TO: Chief Administrator of DEA/NFLIS Database
FROM: Team # 1926111
RE: Summary of Opioid Crisis Data Results and Insights

Data from the individual year files was parsed by county and put into a large heat map of the area to see which regions had the most drug reports each year. The maps showed that for almost all of the counties, the ones in or right around major cities had more drug reports on a year to year basis. This finding was unsurprising, but helpful as it meant research could be focused on those counties since they had a larger data pool to work with which could give more accurate results. Data from the NFLIS file was then taken and compared to the yearly files based on the counties that were singled out from the heat maps. These counties data in the yearly files were compared and run through TSFresh and Scikit Learn, two separate Python libraries that house Machine Learning algorithms, to find which socioeconomic data points were most likely to indicate the use of opioids based on a correlation value that ranged from one to negative one, with one being highly correlated and negative one being highly inversely correlated. 14 data points were originally found to have more than a 0.8 correlation value to the drug reports but many had to be dropped as some of the data from the yearly reports were too general to use a quantifying data point, such as total number of foreign born people in the county. While this might mean that foreigners are more likely to be using opioids, because the data went into more detail about everyone's nationality and no individual nationality was as highly correlated the data point was dropped in favor of more relevant and specific data.

These points are gone over completely in the full report but the main data points that had a high positive correlation were counties that had high amount of veterans, a large population that had some non-institutionalized disability, a large population of people under 18 years old that lives with their grandparents, and the data point with the highest correlation was a large population of people under 18 that have their primary caregiver be their grandparents. There were only a few data points that had any negative correlation which were households that had a married couple or households that are of the typical "nuclear" family idea. This is compatible with the other correlated data since a person that only has their grandparents as their caregiver has a high correlation with opioid use. There was also a data feature that ran inversely to the number of drug reports per year. This was the percent of people over the age of 3 enrolled in school; as this number

went down, the number of drug reports went up. The data points that were found to be highly correlated with opioid use are not automatic indicators that any one person in the world will be an opioid user if these things are true for them. This data merely shows what was most commonly found in the specific counties from this specific time range. More research will need to be done on a wider scale to ensure this data is accurate across the country but this report should provide an excellent starting point for any government policies and regulations that might be made to help limit the spread and use of opioids.

Causes for Opioid Overdose

Team # 1926111

January 27, 2019

Abstract

In this paper we explain our methods behind trying to solve Problem C. We first explain how we approached the problem and how responsibilities for each project were delegated. We then discuss our reasoning behind the programming languages we used to parse and understand the big data sets. This section also explains our thought process behind why we chose to shrink the data from close to 600 data features (columns) into something more manageable like 150 data features. We present our visualizations of the data and make observations on possible correlations between drug report data and socioeconomic data. Finally we present our model and suggestion for solving the opioid crisis based on the correlations we found in our visualizations of the data.

Table of Contents

1 Introduction

Every day, 130 people in the United States die from opioids after overdosing and ever since the 1990s, when pharmaceutical companies were convincing people they would not get addicted to opioids, this number as been steadily increasing [3, 6]. In fact, the death toll by opioid overdose today is already 5 times higher than it was in 1999 [5]. Even heroin, an opioid related drug, has seen increase in overdose deaths by 400% between the years of 2010 and 2017 [4].

The heroin and opioid crisis has affected many people from all walks of life and something needs to be done to find a solution to this problem. Data is available for reading and understanding, but finding a way to quantify the data, identify key factors, and introduce new possible solutions is no easy task. The United States government is making good first steps to try to combat this epidemic, but more can be done proactively to stop the exponential rise of opioid overdose. The first step in treating this epidemic is finding out what factors are most likely to lead to heroin and opioid use so that users can be identified and treated in the early stages. Finding these factors can also best help the government start to create programs and policies that can stop such drug use before it begins.

The most common problem with data arises when there is too much data to look through. This becomes overwhelming for an individual person to try to understand. Technology has reached a point where we can program the reading and understanding of the data and finally make conclusions from our observations. In our paper, we attempt to do just that by parsing and cleaning the data in a way that we can visualize and understand, producing a model that will suggest ways to help solve the opioid crisis.

1.1 Assumptions

– We are assuming that age, gender, and race do not have an effect on opioid usage numbers. While these statistics were not provided to start, the Centers for Disease Control also states that the Opioid Crisis affects people of all types of backgrounds, not just specific groups [3]. – The population of a certain area must also be considered when reading the data. An area with a larger population will always have more drug reports than a less populated area. This also applies to the amount of households in each county.

1.2 Key Terms

– **Features** - The columns of the data sheets, these are called "features" when talking in the context of Machine Learning algorithms.
– **Python** - A scripting programming language that has many use cases including Machine Learning, building web applications, and even desktop applications.
– **Pandas** - A Python library that has functions for manipulating and cleaning up large data sets [2].
– **"Plotly" or "Plot.ly"** - an online software tool that provides an API, or application programming interface, that allows for the visualization and analysis of data.
– **TSFresh, Scikit Learn** - Machine Learning libraries in Python that help with making predictions on given data features [1].
– **FIPS** - Federal Information Processing Standard. All counties within the United States have a unique FIPS code that was used to match our drug report data with our socioeconomic data.
– **SQL** - Structured Query Language. A popular programming language for storing and retrieving data.

1.3 Plan of Attack

Data collection and sorting is always a big undertaking, especially when the data sets get very large or broad. The data given was originally separated into two major sets, the first being an overall drug report that had how many opioid related drug reports were made by county in a specific year from 2010 to 2016. The second set was filled with socioeconomic data about

each individual county for the years 2010 to 2016. A great amount of work needed to be done in order to make any observations on all the data given. The Plan of Attack was as follows:

- Split the drug report data into separate data sheets by year for easier viewing
- Decide on what columns from the socioeconomic data sheets were most relevant and helpful for our project
- Merge each county in each drug report year sheet with it's corresponding socioeconomic data features
- Visualize what data we had so far. We originally decided to use a heat map of the section of the United States containing our data.
- Use Python Pandas to produce correlations on, or relationships between, the data features
- Use all of the given data and results to produce a model that would help fight these causes

2 Organizing the Data

2.1 Splitting by Year

In order for the data to be more easily read, we needed to split it up in some way. We decided that splitting the drug report data up by year. This allowed us to perform tasks like calculating the total drug reports for the year for a state much easier. Python Pandas was able to do all of this with just a few lines of code and in a couple of seconds.

2.2 Deciding on Features

As a team, we decided that our algorithms and model would be more accurate if we based each counties socioeconomic data on percentages instead of estimates. We started with all 150 features in our data set.

2.3 Merging

Finally, the 6 resulting data sets representing 2010 through 2016 were merged with the socioeconomic data about each county for each year. This was done with the Pandas.merge() function that mocks a SQL database join. In the end, we were left with 6 data sheets, each representing a year of drug report data for Ohio, Pennsylvania, Kentucky, Virginia, and West Virginia; as well as each count of each state's socioeconomic data matched to it.

3 Visualization

Using Plotly's API, we were able to send our data to their visualization service and receive heat maps for any of the parameters we provided it. Figures 1 and 2 below show the results for some of the data sets.

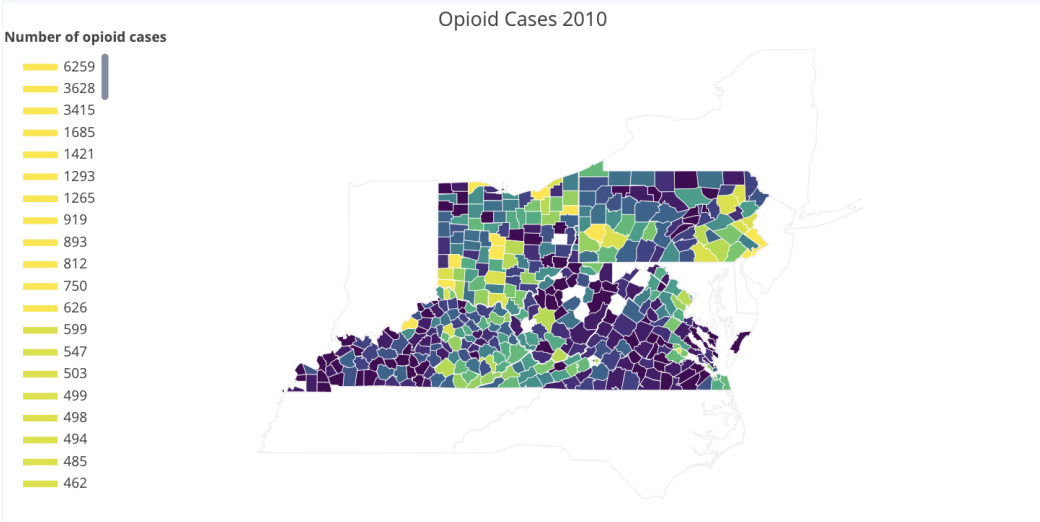

Figure 1: 2010 Drug Report data by county.

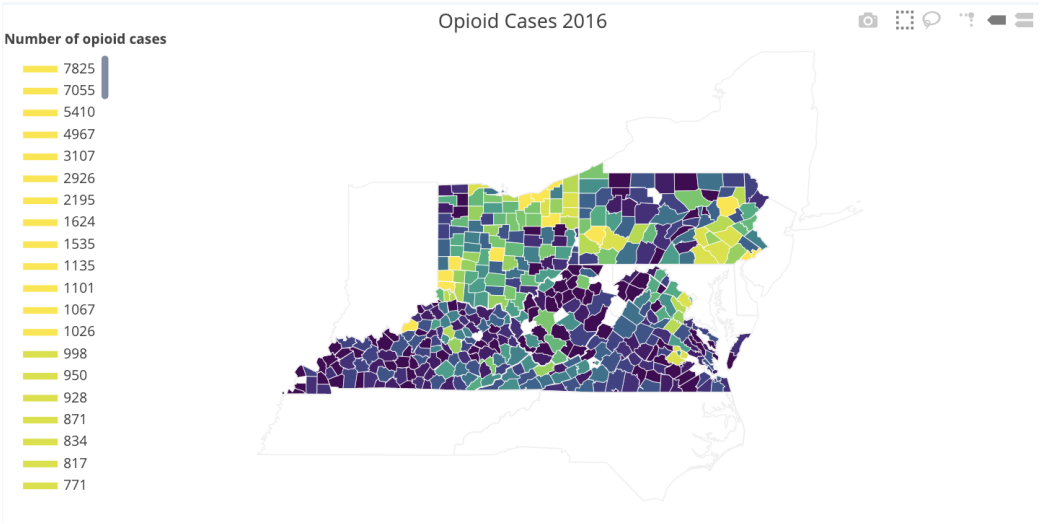

Figure 2: 2016 Drug Report data by county.

These original 7 models provided a great reference for comparison when looking for correlations.

4 Correlations

The goal of correlating data is to be able to easily identify relationships between the features in our data set and the amount of drug reports for a county. We were able to achieve this by using another Pandas function called Pandas.corr() that provided a correlation score between -1 and 1 for each of the features of our data set. With this information, we could now easily pick out which features were closely tied to possibly causing drug reports in each county.

4.1 Explanation

The first models made were heat maps of the counties based solely on the opioids drug reports for each individual year. This was done to give us an estimate of where drug incidents are most often happening as well as baselines for any other data comparisons we will use in the future. Heat maps were then generated based on each different value point of the socioeconomic data

so that each data point could been seen and compared to the total drug report. These heat maps were originally done solely based on the total amount of each data point in the county but we realised this was an error in judgement once a normal map was brought up and it was found that most of the higher data points were centered around more heavily populated areas such as main cities. New heat maps were then generated based on the percent value data instead which turned out to be a lot more helpful and accurate, in that the data was more evenly spread out and changes were able to be properly seen over time.
Once these heat maps were established the drug report data and the socioeconomic data was put together in a correlation algorithm to compare value to see which socioeconomic data points were most correlated to the drug report data. The correlation is done on a simple one to one basis in which the data in the socioeconomic report was compared to the data in the drug report. If both value rose and fell at roughly the same rate then the algorithm would say that the drug report data and the socioeconomic data are correlated. Since this is not a perfect system to determine correlation the data was looked over and discussed to then get a group consensus on whether or not that data point is actually viable as a source of opioid use or if the data points were inconsequential. Once the data had been separated then yearly data was compared to determine which factors had a greater influence over time. The resulting data was then looked at and compared to see which points had the greatest effect on opioid use. The socioeconomic data points that increased or decreased in time with the drug reports are seen as data points that have the highest correlation value while ones that did not follow trends closely with the drug reports were considered to have a smaller correlation. In either case, because unrelated data had already been expunged, all of these data points are considered to have some sort of correlation to opioid use but the ones that have higher correlation should be the focus of research and policies if the government wants to limit or stop the spread of opioid use.

4.2 Most Correlated Features

After the data sets had been correlated against the drug reports for each year, it was time to pull out the features that seemed most related. We came up with the following socioeconomic features:

- Positive Correlations for each County
  - Percent of Veterans over 18
  - Percent of Population that has a disability and is not Institutionalized
  - Percent of Population over 3 years old that is in school
    - Inverse Correlation
  - Percent of Population where Grandparents are living with their Grandchildren
  - Percent of Population where Grandparents are responsible for their Grandchildren
    - Highest Correlation
  - Percent of Population that is 3 years and older and enrolled in school
- Negative Correlations for each County
  - Total Percent of Family Households in the County
  - Total Percent of Females over 15 that are married
  - Total Percent of Households that have a Married Couple

5 Model

5.1 Building the Model

Based off of the data received from the Pandas.corr() models, the correlated data points with a value of 0.8 or greater were taken into consideration for a positive score adjustment, while values that had a value of -0.2 we evaluated to see if they would in fact make a difference in the model. Some of the data points that were in the 0.8 range were deemed to be to broad or unnecessary to the building of the model function and so were excluded from the data model. The data points that gave a negative correlation are considered to be points of int rest that are likely to lower the chance of opioid use but because of their middling correlation score they cannot be said to have any great affect on whether or not opioid use can be stopped or slowed. Some of the data points that were found to have a slight negative correlation were things that seem to be consistent with the rest of the correlation findings, in that families that are married or families that live together in a "nuclear" family dynamic make it so there is a smaller chance of opioid use. These ideas work well with the rest of the data that says children who live with or whose primary care giver is their grandparents will be more likely to use opioids. The family factor cannot help out with another of the big factors in opioids use, which is anyone, regardless of age, that has some form of a disability but are not institutionalized will also be likely to abuse opioids. This idea is consistent with the CDC data in that many people who abuse opioids are people that were originally prescribed them for some form of medical treatment in the first place [3].

5.2 The Proposed Model

The proposed model is for individual counties and is as follows:

$$C_{n+1} = C_n + W_0 * [C_n * \Delta P + W_1 * (C_n * \Delta G_r) + C_n * \Delta V \quad (1)$$
$$+C_n * \Delta D + C_n * \Delta G_l + C_n * \Delta S] - W_2 * [-C_n * \Delta M \quad (2)$$
$$-C_n * \Delta M - C_n * \Delta H - C_n * \Delta R] \quad (3)$$

Where the variables are defined as:

$$n = \text{the year} \quad (4)$$
$$C_{n+1} = \text{total drug reports for next year} \quad (5)$$
$$C_n = \text{amount of drug reports for the current} \quad (6)$$
$$W_0 = \text{Positive correlation weight} \quad (7)$$
$$W_1 = \text{Specific feature weight} \quad (8)$$
$$W_2 = \text{Negative correlation weight} \quad (9)$$
$$M = \text{Marital status} \quad (10)$$
$$H = \text{Married couples} \quad (11)$$
$$R = \text{has spouse} \quad (12)$$
$$P = \text{Total Population} \quad (13)$$
$$G_r = \text{Grandparents responsible for grandchildren under 18} \quad (14)$$
$$G_l = \text{Grandparents living with grandchildren under 18} \quad (15)$$
$$V = \text{Veterans 18+} \quad (16)$$
$$D = \text{Population of people with disabilities} \quad (17)$$
$$S = \text{School enrollment for people over 3 years old} \quad (18)$$

6 Results and Conclusion

6.1 Results

The formula presented in section 5.2 encapsulates the change in correlated features and unique weights. Before we go any further with explaining the model further, it is worthy to note that the model is moderately volatile to the unique weights. That is, when changing the weights value, the models results will moderately change. Due to the time constraint, we were only able to test the model on a limited number of weight variations. The weights given in section 5.2 may not be the best weights for the model, but the accuracy obtained from the given weights resulted in the best readings that we could generate. We tested the model starting at 2012. Accuracy readings were obtained by dividing the actual number of drug reports by the models predicted number of drug reports:

- 2012: 93%
- 2013: 94%
- 2014: 92%
- 2015: 95%
- 2016: 99%

We added a feature weight to the G_r variable since that feature showed more correlation to the opioid cases than any other feature we saw. All correlated features showed very similar correlations to the opioid cases, so adding, changing, or removing the feature weight for the G_r variable may result in a more accurate model. The model predicts that the amount of opioid cases for 2017 for the five states is 89,811, which is a slight decrease from the amount of cases in 2016.

6.2 Conclusion

In summary, this paper details our team's attempt at solving the Opioid Crisis. We started by cleaning the data and looking for correlations between the features and the drug reports data. Once finding the correlations and seeing what actually contributes to the increase of opioid related drug reports

every year, we set out to develop a mathematical model that could predict that. We verified our model by running it through the available data we had and seeing how closely it could predict the values. We had some successes, but not enough to call our model the "Solution to the Opioid Crisis."

The greatest contribution our findings give is the highly correlated features found in the data. These highly correlated features were columns out of the data set that were found to rise, or fall for the inverse correlation, every year along with the drug report data. In general, a person's education level and their home life significantly impacts their likeliness to take opioids and either overdose or be caught. This means that the efforts put in by local law enforcement and government are taking to decrease crime and increase education have twofold benefits. These local bodies should double their efforts in those respective areas because it will provide benefits in multiple ways.

Something can be said about the application of Machine Learning algorithms for this problem. Our team originally ran our data sets through 5 or 6 machine learning algorithms in order to produce possible predictions on the data. After doing this, we realized we couldn't use these models as our own so, we were forced to essentially ignore our findings for that. The Machine Learning algorithms did come up with very accurate predictions for some of the models and we would have been able to predict, accurately, drug report statistics for later years if we were given the socioeconomic data for those years. This is, without a doubt, a problem that can be solved by Machine Learning algorithms and our team's suggestion is for the government to also focus some of their effort in that direction. In context of our model, we believe that, while our it is not 100% reliable, the correlations found are a great starting point to helping curb this epidemic.

References

[1] Documentation of scikit-learn 0.20.2¶.

[2] pandas: powerful python data analysis toolkit¶.

[3] Cdc features, Apr 2018.

[4] Opioid overdose, Dec 2018.

[5] Prescription opioid data, Dec 2018.

[6] National Institute on Drug Abuse. Opioid overdose crisis, Jan 2019.

7 Appendix A

Python for cleaning and merging data sets

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np


mainDF = pd.read_csv('MCM_NFLIS_Data.csv')
year = 2010

# Split the data into sheets by year
for x in range(2010, 2017):
        print "Creating " + str(x) + "_Report_Data..."
        yearDF = mainDF[mainDF['YYYY'] == x]
        yearDF.to_csv(str(x) + "_Report_Data.csv")

# Merge latitude and longitude data by FIPS code
latLongDF = pd.read_csv("latlong_data.csv")
latLongDF['FIPS_Combined'] = latLongDF['FIPS_Combined'].astype(str)

# Merge them based on matching FIPS combined codes
for x in range(10,17):
        print "Reading 20" + str(x) + "_Report_Data..."
        yearDF = pd.read_csv('20'+ str(x) +'_Report_Data.csv')
        print "Reading ACS_" + str(x) + "_5YR..."
```

```
tempDF = pd.read_csv("ACS_" + str(x) + "_5YR_DP02/ACS_" + str(x) +"_5YR_DP02_with_ann.csv")


# Match keys with year sheets
tempDF.rename(columns={'GEO.id2': 'FIPS_Combined'}, inplace=True)

# Account for columns that are off by one in the other sheets
if(x>12):
        tempDF.rename(columns={'HC01_VC21': 'HC01_VC20'}, inplace = True)
        tempDF.rename(columns={'HC01_VC26': 'HC01_VC25'}, inplace = True)
        tempDF.rename(columns={'HC01_VC101': 'HC01_VC99'}, inplace = True)
        tempDF.rename(columns={'HC01_VC109': 'HC01_VC104'}, inplace = True)
        tempDF.rename(columns={'HC01_VC22': 'HC01_VC21'}, inplace = True)

# Make sure everything is a string
yearDF['FIPS_Combined'] = yearDF['FIPS_Combined'].astype(str)
tempDF['FIPS_Combined'] = tempDF['FIPS_Combined'].astype(str)


# Merge specific columns from each sheet and on the shared key
print "Merging for 20" + str(x) + "_Report_Data..."
combinedYearDF = pd.merge(yearDF,tempDF[ \
        ['FIPS_Combined','GEO.display-label','HC01_VC03','HC01_VC07','HC01_VC09','HC01_VC11', \
        'HC01_VC14','HC01_VC20','HC01_VC21', \
        'HC01_VC25','HC01_VC39','HC01_VC40','HC01_VC62','HC01_VC79','HC01_VC80','HC01_VC85', \
        'HC01_VC86','HC01_VC87','HC01_VC88','HC01_VC89','HC01_VC90', \
        'HC01_VC91','HC01_VC99','HC01_VC104','HC01_VC106']], \
```

```
 on='FIPS_Combined', how='left')

print "Merging Lat-Longs..."

combinedYearLatLongDF = pd.merge(combinedYearDF, \
latLongDF[['FIPS_Combined','INTPTLAT','INTPTLONG']], on='FIPS_Combined', how='left')

# drop random extra index column
combinedYearLatLongDF = combinedYearLatLongDF.drop('Unnamed: 0', axis=1)

# Write each to their own data sheet
combinedYearLatLongDF.to_csv("20" + str(x) + "_Report_Data_Merged.csv")
```

8 Appendix B

Function for processing data for Opioid heatmaps

```
posCorrelation = pd.DataFrame()
negCorrelation = pd.DataFrame()
def ProcessOpioidHeatmaps():
    print "Processing opioid data..."
    for x in range(0,7):
        opioidData = pd.read_csv("data/opioid-data/201" + str(x) + "_Report_Data.csv")
        latLong = pd.read_csv("data/latlong_data.csv", low_memory=False)
        heatmapData = pd.merge(opioidData,latLong[['FIPS_Combined','INTPTLAT','INTPTLONG']], \
        on='FIPS_Combined', how='left')
        heatmapData = heatmapData[['FIPS_Combined','DrugReports','INTPTLAT','INTPTLONG']]
        heatmapData.to_csv("data/opioid-heatmap-data/heatmapData201" + str(x) + ".csv",index=False)
        print("Just created the heatmap dataset for data/heatmapData201" + str(x) + ".csv")

        aggregation_functions = {'DrugReports': 'sum', 'FIPS_Combined': 'first', \
        'INTPTLAT': 'first', 'INTPTLONG': 'first'}
        heatmapData_new = heatmapData.groupby(heatmapData['FIPS_Combined']).aggregate(aggregation_functions)
        print "Generating heatmap for opioid data in 201" + str(x) + "..."


        fips = heatmapData_new.FIPS_Combined.tolist()
        values = heatmapData_new.DrugReports.tolist()
        colorscale = [
            'rgb(68.0, 1.0, 84.0)',
```

```
            'rgb(66.0, 64.0, 134.0)',
            'rgb(38.0, 130.0, 142.0)',
            'rgb(63.0, 188.0, 115.0)',
            'rgb(216.0, 226.0, 25.0)'
        ]

        fig = ff.create_choropleth(
                fips=fips,
                values=values,
                legend_title='Number of opioid cases',
                title='Opioid Cases 201' + str(x),
               # colorscale=colorscale,
                county_outline={'color': 'rgb(255,255,255)', 'width': 0.5},
                scope=['OH', 'KY', 'WV', 'VA', 'PA', 'IN', 'TN', 'NC', 'MD', 'DE', 'NJ', 'NY'])

        fig['layout']['legend'].update({'x': 0})
        fig['layout']['annotations'][0].update({'x': -0.12, 'xanchor': 'left'})


        py.plot(fig, filename='math comp ' + str(x))
        print "Heatmap generated."
    print "Done processing opioid data."
```

9 Appendix C

Function for producing Opioid heatmaps

```
def GenerateHeatmap(code1, code2, seData, year, title):

    c1 = 'HC03_VC' + code1
    c2 = 'HC03_VC' + code2

    if year > 2:
        aggregation_functions = {c1: 'sum', 'FIPS_Combined': 'first', 'INTPTLAT': \
        'first', 'INTPTLONG': 'first'}

    else:
        aggregation_functions = {c2: 'sum', 'FIPS_Combined': 'first', 'INTPTLAT': \
        'first', 'INTPTLONG': 'first'}

    heatmapData = seData.groupby(seData['FIPS_Combined']).aggregate(aggregation_functions)
    print "Generating heatmap for " + title + "..."

    if year > 2:
        values = heatmapData[c1].tolist()
    else:
        values = heatmapData[c2].tolist()

    fips = heatmapData.FIPS_Combined.tolist()
```

```
colorscale = [
    'rgb(68.0, 1.0, 84.0)',
    'rgb(66.0, 64.0, 134.0)',
    'rgb(38.0, 130.0, 142.0)',
    'rgb(63.0, 188.0, 115.0)',
    'rgb(216.0, 226.0, 25.0)'
]

fig = ff.create_choropleth(
        fips=fips,
        values=values,
        legend_title='',
        title=title + " 201" + str(year),
        county_outline={'color': 'rgb(0,0,0)', 'width': 0.5},
        scope=['OH', 'KY', 'WV', 'VA', 'PA', 'IN', 'TN', 'NC', 'MD', 'DE', 'NJ', 'NY'])

fig['layout']['legend'].update({'x': 0})
fig['layout']['annotations'][0].update({'x': -0.12, 'xanchor': 'left'})

py.plot(fig, filename='math comp se ' + str(year))
print "Heatmap generated."
```

10 Appendix D

Function for processing data for Opioid heatmaps

```
def MLModels(seData, year):

    seData = seData.sort_values(by=['DrugReports'])

    primaryColumn = 'DrugReports'

    msk = np.random.rand(len(seData)) < 0.8
    train = seData[msk]
    test = seData[~msk]
    x_train = train.drop([primaryColumn], axis=1)
    y_train = train[primaryColumn]
    x_test = test.drop([primaryColumn], axis=1)
    y_test = test[primaryColumn]

    algos = [#{"algo": linear_model.LinearRegression(), "scaled": False, "name": "Linear Regression"},
        #{"algo": linear_model.Ridge(alpha=5.5), "scaled": False, "name":"Ridge Regression"},
        #{"algo": svm.SVR(kernel='rbf'), "scaled": True, "name":"SVR rbf"},
        #{"algo": svm.SVR(kernel='linear'), "scaled": True, "name":"SVR linear"},
        #{"algo": linear_model.Lasso(alpha=0.5), "scaled": True, "name":"Lasso"},
        #{"algo": ensemble.RandomForestRegressor() , "scaled": False, "name":"Random Forest"},
        #{"algo": linear_model.ElasticNet(), "scaled": False, "name":"ElasticNet"},
        #{"algo": DecisionTreeRegressor(max_depth=100), "scaled": False, "name":"Decision Tree"}
        #{"algo": AdaBoostRegressor(DecisionTreeRegressor(max_depth=100),n_estimators=500, \
```

```
    random_state=rng), "scaled": False, "name":"AdaBoost"}
]
results = []
for algo_obj in algos:
    algo = algo_obj["algo"]
    for i in range(10):
        msk = np.random.rand(len(seData)) < 0.85

        train = seData[msk]
        test = seData[~msk]
        x_train = train.drop([primaryColumn], axis=1)
        y_train = train[primaryColumn]
        x_test = test.drop([primaryColumn], axis=1)
        y_test = test[primaryColumn]

        algo.fit(x_train, y_train)
        y_pred = algo.predict(x_test)

        result = []
        variance = explained_variance_score(y_test, y_pred, multioutput='uniform_average')
        mse = mean_squared_error(y_test, y_pred)
        r2 = r2_score(y_test, y_pred)
        result.append(algo_obj["name"])
        result.append(i)
        result.append(r2)
        result.append(mse)
        result.append(variance)
```

```
            results.append(result)

            plt.figure(figsize=(10,4))
            plt.title(algo_obj["name"] + " Sample Train/Test")
            plt.plot(y_pred, color='red', linewidth=1)
            plt.plot(y_test.tolist(), color='blue', linewidth=1)
            plt.savefig("charts/201" + str(year) + "/" + algo_obj["name"]+ str(i) +".png")
            plt.clf()


df = pd.DataFrame(data=results)
df.to_csv("ml-results.csv")
```

11 Appendix E

Functions for building the correlation charts

```
def UpdateCorrelationCharts(seData, year):
    #generate correlations in the data with DrugReports
    seData.corr().to_csv("data/correlations/correlations_201" + str(year) + ".csv")
    corrSEData = seData.corr()

    slightlyCorrelated = corrSEData.loc[corrSEData['DrugReports'] >= 0.5]
    moderatelyCorrelated = corrSEData.loc[corrSEData['DrugReports'] >= 0.7]
    highlyCorrelated = corrSEData.loc[corrSEData['DrugReports'] >= 0.8]

    slightlyCorrelated.to_csv("data/package/slightly_correlations_201" + str(year) + ".csv")
    moderatelyCorrelated.to_csv("data/package/moderately_correlations_201" + str(year) + ".csv")
    highlyCorrelated.to_csv("data/package/highly_correlations_201" + str(year) + ".csv")

    slightlyNegCorrelated = corrSEData.loc[corrSEData['DrugReports'] <= -0.1]
    moderatelyNegCorrelated = corrSEData.loc[corrSEData['DrugReports'] <= -1.5]
    highlyNegCorrelated = corrSEData.loc[corrSEData['DrugReports'] <= -0.2]

    slightlyNegCorrelated.to_csv("data/package/slightly_neg_correlations_201" + str(year) + ".csv")
    moderatelyNegCorrelated.to_csv("data/package/moderately_neg_correlations_201" + str(year) + ".csv")
    highlyNegCorrelated.to_csv("data/package/highly_neg_correlations_201" + str(year) + ".csv")
    global posCorrelation
    global negCorrelation
```

```
    highlyCorrelated['year'] = year
    highlyNegCorrelated['year'] = year
    if len(highlyCorrelated) != 0:
        posCorrelation = posCorrelation.append(highlyCorrelated)

    if len(highlyNegCorrelated) != 0:
        negCorrelation = negCorrelation.append(highlyNegCorrelated)


def CreateCorrelationCharts():
    yearlyValues = []
    yearlynValues = []
    for i in range(0,7):
        #read and cleanse data
        seData = pd.read_csv("data/socio-economical-data/ACS_1" + str(i) + "_5YR_DP02.csv")
        seData.rename(columns={'GEO.id2': 'FIPS_Combined'}, inplace = True)

        #get percentages only
        for x in range (3,219):
            if "HC02_VC0" + str(x) in seData.columns or "HC01_VC0" + str(x) in seData.columns or \
            "HC04_VC0" + str(x) in seData.columns or "HC02_VC" + str(x) in seData.columns or \
            "HC01_VC" + str(x) in seData.columns or "HC04_VC" + str(x) in seData.columns:
                if x > 9:
                    seData = seData.drop(columns="HC02_VC" + str(x))
                    seData = seData.drop(columns="HC01_VC" + str(x))
                    seData = seData.drop(columns="HC04_VC" + str(x))
                else:
```

```
            seData = seData.drop(columns="HC02_VC0" + str(x))
            seData = seData.drop(columns="HC01_VC0" + str(x))

            seData = seData.drop(columns="HC04_VC0" + str(x))


heatmapData = pd.read_csv("data/opioid-heatmap-data/heatmapData201" + str(i) + ".csv")

#engineer the data for heatmap generation
aggregation_functions = {'DrugReports': 'sum', 'FIPS_Combined': 'first', \
'INTPTLAT': 'first', 'INTPTLONG': 'first'}
heatmapData_new = heatmapData.groupby(heatmapData['FIPS_Combined']).aggregate(aggregation_functions)

#merge data set with socio-economic data
seData = pd.merge(seData,heatmapData_new[['DrugReports']], on='FIPS_Combined', how='left')


#handle null/non values
seData = seData.drop(columns="GEO.id")
seData = seData.drop(columns="GEO.display-label")
seData = seData.replace('(X)', 0)
seData = seData.replace('-', 0)
seData = seData.replace('NaN', 0)
seData = seData.replace(np.NaN, 0)


#print seData
```

```
    if i < 3:
        yearlyValues.append(seData[['HC03_VC69','HC03_VC98','HC03_VC103','HC03_VC61','HC03_VC75' \
        ,'HC03_VC128','DrugReports']].mean())
        yearlynValues.append(seData[['HC03_VC44','HC03_VC07','HC03_VC27','HC03_VC128',\
        'DrugReports']].mean())
    else:
        yearlyValues.append(seData[['HC03_VC70','HC03_VC100','HC03_VC105','HC03_VC62','HC03_VC76', \
        'HC03_VC130','DrugReports']].mean())
        yearlynValues.append(seData[['HC03_VC45','HC03_VC06','HC03_VC28','HC03_VC130',\
        'DrugReports']].mean())


#all data is in df
prev = yearlyValues[0]
print yearlyValues
changes = pd.DataFrame()
for current in yearlyValues:
    print "-------------"
    for index in current.index:
        if index in prev:
            change = (current[[index]] - prev[[index]])/current[[index]]
            changes = changes.append([change])
    prev = current

prev = yearlynValues[0]
changes2 = pd.DataFrame()
for current in yearlynValues:
```

```
    print "--------------"
    for index in current.index:
        if index in prev:
            change = (current[[index]] - prev[[index]])/current[[index]]
            changes2 = changes2.append([change])
    prev = current

print "Average changes for each year of positively correlated features:"
print changes.mean()
print "Average changes for each year of negatively correlated features:"
print changes2.mean()

for i in yearlyValues:
    print i
for i in yearlynValues:
    print i
```