FINAL REPORT: PREDICTING GUN VIOLENCE IN THE UNITED STATES

BY MATHIAS SCHMUTZ AND OLIVIA COLE

DATA AGGREGATION

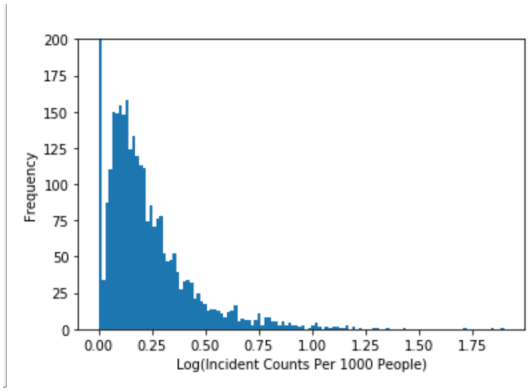
We started with a dataset of over 260k incidents of gun violence in the United States that occurred between 2013 and 2017. Each incident was tagged with its location, the number of injuries and deaths, the age and gender of the shooter, as well as additional information. In addition, we had a list of census data from the American Community Survey 2015 5-year estimate, which contained a large number of demographic and economic attributes pertaining to specific geographical regions. We looked at the data broken down by county and census tract (an area of about 4,000 inhabitants).

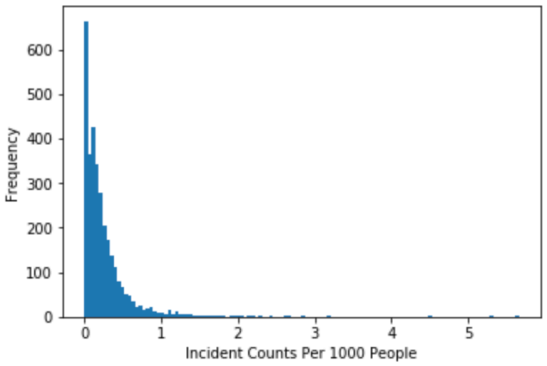
Next, we mapped each of these shootings to a county and census tract from the census data. To do so, we used the Python library *censusgeocode*. This library provided a method that took in a pair of latitude/longitude coordinates and returned the corresponding census county and tract. Once each incident was mapped, we went through each tract and county and counted the number of incidents occurring in each one of the regions. The aggregation code that we used to perform all of this can be found in Appendix A.

After looking at these counts, we determined that we were only interested in shootings in which there was a fatality or injury. Some of the tracts with the highest number of incidents actually had very low injury counts. We assumed this to be discrepancies in reporting and/or policing between regions, so we decided to eliminate incidents in which nobody was harmed to help reduce this potential source of noise.

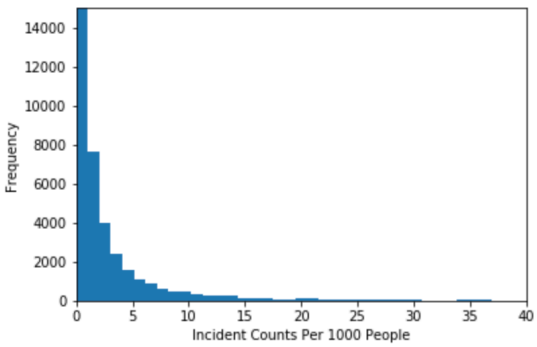
After doing this and removing census areas that were missing info, we obtained the following distributions:

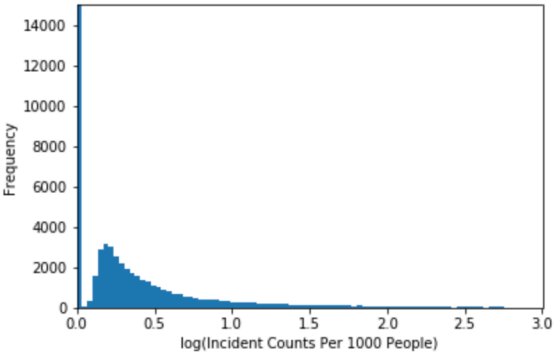
COUNTY-LEVEL





TRACT-LEVEL





As demonstrated by the histograms, over half of the census tracts had 0 shootings. Because of this heavily skewed log distribution, we found it very difficult to find a model that came close to reflecting this, and pursued just a model that produced county level results.

RESULTS: LINEAR REGRESSION

Initially, we performed basic linear regression on the county-level data. Because we were able to obtain an output distribution that looked mostly normal (by taking $\ln\left(\frac{incident\ counts}{1000\ people}\right)$), we decided to use this as our target variable (rather than the raw incident counts).

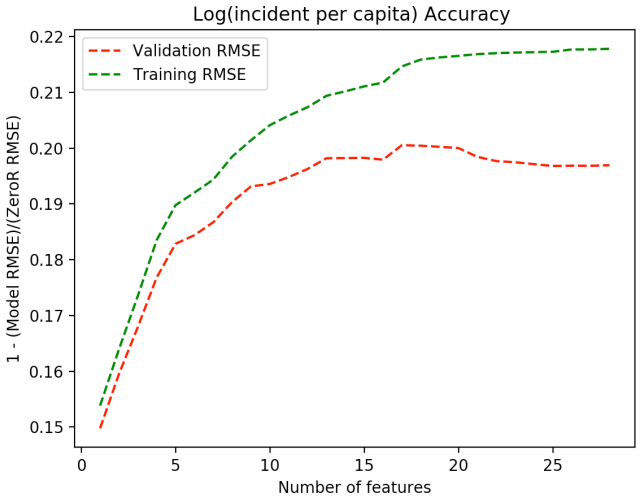
To evaluate our model's performance, we used two different accuracy measures, log accuracy and raw accuracy. For the first, we compared the root mean squared error (RMSE) of the predicted target versus the true target values, and compared this against a baseline. For the second, we convert our model's predictions into raw incident counts, and evaluated this against a baseline value for raw incident counts. The code to produce the baseline accuracy (zeroR) can be found in Appendix B.

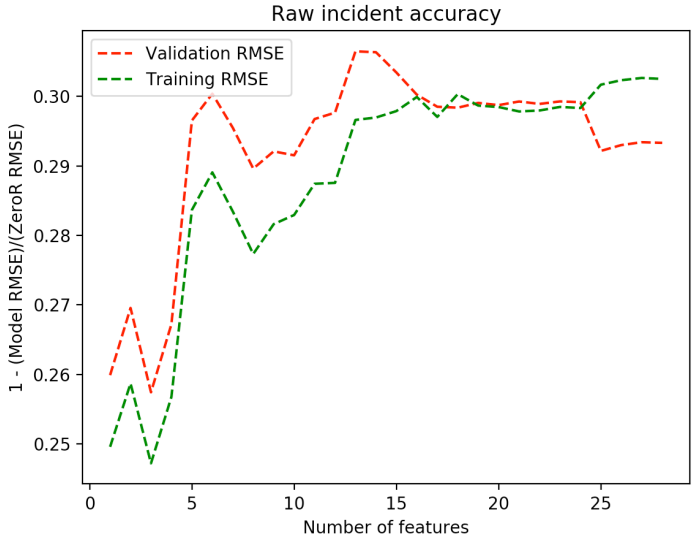
Log RMSE Accuracy	Raw RMSE Accuracy
$y = \ln\left(\frac{incident\_counts}{population} * 1000 + 1\right)$	$y = incident\_counts$
$y_{mean} = \text{mean(y)}$	$y_{mean} = \text{mean(y)}$
$y_{pred} => model\ prediction\ of\ y$	$y_{pred\_raw} = \frac{(e^{y_{pred}} - 1) * population}{1000}$
$RMSE = \frac{\sum_0^n (y_n - y_{pred\_n})^2}{n}$	$RMSE = \frac{\sum_0^n (y_n - y_{pred\_raw})^2}{n}$
$ZeroR = \frac{\sum_0^n (y_n - y_{mean})^2}{n}$	$ZeroR = \frac{\sum_0^n (y_n - y_{mean})^2}{n}$
$\boldsymbol{Accuracy = 1 - \frac{RMSE}{ZeroR}}$	$\boldsymbol{Accuracy = 1 - \frac{RMSE}{ZeroR}}$

This accuracy evaluated our model's ability to predict the raw incident counts of a census region, while the first accuracy graded our ability to predict the log transformed output. Obviously, positive values indicate the model outperforming zeroR.

To perform initial feature selection, we wrote an algorithm that started with zero features and greedily added the feature that most improved testing accuracy. On each subset of features, we performed basic linear regression with k-Fold validation using 3 folds. To get the training and testing accuracies for each subset, we averaged over the 3 folds.

Below are the plots displaying the accuracy values as a function of the number of features that the algorithm had added. Both error plots are generated with same linear model for each set of features, but using the different accuracy metrics described above. The code we used to perform feature selection can be found in Appendix C.





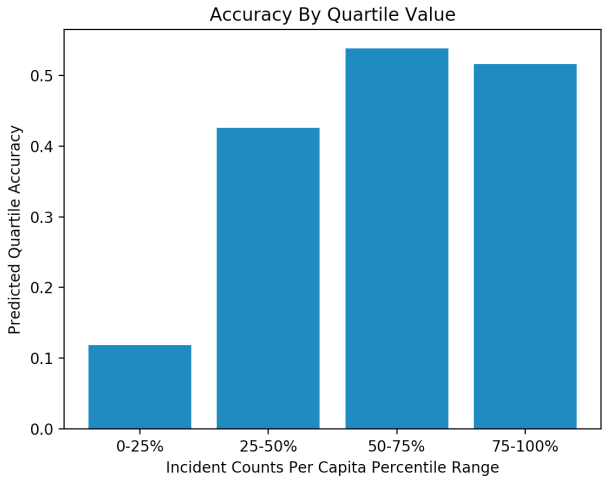
Below displays the input features along with their ranking according to the greedy algorithm described above:

Feature Name	Description	Rank
African-American	% of population that is African American	1
SelfEmployed	% of population that is self employed	2
MeanCommute	The average commute time for workers	3
PercentMen	% of population that are men	4
Drive	% of workers that drives to work	5
Office	% of employed people who work in an office	6
Hispanic	% of population that is hispanic	7
ChildPoverty	% of children that are below the poverty line	8
Walk	% of workers that walk to work	9
Production	% of workers that work in production	10
Poverty	% of people below the poverty line	11
White	% of population that is white	12
PublicWork	% of workers that work do public work	13
PrivateWork	% of workers that work do private work	14
WorkAtHome	% of workers that work form hom	15
Asian	% of population that is Asian	16
Native	% of population that are Native Americans	17
Pacific	% of population that are Pacific	18
OtherTransp	% of workers that does not commute via walking, transit or car	19
Service	% of workers that work in the service industy	20
IncomePerCap	Income per capita	21
Unemployment	Unemployment Rate	22
PercentEmployed	Percentage of People that are employed	23
FamilyWork	% of workers who work in a family business	24
Carpool	% of workers who carpool to work	25
Transit	% of workers who take public transit to work	26
Construction	% of workers who carpool to work	27
Professional	% of people who are professionals	28

The regression looks to fit the data best between 10—15 features. As expected the training error continued to increase with additional features. However, the validation error actually surpassed the training error when using the accuracy metric on raw incident counts. This was surprising, but since our linear model was trained using the log distribution and not the raw distribution, it is definitely possible.

We also ran the same feature selection algorithm using combinations of features. This technique did not produce any change in our accuracy metrics, and the top 5 features did not change using this approach, so we decided not use any of these combinations in future tests.

The plot below displays where our linear regressor made mistakes. To generate this plot, we used our linear model trained on the full subset of features. We broke the incident counts per capita output into quartile bins and generated a new accuracy by measuring whether the regressor's predicted value was in the same quartile as the actual value. Unlike the accuracies used in the previous plot, this accuracy is more of a true accuracy, representing the percentage of correct classifications. We broke down the accuracies by the quartile of the actual value.



The regressor makes most of it's errors on the lowest quartile. This is because our data is skewed to the right, pulling the mean above the median. While the majority of the counts are closer to 0, in order to minimize the least-squares error the regressor must overestimate these values to stay closer to the mean. This effect was actually somewhat negated by taking the natural log of the output, but even taking the natural log was not enough to fully correct for it, as indicated by the plots. It's possible that using a base 10 log instead of the natural log may have been a better approach.

The overall accuracy for this technique was 40%. This outperforms the zeroR approach, which gives a 25% accuracy (each data point has a 25% of being in each quartile bin).

RESULTS: NEURAL NETWORK

Using the Keras library to build our model, we experimented with the following characteristics (parameters, features, tuning, etc.) in our neural network:

- Scaling: We tried two different scalers from the SciKit-Learn Preprocessing library: StandardScaler (which simply removes the mean and scales to unit variance) and MinMaxScaler with a feature range of -1 to 1.
- Hidden Layers: We tried different numbers and sizes of hidden layers. The numbers of layers we tried ranged from one to four and the sizes (number of neurons per layer) ranged from two to fifteen.
- Activations: We focused on using 'relu' and 'linear' activation functions. Generally, we tried to use the same one in every layer, but in some runs we used a combination of both.
- Epochs: We varied the number of epochs we ran for each model, generally running in the range of 50--200.
- Feature Transformation: We tried runs with the raw values from the data and other runs with transformations of the data. As mentioned above, we transformed the target variable from raw incident counts to log(1 + raw_count). We also divided any raw population counts to percentages of the total population.
- Batch size: We tried batch sizes of 1, 10, and 5.

- Attributes: Although initially we used all of the attributes available to us in the data, we later limited the attribute set to only 15 and then 10 attributes. We selected the limited attribute set using the rankings that we generated using the greedy algorithm during our experimentation with linear regression.

After changing different characteristics across several runs, we noticed that an improvement or a decline in the model's performance as a result of an adjustment to one parameter could be offset completely by an adjustment to another. In other words, an adjustment to one parameter did not necessarily have a consistent effect on the model's performance--the effect of the same adjustment varied depending on the other parameters or features of the model. Generally, though, our findings regarding the characteristics of best performing models are detailed as follows:

- Scaling: StandardScaler
- Hidden Layers: Four to five layers (including the input and output layers) with around the following numbers of neurons (listed from input layer to output layer): N, N/2, N/4, 1 (where N = the number of attributes)
- Activations: All 'linear' was more consistently low with the right parameters. Although several models that used both 'linear' and 'relu' layers performed quite well, the 'relu' models were less predictable (usually performing either very well or very poorly).
- Epochs: The best number of epochs varied greatly from model to model. Generally, though, around 100 epochs gave us a good idea of the model's predictive value.
- Feature Transformation: Target variable transformed to log(1 + raw_count) and all raw population counts transformed to percentages of the total population gave us the best results
- Batch size: 5
- Attributes: Limiting the number of attributes did not seem to have consistently positive results. Varying the number of attributes that we fed the model did not seem to have a large impact on the model's performance.

Our code for our neural network can be found in Appendix D. Below are a selection of mostly well-performing runs, with the associated model details.

# of Layers	Layer Sizes	Activations	Epochs	Attributes	Target	Raw RMSE Accuracy
4	N, N/2, N/4, 1	all 'linear'	150	top 20	log(1 + raw count)	40.72%
5	N, N/2, N/2, N/4, 1	all 'linear'	200	top 20	log(1 + raw count)	42.72%
4	N, N/2, N/4, 1	all 'linear'	100	all	log(1 + raw count)	47.95%
4	N, N/2, N/4, 1	all 'linear'	150	all	raw count	37.0%
5	N, N/2, N/2 N/4, 1	all 'linear'	100	top 15	log(1 + raw count)	45.77%
4	N, N/2, N/4, 1	all 'linear'	50	top 15	log(1 + raw count)	45.21%
5	N, N/2, N/2 N/4, 1	'linear' + 'relu' (last layer)	50	top 10	log(1 + raw count)	41.7%
4	N, N/2, N/4, 1	all 'relu	200	top 15	log(1 + raw count)	46.53%
4	N, N/2, N/4, 1	'relu' + 2 'linear' + 'relu'	200	top 20	log(1 + raw count)	48.52%
5	N, 10, 10, 10, 1	'linear' + 3 'relu' + 'linear'	20	top 15	log(1 + raw count)	56.53%
4	N, N/2, N/4, 1	all 'relu'	100	All attributes	raw count	-3.23%

Overall, our neural network did not seem to have much more predictive value (if any) than our regressor. Moving forward, we would like to explore the KerasRegressor wrapper, which actually uses SciKit-Learn's regressor interface. We believe this might be a good way to integrate the two approaches that we explored separately in our project. Additionally, we would like to incorporate an additional feature into our model: the number of gun licenses issued per county. We found gun license data during our initial search for our dataset and believe it could add predictive value to one of our models.

PROJECT WORK BREAKDOWN

- Mathias focused on data preparation and aggregation, linear regression, and feature selection
- Olivia focused on the neural network

APPENDIX

A: Data Aggregation Code: https://github.com/olcole/GunViolencePredictionUSA/blob/master/dataAggregation.pdf

B: Baseline Accuracy Generation Code: https://github.com/olcole/GunViolencePredictionUSA/blob/master/baseline.py

C: Feature Selection Code: https://github.com/olcole/GunViolencePredictionUSA/blob/master/featureSelection.py

D: Neural Network Code: https://github.com/olcole/GunViolencePredictionUSA/blob/master/neuralNet.py