County-wise Coronavirus Classifications

By Helen X. And Chris F.

CS-UY 4563 Intro to Machine Learning

May 11, 2020

Introduction

Currently, the COVID-19 pandemic is a hot topic for data scientists, and multiple models have been developed to analyze and predict different aspects impacted by the virus, such as the number of projected cases or high-risk demographics using social media data.

This project seeks to use machine learning techniques to create a model that can predict whether a county in the US falls above the death rate median (cumulative ratio of deaths to cases) or not, based on county statistics such as demographics, economics and climate.

Data/Data Processing

For this project, we used four different datasets, which represented two types of data: cumulative COVID-19 cases per county and miscellaneous county statistics.

The COVID-19 data is provided by New York Times and is updated daily; we did some digging online for the other datasets, which proved harder than we initially thought. In the end, we (well, Professor Musco) found U.S. Census data for the demographics, which provided various different statistics such as racial demographics, education level, and household data; North American Land Data Assimilation System (NLDAS) data for climate, which provided average temperatures across counties; and Bureau of Economic Analysis data for economic data (GDP per capita 2015). Each dataset was missing information for some counties, but still managed to produce a merged dataset with over 2500 entries, which was an ample amount to test on. Any counties that were missing data were simply dropped from the dataset, but there were still 2500 perfectly usable data points.

When plotting the data, it was unclear what the trend between death per county and some of the features was as seen in the following graphs of features against death rate:

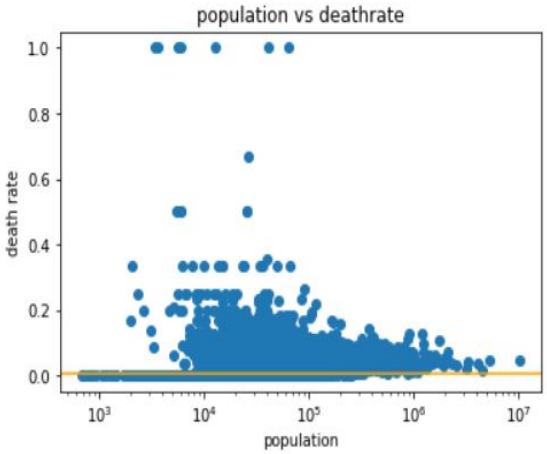


Figure 1: Linearly graphed data

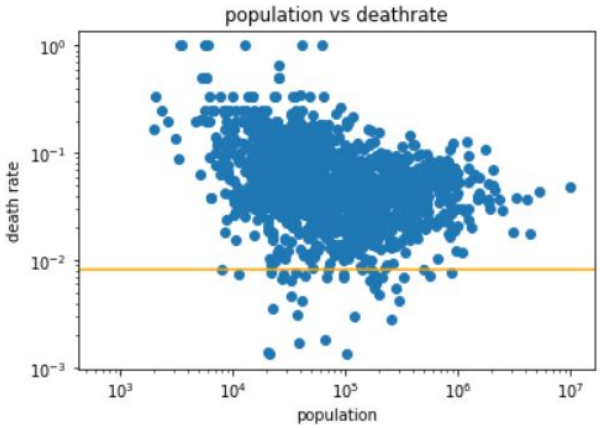


Figure 2: Data graphed logarithmically

Graphing on a logarithmic scale reveals a better and clearer representation of the data, that being said there is a loss of 0 valued death rate. In order to have a better view of all data points we incremented each death rate by a very small value:

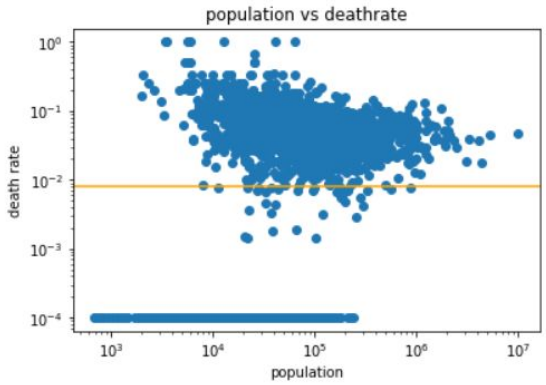


Figure 3: Logarithmically graphed death rate vs population (with incrementation)

Here the large number of counties with a death rate of zero is immediately noticeable, which would explain the very low median (about 0.08).

In order to create a consistent dataset we could easily extract features from, we merged the four datasets together using the county FIPS codes provided in each dataset. Since the NY Times data has data from each day starting from around February, we selected the data from the most recent day (all tests were done on data from April 30, 2020). We then added a column for death rates, which divided the number of cases in a county by the number of deaths.

In addition, due to the difference in how values were scaled (such as the population numbers, GDP and some of the demographics), there was a need to normalize data. To do this the values were mean-centered and divided by standard deviation.

Approach

The original plan was to figure out whether certain features would be correlated to the impact of COVID (measured by death rate) in communities and classify the features (whether or not a feature would correlate well with COVID deaths or not), but we have since changed the goal to classifying counties using said features, since the performance measures would be more concrete that way. We opted to use the median since it would provide a concrete baseline to test against.

In order to create the models, we selected features and then performed a train-test split on the dataset. We then trained on the models, using a mix of self-implemented algorithms and built-ins from Scikit-learn. We used three types of classification models: k-nearest neighbors, naive Bayes, and logistic regression.

As a baseline, we used an all-zeros classifier on the test set. That is, if we predict all values to be below the median, we are guaranteed to get 50% since half of the values always lie

below the median. Our goal is to use the aforementioned data to create a model that performs better than this baseline. That said, we used accuracy as a metric to evaluate the performance of the model.

There are multiple factors that we experimented with to create the most accurate model. The two main factors we tuned were the algorithm used and the number of features. Additionally, as a test we also decided to implement an algorithm to select features based on their individual accuracy in a naive Bayes model, and determine if that would create a better-performing model than our arbitrarily-selected features.

Interestingly enough, all models tested, even with only one feature, yielded an accuracy rate of above 50%, which means there is some sort of correlation in the data that the models were able to use to improve accuracy.

	Population	Population + avg. min temperature	5 features (arbitrarily selected*)	5 features (algorithmically selected**)
K-Nearest Neighbors	0.693145	0.692057	0.684440	0.700762
Naive Bayes	0.627856	0.635473	0.655060	0.620239
Logistic (lbfgs)	0.701850	0.707291	0.748640	0.723613
Logistic (lbfgs, l2 regularized)	0.697497	0.704026	0.745375	0.717084

Figure 4: Accuracies of different models

* Features used: population, avg. min temp, gdp per capita, % black demographic, population above 65. Selected based on what we personally thought would be better predictors

** Features used: % population with bachelor's degree, % high school graduate, mean travel time to work, % housing units in multi-unit structures, 2014 population. Selected based on best individual performance.

However, this table may not be a definitive measure of performances, as percentages tended to jump; the fewer features tested, the more percentages seemed to fluctuate. For example, testing naive Bayes on one feature yielded testing accuracies varying from 60% to 65%.

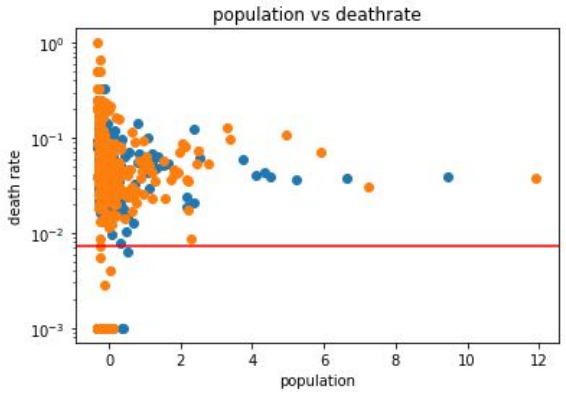
When testing logistic regression, several different algorithms provided by Scikit-learn were tested, and the L-BFGS algorithm seemed to perform the best on our dataset. Using regularization didn't seem to impact the model by much (in fact, it even seemed to perform worse in some trials).

What's also interesting is that the arbitrarily selected features consistently outperformed the algorithmically selected features, meaning that individually well-performing features did not necessarily translate to doing well with other features. That's not to say that an algorithm can't be trusted to select features, however; it just means the algorithm we implemented wasn't very good for selecting features for this particular dataset.

Final Model Performance

The current best-performing model we have, judged solely on accuracy, is the logistic regression model with 5 arbitrarily picked features and no regularization Its final accuracy was 74.8 percent, which is much better than the original 50 percent.

Just to show how well it does, we decided to graph the classifications in relation to features and death rate:



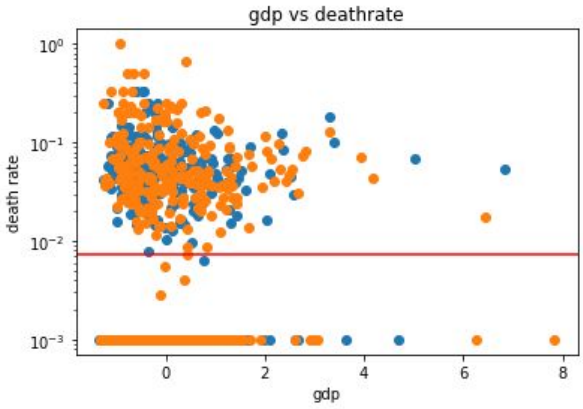


Figure 5: Data classification of Pop VS DR Figure 6: Data classification of Pop VS DR

(orange is 0, blue is 1; for some reason logarithmic graphing wouldn't work for population)

Looking at the graph, it's actually difficult to judge how accurate the model really is, due to the scatterplot looking seemingly random. It's also difficult to judge the accuracy of the model on counties with a death rate of zero, as the points down there are so concentrated that the 0's might just cover up the 1's almost completely.

```
Accuracy on data = 0.748640
Recall: 0.6460176991150443
Precision: 0.8044077134986226
```

Figure 7: Accuracy, recall, and precision of final model.

It's also important to note that the models generally had higher precision than recall, meaning that it produced a lot of relevant results, but it also produced a lot of false negatives.

Conclusion

In the end, we managed to achieve a model that outperformed the baseline by 25%, which was way better than what we expected with such random data with seemingly no correlation to the death rate.

Unfortunately due to time restrictions there were a few additional features/methods that didn't make it into this project. For example, we could potentially try to divide the data into even more classes for more specific classification. The main difference here would have been a middle ground between extremely high death rates and no deaths, since it was shown in the dataset there was an astronomical number of counties with a death rate of zero (although it's uncertain whether that is reflection on the accuracy of the dataset itself or not).

That said, a lot can be done to improve the accuracy of the actual models used; one thing would be to find a larger variety of features to test the model so there's more material than just those three datasets. Additionally more machine learning methods could be used; for example, the use of support vector machines and convolutional neural networks would have been interesting and would hopefully have improved results greatly. Ideally, we could have found more up-to-date data, since most of our county statistics were from around 2010-2015 and may have affected accuracy in some places.

In addition, using cross validation to help with the overall tendency of accuracies to jump around but that wasn't tested again due to time restrictions. Also, the feature selecting algorithm we implemented may have performed poorly in this case, but we believe there are other ways we can implement the algorithm to do better with feature selection.