

CIS 520 Machine Learning Final Project

Yijie Zhao; Xin Gu; Manqiu Zhang
L3 NORM

December 10th 2018

1 Description of The Leaderboard Submission

– Methods And What Made The Most Difference

We tried various methods, including KNN regression, Gaussian mixture model, K means, ridge regression (we later modified this to a linear model with an elastic net regularization to remove any redundant predictors), Gaussian process regression, and support vector machine regression. We tuned the (hyper)parameters of the models mentioned above through cross validation. We originally focused on KNN regression as well as Gaussian mixture model; however, our testing error stuck at around 0.12, and tuning the (hyper)parameters did not help much.

The first major difference occurred when we switched to using a linear model with an elastic net regularization with the built-in function Lasso. We considered adding this regularization term because the number of observations far exceeds that of the features and we would therefore want to use the regularization to simplify the model to avoid any over-fitting. By tuning α , which indicates how much of a lasso or ridge the regularization is, and λ , the penalty coefficient, we were able to achieve a model that produced a testing error of 0.0821. We realized that by tuning α and λ to an optimal value (we used $\lambda = 6 * 10^{-6}$ and $\alpha = 0.65$ in the end, meaning that the regularization is more towards lasso than ridge), the cross-validation and testing error was able to be improved by a lot.

The second major difference occurred when we switched to a Gaussian process regression model. By tuning the basis function and kernel function (we also tried a custom squared exponential kernel function), we were able to achieve the best test error mentioned above (0.0733). The distribution of a Gaussian process is the joint distribution of all those (infinitely many) random variables, and as such, it is a distribution over functions with a continuous domain, e.g. time or space.¹ We believed the reason behind this improvement was because the data given fits this distribution well. Additionally, we also tried to ensemble the results from various models by averaging them; however, the accuracy was not significantly improved. We believed that this might be because the methods we tried to ensemble have correlated errors.

– Training and Testing Error

We are ranked the 12th on the leader-board with a test error of 0.0733. The training error of the model we uploaded was 0.0314.

2 Description of Dimension Reduction Methods

- For the data-set we are given, there are 1019 training samples, each represents a county, and 2021 features, 21 of which describe demographics, socio-economic status (SES), and 2000 of which are LDA Twitter topics of the people from each county. Since the number of features (p) is much larger than

¹<http://www.gaussianprocess.org/gpml/chapters/RW2.pdf>

the number of instances (n), we decide to reduce the dimension of the data, and we did so through two approaches.

- Principal Component Analysis (PCA)

We selected the number of principal components kept based on two methods:

Method 1: select the number of PCs based on the percentage of variance explained: we ran a for loop testing variance explained ranging from 70 to 99.9 percent and checked the resulting error using a linear regression model with an elastic net regularization with a 5-fold cross validation.

Method 2: try keeping 60 to 140 PCs left: we ran a for loop testing the error (same procedure as method 1).

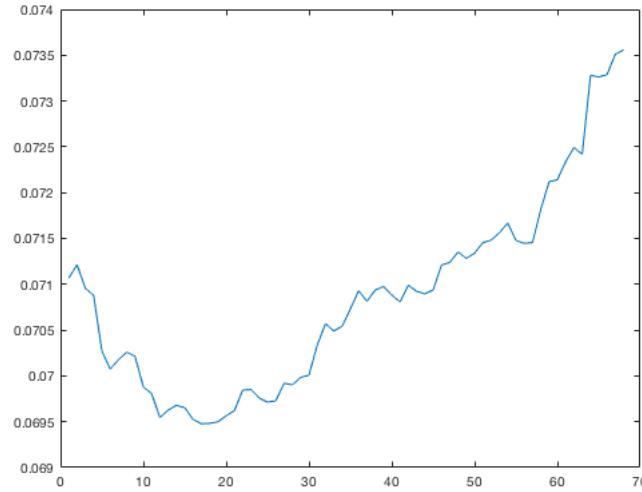


Figure 1: Training Error Curve W.R.T. PC Kept

The x-axis indicates the the number of principal components (with $x = 0$ indicates that the first 60 principal components are kept, and $x = 70$ indicates that the first 140 principal components are kept) that are kept in the model.

From the two methods above, we ultimately decided to choose 97 PCs (using method 2), which leads to the least L2 error on the held-out test set). We believed that since most of the features are highly correlated, performing a PCA would be preferred.

- Feature Selection

By looking at the description of each of the demographics as well as SES features, we realized that there are three features: small-metro, medium-metro, and large-metro that consist repetitive information. Therefore, we tried taking off one or two and see how one model (linear regression with an elastic net) performs. We saw that taking off medium-metro and small-metro leads to the best result (lowest cross-validation error), and therefore decided to take these two features off from the model.

3 Generative Method - Gaussian Mixture Model (GMM)

- Description

The model takes train inputs, train labels and test inputs as arguments and returns predicted labels of test data.

After dimension reduction, the model first uses K-means methods to divide training points into K clusters. Then we use mean values of each cluster as initial parameters in GMM model and set co-variance as a diagonal matrix, because random initial parameters may not reach optimum.

After the model is trained with reduced training data (by doing a five-fold cross validation on the training data-set, each time we trained the model on 918 labeled observations and tested it on the left-out 101 observations), we use the GMM model to get probabilities of each data points belonging to each cluster. Assign each training point to its most possible cluster. Then calculating mean values of all labels in a certain cluster and set the mean values as the label of the certain cluster.

Then we find the closest training point to each test point (using L2 norm), and assign the test point to the cluster which this training point belongs to and assign the label of the cluster to this test point. That is how labels of test points are predicted.

- Analysis of methods

The parameter changed in the model is number of clusters, which changed from 1 to 15. First we plot the distortion within clusters using K means to find out suitable number of clusters. Based on the decreasing trend, nine to thirteen clusters seems to be proper.

K	CV Error	Training Error	Test Error
1	0.1684	0.1667	N/A
2	0.1601	0.1553	N/A
3	0.1505	0.1465	N/A
4	0.1510	0.1418	N/A
5	0.1475	0.1430	N/A
6	0.1453	0.1422	N/A
7	0.1445	0.1408	N/A
8	0.1446	0.1389	N/A
9	0.1413	0.1392	N/A
10	0.1393	0.1386	Around 0.14
11	0.1401	0.1389	N/A
12	0.1406	0.1388	N/A
13	0.1422	0.1376	N/A

CV error, training error, and test error of each K

Then we perform GMM with K change from 1 to 15, calculating training and cross validation errors. Form the plot, we can see that training error goes down as K increases, while CV error also decreases as K increases, but their trends are little bit different when K goes bigger (than 13).

Based on distortion within clusters of K-means and errors of GMM, and considering of under-fitting and over-fitting (If K is too small, chances are it will be under-fitting; if K is too big, chances are it will be over-fitting), we finally choose K=10 to create the GMM model, trying to reach a balance.

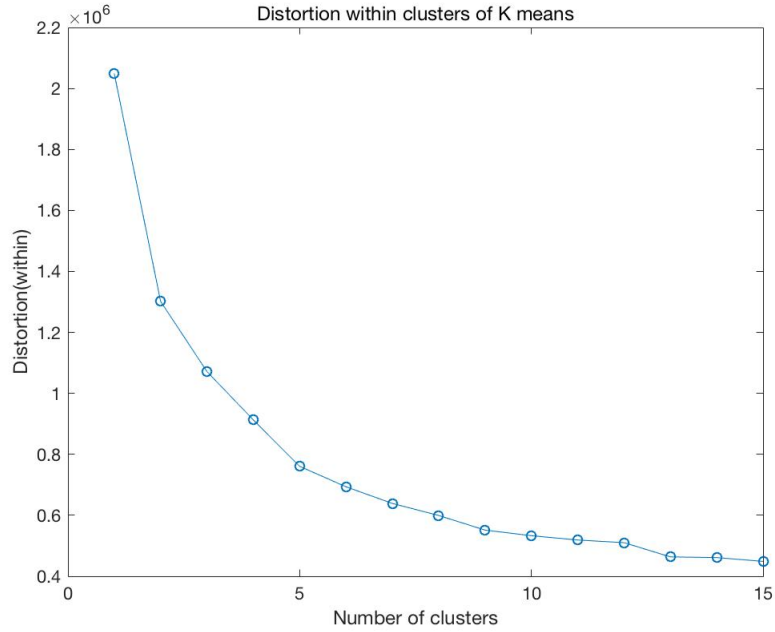


Figure 2: Distortion of KMeans W.R.T. K

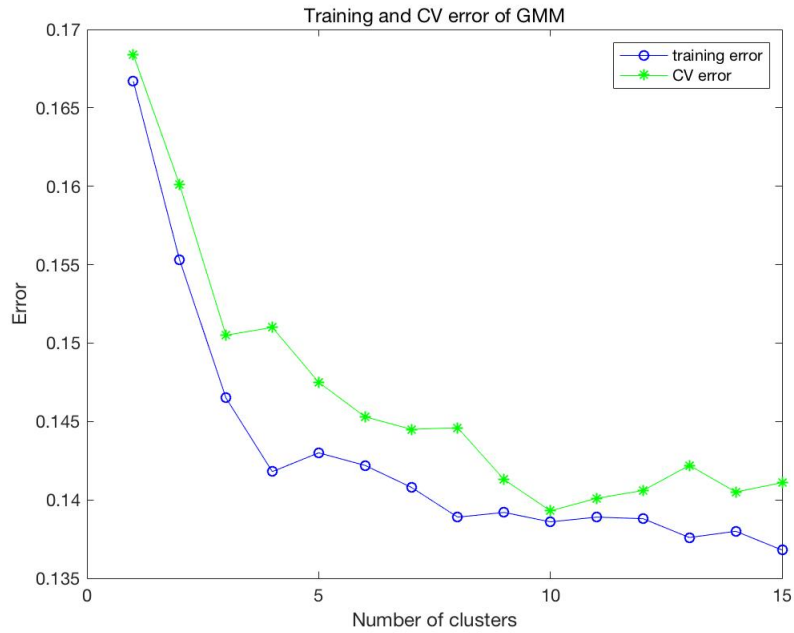


Figure 3: CV error, training error W.R.T. the number of clusters

– Interpretation

The test error is not so ideal when $K=10$, which is approximately 0.14. The reason we believe that causes such result is that GMM is more suitable for classification problems while a majority of our input features and labels are continuous instead of discrete, so it doesn't work so well. While K increases, the K means model becomes more complex, and training error drops, which is also proven

by the plot below. On the other hand, as the model becomes more and more complex, the CV error should first drop, as the model goes from an under-fit towards optimality, and then increases when the model becomes an over-fit. This trend can also be seen in the plot below.

From result of dimensional reduction, we can tell that demographic and SES features are very predictive, and a part of the topic features is also quite predictive.

The table below represents the correlation among nine labels, we can find some correlations from it. For example, Label 5 has negative correlation with all other labels; Label 2 has strong positive correlation with Label 8; Label 3 has strong positive correlation with Label 8. And all other labels has positive or negative correlation to some extent. But in this model, the correlation of labels is not taken into consideration, which is also a reason preventing better performance.

	1	2	3	4	5	6	7	8	9
1	1	0.7306	0.6914	0.7073	-0.6540	0.7424	0.7368	0.7552	0.6466
2	0.7306	1	0.7534	0.7841	-0.6962	0.6128	0.7220	0.9059	0.5365
3	0.6914	0.7534	1	0.6533	-0.6572	0.5528	0.7339	0.9073	0.4562
4	0.7073	0.7841	0.6533	1	-0.7148	0.6415	0.5595	0.7199	0.5737
5	-0.6540	-0.6962	-0.6572	-0.7148	1	-0.6054	-0.5798	-0.7176	-0.4276
6	0.7424	0.6128	0.5528	0.6415	-0.6054	1	0.6950	0.5902	0.7553
7	0.7368	0.7220	0.7339	0.5595	-0.5798	0.6950	1	0.7767	0.6484
8	0.7552	0.9059	0.9073	0.7199	-0.7176	0.5902	0.7767	1	0.5003
9	0.6466	0.5365	0.4562	0.5737	-0.4276	0.7553	0.6484	0.5003	1

Figure 4: Correlation matrix of the nine predicted health outcomes

– Conclusion

With K changing from 1 to 15, its performance first becomes better and then turns down when K gets too big. The best performance is when $K = 10$, considering both training and cross validation error. The worst model is of course when $K=1$, but it is not likely that someone will use it in a model.

I changed my K and changed initial parameters of GMM to improved my accuracy, but change of Component Proportion in GMM failed to help, which means this parameter has little effect on the outcome.

When K goes bigger, from 1 to 10, training error and testing error both decrease. When K keeps increasing from 10, training error generally keeps decreasing, while testing error begins to increase, which indicates over-fitting.

More effort on dimension reduction may be a potential opportunities for future improvements. Also, adjusting other parameters in GMM model, such as co-variance type, is also a way to pursue additional improvements. Finally, ensemble with other models may also improve performance.

4 Discriminative Method - Linear Model with Elastic Net Regularization

– Description And Analysis of Method

We use the given train inputs (1019 samples) to train our discriminative linear model with an elastic net regularization. There are two parameters we tuned upon - α (which is a positive scalar value in the interval $(0,1]$ and means the weight of lasso (L1) versus ridge (L2) optimization) and λ (which is

the regularization coefficients of the regularization).

$$\hat{\beta} \equiv \underset{\beta}{\operatorname{argmin}}(\|y - X\beta\|^2 + \lambda_2 \|\beta\|^2 + \lambda_1 \|\beta\|_1)$$

Figure 5: Formula for estimates from the elastic net method

When tuning α , we fixed λ and tried α from 0.1 to 1, with a step size of 0.1, gradually shifting towards lasso and did a 5-fold cross-validation on the training data. We found that when α equals 0.65, the model did best with the smallest CV error (As the graph below shows, the CV error is the lowest and equals to 0.0759). As the picture shows, the training error rarely changed with λ , but the CVerr (can be treated as the test error) did change along. As λ becomes smaller, the model is more like a ridge regression, which does not zero out any of the features and only shrinks all of the features. As λ becomes larger, the model is more like lasso regression, which zeros out some of the features and shrinks the others.

And that is why we found the optimal model in the middle of ridge and lasso. A complete ridge regression is not optimal most likely because of the existence of noise within the predictors, some of which might also be redundant. On the other hand, a complete lasso is also not optimal because while it zeros out the least correlated features, it shrinks the rest down by an equal amount. Additionally, if there is a group of highly correlated variables, then lasso would often only choose one of them to zero out while leaving the else untouched, which is undesirable.

We can also see that as the model transfer from ridge to lasso, there is a main trend in the reduction of the CV error. This might be due to some of the noisy/redundant features being zeroed out and allowing the model to become a better fit. The training error, as α changes, seems to not have changed by much. This, we believe, might be because both types of regularization penalties, lasso and ridge, penalize the original model and make the training error higher as compared to a linear model that is not regularized, but they penalize the model by the same extent that there would not be a clear different to the training error while tuning α and changing in between the two penalty types.

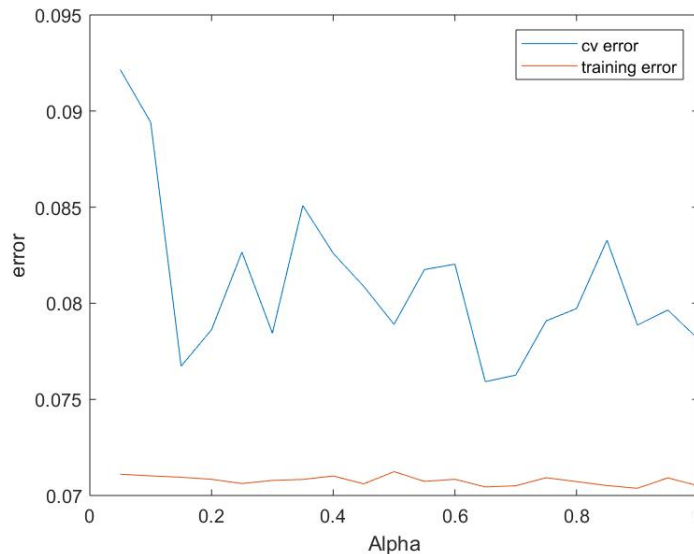


Figure 6: Training and CV error W.R.T. α

α	CV Error	Training Error	Test Error
0.05	0.0921	0.0711	N/A
0.10	0.0708	0.0775	N/A
0.15	0.0767	0.0709	N/A
0.20	0.0708	0.0776	N/A
0.25	0.0827	0.0706	N/A
0.30	0.0705	0.0774	N/A
0.35	0.0851	0.0708	N/A
0.40	0.0711	0.0776	N/A
0.45	0.0809	0.0706	N/A
0.50	0.0707	0.0776	N/A
0.55	0.0817	0.0707	N/A
0.60	0.0709	0.0776	N/A
0.65	0.0759	0.0704	N/A
0.70	0.0706	0.07725	N/A
0.75	0.0791	0.0709	N/A
0.80	0.0712	0.0780	N/A
0.85	0.0833	0.0705	N/A
0.90	0.0703	0.0772	N/A
0.95	0.0796	0.0709	N/A
1.00	0.0705	0.0774	N/A

CV error, training error, and test error of each Alpha

Then we fix the α to 0.65, and tune λ afterwards. We initially tried λ at a relatively large range from 0 to 1, and then based on the result further narrow down the testing range of λ . As can be seen below in the graph, we tried λ from 10^{-6} to 10^{-5} with a step size of 10^{-6} and achieved the lowest CV error at $\lambda = 6 * 10^{-6}$.

The regularization coefficient, λ determines how much the coefficients will be shrunk. As λ increases, the CV error, though not extremely obvious, does show a decreasing and then increasing trend, reaching a minimum at around when $\lambda = 6 * 10^{-6}$. This trend is because while initially the penalty is too small, after reaching the optimal penalty, the penalty then becomes too large and causes the CV error to raise again.

The training error overtime shows a slight increasing trend, which is due to the increasing penalty and the less fit of the model to the training data (this is what we are striving for though, since we want to introduce the regularization to avoid over-fit). It is interesting that in this exercise, the optimal λ is a very small value, and this took us a lot of tries to find.

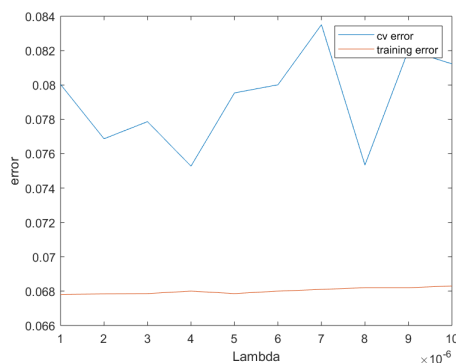


Figure 7: Training and CV error W.R.T. λ

Lambda	CV error	Training error
1.0e-06	0.080	0.0678
2.0e-06	0.077	0.0678
3.0e-06	0.078	0.0679
4.0e-06	0.075	0.0680
5.0e-06	0.080	0.0679
6.0e-06	0.080	0.0680
7.0e-06	0.084	0.0681
8.0e-06	0.075	0.0682
9.0e-06	0.082	0.0682
1.0e-05	0.081	0.0683

Figure 8: Training and CV error W.R.T. λ

9x9 double									
	1	2	3	4	5	6	7	8	9
1	1	0.8283	0.8289	0.8073	-0.8338	0.8791	0.8885	0.8809	0.8129
2	0.8283	1	0.7919	0.8280	-0.7788	0.6791	0.7475	0.9236	0.6342
3	0.8289	0.7919	1	0.7049	-0.7573	0.6536	0.8020	0.9371	0.5708
4	0.8073	0.8280	0.7049	1	-0.8147	0.7372	0.6168	0.7687	0.6934
5	-0.8338	-0.7788	-0.7573	-0.8147	1	-0.7417	-0.6914	-0.8089	-0.6119
6	0.8791	0.6791	0.6536	0.7372	-0.7417	1	0.8076	0.6855	0.9182
7	0.8885	0.7475	0.8020	0.6168	-0.6914	0.8076	1	0.8331	0.7856
8	0.8809	0.9236	0.9371	0.7687	-0.8089	0.6855	0.8331	1	0.6044
9	0.8129	0.6342	0.5708	0.6934	-0.6119	0.9182	0.7856	0.6044	1

Figure 9: Correlation matrix of the nine predicted health outcomes

5 Instance-based Method - KNN Regression

- Description

This model implements KNN regression of the instance-based category. This is a simple algorithm that stores all available cases and predicts the numerical target based on a similarity measure. There are two parts in this file.

Part one implements PCA for the purpose of dimension reduction (as explained above in the PCA section).

Part two performs KNN regression based on two feature blocks: demographics and LDA topics. This step is performed since, as shown clearly from the data, the features are split to two categories, and we believed that analyzing the two categories of the features given separately would help improve the prediction. In Part two, step one was to min-max rescale the training and testing input. The standardization of distance ensures that, despite the different measurement scales of the features, they all have about the same influence on the distance calculated. Step two calculates two distance matrices of dimension $m * n$ (m : number of testing instances; n : number of training instances), representing the euclidean distance of each of the m testing instance to each of the n training instance for the two feature blocks (demographics, LDA topics). Step three finds the closest k training instances to each of the testing instance (the number k is found by trying various k of a wide range: 50 to 500, which is about the number of features left in the model, and using the one that gives the least L2 error on the held-out test set), and calculates an inverse distance weighted average of the k nearest neighbors found above as a prediction result. The same procedure is done on all nine of the health outcome predicted variables through a for loop. By doing a five-fold cross validation on the training dataset, each time we trained the model on 918 labeled observations and tested it on the left-out 101 observations.

Training procedure

KNN Regression model consists just the labeled training data and thus does not have a traditional training step as many other machine learning models. However, there are several places in particular that are tuned for this model:

1. Distance functions

Two distance functions are tried:

a. Euclidean

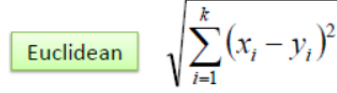
The figure shows the Euclidean distance formula. On the left, the word "Euclidean" is enclosed in a light green rectangular box. To the right of the box is the mathematical formula for Euclidean distance: the square root of the sum from i=1 to k of (x_i - y_i) squared.

Figure 10: Euclidean distance formula

b. Manhattan

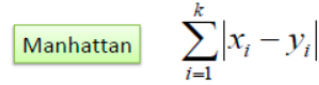
The figure shows the Manhattan distance formula. On the left, the word "Manhattan" is enclosed in a light green rectangular box. To the right of the box is the mathematical formula for Manhattan distance: the sum from i=1 to k of the absolute value of (x_i - y_i).

Figure 11: Euclidean distance formula

Euclidean distance function is chosen ultimately through cross-validation (result shown below):

Distance Method	Training Error	CV Error	Test Error
Euclidean	0.1187	0.1265	0.1287
Manhattan	0.1243	0.1289	N/A

Training, CV, and test error using the two distance methods

2. Distance Standardization Method

As explained above, the data-set is standardized, and two methods are tried:

a. Maximum-Minimum Normalization

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Figure 12: Maximum-minimum normalization formula

b. Mean Normalization

$$x' = \frac{x - \text{average}(x)}{\max(x) - \min(x)}$$

Figure 13: Mean normalization formula

Note: X is an original value, $X\hat{a}$ is the normalized value.

Maximum-Minimum Normalization is chosen ultimately through cross-validation (result shown below):

Distance Method	Training Error	CV Error	Test Error
Maximum-Minimum	0.1187	0.1265	0.1287
Mean	0.1189	0.1322	N/A

Training, CV, and test error using the two distance standardization methods

3. K in KNN (hyper-parameter)

K, the hyper-parameter indicating the number of nearest neighbors (using the euclidean distance) to each of the testing data to be considered for the prediction of the label of this testing data, is tuned through cross validation. A wide range from 40 to 140, with a step size of 20, is tried, and a 5-fold cross validation error for each K is recorded.

– Analysis of Methods

K	Training Error	CV Error	Test Error
40	0.1134	0.1498	N/A
60	0.1142	0.1456	N/A
80	0.1176	0.1298	N/A
100	0.1189	0.1265	0.1287
120	0.1201	0.1278	N/A
140	0.1262	0.1389	N/A

training, test, and CV error when tuning K in KNN Regression

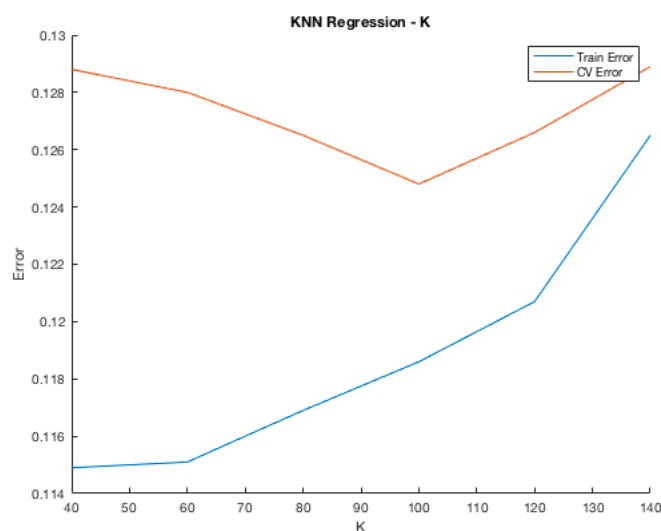


Figure 14: training and CV error when tuning K in KNN Regression

– Interpretation

Below is a matrix of the pairwise correlation of the 9 health outcome variables predicted by the KNN

	1	2	3	4	5	6	7	8	9
1	1	0.9135	0.8452	0.9499	-0.9335	0.9197	0.9263	0.9364	0.8804
2	0.9135	1	0.8505	0.9228	-0.9093	0.7377	0.8106	0.9635	0.7065
3	0.8452	0.8505	1	0.8052	-0.8668	0.6873	0.8644	0.9377	0.6544
4	0.9499	0.9228	0.8052	1	-0.9251	0.8652	0.8320	0.9081	0.8268
5	-0.9335	-0.9093	-0.8668	-0.9251	1	-0.8571	-0.8725	-0.9230	-0.8067
6	0.9197	0.7377	0.6873	0.8652	-0.8571	1	0.8797	0.7704	0.9607
7	0.9263	0.8106	0.8644	0.8320	-0.8725	0.8797	1	0.8935	0.8890
8	0.9364	0.9635	0.9377	0.9081	-0.9230	0.7704	0.8935	1	0.7353
9	0.8804	0.7065	0.6544	0.8268	-0.8067	0.9607	0.8890	0.7353	1

Figure 15: Correlation matrix of the nine predicted health outcomes

regression model described above.

As shown in the table above, all 9 of the health outcome predicted labels seem to be strong correlated either positively or negatively as the absolute value of the pairwise correlations indicated above are very close to 1. However, the model constructed did not take the correlation of the 9 health outcomes into consideration, which caused certain inaccuracy in the prediction.

From feature selection and seeing that taking out both the medium-metro and large-metro yields the best result, we believe that among all three metro-size-wise features, small-metro has the most predictive power and is the most influential one in the health outcomes.

As can be seen in the plot above, as K increases, the model becomes less complex, and thus it is justifiable that the training error goes up. At the same time, CV error first decreases, as the model goes from an over-fit towards optimum, and then increases, as the model then moves towards an under-fit. This trend is also seen in the graph above.

– Conclusion

In this model, K , the number of nearest neighbors that are taken into account when predicting the test labels, are tuned. We realized that, as mentioned above, as K increases, the error first decreases and when increases with the minimum occurred at around when $K = 100$. Originally we did not standardize the distance, which resulted in a high error. To improve the accuracy, we used max-min normalization to re-scale the data-set (reason mentioned above), and thus resulted in a higher accuracy for this model. We tried splitting the model to more than two feature blocks by further splitting the demographics SES features, performing a KNN regression on each one of the feature blocks and average the results in the end. However, this further splitting did not do much of a help. The error curves above have a parabola shape since a KNN model is over-fitting when k is small and under-fitting when k is large. In the future, instead of simply averaging the results gotten from the two feature blocks (demographics LDA topics), we can try more advanced ways to ensemble the two sets of results (e.g. based on the inverse of the variance) and see if this could bring up the accuracy by even more. A potential future research area might be to further narrow down on the LDA topics and see which subsets of the 2000 topics are better at predicting the health outcomes of people in each county.

6 Innovative Method - Gaussian Process Regression with Custom Kernel Function

– Description

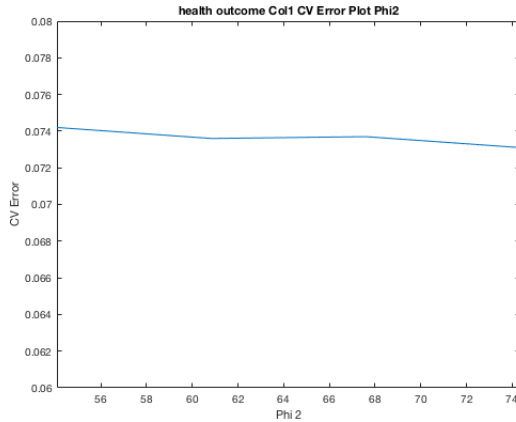
This model implements a custom kernel of the innovative category. There are two parts in this file.

Part one implements PCA for the purpose of dimension reduction (explained above).

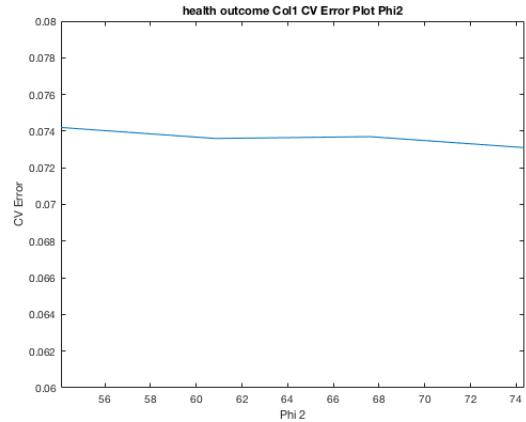
In part two, we have used the regression learner, a build-in application in Matlab, to have a general picture of which types of regression to use on the data. We saw that among all nine health outcomes, exponential Gaussian process regression consistently gives the lowest RMSE among all regression models, such as linear, fine tree, linear SVM, boosted tree, bagged tree. A Gaussian process regression (GPR) models are non-parametric kernel-based probabilistic models. In order to capture the non-linear pattern in the data, we build a custom kernel function to make linear models work in non-linear settings by mapping (changing the feature representation) data to higher dimensions where it exhibits linear patterns and thus applying the linear model in the new input space. The kernel function we have is a custom squared exponential kernel function mentioned below.

We tuned the basis function ("constant", "none", "linear", "pure-Quadratic"), the kernel function ("exponential", "matern32", "matern52", and a custom squared exponential kernel function) as well as the kernel parameter phi, a 2-by-1 vector for the kernel functions we tried upon and whose first element represents the lengths scale and the second component requirements the signal standard deviation. These are some of the most important name-value pairs of arguments suggested by Matlab, which is the reason we decided to tune these arguments to see if it would yield a significant change to our result.

We tuned the first element of ϕ around $\text{mean}(\text{std}(\text{train inputs}))$ and tuned the second element around $\text{std}(y)/\sqrt{2}$, where y represents each of the health outcomes. These are the default inputs for ϕ suggested by Matlab and is the reason we decided to tune these hyperparameters around these two values to begin with. However, there did not seem to be a significant change in the resulting accuracy by changing the ϕ_1 and ϕ_2 . For example, as could be seen from the two plots below that show the errors of just the predicted first health outcome label when tuning ϕ_1 and ϕ_2 values respectively, the error does not seem to be changing by a significant amount. Therefore, we decided to stick to the default initial theta values (as calculated by the formulas mentioned above in the paragraph) for each one of the nine health outcomes.



(a) CV Error W.R.T. ϕ_1



(b) CV Error W.R.T. ϕ_2

Figure 16: CV Error W.R.T. ϕ

By doing a five-fold cross validation on the training data-set, each time we trained the model on 918 labeled observations and tested it on the left-out 101 observations.

The custom squared exponential kernel function we implemented in the Gaussian process regression

model:

$$\text{kfcn} = @ (XN, XM, \phi) (\exp(\phi_2)^2) * \exp(-(pdist2(XN, XM).^2)/(1.9 * \exp(\phi_1)^2));$$

Initial θ of each of the nine health outcomes:

Health Outcome	ϕ_1	ϕ_2
1	1.0954	67.5964
2	1.0954	3.2977
3	1.0954	0.3956
4	1.0954	0.4995
5	1.0954	2.3685
6	1.0954	4.0430
7	1.0954	2.4861
8	1.0954	0.4833
9	1.0954	3.2455

Initial θ of each of the nine health outcomes

As described in details above, tuning the kernel parameter did not seem to be affecting the result by much, therefore, for this model, we only tuned upon the basis and kernel functions, of which the cross-validation and training errors are listed below.

When tuning the basis function:

Basis Function	Training error	Testing error
"linear"	0.0620	0.0838
"none"	0.0634	0.0895
"constant"	0.0515	0.0738
"pure-Quadratic"	0.0554	0.1178

CV error W.R.T various basis functions

We can see from the result that when we set Basis Function as linear and pure Quadratic , it obviously causes over-fitting. And between constant and none, constant has lower training and testing error.

When tuning the kernel function:

Kernel Function	Training error	Testing error
"exponential"	0.0621	0.0756
"matern32"	0.0601	0.0701
"matern52"	0.0520	0.0788
"kfcn"	0.0721	0.0764

CV error W.R.T various kernel functions

From the result, we can tell that exponential causes over-fitting, and matern32 has the best perform.

As a result, we chose the basis function to be constant and the kernel function to be matern32 for our final model.

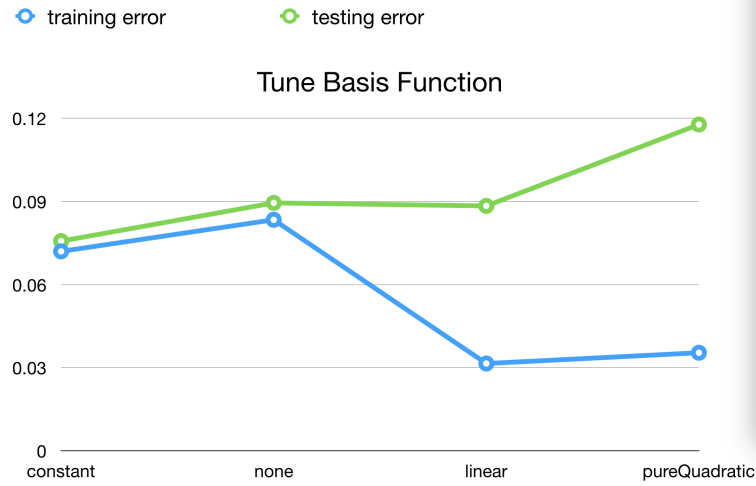


Figure 17: CV error W.R.T various basis functions

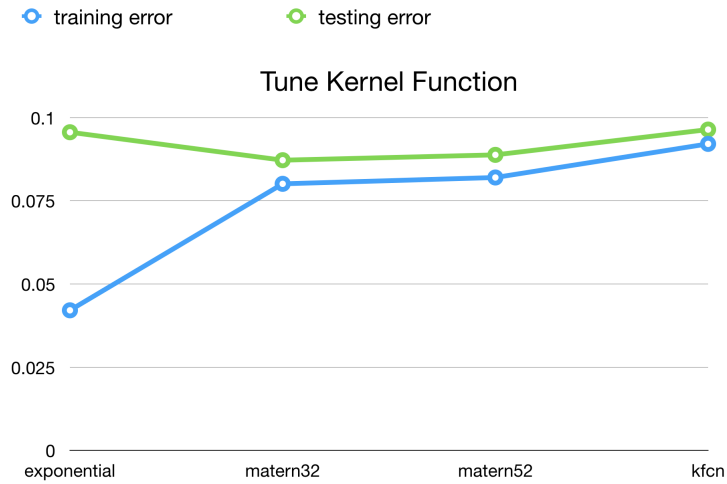


Figure 18: CV error W.R.T various kernel functions

- Interpretation

All labels are correlated to some extent. Label 5 has negative correlation with all other labels; Label 2 has strong positive correlation with Label 8; Label 3 has strong positive correlation with Label 8. There are two groups of features, regarding of the number of feature in both groups, we chose to perform PCA on topic features. It turns out that the demographics/SES features and the selected principal components of topics features and have strong predictive power.

- Conclusion

We tuned the basis function ("constant", "none", "linear", "pure-Quadratic"), the kernel function ("exponential", "matern32", "matern52", and a custom squared exponential kernel function) as well as the kernel parameter ϕ , a 2-by-1 vector for the kernel functions. It turns out that Kernel parameter ϕ has little effect on the outcome, and when Basis Function is constant and Kernel Function is matern32, the model has the best performance. When we set Basis Function as linear and pure Quadratic, it causes over-fitting, so the training error is low while testing error is high. Tuning other parameters of

the model may lead to better performance, and more effort on dimension reduction and ensemble with other models may also yield better outcome.

7 Conclusion

Method	Training error	CV error	Test Error
GMM	0.1378	0.1392	0.1443
Linear with Elastic Net Regularization	0.0710	0.0759	0.0791
KNN Regression	0.1187	1265	0.1287
Gaussian Process Regression with Custom Kernel Function	0.0515	0.0735	0.0733

Training, CV, and test error of each method mentioned above

As can be seen above, the Gaussian process regression model works the best as it has the lowest CV error as well as test error among all four methods.

The GPR model performed the best, while the KNN regression model and the GMM model both performed relatively mediocre.

We believe that the GPR model performed the best since the distribution of the data given fits better that of a GPR model. Among all the GPR models we tried, the best results occur when we used an exponential kernel function. This suggests a hint of non-linearity in the given data-set and suggests the need of a kernel function introduced in the model to capture this non-linearity. We tried to improve the accuracy by tuning the kernel parameters, changing the basis as well as kernel functions. We realized that tuning the kernel parameters (theta as mentioned above) failed to change the result significantly, and we believed that this might be because we have only tried on a very narrow range, and the result could have been different if we were to try on a wider range, and this is something we can improve upon in the future.

The KNN Regression model is a very simple model, and it is not surprising that it does not perform as well since KNN regression is most preferred and powerful when the data-set is small, clean, and with a not-very-wide range, which our data-set does not agree with as well and causes this model to not work as well as some of the other models we tried. As mentioned above, we tried splitting the features to more than 2 feature blocks by further splitting the demographics and SES features. However, this did not do much of a help. The reason behind this might be that there are only 21 demographics and SES features given and this small a number would not fit if they were to be split further down to more feature blocks. The error curves above have a parabola shape since a KNN model is over-fitting when k is small and under-fitting when k is large.

GMM model also doesnât work as well since we believe GMM might work better for classification than regression, while we are given features that are mostly continuous in this data-set. To improve the accuracy, we performed K means first on the data-set to get the initial K centroids as a preparation for GMM. We also tuned K, the number of clusters, as well as the co-variance matrices, seeking for an improvement in the accuracy. We realized that while tuning K yields a significant change in the resulting accuracy, tuning the co-variances matrices did not do much of a help. We tried K from 1 to 15 and noticed yet again a parabola-shaped error curve for both the training and CV error (the training error curve is below the CV error curve since the model would be better at predicting the data it was trained on as compared to the held-out test set). The GMM model is most likely under-fitting while K is small (too few clusters) and and over-fitting when K is large (too many clusters), and thus itâs justifiable that the optimal K (K = 10) occurs in the middle.

For the linear regression with elastic net, we tried to tune α (how much of a Lasso or a Ridge the model is) and λ , the penalty coefficient, seeking for an improvement in the accuracy. We realized that with a regularization more towards lasso (L1 with more features zeroed out as compared to ridge) works well. As compared to a ridge regression, this model we have achieved would zero out more features. This suggests the existence of noise in the reduced training inputs, which would explain why the model is better close to lasso

than ridge. We also tried performing a elastic net model on two feature blocks (one demographics and one LDA topics); however, the resulting accuracy was not affected significantly. We believed that this might be due to the small number of demographics and SES features we are given, which caused the elastic net model ran on just these features to be incompetent. Additionally, it might be because that the original elastic net model has already successfully captured all the predictive power of both feature sets.

We determined the importance of the features based on the elements of the first principal component (i.e. the biggest loading with the most effect on the prediction of y).² We saw that among the demographics/SES features, the most important features are:

1. ses pcrural (percentage of the county which is considered rural)
2. demo pcwht (percent white population)
3. demo 65over (percent population over 65 years)
4. nchs 2013 (counties classified as large urban, suburban and so on, based on population statistics)
5. ses pcunemp (percentage unemployed)

The least important demographics/SES features:

1. ses income ratio (ratio of household income at the 80th percentile to that at the 20th percentile)
2. ses log hhinc (median household income (logged))
3. smallmetro (counties with 50k - 250k residents)
4. ses foodenvt (access to a healthy food environment (e.g. ability to buy vegetables nearby))
5. micro(central and non-core counties within metropolitan areas)

This result is expected since in rural area, people often find it hard to get a highly-paid job. The most common jobs they can find are labour-focused and are often hurting their bodies. Also it's hard to find organic food markets in rural area. People can't eat healthy and often live in poor conditions, which can explain why they tend to have more health problems. On the contrary, in big cities with more people, there are more resources like organic food markets, gym, good hospital, and even good personal health consultant, which help people build a healthy life. Besides, white people are more likely to be wealthy and well-educated, which will help them build a healthy and balanced life style and eventually lead to less health problems. And it is very obvious that old people who are over 65 are more likely to have health problems since as people are aging, they often become weaker and body often doesn't function well. As for unemployed people, they are lack of income, so they can only survive on poor-quality food, some of them are even homeless because they cannot afford the rent. And that's why they are much more likely to be involved in health problems.

Among the LDA Twitter topics, many seem to be negative (pessimistic as could be seen from the many curse words), and some seem to be vaguely relevant to racism or even sexism). Some seem to be dominant of negative cursing words, while others are short hands or idioms that are frequently used online, such as dat and ppl. It is potentially possible that using social media to express negative feelings have a correlation with the health outcome of people. There might exist latent variables such as anger that potentially has an effect on both people's health and their inclination of expressing negativity online, and this might be the reason why these topics are ranked the most important among all LDA topics. a couple of most important several topics' visual representations can be seen below:

²<https://onlinecourses.science.psu.edu/stat505/node/54/>



Overall, looking at both the demographics/SES and LDA Twitter topic features together, among the top 10 most important several features, 6 are demographics/SES features (e.g. ses pcrural, demo pcwhit, demo 65over, etc.) and only 4 are of the Twitter topics about, for example, curse words, religion, and health. This suggests the possibility that demographics/SES features have a stronger predictive power to health outcomes of the residents in each county as compared to the Twitter topics.

17