CIS 520 - Project Report

Section 1: Data Processing

1.1 Skew Correction

The distribution of the labels is skewed. We know that regression works better on normal distribution as regression models the mean. In a skewed distribution, the mean is not a good indicator or measure of central tendency, so in order to perform regression, we skew corrected it.

The table below shows the original skewness of each label and the skewness after calling the function transformLabels().

Outcome Feature	Original Skew	Skew of transformed labels
health_aamort	0.4557	0
health_fairpoor	0.6989	0.0625
health_mentunh	0.0336	0.0336
Health_pcdiab	0.4615	0.0513
Health_pcexcdrin	0.0761	0.0761
Health_pcinact	-0.0886	-0.0886
health_pcsmoker	0.3072	-0.1104
health_physunh	0.3191	0.0348
Heath_pcobese	-0.4445	0.0563

After we predict the labels for the test set, we reversed the skew correction.

1.2 Dimensionality Reduction:

We performed Principal Component Analysis on our 2000 Twitter features as we wanted to capture the variation in our data whilst removing any possible correlation. The table below shows the percentage of the total variance explained by selecting certain numbers of PCA components.

Number of PCA Components	**Percentage of Variance Explained**
30	89.2919
40	90.7310
80	93.7676
90	94.2255

As we can see, out of 2000 Twitter features, 90 principal components are enough to explain 94.2255% of the variance in the data.

Section 2: Leaderboard Method

We used the same method described in our generative model for our leaderboard submission.

Section 3: Other ML Techniques

3.1 Generative Model: Gaussian Process Regression

This is the model that we submitted to the leaderboard.

We used the inbuilt MATLAB function fitrgp() for implementing Gaussian Process Regression.

```
fitrgp(X,Y,'KernelFunction','rationalquadratic', 'Standardize',1, 'Sigma',
obj.sigmaParams{index}, 'BasisFunction', 'none', 'FitMethod', 'sr')
```

We tried ‘exponential’, ‘squaredexponential’, ‘matern32’, ‘matern52’, and ‘rationalquadratic’ kernel functions.

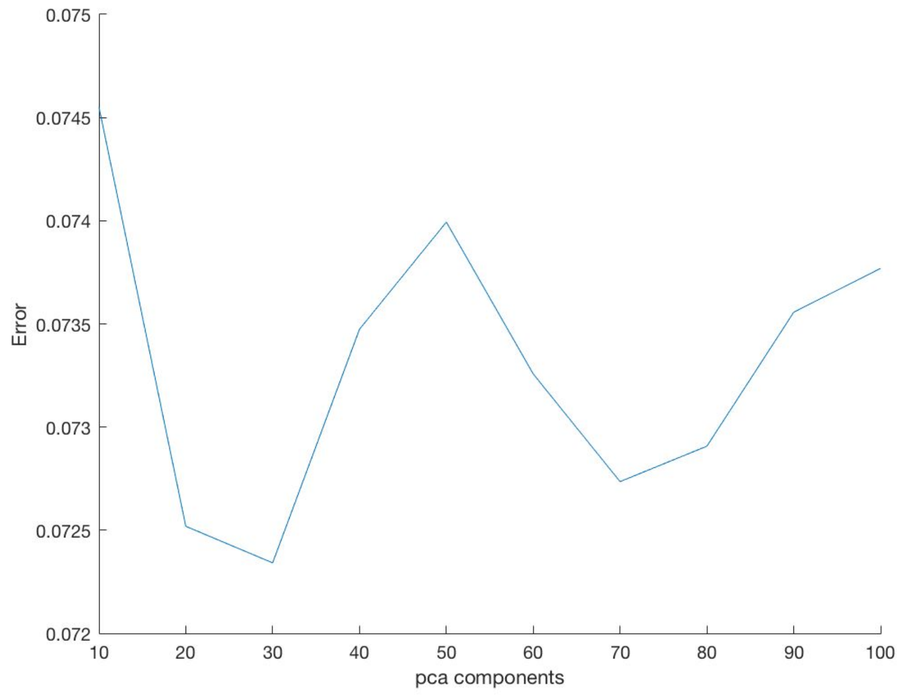
Kernels	**Training Error**	**Test Error**
exponential	0.0025	0.0733
squaredexponential	0.0430	0.0739
matern32	0.0327	0.0731
matern52	0.0385	0.0731
rationalquadratic	0.0382	0.729

As we can observe from the above table, “rationalquadratic” gives a good tradeoff between the training and test error, so we chose that as our kernel function.

For regularization, we passed the FitMethod as “sr” (takes the standard deviation of the labels) and we tried using exact but that was overfitting the data.

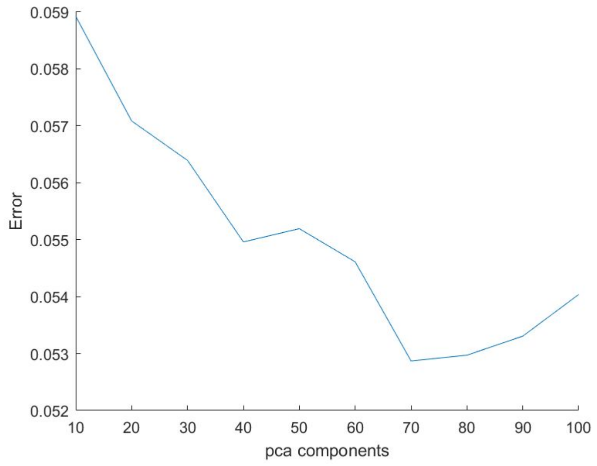
To improve the accuracy of our model, we analyzed the PCA component vs Error graph for every outcome label and chose the number of principal components that gave us the least error. Below are the plots for the same.

LABEL 1 :



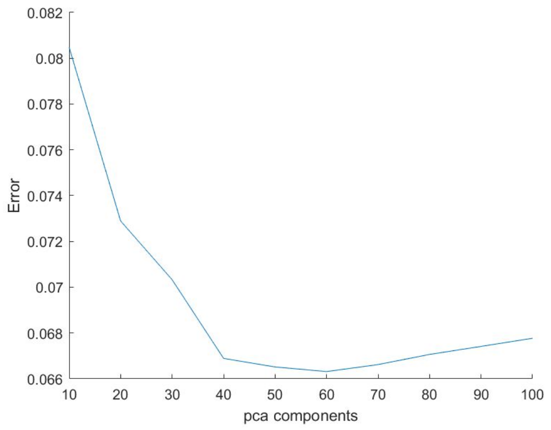
As we can see from the above plot, we get the minimum error for 30 PCA components. Hence, number of PCA components chosen for Label 1 = 30.

LABEL 2 :



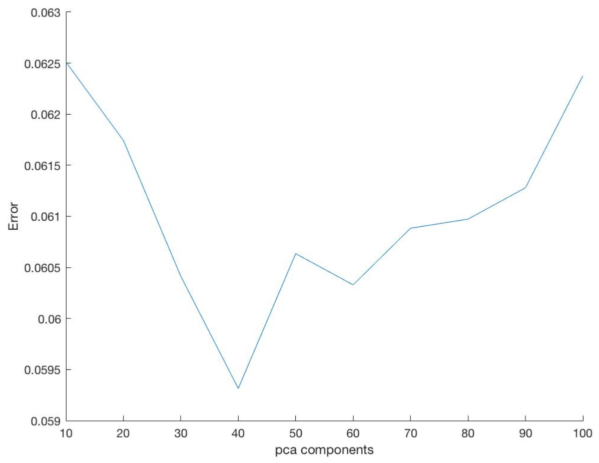
As we can see from the above plot, we get the minimum error for 70 PCA components. Hence, number of PCA components chosen for label 2 = 70.

LABEL 3 :



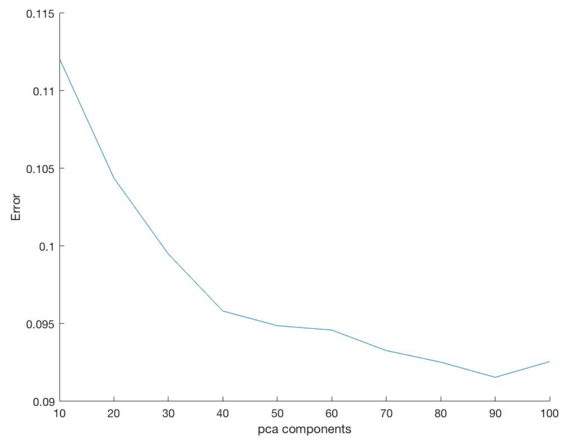
As we can see from the above plot, we get the minimum error for 60 PCA components. Hence, number of PCA components chosen for label 3 = 60.

LABEL 4 :



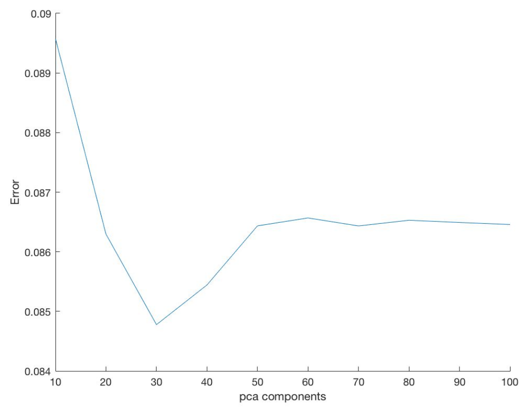
As we can see from the above plot, we get the minimum error for 40 PCA components. Hence, number of PCA components chosen for label 4 = 40.

LABEL 5 :



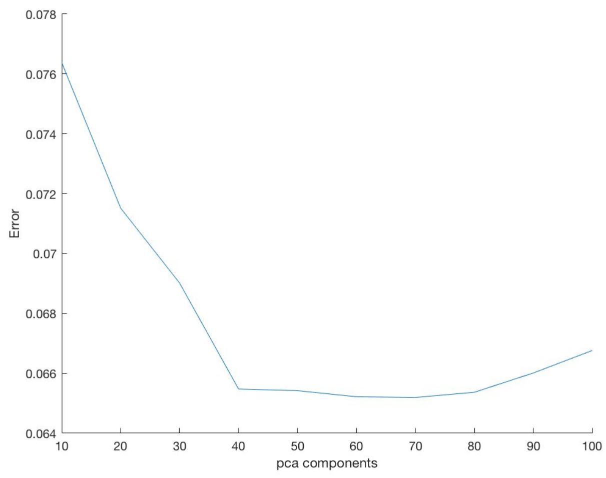
As we can see from the above plot, we get the minimum error for 90 PCA components. Hence, number of PCA components chosen for label 5= 90.

LABEL 6 :



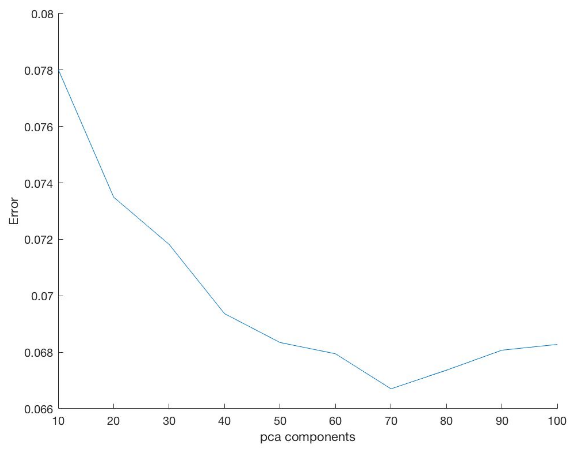
As we can see from the above plot, we get the minimum error for 30 PCA components. Hence, number of PCA components chosen for label 6 = 30.

LABEL 7 :



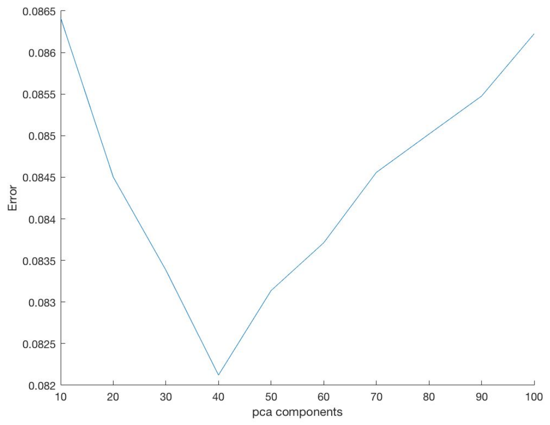
As we can see from the above plot, we get the minimum error for 40 PCA components. Hence, number of PCA components chosen for label 7 = 40.

LABEL 8 :



As we can see from the above plot, we get the minimum error for 70 PCA components. Hence, number of PCA components chosen for label 8 = 70.

LABEL 9 :



As we can see from the above plot, we get the minimum error for 40 PCA components. Hence, number of PCA components chosen for label 9 = 40.

3.2 Discriminative Model: ElasticNet Regression

3.2.1 Model

We created a class implementing the ElasticNet model with the following parameter values:

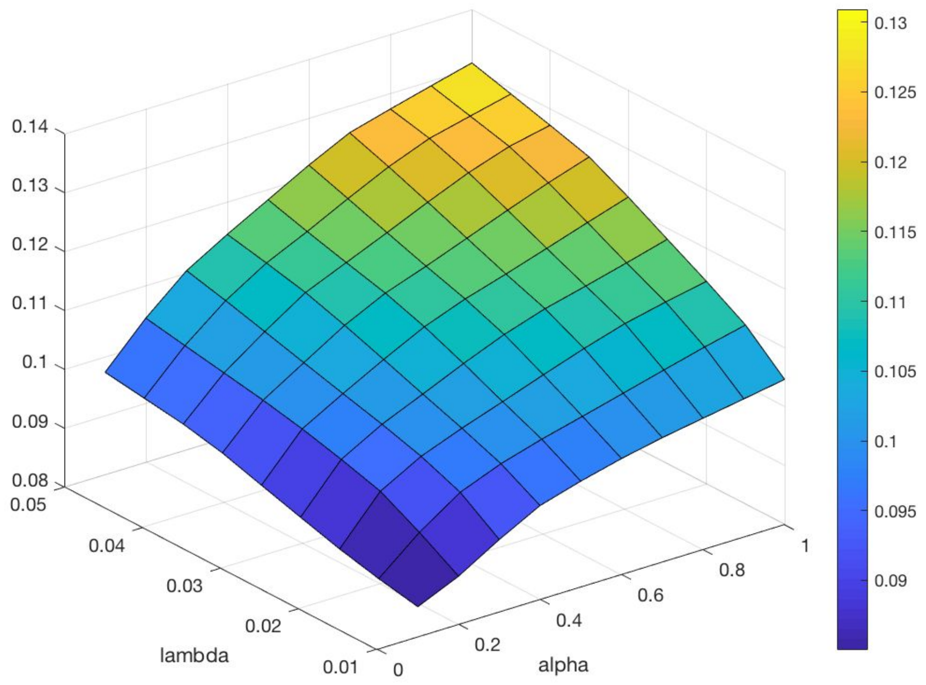
- α = 1.0
- λ for each outcome:
  - health_aamort = 0.0085
  - health_fairpoor = 0.0028
  - health_mentunh = 6.8172e-05
  - Health_pcdiab = 9.4965e-4
  - Health_pcexcdrin = 3.053e-04
  - Health_pcinact = 0.0012
  - health_pcsmoker = 0.0026
  - health_physunh = 0.0014
  - Heath_pcobese = 5.4241e-04

where α is the ratio of weight of lasso regularization vs weight of ridge regularization.

If we try to optimize same alpha/lambda for all the labels,

Surface Plot

for Cross Validation Error:

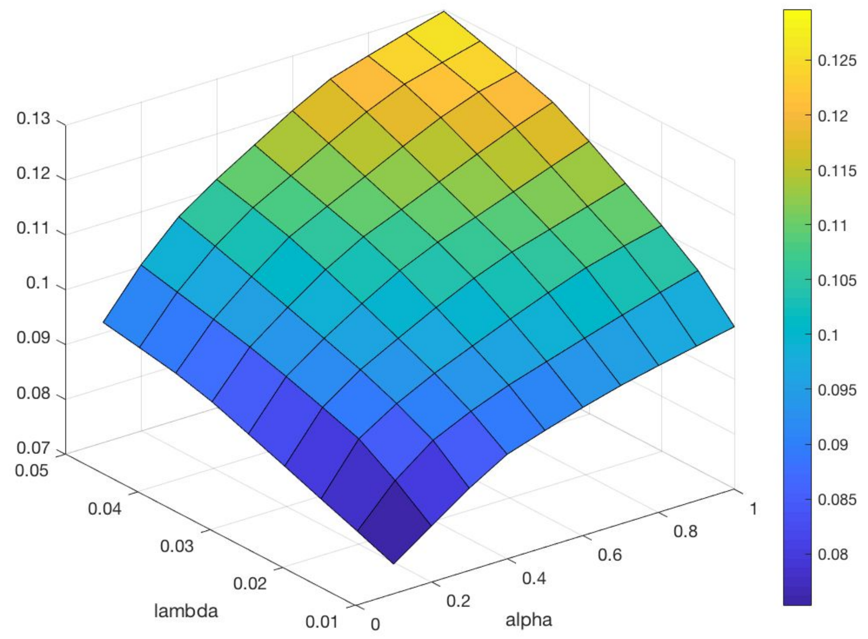


Cross Validation Errors :

Alpha\Lambda	0.01	0.015	0.02	0.025	0.03	0.035	0.04	0.045	0.05
0.1	0.08508179614	0.08640405856	0.0880069137	0.08995289962	0.09195539853	0.09399922298	0.09546690855	0.09642912038	0.09738080436
0.2	0.0881503512	0.09215657685	0.09570034999	0.09767355518	0.09941409353	0.1008454686	0.1020458972	0.1032403966	0.1044657499
0.3	0.09226223227	0.09684986441	0.09951916615	0.1015243262	0.1032901482	0.1051031835	0.1069433602	0.1087604017	0.1104191769
0.4	0.09592264261	0.0995919611	0.1021759103	0.1045378866	0.1069760507	0.1093330851	0.1114705623	0.1130446431	0.1144721466
0.5	0.09795297238	0.1016387204	0.1045734667	0.107615202	0.1104215179	0.1126956378	0.1144422434	0.116274902	0.1182337574
0.6	0.099683	0.103411	0.107028	0.110436	0.113033	0.115151	0.117413	0.119749	0.122001

	68887	6628	0139	5513	3548	4452	7528	0571	6477
0.7	0.1011183559	0.1052246354	0.1093861624	0.1127079006	0.115149238	0.1177874925	0.120475555	0.1230421194	0.1253845647
0.8	0.1023084786	0.1070637357	0.1115033638	0.1144400264	0.1173933585	0.1204474314	0.1233634129	0.1257527089	0.1273035479
0.9	0.1034783141	0.1088603349	0.1130571346	0.1162568076	0.1196739133	0.1229790703	0.1257080726	0.127430708	0.129130739
1.0	0.1046801742	0.1104807134	0.1144564816	0.1181574123	0.1218825604	0.1252538003	0.1272009037	0.1290728832	0.1309096888

Surface Plot for Training Error:



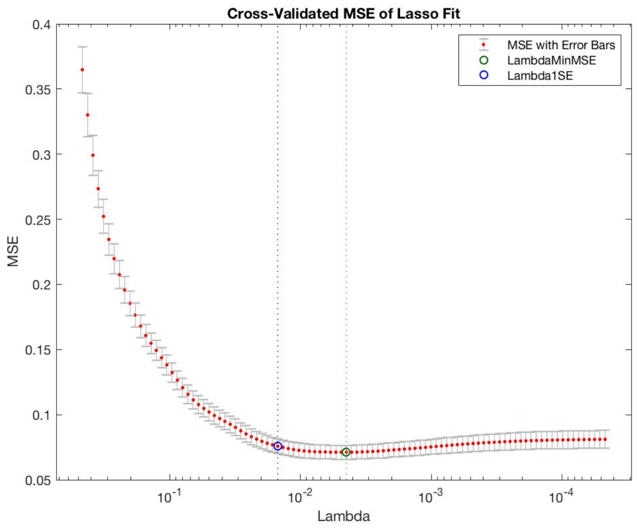
Alpha\Lambda	0.01	0.015	0.02	0.025	0.03	0.035	0.04	0.045	0.05
0.1	0.07534265156	0.07790707241	0.0803544146	0.08288570178	0.08533568463	0.08774425991	0.08953762724	0.09075829905	0.09191342568
0.2	0.08016006482	0.08505107536	0.08919831957	0.09150728504	0.09367918961	0.09563296823	0.09724406553	0.09875770409	0.1002723768
0.3	0.08495881994	0.09025930432	0.09352087925	0.09625919532	0.09851872303	0.1007459644	0.1029240595	0.1050323405	0.1069732313

0.4	0.08903176374	0.09344160444	0.09692855964	0.09987310235	0.1027687638	0.1055164562	0.1080380755	0.109954005	0.1116556275
0.5	0.09126503126	0.09609974951	0.09979378189	0.1033787129	0.1066652607	0.1094055268	0.1115053476	0.1136385311	0.1158306141
0.6	0.09336283332	0.09827706148	0.1026122586	0.1065888937	0.1097268743	0.112255538	0.1148338407	0.1174513038	0.1199577087
0.7	0.09523547761	0.1004251732	0.1052810371	0.1092538222	0.1121821732	0.1151898095	0.1182045651	0.1210565972	0.1236179955
0.8	0.09676722908	0.1025339482	0.1077340053	0.111282076	0.1146915935	0.1181254263	0.1213681472	0.1240007869	0.1257383199
0.9	0.09819511951	0.1045605036	0.1095763964	0.1133534455	0.1172117895	0.1209090645	0.1239204545	0.1258544597	0.127725824
1.0	0.09963017321	0.1064368628	0.1112080125	0.1154622467	0.1196585242	0.1233786853	0.1255746742	0.1276455678	0.1296319316

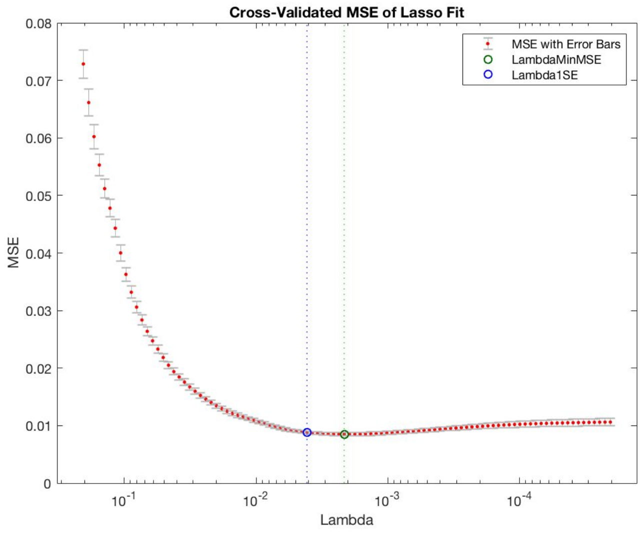
Note: However, for each alpha, optimization leads to unique values of lambda for each outcome label. In such cases, working with to each feature, and optimizing lambda for each alpha, resulted same cross validation error for all alpha's. We pick alpha as 1.0 as we prefer sparse vector

Optimizing specific to each labels,

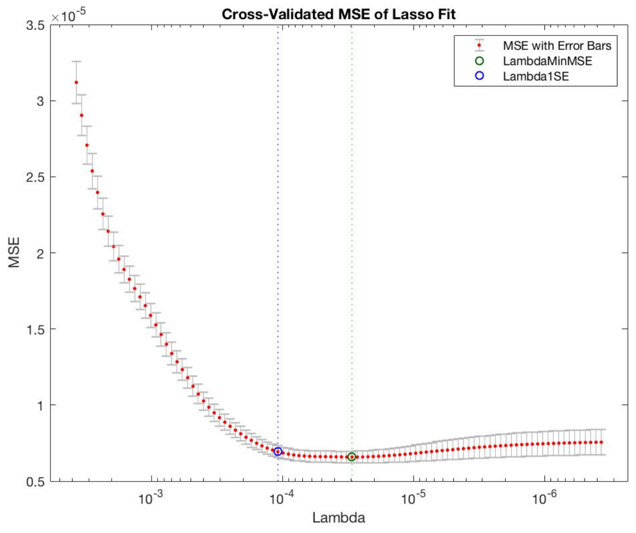
LABEL 1 :



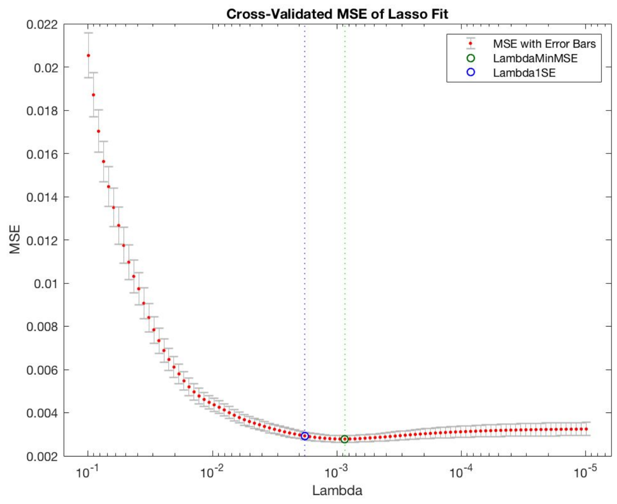
LABEL 2 :



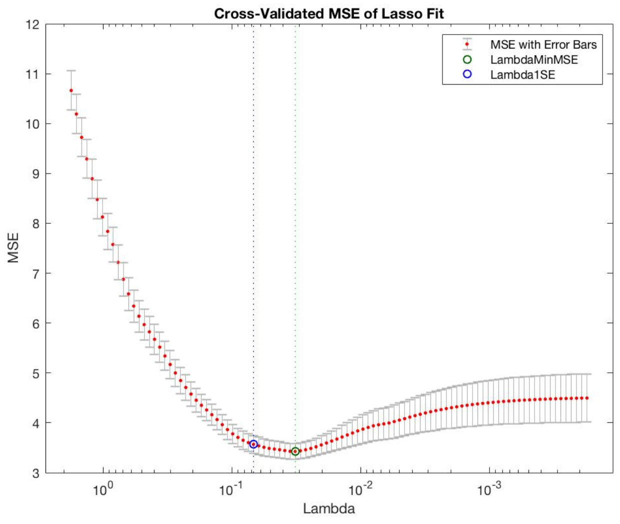
LABEL 3 :



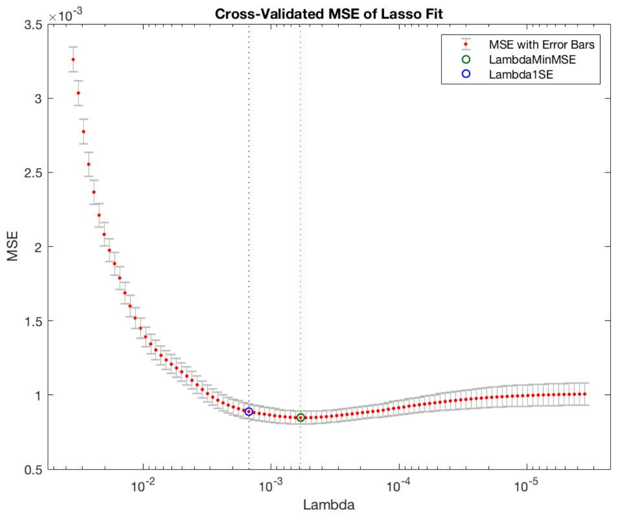
LABEL 4 :



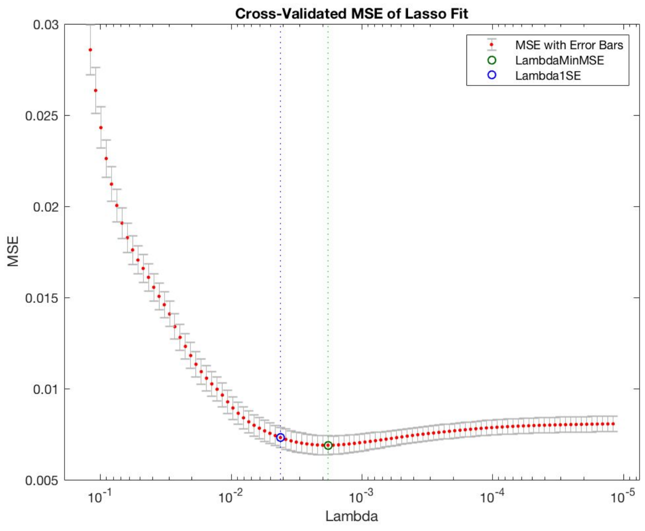
LABEL 5 :



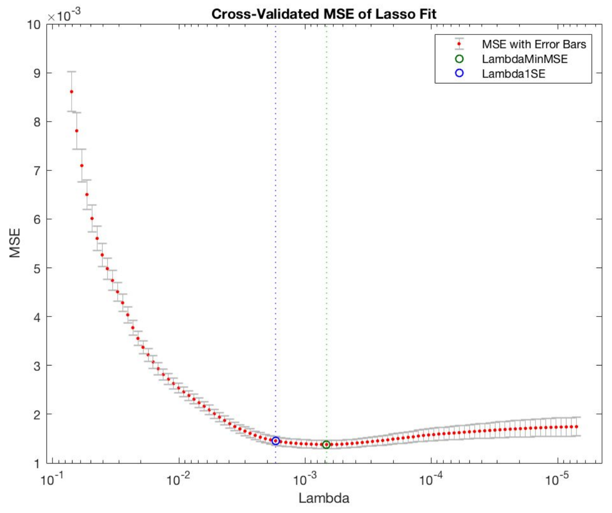
LABEL 6 :



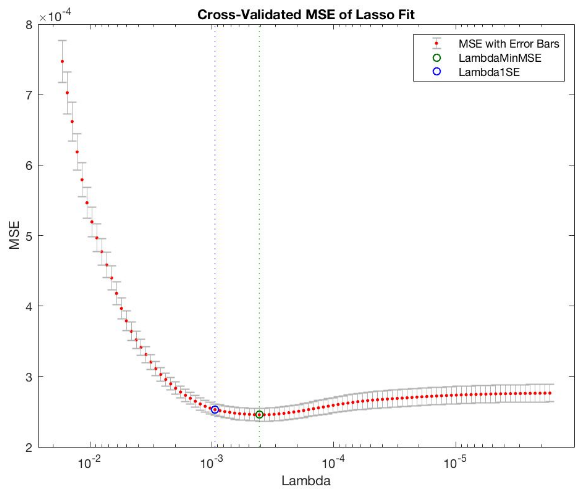
LABEL 7 :



LABEL 8 :



LABEL 9 :



In order to actually create the ElasticNet Regression model with the parameters specified, we called the inbuilt MATLAB function lasso() and passed our data and parameters for each outcome feature.

```
lasso(X, Y, 'Alpha', obj.alpha,'Lambda', lambda)
```

3.2.2 Development

To develop a discriminative model for our data, we first tried using ridge regression, as we had a large number of features and wanted to prevent overfitting. However, the least value of cross validation error after performing 10-fold cross validation using Ridge regression was 0.0808. Clearly, simply adding a penalty was not good enough for the model.

However, as we have already performed PCA, the features that we are using will not be correlated, so instead of performing Lasso regression, we decided to try the ElasticNet regression model which led to a mean cross validation error of 0.0799 after performing 10-fold cross validation.

3.2.3 Results

Mean Cross Validation Error = 0.0811 (Optimizing for each label)
Train Error = 0.0702

3.3 Instance Based Model: K-Nearest Neighbors

3.3.1 Model

We have created two versions of KNN based on how we developed and tuned the models.

3.3.1.1 Model 1:

Outcome Feature	**K**	**Distance Metric**
health_aamort	187	Correlation
health_fairpoor	55	Euclidean
health_mentunh	58	Seuclidean
Health_pcdiab	6	Seuclidean
Health_pcexcdrin	6	Correlation
Health_pcinact	40	Euclidean
health_pcsmoker	335	Euclidean
health_physunh	1	Seuclidean
Heath_pcobese	95	Euclidean

3.3.1.2 Model 2:

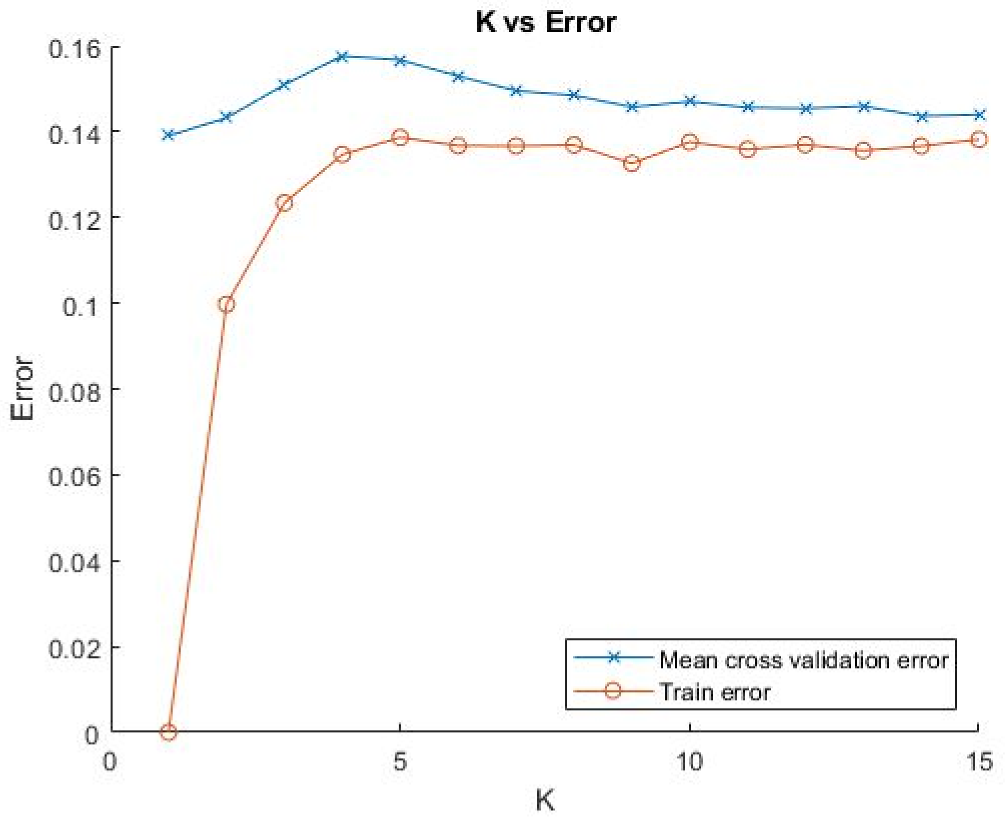
K-Nearest Neighbors has hyperparameters K = 1, and Distance Metric = Euclidean.

In order to create both these models, we used the inbuilt MATLAB function fitcknn(), where we passed these hyperparameters.

3.3.2 Development

To develop Model 1, we used the “OptimizeHyperParameters” option with the value “auto” which means that we only optimize K and the distance metric. This performed 5-fold cross validation on the training data and found the hyperparameters that minimized loss. As we used more than 10 feature columns, the "exhaustive" search method was used. This means that for every point, we find the distance from every other point to identify the K-Nearest Neighbors. The other option is to use the "kdtree" option in which the algorithm creates a tree like structure for storing and finding the k-nearest neighbors of any new point.

To develop Model 2 on the other hand, we manually found the lowest cross validation and train set errors for values from K=1 to K=15. We performed 10-Fold Cross validation in each case. The lowest cross validation error was for K=1, so our model performs 1-Nearest Neighbor classification.



3.3.3 Results:

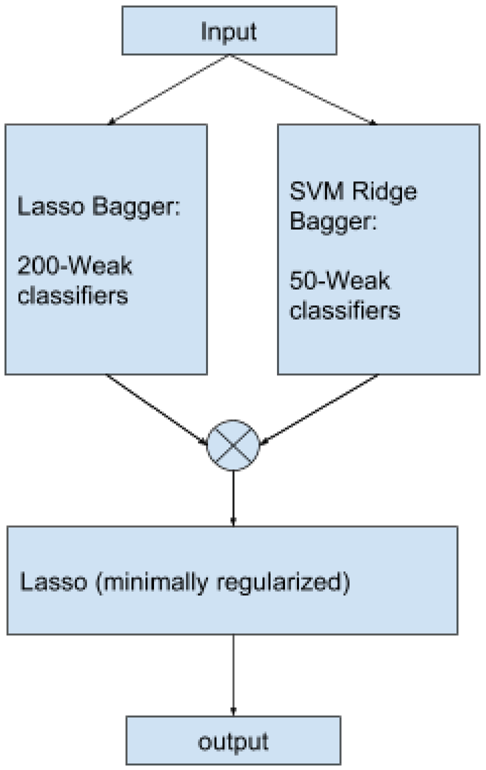
Model 1:

Cross Validation Error = 0.1441
Train Error = 0.1226

Model 2:
Cross Validation Error = 0.1390

Train Error = 0

3.4 Novel Method: Ensemble - Stacking



Stacking requires weak learners so \lambda is 0.0002. Will not work with strong classifiers

This was our 3rd submission to the leaderboard., Ensemble Error on Leader Board - 0.0775

As we can observe from the above table, for Lasso, though the error is decreasing as we increase the number of classifiers, but we chose 50 classifiers as increasing the number of classifiers increases the time complexity.

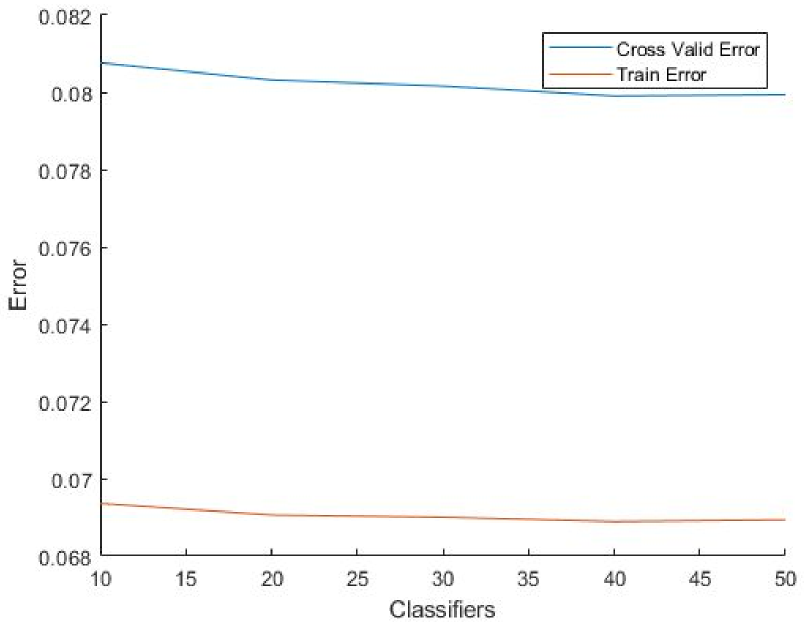
For Lasso, we selected 200 features out of 2000 features because the probability of missing out a feature comes out to be 0.00051 which is very less and affordable.

For Ridge, we selected 200 features out of 2000 features because the probability of missing a feature comes out to be 7*(10^{-10}).

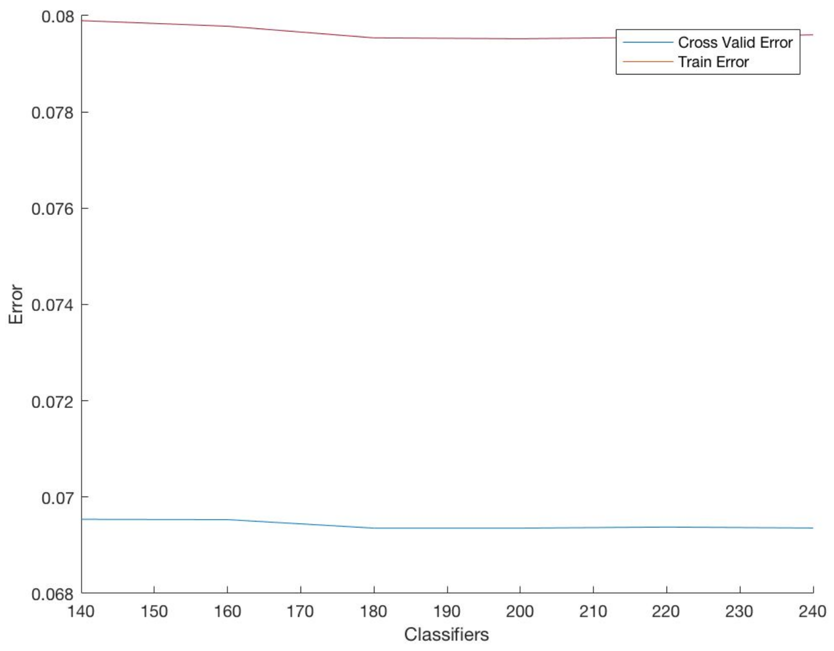
The reason why we chose 200 features out of 2000 features, 50 and 200 classifiers in case of lasso and ridge respectively because, increasing any of the values would make it computationally time consuming and with the chosen values, the probability of missing a feature is very less.

We did some experiments on choosing the number of classifiers for lasso and Ridge. Keeping the feature count limit to 200.

Cross Validation Errors:
Elastic Net Config(0.001, 0.002):



Ridge Config(0.001, 0.002):



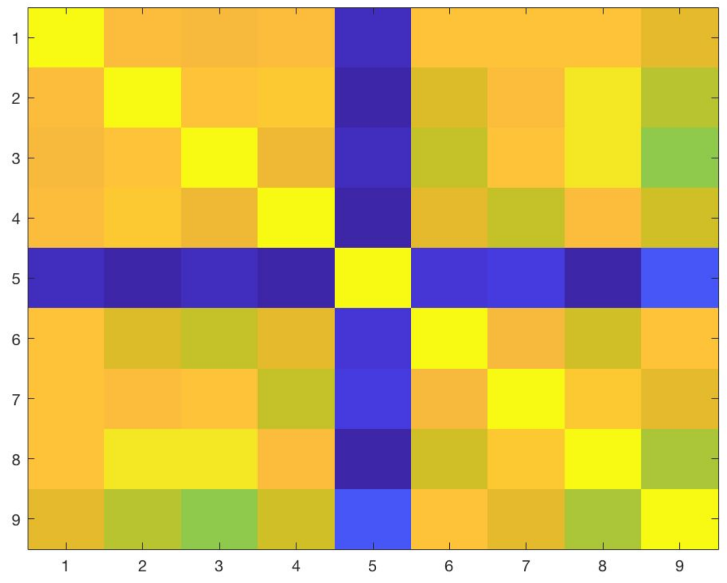
Results:

Cross Validation Error: 0.0799
Training Error: 0.0684

Section 5: Interpretation

5.1 Correlation Between The Labels

The heatmap showing the correlation between the different outcome variables is given below.



In this heatmap, the more yellow the block, the more correlation between the two labels.

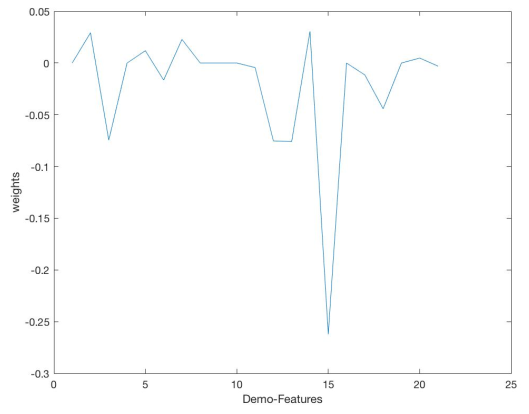
The most surprising fact is that outcome label 5, 'health_pcexcdrin', which is the percent of adults that report excessive drinking seems to have no correlation to any other outcome label. This label was also the hardest to classify as it gave us the maximum error. According to the heatmap, label 5 had the most correlation to label 9, which indicates the percent of adults who reported being obese. This makes sense, as there is medical evidence that alcohol causes weight gain.

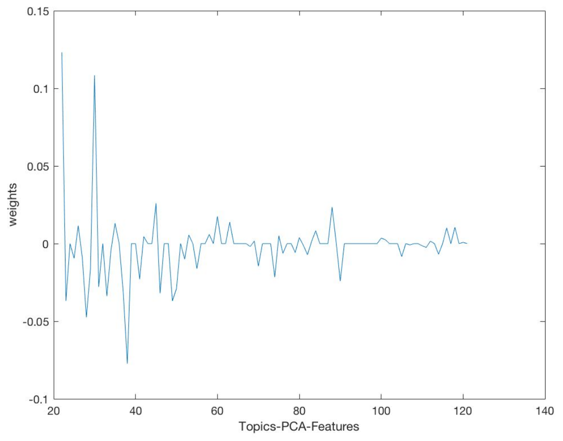
There also appears to be a correlation between label 8, average number of reported physically unhealthy days, and labels 2 and 3, which are the percent of adults reporting fair or poor health and the percent of adults reporting mentally unhealthy days in a month. This mean that more physical activity results in better health and better mental health too, or at least, there is some correlation between these variables.

5.2 Comparison between Feature and Labels

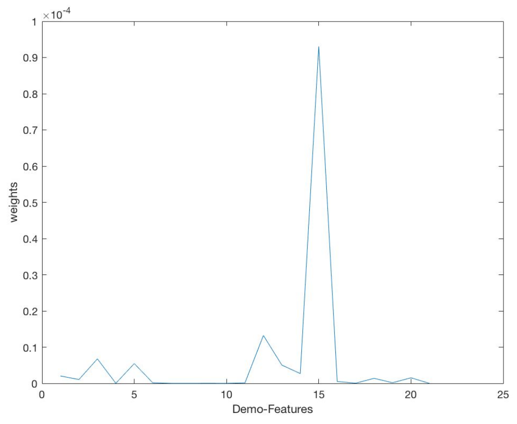
health_aamort

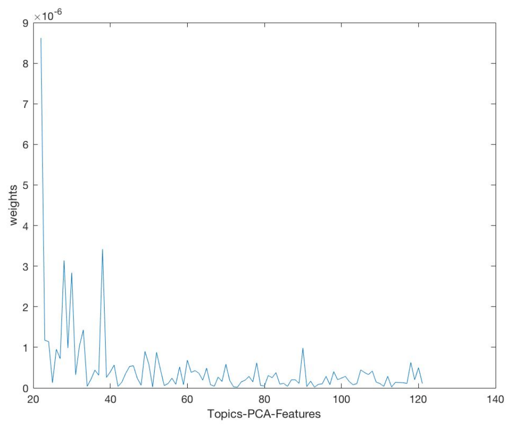
Elastic Net Weights





Regression Tree Weights (fraction representing the importance of feature.)

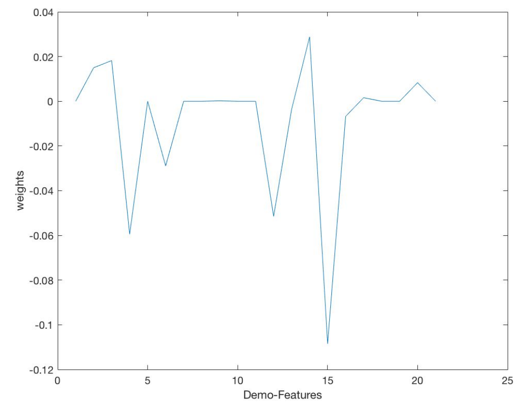


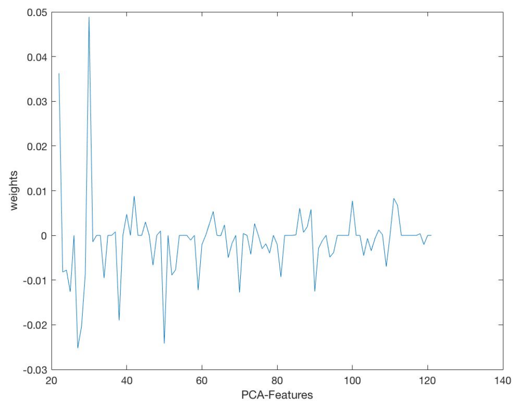


We see that for **health_aamort,** which predicts the years of potential life lost, Demographic Feature, ses_log_hhinc, which is the median household income, has more importance. This could be because a lower median income results in lack of facilities and perhaps inability to afford healthcare. Principal Component 1 has more weight. Followed by 9, 17.

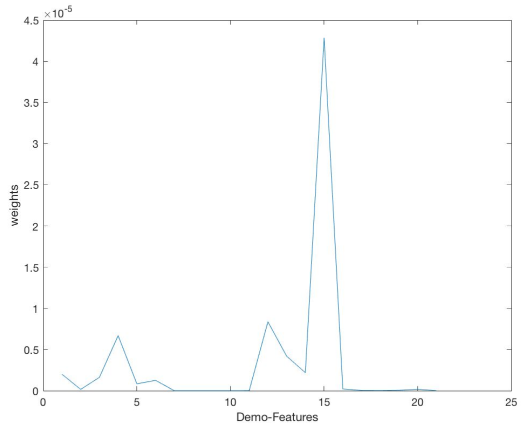
health_fairpoor

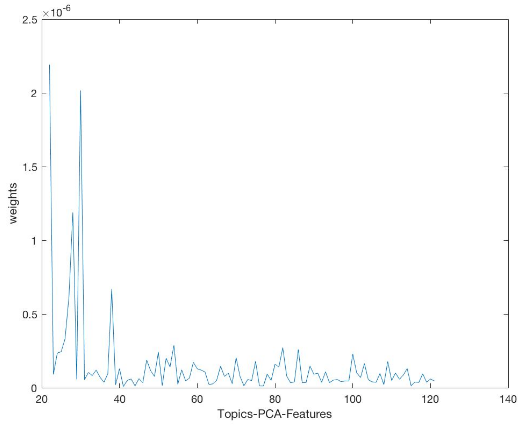
Elastic Net Weights





Regression Tree Weights (fraction representing the importance of feature.)

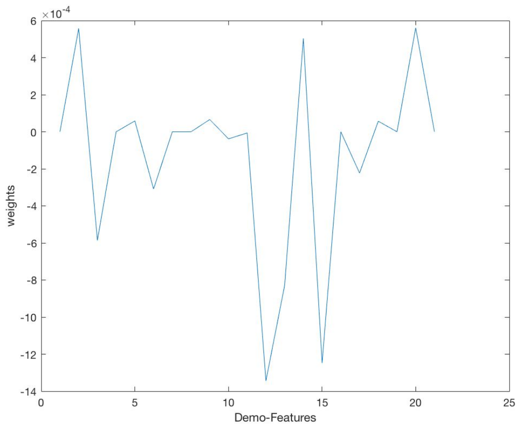


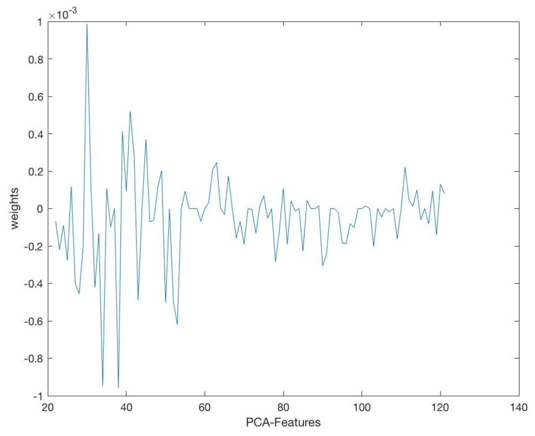


We see that for **health_fairpoor,** Demographic Features, household income, percentage of population with a college degree, and the percentage of the population that is white have more importance. Income has been given the highest weight. Clearly, income plays a large role in determining the health statistics of a county. And the Principal Component 1 has more weight. Followed by 9, 7.

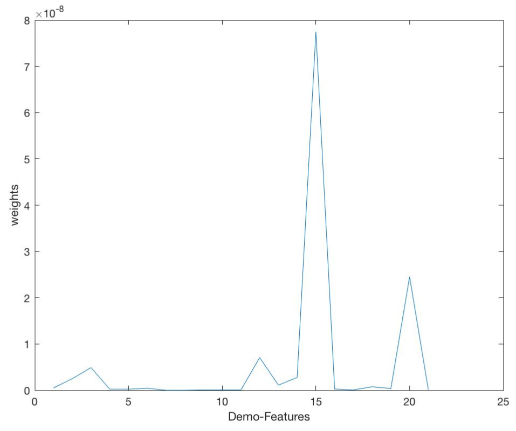
health_mentunh

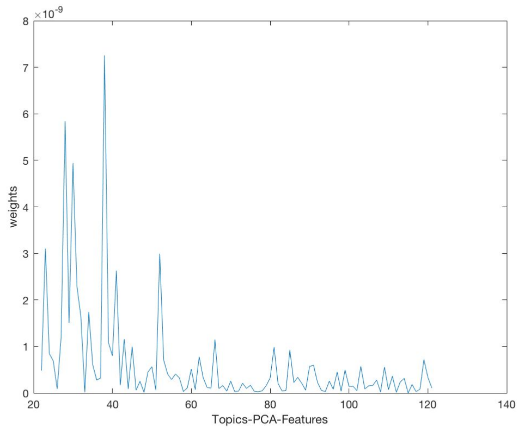
Elastic Net Weights





Regression Tree Weights (fraction representing the importance of feature.)

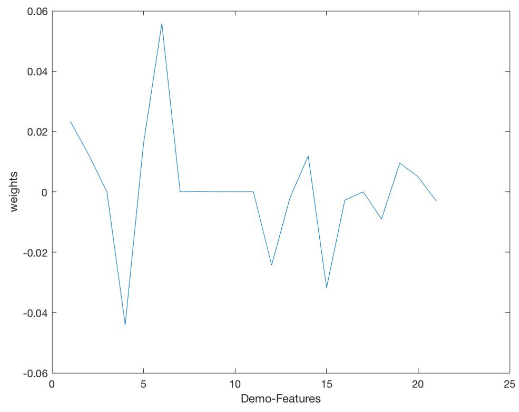


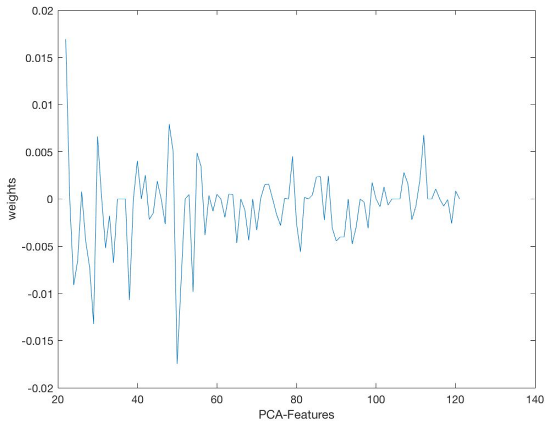


We see that for **health_mentunh,** Demographic Features, household income, the percentage of population with a college degree and the percentage of unemployed people has more importance. And the Principal Component (7,9, 17) has more weight.

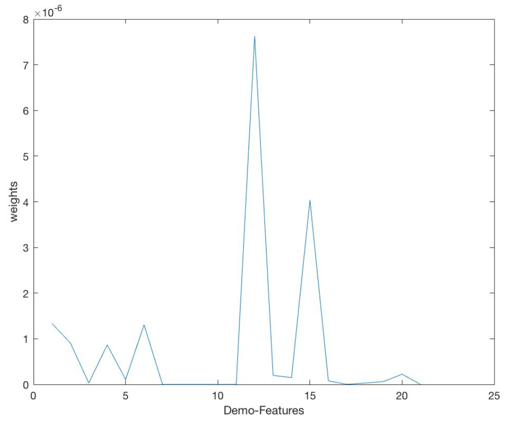
Health_pcdiab

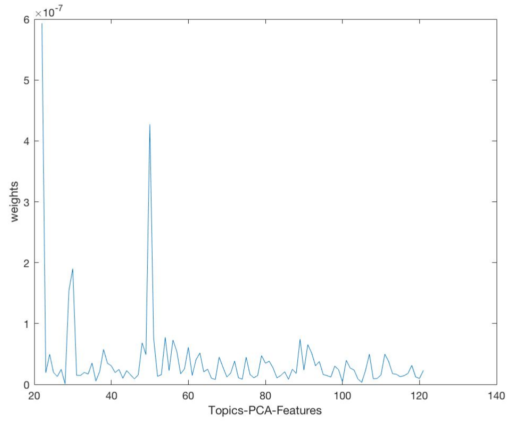
Elastic Net Weights





Regression Tree Weights (fraction representing the importance of feature.)

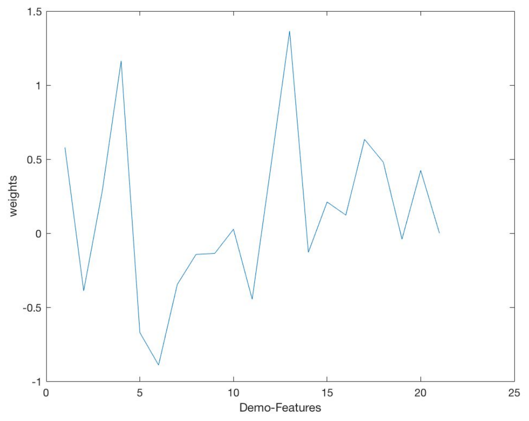


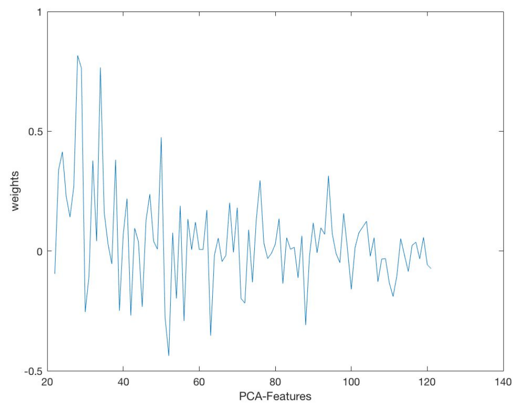


We see that for **Health_pcdiab,** Demographic Features, the percentage of white population, percentage of people over 65, income and percent of college degree holders have more importance. And the Principal Component (1, 29, 9) has more weight.

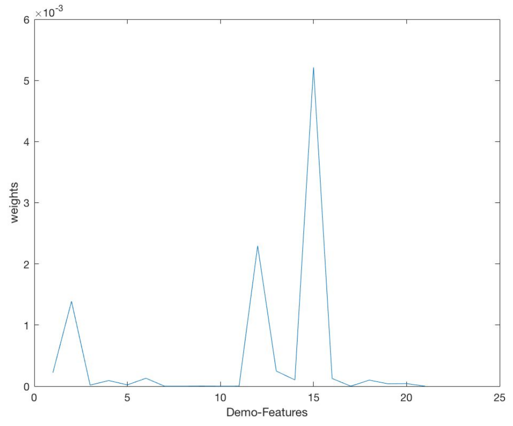
Health_pcexcdrin

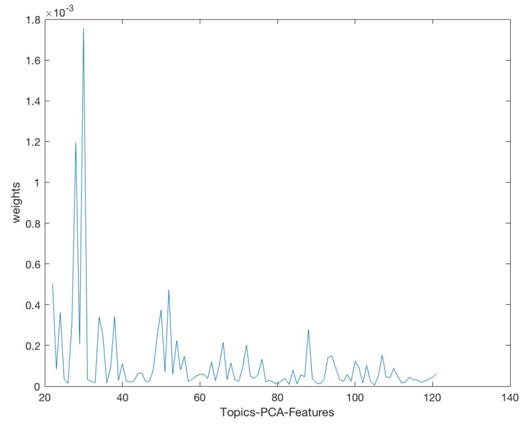
Elastic Net Weights





Regression Tree Weights (fraction representing the importance of feature.)

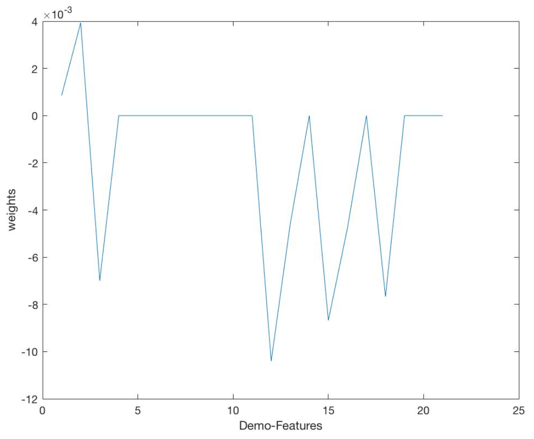


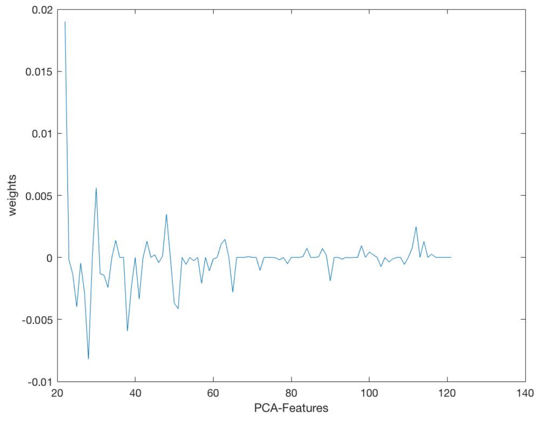


We see that for **Health_pcexcdrin,** Demographic Features, the percentage of white population, percentage of people over 65, income and percent of college degree holders have more importance. And the Principal Component (7, 9, 13) has more weight.

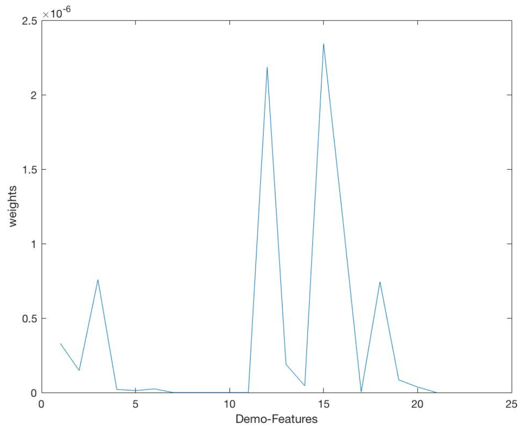
Health_pcinact

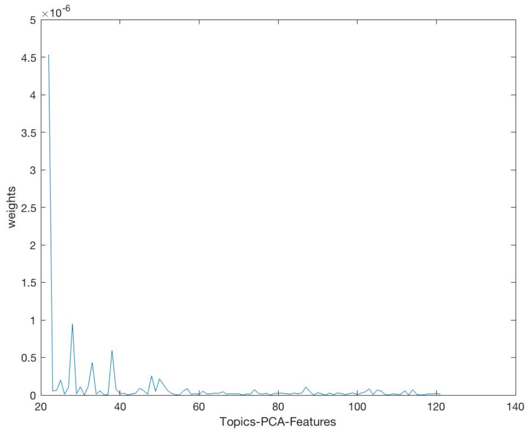
Elastic Net Weights





Regression Tree Weights (fraction representing the importance of feature.)

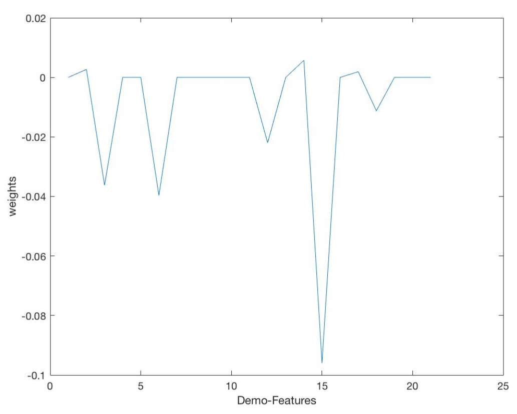


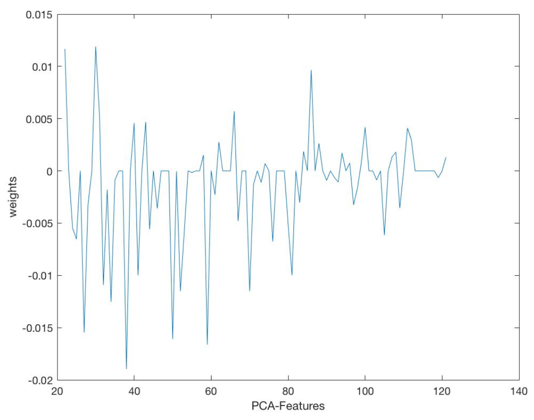


We see that for **Health_pcinact,** Demographic Features, household income and percentage of population with a college degree has more importance. And the Principal Component (1,7) has more weight.

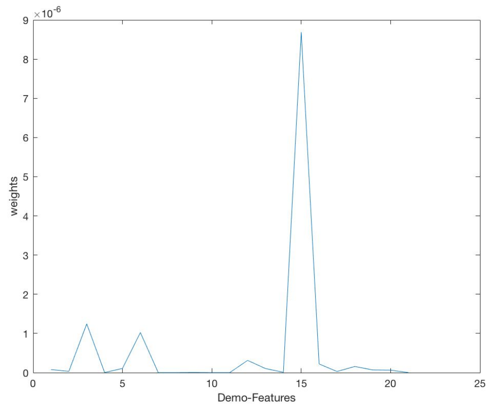
health_pcsmoker

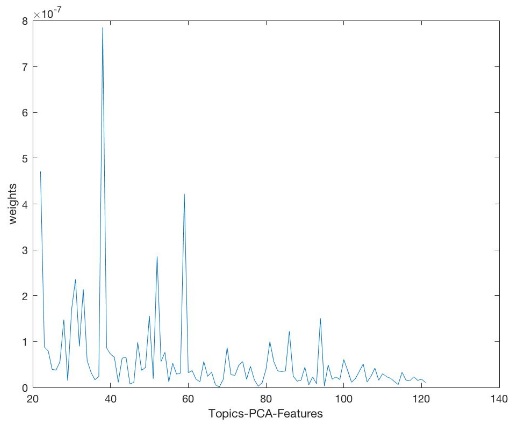
Elastic Net Weights





Regression Tree Weights (fraction representing the importance of feature.)

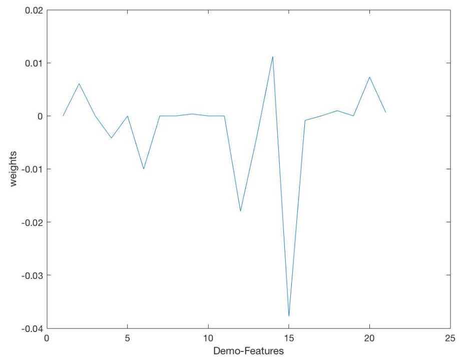


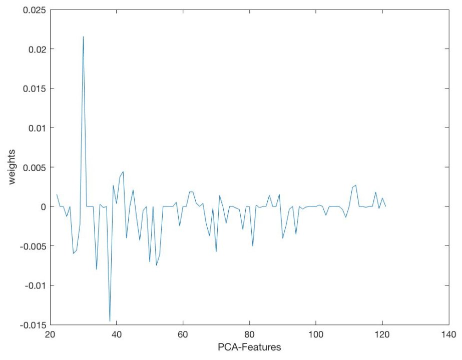


We see that for **health_pcsmoker,** Demographic Feature, household income, percentage of hispanic population, and percentage of people under the age of 18 have more importance. And the Principal Component (17, 38) has more weight.

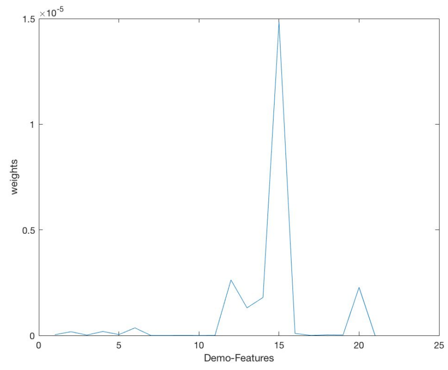
health_physunh

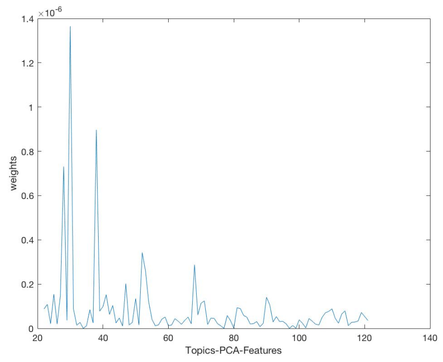
Elastic Net Weights





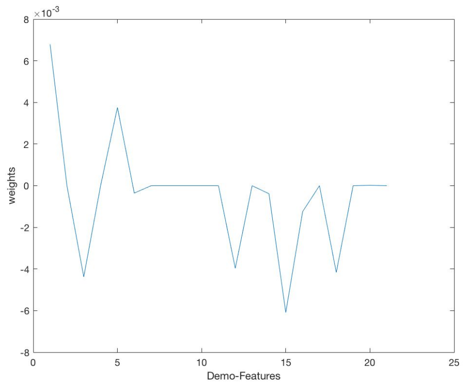
Regression Tree Weights (fraction representing the importance of feature.)

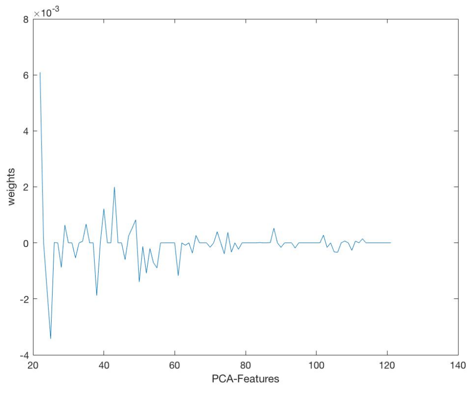


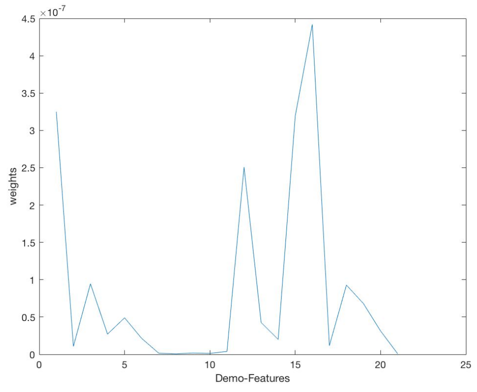


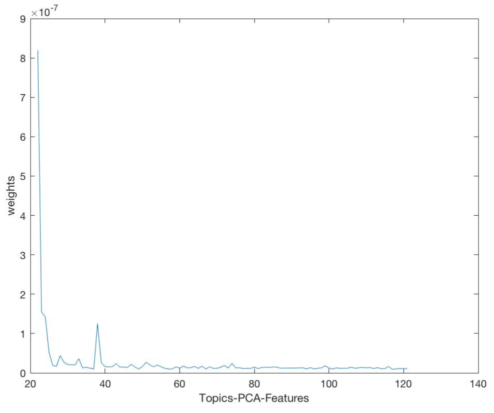
We see that for **health_physunh,** Demographic Features household income and percentage of college degree holders have more importance. And the Principal Component (9, 7, 17) has more weight.

Heath_pcobese









We see that for **Heath_pcobese,** Demographic Features, household income, percentage of college educated people and percentage of black population have more importance. And the Principal Component (1, 2) has more weight.

Conclusion

Model Comparison

Model	Cross-Validation Error

Gaussian Process Regression, Fit-Grp with non-linear kernel	0.725
Stacking	0.0799
ElasticNet	0.0811
KNN	0.1441

Clearly, non-linear models work better on the data. Stacking and ElasticNet are linear models (as we have used Lasso and Ridge Regression models in our stacking algorithm). While KNN can model non-linear relationships, in this case, it is hard to say what metric is suitable as a distance metric and how we should calculate the K-nearest neighbors.

Therefore, the Gaussian Process Regression Model gave us the lowest error.

Improving Accuracy:

To improve the accuracy of our model, we analyzed the PCA component vs Error graph for every outcome label and chose different number of principal components for every label that gave us the least error.
Skew correction of labels has helped to increase the accuracy. Used Log, square root and cubic-root for skew correction of labels. Linear regression gives better accuracies when the data distribution is similar to normal distribution. We also used PCA to reduce the dimensionality of the data and

DIFFERENT MODELS TRIED AND WHY IT DIDN'T WORK :
K nearest neighbours uses distance metric for learning, but this is a regression problem, so standardization is not a good thing as we do not know the metrics of the features. That is why KNN performed bad.

Neural Networks didn't work because the data that was given to us is very less (around 1000 samples), and the neural net had to learn a lot of parameters(weights), but since the data was not enough it couldn't learn the optimal parameters.

For Boosting Trees the error was 0.08. It wasn't working bad but gaussian regression process was working better. Since data can be modeled properly with a kernel, fitgrp finds kernel parameters such that it gives an optimal fit. Whereas for boosting trees, there is a trade-off between over-fitting and under-fitting and it was difficult to generalize as we approximate too much it was leading to over-fitting. Even after finding a trade-off, gaussian regression process was giving better results as compared to boosting trees.

The data given to us is non-linear and stacking and elastic net are linear regression methods, whereas gaussian regression process is non linear. So gaussian regression process worked the best among the models that we tried.

DATA PROCESSING THAT DIDN'T WORK:
Skew correction on input data didn't work because we are doing skew correction independent of every column feature . Even though the overall skew was less, it could have been that it could have missed some correlation and so we were getting better training error without performing skew correction on the input data.

We also tried CCA for dimensionality reduction as performing only PCA on the input set may lead to loss of information that is correlated with our labels. However, that did not work because the rank of our labels is 9 and CCA reduces our Twitter data to a 9-dimensional subspace from 2000 dimensions. Therefore, it is not able to capture the correlation between the features and the labels very well.

Future Work

In the future, finding an approximate non-linear transformation could help model the data better.