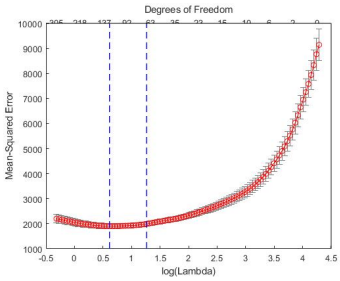
Project Report

Vatsal Chanana, Dewang Sultania, Shikhar Brajesh
Group Name: Overfitting is coming

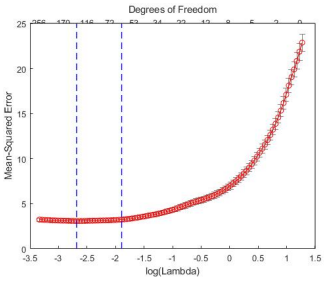
December 2018

1 Leaderboard Submission

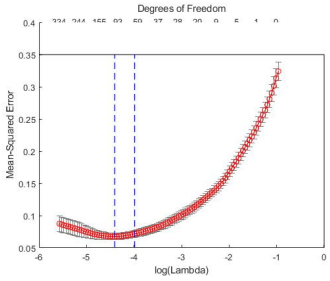
1. **Method 1**: As our first model we used an ensemble of glmnet, Gaussian Process Regression and Random Forest. Glmnet is an implementation of the elastic net developed by Qian, J., Hastie, T., Friedman, J., Tibshirani, R. and Simon, N.(Stanford University). It is a combination of lasso and ridge regression. The code is implemented in such a way that it learns the optimal regularization parameters and which features to use(L0-penalized loss) using cross validation. We split the data into training and testing set with 70% data for training and 30% used for testing. So we used that testing error as our measure of performance before submitting to the leaderboard. Using only glm was giving a testing error of 0.0805 and that combined with an ensemble of Gaussian Process Regression and Random Forest gave an error of 0.0769 on our test set. Before running Gaussian Process Regression we used PCA to reduce the dimensionality of dataset to 250 components. The number was chosen based on how well the variance of the data is explained using a particular number of components. We kept the first 21 rows as it is and applied PCA only on the tweets. The model was trained in such a way that it made predictions for each of the y's separately. The ensemble gives 70% weightage to GLM, 20% to Gaussian Process Regression adn 10% to Random Forest. These weights were chosen based on how well these models performed individually, we started with random weights to model predictions and chose the ones which gave the least error on 30% seperated out data. On submission the score we got was 0.0770 which was enough to beat the first baseline of 0.1479 by a margin.



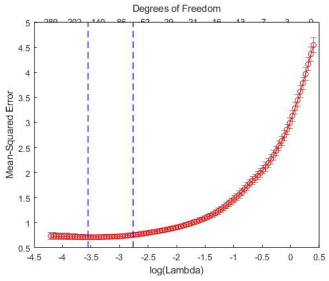
(a) Prediction 1



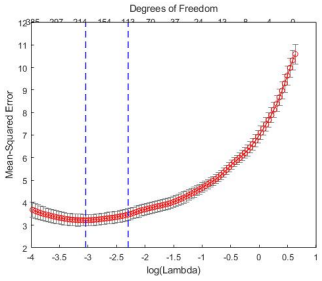
(b) Prediction 2



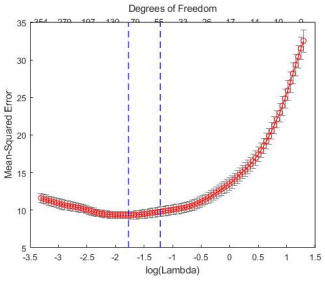
(c) Prediction 3



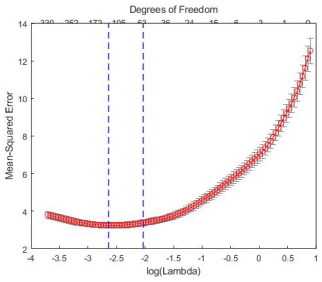
(d) Prediction 4



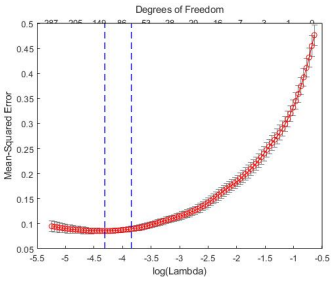
(e) Prediction 5



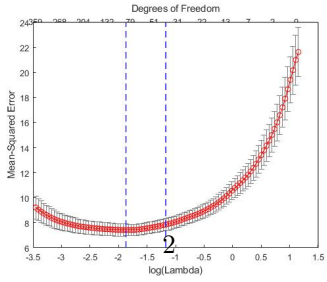
(f) Prediction 6



(g) Prediction 7



(h) Prediction 8



(i) Prediction 9

Figure 1: GLM using Cross Validation to determine optimal λ for different y's

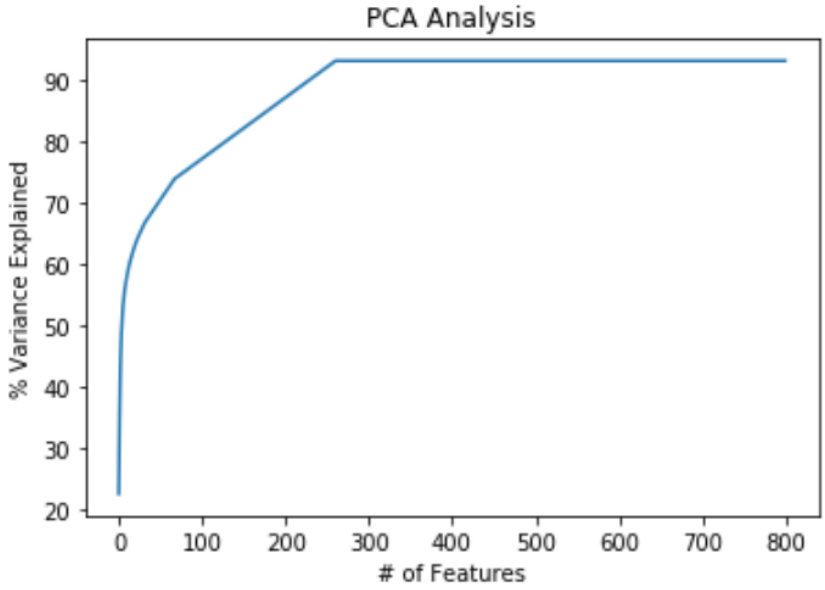


Figure 2: Variance and PCA

2. **Method 2**: For improving our accuracy further, we used the concept of learning the residuals as well. The motivation behind this was that models like glm and Gaussian Process Regression are able to capture the linearities in data quite well but are not able to capture the non-linearities, so for that purpose tried training the residual errors of the model obtained and adding the learned residuals back to the test predictions, so after glm and GPR make their predictions we trained a Random Forest on the residual errors of the training set and then used that to predict the residuals of the test set. Then we trained an ensemble of GLM with residual learned using Random Forest , Gaussian Process Regression with Residual learned using Random Forest and glm with residuals learned using ridge. This ensemble gave weights of 0.5, 0.25 and 0.25 respectively to GLM Random Forest, Gaussian Process Regression with Random Forest and GLM with ridge. This improved our test error significantly giving us a score of 0.0751 on the leaderboard.

3. **Method 3**: Disclaimer: The method tried here happened in the heat of the contest, this method is not something we can explain, it is something that worked and earned us a place in top-10, so we will just describe what it was but can't really explain why it worked. I guess sometimes in contests, it boils down to just trying various things till one of them works and the method we are going to describe is one such thing. Our final submission which yielded us a score of 0.0732 and entry into the prestigious top 10 is an ensemble of ensembles, it combines algorithms like PCA, Random Forest,

Lasso, Glm, Ridge and Gaussian Process Regression. What we did here was basically combine the above two methods. For each y-column in the dataset, we first trained an ensemble of glm, ridge and lasso and learned the residuals of that ensemble using Random Forest, we also trained an ensemble of glmRandomForest, glmRidge and Lasso (glmRandomForest and glmRidge refer to the residual model described in Method 2) and then did a weighted average of these two ensembles along with Gaussian Process Regression combined with residual learning. This method is kind of true to our name (Overfitting is coming) but it seems like it did not overfit that much and gave a testing error of 0.0732 along with Rank 10 on the leaderboard.

2 Discriminative Methods

Errors:

Model	**Training Error**	**Testing Error**
Ridge Regression	0.0639	0.0796
GLM (ElasticNet)	0.0724	0.0805
Gaussian Process Regression	0.0511	0.0819
Lasso Regression	0.0686	0.0806
Random Forests	0.0448	0.0905
SVM Regression	0.0656	0.0809
Shallow Neural Network	0.1122	0.1184

Ridge Regression:

The training data was first preprocessed by mean centering and scaling the data and then performing PCA on the data. For each column in y, we trained a different linear regression model with L2 penalty on the weights.
Hyperparameter tuning:
A different penalty λ was chosen for the ridge regression model to predict each type of output label (i.e. column in y) by 10-fold cross validation. We tried λ values ranging from 10^{-5} to 10^{3} ($[10^{-5}, 10^{-3}, \ldots 100, 500, 1000]$) for each of the 9 models.
PCA + Ridge regression seemed to be the best individual model with the lowest testing error.

Elastic Net (GLMNet):

Glmnet is an implementation of the elastic net developed by Qian, J., Hastie, T., Friedman, J., Tibshirani, R. and Simon, N.(Stanford University). It is a combination of lasso and ridge regression. The code is implemented in such a way that it learns the optimal regularization parameters and which features to

use(L0-penalized loss) using cross validation. We split the data into training and testing set with 70% data for training and 30% used for testing. We did not perform any dimensionality reduction prior to GLMNet, GLMNet does the feature selection by having an L1 penalty on the weights.
GLMNet was the second best individual model but worked well with ensembles and was a part of the final model when improved by training the residuals.

Random Forests

Since, random forest is one of the most effective and sought after machine learning models in data science competitions, we tried to predict each label, Y with a separate random forest model. We tuned the number of trees used by each of the 9 TreeBagger models using 10 fold cross validation. Values (for the number of trees) in the range of 3 to 1000 were used in the cross validation process.

Gaussian Process Regression

Gaussian process regression (GPR) models are nonparametric kernel-based probabilistic models. We hoped to capture any non linearities in the data with GPR models. It was used with residual training as part of our final model ensemble as well.
We tuned the 'Regularization' hyperparameter by 10 fold cross validation. We tried values ranging from 10^{-5} to 10^{3} ($[10^{-5}, 10^{-3}, \ldots 100, 500, 1000]$) for each of the 9 models. 0.001 seemed to be the best regularization parameter for most of the models.

SVM Regression

We tried using SVM regression with a linear kernel, tuning the values of C's using 10-fold cross validation. Values ranging from $10^{-3} to 10^{3}$ were tried for kernel widths. SVM was inferior to ridge, lasso or glm models and wasn't included in our final model.

Shallow Neural Networks

Since there was very less data to train a deep neural network, we tried training a shallow neural network with 10 hidden layers and producing 9 outputs. Our intuition behind trying a neural network was that it would capture the non linearities in the model and some of the correlations between the outputs. Because of the scarcity of sufficient data to train a neural network, the model seemed to underfit the data and thus, performed poorly.

3 Instance Based Methods

We implemented the following models in this category:
K Nearest Neighbors
Kernelized Nearest Neighbors

We performed 10-fold cross validation to identify the best value of hyperparameters (K for K Nearest Neighbor and σ for Kernelized Nearest Neighbors. We initially tried to use all features in the dataset and then performed dimensionality reduction using Principal Component Analysis. For different number of principal components, we determined best value of hyperparameters.

K-Nearest Neighbors

K vs cross validation error	
K-values	CV-Error
1	0.1435
3	0.1529
5	0.1557
7	0.1587
9	0.1605
11	0.1606

Test Error without PCA : 0.1370

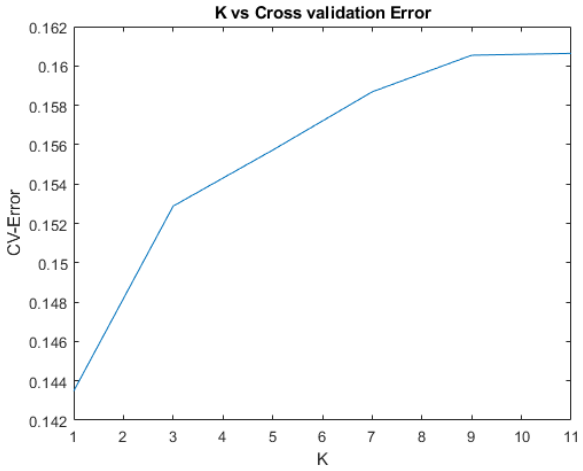


Figure 3: Cross validation error for different K values

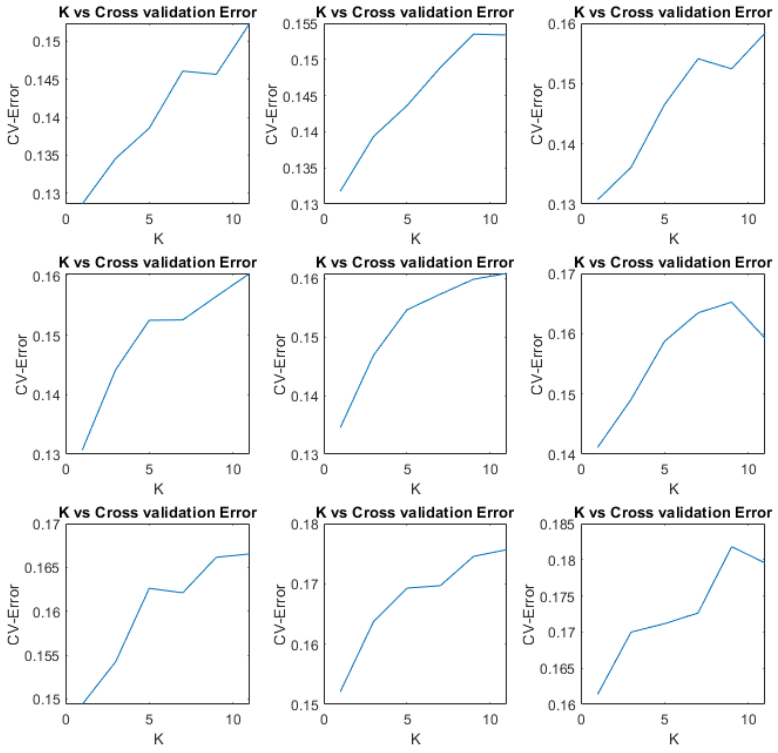


Figure 4: Cross validation error on PCA components : 10,30,50, 70, 100, 150, 200, 250, 300 (top to bottom)

We observed than dimensionality reduction does not help in our case as there is no considerable decrease in CV-error and K value of 1 is optimal in all cases.

Kernelized Nearest Neighbors

σ vs cross validation error	
σ-values	CV-Error
0.0001	0.1945
0.001	0.1945
0.01	0.1945
0.1	0.1945
1	0.2012
10	0.2572

Test Error without PCA : 0.1876

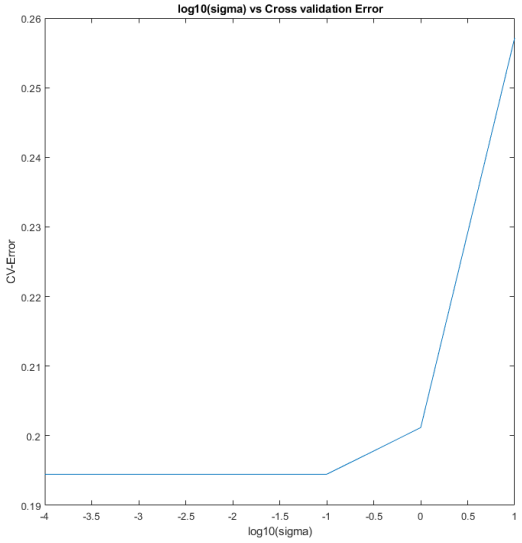


Figure 5: Cross validation error for different σ values

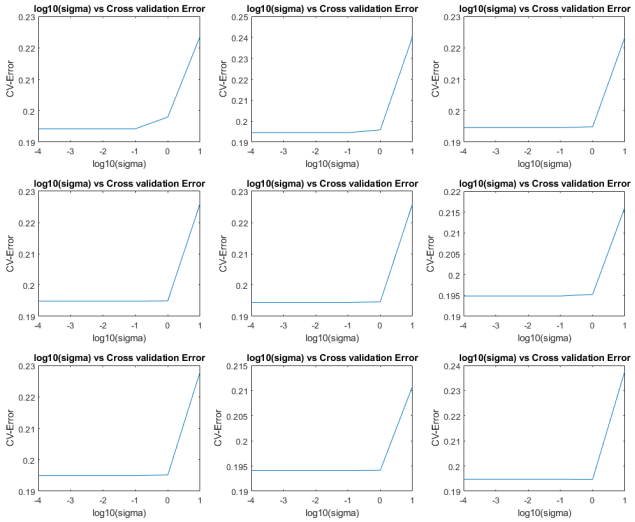


Figure 6: Cross validation error on PCA components : 10,30,50, 70, 100, 150, 200, 250, 300 (top to bottom)

We observed than dimensionality reduction does not help in this case also.

There is no considerable decrease in CV-error and σ value of 0.1 is optimal in most cases.

Reducing the number of dimensions in instance based method did not improve our error. We decided, it would be better to apply different models with different number of PCA components to predict each type of label. Therefore we abandoned this approach.

4 Generative Methods

K-Means Clustering: We tried clustering the dataset into K clusters prior to regression. We partition our dataset into K sets (each set representing a cluster) and train K X 9 ridge regression models (one of each column in y) on these sets. We obtained the K predictions for each column from these K linear regression models and formed the final predictions by weighting the individual predictions with the distance of the test data point from the respective cluster center.
To get the predictions for a test data point x_i, for each column j in y_i, the weighted predictions were obtained as:

$$y_j^{(i)} = \frac{\sum_{c=1}^{K} \hat{y}_{j,c}^{(i)} d_c(i)/}{\sum_{c=1}^{K} d_c(i)}$$

where $d_c(i)$ is the L2 distance of x_i from the cluster center for the c^{th} cluster.

We performed 10-fold cross validation to find the optimal value of K. For $K > 1$, we obtained the minimum validation error on K=4.

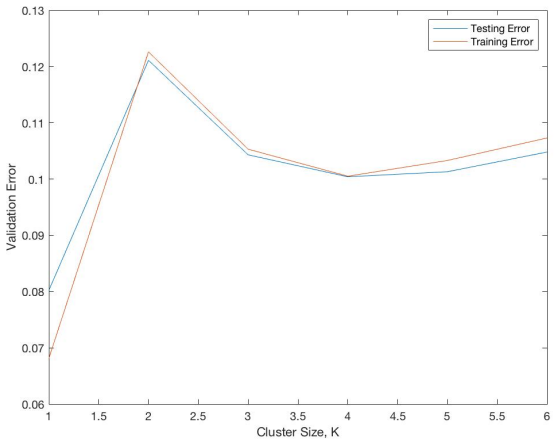


Figure 7: Error in K means clustering

The above plot shows that segregating the data into clusters and weighting the predictions based on the relative distance actually increases the error. K = 1

i.e. the entire data in a single cluster has the least error. This can be attributed to the fact that the training data is relatively small and when we partition the training data into multiple clusters and train a separate model on each of these clusters, the data available for these linear regression models is scarce. The K individually trained linear regression models are inaccurate and thus the overall weighted predictions are worse than training a single linear model on the entire training dataset.

5 Novel Methods

The novel methods we tried are described in leaderboard submission section. Errors of our novel methods:

Model	**Test Error**
Ensemble of GLM, Gaussian Process Regression and Random Forests	0.0769
Ensemble of GLM and Gaussian Process Regression with Random Forest and Ridge to learn the residuals	0.0751
Ensemble of ensembles (method 3 in leaderboard section)	0.0732

6 Interpretation

The obesity values predicted by our model on the training set according to metro type, non-metros seem to have the most varied range in when it comes to obesity and the median values for large and medium metros is less than that of small and non-metros showing that according to the model, more people are obese in small metros and non-metros compared to large and medium ones.

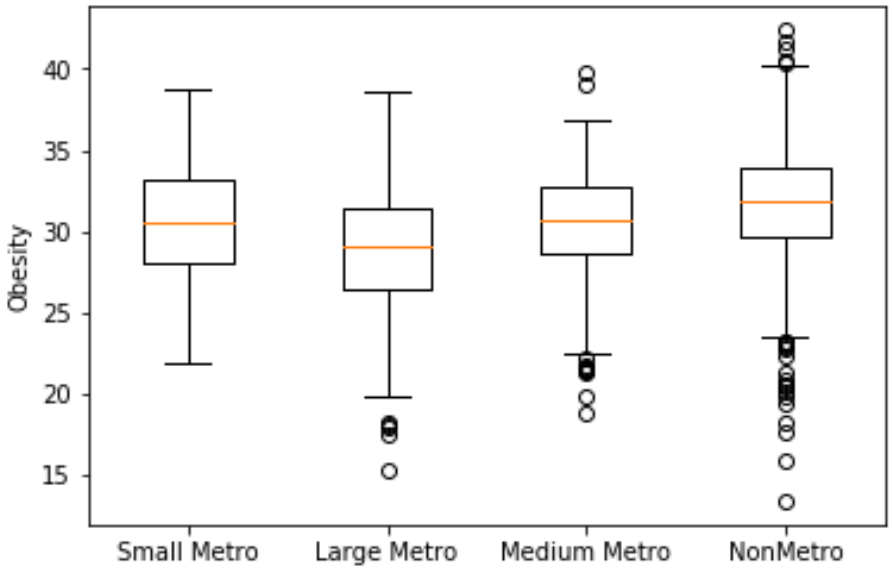
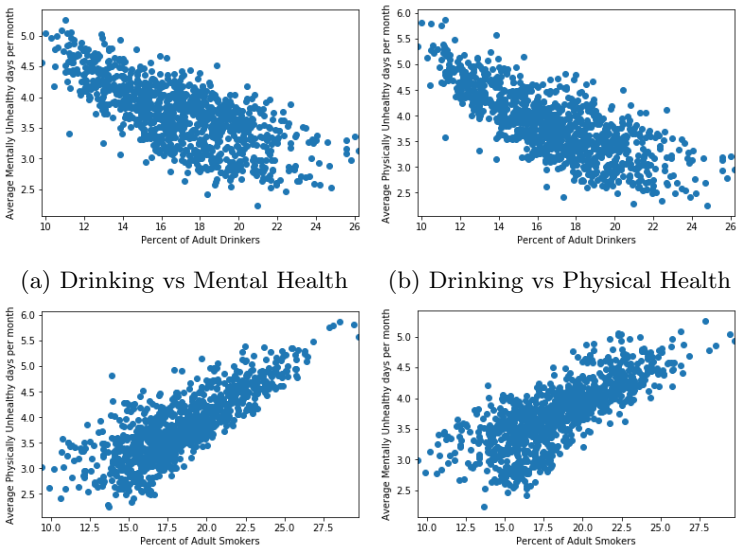


Figure 8: Box Plots of obesities predicted for various types of metros

Now we explored how drinking and smoking predictions made by the model are related to the physical and mental health. These plots show that smoking and physical and mental health are positively correlated while drinking and physical mental health are negatively correlated.



(a) Drinking vs Mental Health (b) Drinking vs Physical Health

(c) Smoking vs Physical Health (d) Smoking vs Mental Health

Figure 9: GLM using Cross Validation to determine optimal λ for different y's

which again shows that correlation and causation are not the same thing, there might be the case that people with high income brackets tend to drink more but are physically healthy, so we explore that further and found that indeed people with more housing income tend to drink more but because they are wealthy and can access places of physical health they are physically healthy despite drinking.

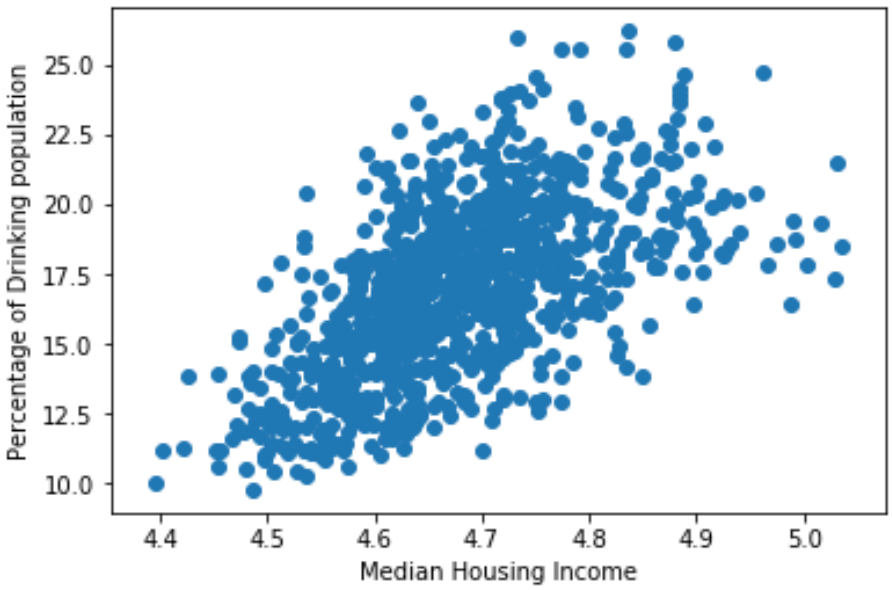


Figure 10: Income vs Drinking

What people are posting where the age adjusted years of potential life lost is greater than 500:

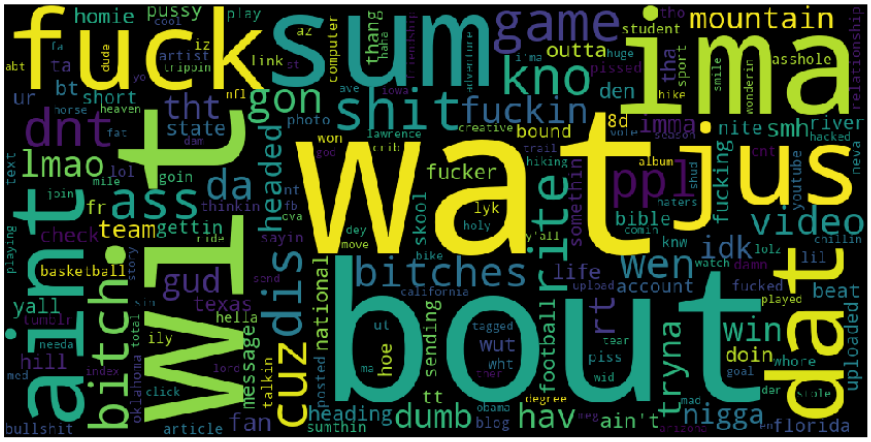


Figure 11: Word Cloud for greater than 500 YPLL

What people are posting where the age adjusted years of potential life lost is less than 200:

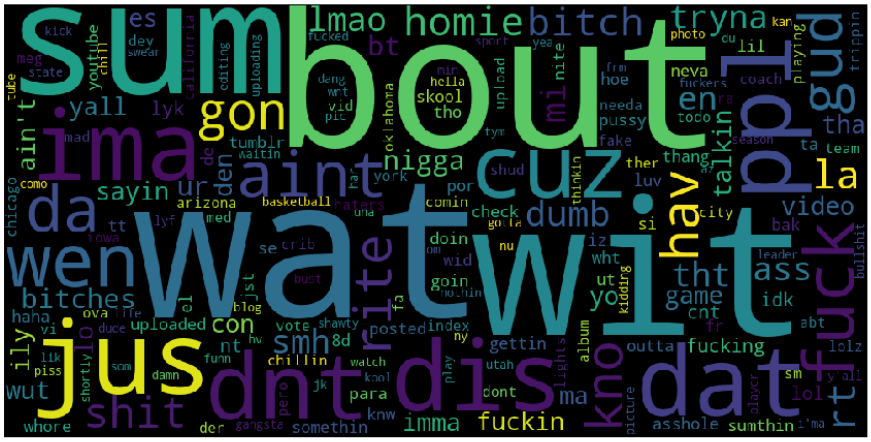


Figure 12: Word Cloud for less than 200 YPLL

7 Conclusion

1. Our ensemble described in leaderboard method 3 performed the best while the Instance based Nearest neighbours method performed the worst.
2. Ensembles and learning the residuals obtained improved our accuracy a lot.
3. We tried using K-means with weighted averaging based on distance(inspired by how recommendations work) but that failed to work in this case.
4. Model can be further improved by more hyperparameter tuning and trying out methods like partial least square regression or by making better ensembles.

References

[1] Glmnet for Matlab (2013) Qian, J., Hastie, T., Friedman, J., Tibshirani, R. and Simon, N.
`http://www.stanford.edu/ hastie/glmnet_matlab/`

[2] LARS Algorithm in MATLAB, Implementation by Sung Soo Kim
`https://www.mathworks.com/matlabcentral/fileexchange/23186-lars-algorithm`