Rhythm Kumar, Hunor Vajda, Neha Bangalore, Alaguvalliappan Thiagarajan

MAD2502

**Multifaceted Analysis of Factors Influencing Overdose Death Risk**

The problem we are investigating revolves around the increase in fentanyl-related overdose deaths, on a national level. With a staggering 71,601 drug overdose deaths attributed to synthetic opioids, such as fentanyl, in 2021 nationally (NIH), it is evident that the opioid crisis is a pressing public health issue. The project aims to utilize computational math, specifically machine learning models such as random forests, to uncover the relationships between factors associated with drug overdoses across the nation and fentanyl-doses in particular for the state of Florida. By investigating fatal overdoses, the study seeks to discern key predictors and non-linear relationships that contribute to the occurrence of fentanyl overdoses in counties across the nation.

Our project will entail finding the relationships between provoking/palliative factors in reported overdose death rates. The manner in which we'll be doing this is finding factors relating to overdose rates and correlating them using the Recursive Feature Elimination and Random Forest models to the importance of different factors in our model's prediction of overdose rates at the county level. The relationships/factors that we'll be analyzing are police presence, access to harm prevention/reduction, demographic information of the location, income levels, education, traffic and road accessibility, and the amount/quality of health services relating to fent OD's (number of police, EMS, ED, and ICU locations). We want to create spatial visualization of how each factor affects fentanyl overdose and death rates with our Random Forest model because it is the best model to show non-linear relationships with minimal data. We will be using pandas for working with the data and preparing it for training of a model. The model will be from Sci-kit-Learn and our inputs will be the factor we describe obtaining below and the model output

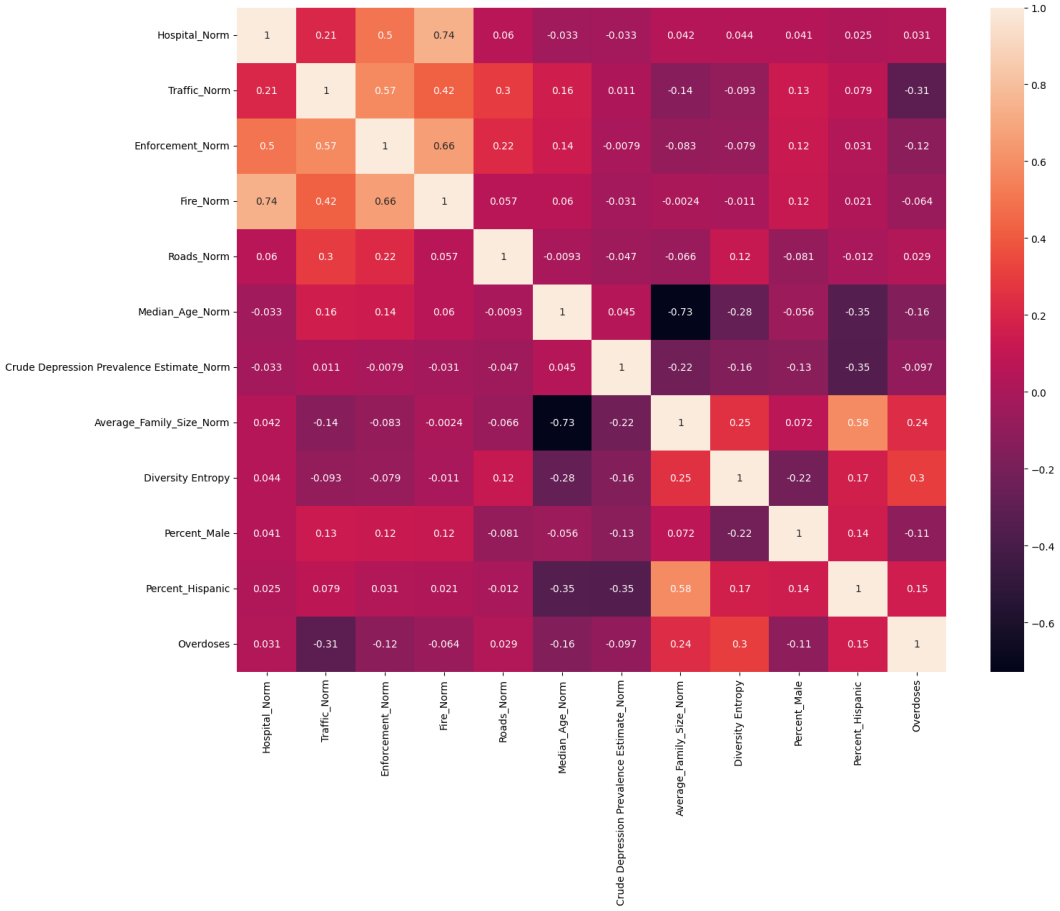
would be the predicted overdose and death rates. Our visualizations will be using matplotlib and ArcGIS so we can plot by county.

In terms of previous work, the group members started working on a similar project with the Gator AI club before the formation of the capstone groups. The AI club helped us source the initial Florida overdose rates through the UF Frost program. However, the work became parallel so we used similar models and visualizations.

To justify our use of Random Forest Regressors, we need to talk about our initial analysis of the data. For our initial analysis, we computed a correlation matrix of all of our data using the pearson method. The formula for the Pearson method correlation calculation is below, where x and y represent two separate features (the correlation matrix computes the correlation between all the features).

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \quad \textbf{(Eq.3)}$$

For the correlation values, values close to -1 or 1 indicate a strong linear relationship between two variables, values close to 0 indicate a non-linear or no relationship. As all the computed values for our target variable overdose deaths were close to 0, it was clear that our features only shared a non-linear relationship with the target if any relationship.




This necessitated the use of a non-linear model. We initially only had Florida county data, data which only had ~50 samples. This made it necessary to work with a model which could handle a small sample size as well as have computable feature importances as the main point of our analysis is highlighting which features affect the overdose death rate. We decided to use a Random Forest regression model which combines predictions from multiple decision trees to make more accurate and robust predictions. “Decision tree algorithms are known for the ability of detecting the features that are important for classification … Inversely: the trees’ capabilities can be used for feature selection” (Grabczewski)

Decision trees work by partitioning feature spaces and associating those spaces into “decisions”, ending with a final label or category. The structure of a tree resembles a binary

tree, where the topmost node is the root, representing the entire dataset. Internal nodes, which branch out from the root, correspond to decisions based on specific features, while the leaf nodes represent the final predictions or target values (Saini). At each internal node, the algorithm evaluates different features and selects the one that optimally separates the data, aiming to maximize homogeneity within resulting subsets. This process continues recursively until a stopping criterion is met, such as reaching a predefined depth or a minimum number of samples in a leaf. For regression tasks, the decisions minimize the variance within each subset to ensure the predicted values are as consistent as possible (Saini). Decision trees have interpretability because they provide a transparent representation of the decision logic, allowing one to trace the path to a leaf and understand the conditions leading to a particular prediction.

A random forest takes the average score from a series of decision trees and the reason it's called a random forest, rather than just a forest is because of the fact that each decision tree is given a random subset of features and trained on a random sample of data points with replacement, making it so that our limited data points can be reused over different decision tree models (Biau).

This makes a random forest far more equipped to handle limited data compared to a deep learning model. However, because of the limited number of sample points, to keep a decent training performance of 86 percent, the model was heavily overfit which is a problem with random forests when there isn't a cap on the max depth of each decision tree or the number of trees (Biau).

In order to get a random forest model that could actually generalize, we got more training data that we'd collect by having the model be trained on each county in the United States rather than each county in Florida. Even after collecting more data, a random forest is still preferred

over a deep learning model because of interpretability and the inherent feature selection. The main goal of this project is to assess what parameters are relevant in determining the rate of drug overdose. This means that the accuracy of a given model is irrelevant if we can't interpret the importance of the features fed into the model. While most popular machine learning models, like deep neural networks, are black box models, random forests are very clear in the classifications and can have the importance of their features evaluated using Gini impurity - which is automatically calculated in Sci-kit-Learn models.

While gini impurity gives us an estimate of the importances of each feature, for the task of feature selection, we wished to use a more refined method. Recursive Feature Elimination is a technique where each combination of features is removed recursively and at each iteration the model performance is tested. The features that preserve accuracy are kept and the features which the model doesn't use or worsen accuracy are dropped (Brownlee). The number of features to eliminate can be a hyper parameter but we chose to make it depend on the model performance. When the number of features to eliminate depends on the model performance, the model keeps iterating until the calculated performance begins decreasing or no longer increases with a smaller subset of features (Brownlee). This ensures that we are not losing any information in our process of selecting features.

For much of our data collection we are working with raw data counts in csv files. To compile and group our data together by counties we use the pandas python which has "fast, flexible, and expressive data structures designed to make working with 'relational' or 'labeled' data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python "(NumFOCUS Inc.). The most common function we used is the groupby function. The groupby function works in three steps. The first step is

splitting up the data by specific combinations of values in a chosen feature. The second step is using an aggregating function such as sum or mean. The final step is regrouping the data using the groups defined in the first step. The regrouped data is outputted as a separate series or extra pandas dataframe (NumFOCUS Inc.).

We want our model to consider information about each county population. To properly understand a population, it is important to understand the distribution of the species in a given location. The simplest way to do this would be to feed the machine learning model the percentages for each race. However, this data would be directly correlated to our target variable (overdose death rates), this means that if mainly white populations are affected by the opioid epidemic, the feature representing the percentage of white population would have a significantly high model importance. This feature would not explain more than what was already known and is not very actionable.

To counteract this while still describing the demographics of each county population, we use a concept that was originally developed for information theory (Konopiński, 2020). Shannon's entropy, first developed in 1948, was created to determine the level of uncertainty of a random event in terms of its distribution. This was quickly adopted to species diversity as treating the diversity as a random event and using the formula " takes into account the proportion of each species in an ecosystem studied; hence, it gives a better description of an ecosystem's diversity than a plain number of species" (Konopiński, 2020). We will use this same concept and apply it to racial diversity.

$$H' = -\sum_{i=1}^{R} p_i \ln p_i$$

Above is the formula for the Shannon-Wiener index, where p_i in our case is the proportion of each population by race. The data we are using for this are the population estimates from the (Census Bureau, 2022) for the most recent year. It should be noted that this diversity score does not measure the percentage of minority population but rather measures a county population's similarity to a uniform distribution.

The population estimate data is in the form of a csv file. The estimates are broken up into age groups, year, whether or not hispanic, gender, and any combinations of races(Census Bureau). To properly handle the counts for the Shannon-Wiener computation, we go through the keys one by one and add up all the variants of the base and store that as a column, then drop the variants.

```
for key in demographics.keys():
    k = key.split("_")
    if len(k) == 2 and k[0] != 'TOT':
        demographics[k[0]] = demographics[k[0] + "_MALE"] + demographics[k[0] + "_FEMALE"]
        demographics.drop([k[0] + "_MALE", k[0] + "_FEMALE"], axis = 1)
```

Above is an example of removing the gender breakdown of each population estimate. Here, we split up the keys of the entire csv file using pandas by the '_' character. Then we add a column of the base key, adding up the male and female variants. Once we used the male and female variants, we removed those keys so as to not iterate over them once more. This process is repeated for each variant type and finally we compute the race percentages and the Shannon Wiener index - known as diversity score for our features by following the formula above.

We seamlessly employed open-source data, referenced below, to procure comprehensive national statistics on the abundance of fire departments, pharmacies, hospitals, and local law enforcement stations. The utilization of the code snippet provided below facilitated the

systematic organization of this diverse dataset, with a particular focus on grouping each variable according to county distinctions.

```python
import pandas as pd

pharmacies = pd.read_csv('PharmaciesSet.csv')

pharmacies.describe()

countycount = ['COUNTY', 'NAME']
countydf = pharmacies[countycount]
countygrp = pharmacies.groupby("COUNTY")
total_groups = countygrp.ngroups
print(f"Total number of counties: {total_groups}\n")

countygrp = pharmacies.groupby("COUNTY").size()
pharmaciesdf = {"County Name": [], "Number of Pharmacies": []}

for name, size in countygrp.items():
    pharmaciesdf["County Name"].append(name)
    pharmaciesdf["Number of Pharmacies"].append(size)
   # print(f"Group {name}: {size} stations")

pharmaciesdf = pd.DataFrame(pharmaciesdf)
print(pharmaciesdf)
```

The provided Python code utilizes the pandas library to analyze and summarize information about pharmacies stored in a CSV file named 'PharmaciesSet.csv'. The first three lines of code import the pandas library and read the CSV file into a DataFrame named 'pharmacies'. The 'describe()' method is then called on this DataFrame to generate descriptive statistics, providing a summary of the numerical data in the dataset, such as mean, standard deviation, minimum, and maximum values.

Following this, the code extracts specific columns, 'COUNTY' and 'NAME', from the 'pharmacies' DataFrame and creates a new DataFrame named 'countydf'. The 'groupby' function is then applied to group the data by the 'COUNTY' column, creating a grouped object called

'countygrp'. The variable 'total_groups' is assigned the number of unique groups or counties present in the dataset using the 'ngroups' attribute. A print statement displays the total number of counties in the dataset. Subsequently, the code performs a further grouping operation on the 'COUNTY' column and calculates the size of each group using the 'size()' method. The results are then stored in a dictionary named 'pharmaciesdf' with keys 'County Name' and 'Number of Pharmacies'. The code iterates through the grouped data, extracting county names and the corresponding number of pharmacies, and appends this information to the 'pharmaciesdf' dictionary. Finally, the code converts the dictionary into a new DataFrame named 'pharmaciesdf' and prints the resulting DataFrame, presenting a clear tabular representation of the number of pharmacies in each county. This code is useful for summarizing and organizing pharmacy data by county, facilitating further analysis and insights into the distribution of pharmacies across different geographical regions. We used a variation of different variables names and CSV files in order to gather the data about local law enforcements, hospitals, and fire departments.

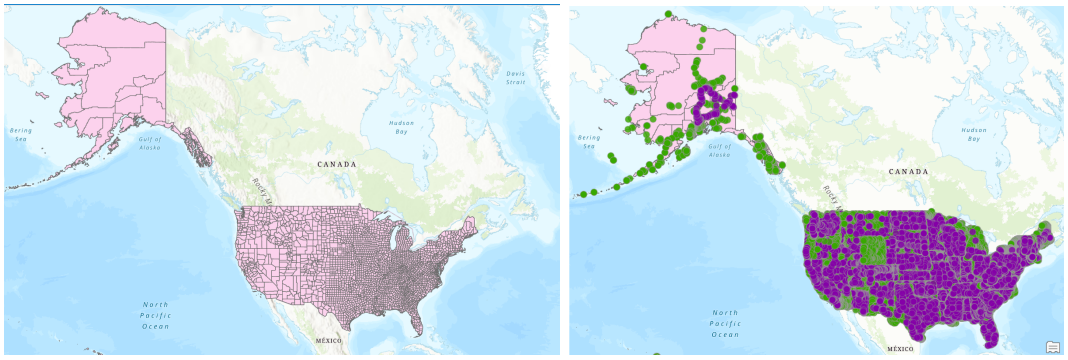
```
[ ]  pharmaciesdf.to_csv('pharmaciesdf.csv', index = False)
```

This line of code was used to use the new data which was gathered and create a new CSV file only concerning the name of the county and the amount in each county.

The data for the number of roads and AADT for Florida came from (ESRI). AADT is the Annual Average Daily Traffic which is the average amount of time drivers lose to traffic per day on a given road.

While each road had a longitude and latitude value, they didn’t have a county as an attribute. Our initial attempt in python was to use geopy to locate the county for a given latitude and longitude. However, it took 6 seconds to get the counties for 10 road nodes. We wondered if

maybe it was just the start up time for the tool that took so long but running the code on 100 roads took 51 seconds implying that the time it would take the tool to locate the county for an additional road would scale linearly. With over 2 million road nodes, this wasn't a feasible strategy so we resorted to using ArcGIS Pro. In ArcGIS we were able to load in both the shapefile for both the roads and the counties in the USA and utilize the spatial join tool to get the county for each road.




At this point, we have many road nodes and AADT points in a csv file labeled with their respective county. To get the total number of roads and traffic per county we use the pandas groupby method and sum up the counts of each by the respective county name. We chose to normalize the number of roads to the area of a given county because the number of roads per square mile would more accurately assess the accessibility of roads. The traffic was normalized to the population because the AADT is calculated for all drivers on the road which would make it such that more drivers would imply a higher AADT whereas the AADT per driver would be a more accurate metric.

Once we are done collecting and managing all of our data, our next step is to perform our analysis. Below is a description of the steps of our analysis.

Our state level county data is formatted so each county has 19 features and our national level county data has 14 features. These features all have the possibility to affect the overdose

rates in these counties. Our goal here was to see which counties have similar feature values and group them together. This could help us determine if those common features contribute to a higher overdose rate.

K-means clustering partitions data into K number of clusters based on the similarity of the feature values. The K-means algorithm in python works by iteratively assigning data points to clusters based on feature similarity and updating cluster centroids until the distance is minimized (NVIDIA, et al). t-Distributed Stochastic Neighbor Embedding (t-SNE) is used for dimensionality reduction in complex datasets. It visualizes high dimensional data in low dimensions (in our case to 2 dimensions) for comprehensive purposes and pattern visualization while keeping local similarities preserved (NVIDIA, et al). Using t-SNE data in a K-Means plot, we were able to condense the multidimensional county data and form visible clusters to understand which features affect the clusters with the highest and lowest overdose rates.

We started this process with the state level county data. Set this data up, we read it as a pandas dataframe into the “data " variable and created a new variable “data_array” to hold a numpy array with the data from that dataframe.

```
data = pd.read_csv("Data_with_Depression.csv").drop(['Pop', 'Unnamed: 0' ], axis=1)

data_array = data.drop(['index', 'Fentanyl_deaths_2022'], axis=1).to_numpy()
```

Then we make a variable called “X” that holds the “data_array” contents and “y” that holds the overdose rates from the original dataframe.

```
X = data_array

y = data['Fentanyl_deaths_2022']
```

The X is what gets fitted and transformed into two dimensional space. Perplexity determines the target number of neighbors for each data grouping for the t-SNE algorithm (NVIDIA, et al), so we set the perplexity to a lower number, 7, as there are not many data points in the state level county database. We also set the number of iterations to be a large number to accommodate for the low data volume. Then, we created a new dataframe with the two resulting dimensions from the t-SNE and the target variable, which is the "y".

```
tsne = TSNE(n_components=2, perplexity=7, n_iter = 10000000000000000000, init = 'random')

X_tsne = tsne.fit_transform(X)

tsne_df = pd.DataFrame(data=X_tsne, columns=['Dimension 1', 'Dimension 2'])
tsne_df['Target'] = y
```

We used that dataframe to create a scatter plot with colors according to the "y" value.

```
plt.figure(figsize=(10, 8))
scatter = plt.scatter(tsne_df['Dimension 1'], tsne_df['Dimension 2'], c=tsne_df['Target'], cmap='viridis')
plt.title('t-SNE Visualization for State Data')
plt.legend(*scatter.legend_elements(), title='Overdose')
plt.show()
```

With this t-SNE dataframe, we can now perform K-means on the t-SNE data to determine the ideal K value and then perform K-means on the original data with that K value.

```
kmeans = KMeans(n_clusters=5, random_state=42)
y_kmeans = kmeans.fit_predict(X_tsne)
plt.scatter(X_tsne[:, 0], X_tsne[:, 1], c=y_kmeans, cmap='viridis')
centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='red', marker='X', s=200, alpha=0.75)
plt.title('k-Means Clustering on t-SNE-transformed Data')
for i, center in enumerate(centers):
    print(f"Cluster {i + 1} Center: {center}")
plt.show()
```

The ideal K value for the state data ends up being six when we display the cluster centers after we run the K-means fit to the t-SNE data. We also want to understand where those center values align. These center values were fitted to the t-SNE data, so now we want to fit the k-Means plot

to the original data because we want to see the groupings in accordance with the original data. Now, we can run the K-means fit to the original data while setting K to six and labeling the clusters in a legend. While doing this process, we ran into an issue with the colors in the plot. The colors for the embedded data points were not matching up to the legend. There was a brown colored cluster while brown did not coordinate to anything on the legend. To attack this issue, we ended up creating our own colormap with unique labels.

```
from matplotlib.colors import ListedColormap
colors_og = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd']
custom_cmap_og = ListedColormap(colors_og)
```

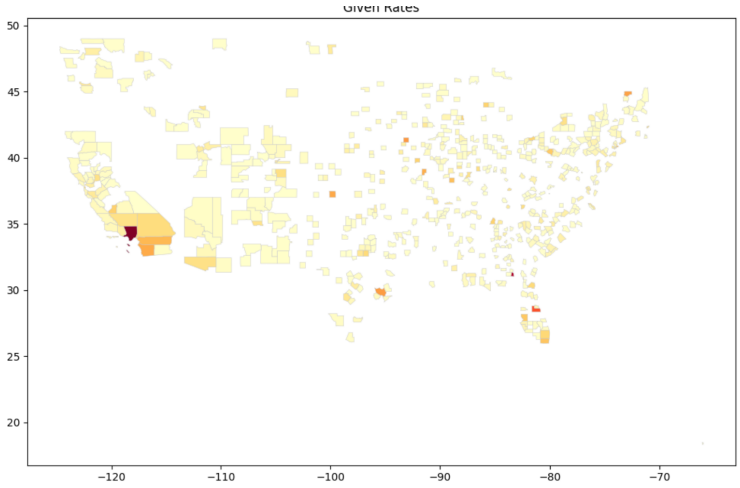
Once we plot the t-SNE data along with the k-means groups and visually determine they are correct, we can determine the characteristics of each K-means cluster by comparing their centers which contain the averages of each feature for each county in the cluster (NVIDIA, et al).

```
kmeans = KMeans(n_clusters=5, random_state=42)
kmeans.fit(X)
labels = kmeans.labels_
unique_labels_og = np.unique(labels)
centroids = kmeans.cluster_centers_
unique_labels_og, counts = np.unique(labels, return_counts = True)
for label, count in zip(unique_labels_og, counts):
    print(f'Cluster {label}: {count} points')
plt.scatter(X_tsne[:, 0], X_tsne[:, 1], c=labels, cmap=custom_cmap_og, s=50, alpha=0.8)
legend_handles_og = [plt.Line2D([0], [0], marker='o', color='w', markerfacecolor=plt.cm.tab10(i), markersize=10) for i in unique_labels_og]
legend_labels_og = [f'Cluster {i}' for i in unique_labels_og]
plt.legend(handles=legend_handles_og, labels=legend_labels_og, title='Clusters')
plt.title('K-means Clustering on Original State Data')
plt.show()
```

We repeated this same process on the national data, while increasing the perplexity to 30 and reducing the iteration count. We were met with a K value of 6, which we implemented into the original national K-means cluster.

We will now use the Random Forest models and Recursive Feature Elimination models we have justified and explained previously. To get the models from SKLearn, we simply had to import the models from the publicly available SKLearn python library. For this analysis, there were too many holes and inconsistencies with the whole US data so we were only able to use the Florida data with the models. To further explain our issues, a large majority of the data we

collected was on a state level, meaning each state generated their data - with their own data conventions and no state labels. This meant that many counties had null values for some of the features at a national level, which would make our Random Forest model incredibly skewed if we were to treat those nulls as 0. This was on top of the fact that any counties that shared a similar name got reduced to a single county when we merged the data as the data had no state labels. Below is a visualization of the given rates for the US data.




Many important counties were lost and this is very clear in this visualization.

Moving back to the Florida data, we instantiate the imported models as a python object with the default initializers. When the RFE model is run, the chosen features are Traffic, Hospitals, Number of Roads, Percent_Hispanic, Diversity Entropy, Median Household Income, Sheriff,  Officers, and Hospitalizations From Depression. Using these chosen features only, we train a final Random Forest model. Once the training is done, we retrieve the feature importances for analysis.

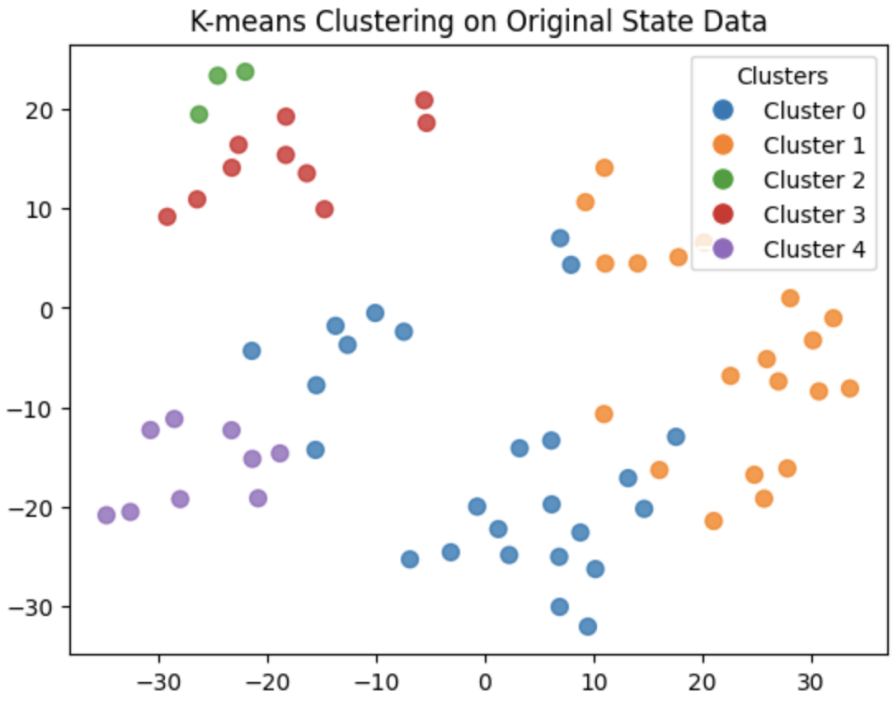
To further analyze the data and utilize our trained model, we decide to use Random Forest's predictive features to predict the data we have with percentage shifts. As we used normalized data (data between 0 and 1) to train our model for each feature, it is very easy to shift

a feature column by a certain percentage (ex: 0.12) and classify that as a 12% shift in our data. We utilize interactive Python features to create a sliding scale which ranges between -0.15 and +0.15. We also create a dropdown to select any of the RFE chosen features to shift.

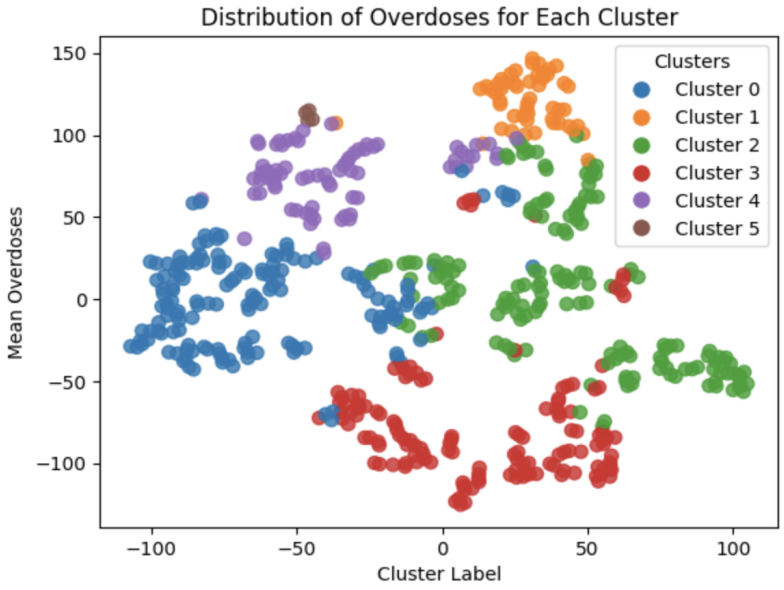
With these predictions, it is important to visualize them. Using geopandas, a pandas-like python library which can handle shape and geographic coordinate data (NumFOCUS Inc.) and a US county or FL county shapefile from (ESRI), we plot the predictions of the model by filling the county area with a color mapped to a specific overdose death rate value gradient.

We will now be talking about the results of our analysis and any conclusions that can be extrapolated from them.

The Florida level K-means resulted in five clusters. Cluster 0 had 25 points, cluster 1 had 20 points, cluster 2 had 3 points, cluster 3 had 10 points, and cluster 4 had 9 points. Cluster 0 had the most counties and cluster 2 had the least.



The national level K-means resulted in six clusters. Cluster 0 had 154 points, cluster 1 had 51 points, cluster 2 had 153 points, cluster 3 had 133 points, and cluster 4 had 72 points. Cluster 0 had the most counties and cluster 1 had the least.



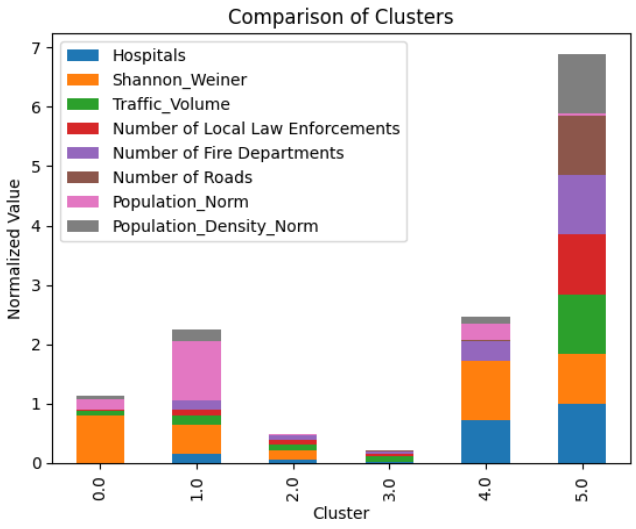

Each cluster represents a group of data points that share similar characteristics based on the features used in the clustering algorithm. The charts show the distribution of these clusters. The characteristics will be analyzed through the two files resulting from these plots.

```
[204…  #create csv with the clusterdata dataframe for analysis
       clusterdata.to_csv("clusterdata.csv", index = False)
```

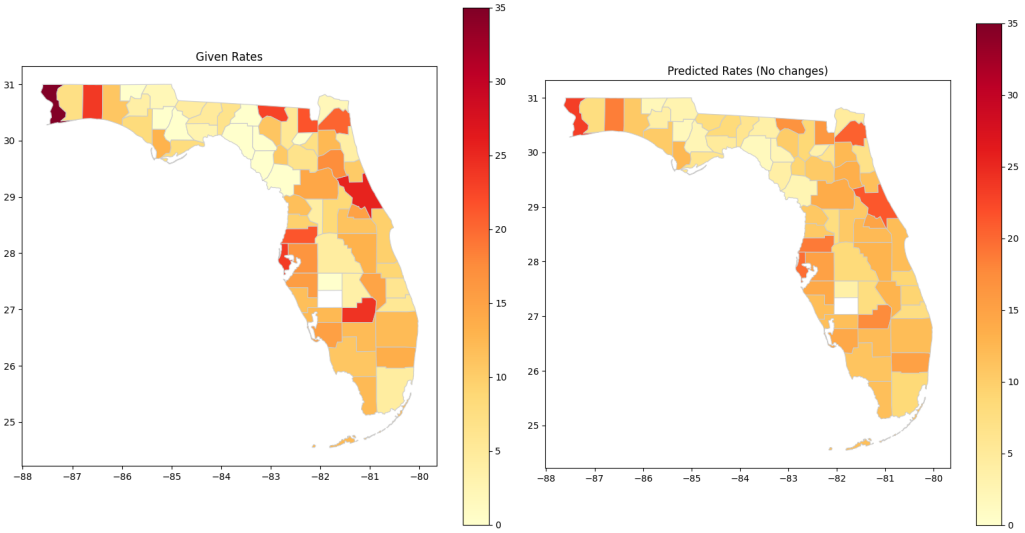
```
[205…  #create csv with new national_data with the y_pred added to it for analysis.
       national_data.to_csv("nationalgyat.csv", index = False)
```

Here we are looking at the 6 clusters created by the K-means algorithm and visualized by the t-SNE algorithm. In the table, we can see the drug overdose per cluster. Both the raw counts from 2020 onward, and the values normalized by population. The bar graph on the right depicts a comparison of a subset of features depicted by the legends. The values in the bar graph are normalized so the scale is readable.

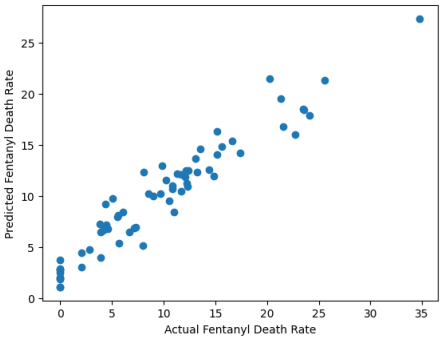
| cluster | Overdoses | Overdoses_Norm |
|---|---|---|
| 0.0 | 4615.175325 | 5.230450 |
| 1.0 | 6392.235294 | 5.944888 |
| 2.0 | 2260.941176 | 3.900423 |
| 3.0 | 2244.278195 | 3.168825 |
| 4.0 | 7603.013889 | 42.407752 |
| 5.0 | 5008.500000 | 7.376519 |



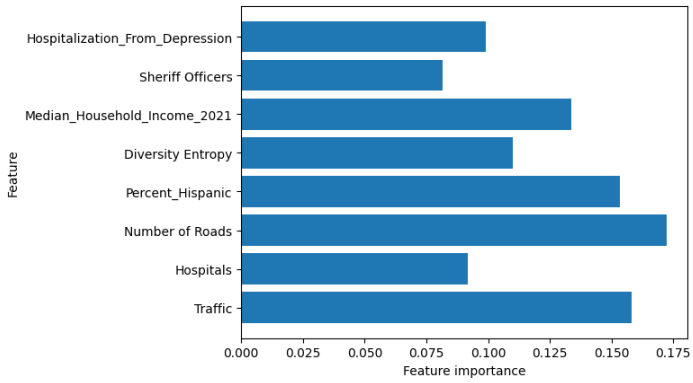

Cluster 5 is the cluster containing outliers, and with New York County as one of its 4 points, it's no surprise it has far higher population densities, number of public services and roads. Excluding the outlier cluster, the trends become a bit more clear. As we can see clusters 2 and 3 by far have the lowest values for drug overdoses and as we can see on the graph on the left, they also have the lowest values for population, population density, and number of services. This implies that they are rural areas and while they are lacking in health care services to help with drug overdoses, the rurality also implies that the cost of living is low with the area likely having a low homeless population, a population highly associated with drug use. In contrast, highly diverse counties with high population densities likely have high drug overdoses because of high cost of living and high crime rates. The war on drugs hits communities of color far harsher than large white communities so it is unsurprising that the cluster with the largest percentage of people of color has the highest values for drug overdoses. It is also interesting to note cluster 4's lack of local law enforcement per capita. While there are arguments for the efficacy of law enforcement agencies, in our case, law enforcement agencies have access to narcan supplies and many agents regularly carry those supplies. Those narcan supplies, explained before, are the key to preventing drug overdose deaths.




The visualizations above depict the Random Forest Regression model when trained with the chosen features from the RFE. This could also be visualized with the scatterplot below, where the x-axis represents the actual input data and the y-axis represents the model output which corresponds to the map visualization on the right. This run of the model had 87% accuracy and we can see an important tendency of machine learning models to point out. That tendency is that models can predict intermediate values very well but fail to predict extreme values well. This is taken into account in our analysis.

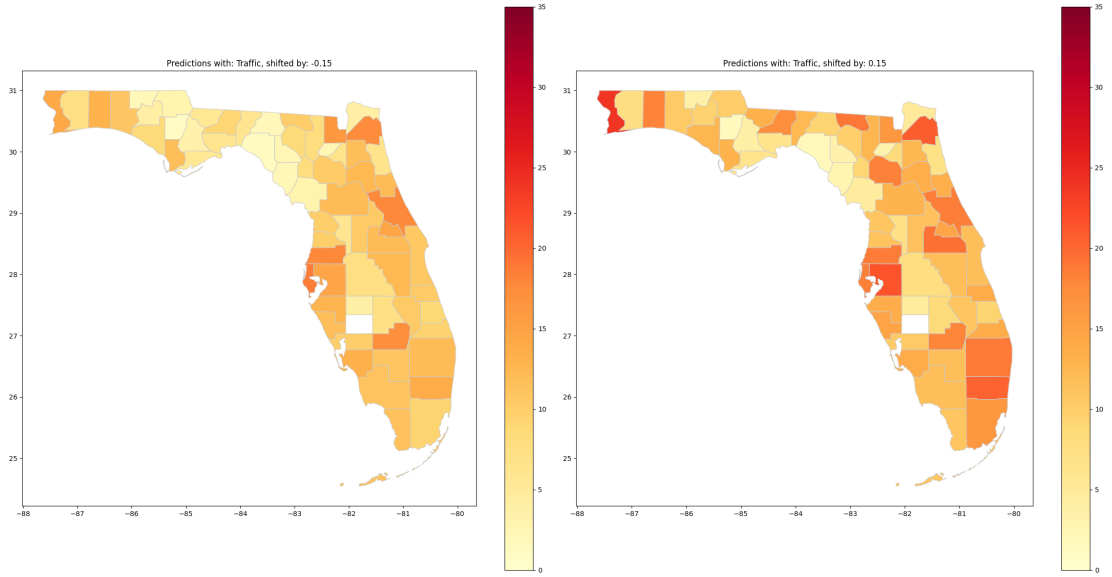



Before we go into the further analysis by shifting, we look into the feature importances of the final trained model below. The calculated feature importances reflect on how much each feature is used by the model for its predictions.




The three most important features for the model were the number of roads per capita, percent hispanic, and the measure of time lost in traffic per capita. This makes a lot of sense as those features are also measures of how dense a county is which correlates to counties with large cities like San Francisco. Large metro areas will always be more diverse than rural ones, will have more road nodes/ intersections per square mile per capita, and will always have high measured traffic. It is important to note, however, that although the three most important features could relate to a simple explanation, the rest of the features were still deemed important by the Recursive Feature Elimination model and the top three features can be viewed in another light.

For example, the measure of traffic is definitely correlated to EMS response times (NIH). To save an individual who is overdosing on a drug like Fentanyl, one must administer a dose of Narcan. Narcan is expensive and generally only carried by emergency response teams or law enforcement (NIH). Depending on the severity of the overdose and the condition of the victim, there is a non-arbitrary time limit on the point in which it is too late to administer narcan. While the sources of narcan like EMS have ways to circumvent traffic, the measure of traffic still affects the response times of those services.




The visualization on the left depicts the model predictions with the traffic feature shifted by -15% while the visualization on the right depicts the model predictions with the traffic feature shifted by +15%. It is important to compare these visualizations together as well as the original data and the non-shifted predictions. The predictions with the largest changes are counties that had an intermediate level of traffic. This most likely has to do with the structure of the decision trees that are aggregated with the Random Forest model. The shifts in the traffic feature switches the ‘decisions’ made by the trees and ultimately changes the tree-path the prediction takes for each tree.

To conclude, our exploratory analysis, while flawed by joining features on county name, a non-unique attribute, using inaccurate and inconsistent CDC data as our target for drug overdoses, and having to aggregate different state conventions into one, provides us with a sense of direction as to how to approach this problem in the future. For example, in our state data, traffic was the most influential parameter but when running the regression on the national dataset it was clear that it was a proxy for population density. Even so, viewing the traffic data in a different light (as a measure for emergency response time) reveals a pattern and a need for better

emergency response services. By adding more parameters we'll be more accurately able to discern the relationships at play. Our current parameter set for the national data was insufficient for the task of regression with poor test results so in the future it would be valuable to both add more parameters and switch to a classification task of high risk, neutral, and low risk rather than predicting the values of overdoses directly. As for the clustering, we were able to see clear distinctions between the average number of overdoses for each cluster with clusters 2 and 3 having far fewer than the rest. By analyzing the attributes of these clusters we illuminated the features that lead to low fentanyl overdose rates but far more analysis needs to be done.

Our multifaceted analysis of factors influencing fentanyl-related overdose deaths on a national scale has provided valuable insights into the complex dynamics surrounding this pressing public health issue. Utilizing computational math, machine learning models, and spatial visualizations, we aimed to uncover the relationships between various factors and fentanyl overdose across counties. The use of Random Forest regression models and Recursive Feature Elimination proved crucial in handling limited and assessing feature importance. Our analysis highlighted the significance of parameters such as the number of roads per capita, percentage of Hispanic population, and traffic measures in predicting overdose rates. K-means clustering further revealed distinct patterns among counties, emphasizing the impact of population density and service accessibility. Despite limitations in data accuracy and the need for further refinement, our study lays the groundwork for future investigations. Moving forward, expanding the parameter set, transitioning to a classification task, and conducting more in-depth analysis of low-risk clusters can enhance our understanding of the underlying factors contributing to fentanyl overdoses. The exploratory analysis provides a valuable direction for future research endeavors in combating the opioid crisis.

Citations:


Biau, Gerard. “Analysis of a Random Forests Model - Journal of Machine Learning Research.” *Journal of Machine Learning Research*, JMLR, Apr. 2012, jmlr.org/papers/volume13/biau12a/biau12a.pdf.

Brownlee, Jason. “Recursive Feature Elimination (RFE) for Feature Selection in Python.” *MachineLearningMastery.Com*, MLM, 27 Aug. 2020, machinelearningmastery.com/rfe-feature-selection-in-python/.

Bureau, US Census. “County Population by Characteristics: 2020-2022.” *Census.Gov*, US Government, 20 June 2023, www.census.gov/data/tables/time-series/demo/popest/2020s-counties-detail.html.

(burnpiro), Kemal Erdem. “T-SNE Clearly Explained.” *Medium*, Towards Data Science, 21 July 2022, towardsdatascience.com/t-sne-clearly-explained-d84c537f53a.

ESRI. “ This Feature Layer Contains 2022 Traffic Counts in the United States.” *Arcgis.Com*, Oct. 2023, www.arcgis.com/home/item.html?id=70507a8779a2470b89c6a8c90394d68e.

Grabczewski, Krzysztof. “Feature Selection with Decision Tree Criterion - Researchgate.” *Conference Papers*, ResearchGate, Dec. 2005, www.researchgate.net/publication/4219423_Feature_selection_with_decision_tree_criterion.

Kelly, Anthony. *A New Composite Measure of Ethnic Diversity*, British Educational Research Association, 10 Oct. 2018, bera-journals.onlinelibrary.wiley.com/doi/full/10.1002/berj.3482.

Konopiński, Maciej K. "Shannon Diversity Index: A Call to Replace the Original Shannon's Formula with Unbiased Estimator in the Population Genetics Studies." *NIH Peer Journal*, U.S. National Library of Medicine, 29 June 2020, www.ncbi.nlm.nih.gov/pmc/articles/PMC7331625/#ref-36.

(LEDU), Education Ecosystem. "Understanding K-Means Clustering in Machine Learning." *Medium*, Towards Data Science, 12 Sept. 2018, towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1.

"Local Law Enforcement Locations." *HIFLD Open Data*, hifld-geoplatform.opendata.arcgis.com/datasets/geoplatform::local-law-enforcement-locations/explore?location=32.128900%2C-90.361607%2C4.26. Accessed 10 Dec. 2023.

NIH. "Drug Overdose Death Rates." *National Institutes of Health*, U.S. Department of Health and Human Services, 25 Sept. 2023, nida.nih.gov/research-topics/trends-statistics/overdose-death-rates#:~:text=Overall%2C%20drug%20overdose%20deaths%20rose,overdose%20deaths%20reported%20in%202021.

NumFOCUS Inc. "Pandas API Reference." *API Reference - Pandas 2.1.4 Documentation*, 2023, pandas.pydata.org/docs/reference/.

*NST-Est2022-ALLDATA: Annual Population Estimates ... - Census.Gov*, www2.census.gov/programs-surveys/popest/technical-documentation/file-layouts/2020-2022/NST-EST2022-ALLDATA.pdf. Accessed 10 Dec. 2023.

NVIDIA, et al. "SKLearn Documentation." *Scikit*, Microsoft, et al ..., Oct. 2023, scikit-learn.org/stable/.

*PQDC*, data.cms.gov/provider-data/topics/hospitals. Accessed 10 Dec. 2023.

Saini, Anshul. "Decision Tree Algorithm - A Complete Guide." *Analytics Vidhya*, 13 Sept. 2023, www.analyticsvidhya.com/blog/2021/08/decision-tree-algorithm/.