessed only the amount of alcohol per drinking occasion, and not drinking frequency patterns. Furthermore, Moderation in drinking is still an important recommendation, together with a healthy lifestyle not conducive to weight gain.

8 Conclusion

Methods	Test Error	Training Error
Our own method	0.0810	0.0805
k-means	0.8969	0.7658
SVM	0.1857	0.1843
knn	0.1880	0.1895
Elastic Net	0.1770	0.1532

Obviously our own method performed the best with the error being 0.0810. What we tried to improved accuracy, what we tried but failed to help can be found in section 2. We think that after ensembling all prediction from linear regression, k-nearest-neighbors, neural net with 25 hidden layers, SVM, and random forest with 450 trees, then performing random forest with 70 trees to make our final label prediction really helped to increase our overall accuracy. Future research should be done regarding the topic list. We are still convinced that our theme clustering, if performed correctly, will undoubtedly generate the most accurate prediction. I think if we can generate the topic lists ourselves, we may be able to achieve a more accurate result.

Out of 4 model categories, the elastic net implementation performed the best while the k-means performed the worst. Originally we wanted to use `lasso` as our regularization method, but after realizing the limitations of `lasso`, which will be listed afterwards, we decided to use `lasso` as our feature selection model first. After selecting features, then we used elastic net to predict the final labels. This works significantly better than if we just used `lasso` alone. We tried to further reduce dimensionality using `pca`. However, adding `pca` did not improve or decrease our accuracy for some reason. The limitations of `lasso` is that: 1. In the $p > n$ case, the `lasso` selects at most n variables before it saturates, because of the nature of the convex optimization problem. This seems to be a limiting feature for a variable selection method. Moreover, the `lasso` is not well defined unless the bound on the L1 norm of the coefficients is smaller than a certain value. 2. If there is a group of variables among which the pairwise correlations are very high, then the `lasso` tends to select only one variable from the group and does not care which one is selected. 3. For usual n¿p situations, if there are high correlations between predictors, it has been empirically observed that the prediction performance of the `lasso` is dominated by ridge regression. We think in general, in order to improve our algorithm's accuracy, one possible solution is to fit more models, namely a series of models. In our implementation we merely used `lasso` for feature selection. We think if we are able to combine `lasso` and ridge regularization as training model, the prediction result will be more accurate.

9 References

1. http://nbviewer.jupyter.org/github/brandomr/document_cluster/blob/master/cluster_analysis.ipynb

2. https://alliance.seas.upenn.edu/~cis520/dynamic/2018/wiki/index.php