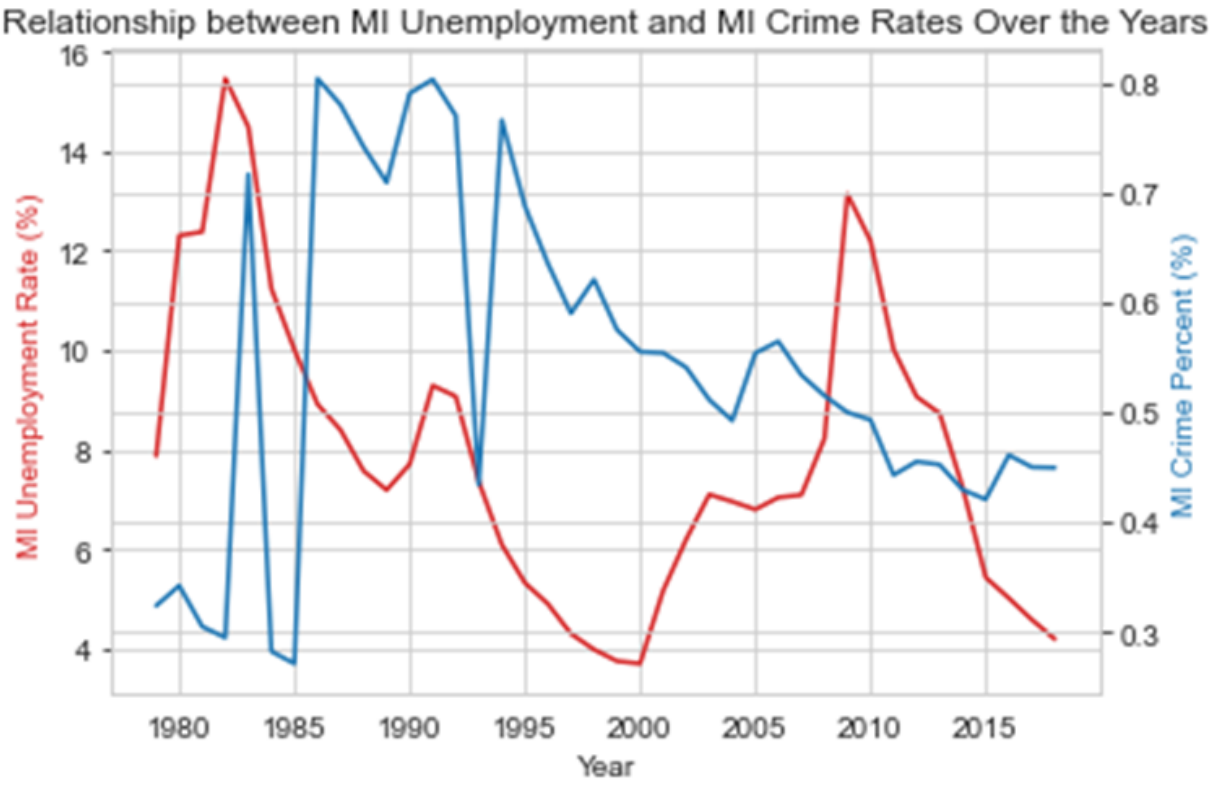
Michigan Crime, Michigan Unemployment, MI Population, US Population Datasets:

The first question we analyzed was looking into how Michigan's unemployment rates affect Michigan's crime rate. We wanted to explore if there is any relationship/association between these two variables. This is an important question because if there is a relationship between Michigan's unemployment and crime rates, this information can be used to develop policies that reduce crime and improve Michigan's economic conditions. For example, if unemployment has a strong relationship with crime rates, then policies around job training programs and targeting reducing unemployment should be created. The results to this question can make it possible to reduce Michigan's crime rates more effectively rather than by simply focusing on Michigan's police force.

To answer this question, we first used data on Michigan's crime rates per 100k people and combined it with data on Michigan's population to obtain the percentage of crime rates in Michigan throughout the years. Afterwards, we combined this data with data on Michigan's unemployment rates from 1979 onwards. Next, we performed a linear regression on the data looking at how Michigan's unemployment rate predicts the state's crime rate. Likewise, we decided to do a simple correlation on these two variables to get further support regarding the strength of their relationship.

Below is a time series plot that shows the results to this question:



Based on the plot above, we can see a very weak relationship between MI unemployment and crime rates over the years. There are times when both lines move in the same direction; for example, in 2005, both lines decreased. We also saw this in 2010 and 2015. However, this relationship is very weak as we see many instances where the lines move in opposite directions, such as in 1980 and 1985. Likewise, our correlation coefficient of -0.251 reveals a weak, negative relationship. There is a slight tendency for when Michigan's unemployment increases, the state's crime rate decreases. Furthermore, our linear regression resulted in a negative coefficient of -0.0130 between Michigan's unemployment rate and crime rate. However, we calculated a p-value of 0.119 resulting in a statistically insignificant model at an alpha level of 5%.

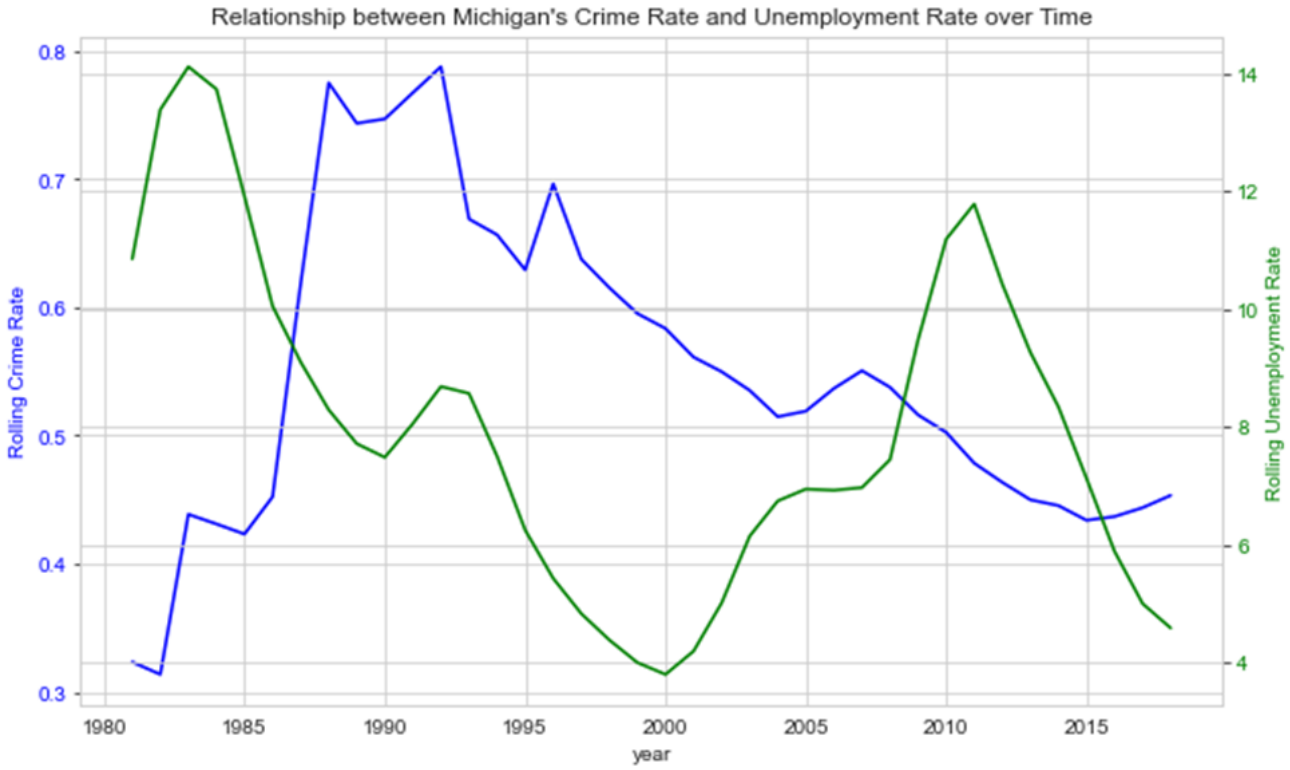
In a research article named Unemployment and Crime by R Tarling, the author writes that "many crimes are committed by employed people and…unemployment is not identified as a powerful determinant of crime"(National Criminal Justice Reference Service, 1983). In addition, the author also states that it appears that crime and unemployment are interrelated in some complex way with other aspects of economic disadvantage"(National Criminal Justice Reference Service, 1983). These findings are related to what we have seen from our analysis as it supports the weak relationship between the variables. We also believe that there are other factors that affect the relationship between these variables such as socioeconomic status and education level

which we would want to dig into deeper. Likewise, it's possible that the relationship between these two variables fit another model better compared to a linear regression. Thus, we would want to further investigate other types of regression models we could perform.

The second question we analyzed was how has the relationship between Michigan's unemployment rate and crime rate changed throughout the years, and are there any cyclical patterns between these rates? This is an important question because if there are trends in how unemployment and crime rates fluctuate, policymakers can use this information to implement programs and crime prevention strategies during the periods where both rates are high. Cyclical patterns suggest that changes in the state's economy and job market are important factors in influencing crime rates and can help forecast periods of high crime and high unemployment rates. On the other hand, if there are no cyclical patterns, it suggests to policymakers that they should consider other factors that contribute to crime and unemployment, such as poverty or inequality.

To answer this question, we decided to perform a time series analysis by taking a rolling mean sample of three years from Michigan's unemployment rate and from Michigan's crime rate. We felt that the rolling mean will help reveal any trends/fluctuations in the data and smooth out noise in the data. We compared the averages for both crime and unemployment rates for each year in the data. Lastly, we did a simple correlation analysis on Michigan's unemployment rate and crime rate to get further evidence on the strength of these variables' relationship.

Below is a time series line plot demonstrating the results of our rolling averages of Michigan's unemployment and crime rates over the years:



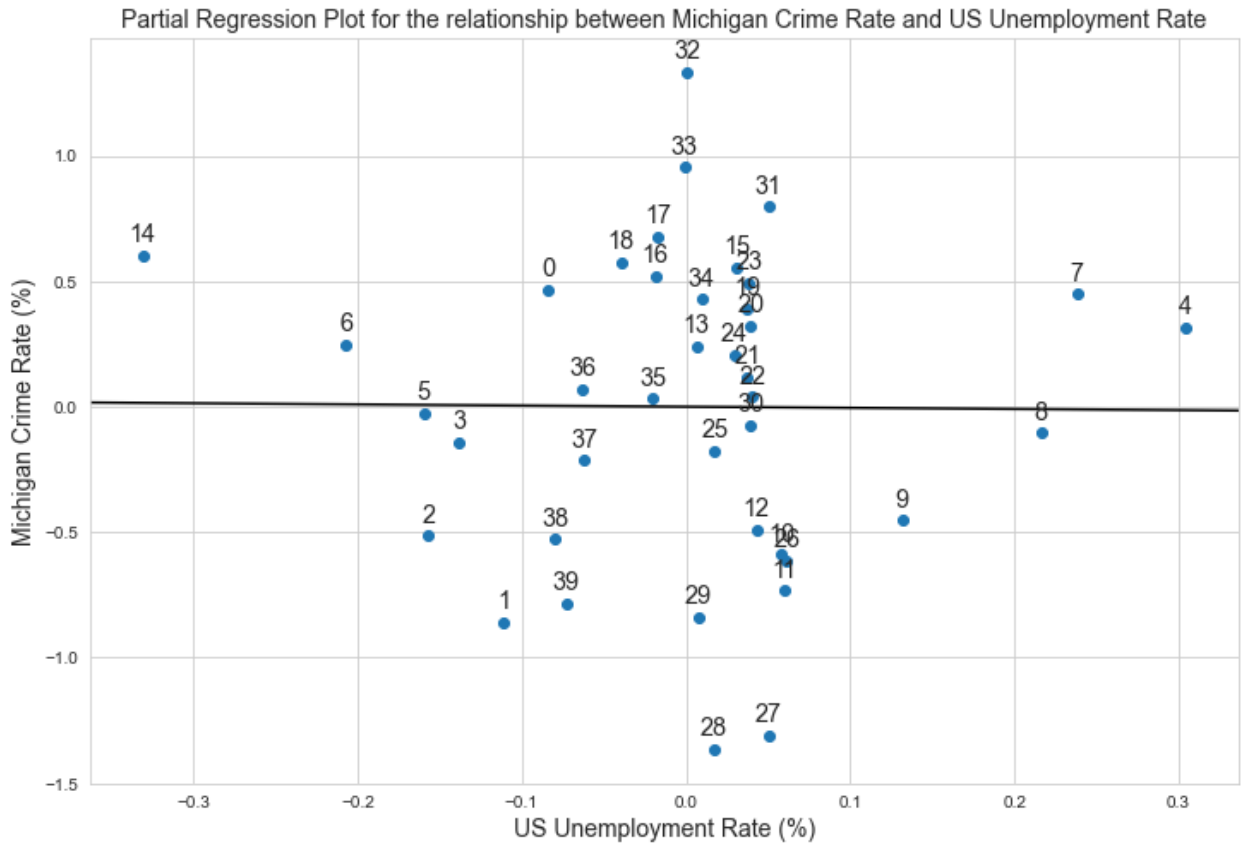
Based on the plot above, there does not seem to be any cyclical patterns or trends between Michigan's crime rate and unemployment rate over time. In the early 1980s, both the rolling crime rate and unemployment rates were increasing with the unemployment rate peaking around 1983 to 1984. Afterwards, unemployment has generally been on a decreasing trend other than around 2009-2010 during the Great Recession. Michigan's crime rates on the other hand have been seen to reach its peak around 1987-1988 (when unemployment was on a decline). There were only a few periods where both rates moved in the same direction such as around 1996-2000; there does not seem to be any strong trends between these two variables. This is further emphasized through our correlation coefficient of -0.392 as this suggests a weak, negative relationship between the variables. Our next step of our analysis would be to possibly look at regional/local trends since Michigan is a large state and crime/unemployment trends in different regions of the state could reveal more specific relationships. Additionally, we would want to example a different time period such as monthly/quarterly to obtain more insight onto the relationship between the variables. By looking at a more granular level, we may see a stronger positive relationship as we originally hypothesized between the variables.

The third question we analyzed was looking into how the United State's unemployment rate affects Michigan's crime rate, while controlling for other variables such as Michigan's unemployment rate, Michigan's population, and US' crime rate. We wanted to explore if there is any relationship/association between these two variables. This is an important question to

explore because by controlling for the other variables, we can see if there is a true effect of US unemployment rate on Michigan's crime rate. This can help Michigan policy makers more effectively reduce crime rates. If there is a strong relationship between these two variables, policymakers can pay more attention to US unemployment rates, and take targeted action based on this variable by introducing more programs to those that lost their jobs--targeting both national and state level problems.

To tackle this question, we used data on US' crime rates per 100k people and combined it with data on US population throughout the years to obtain the percentage of crime rates in the US throughout the years. Afterwards, we combined this data with data on MI unemployment rates from 1979 onwards. Next, we performed a linear regression on the data looking at how the US unemployment rate predicted Michigan's crime rate while controlling for the variables of Michigan's unemployment rate, Michigan's population, and US' crime rate.

Below is a partial regression plot which shows the relationship between Michigan's crime rate and US unemployment rate while controlling for the other variables.



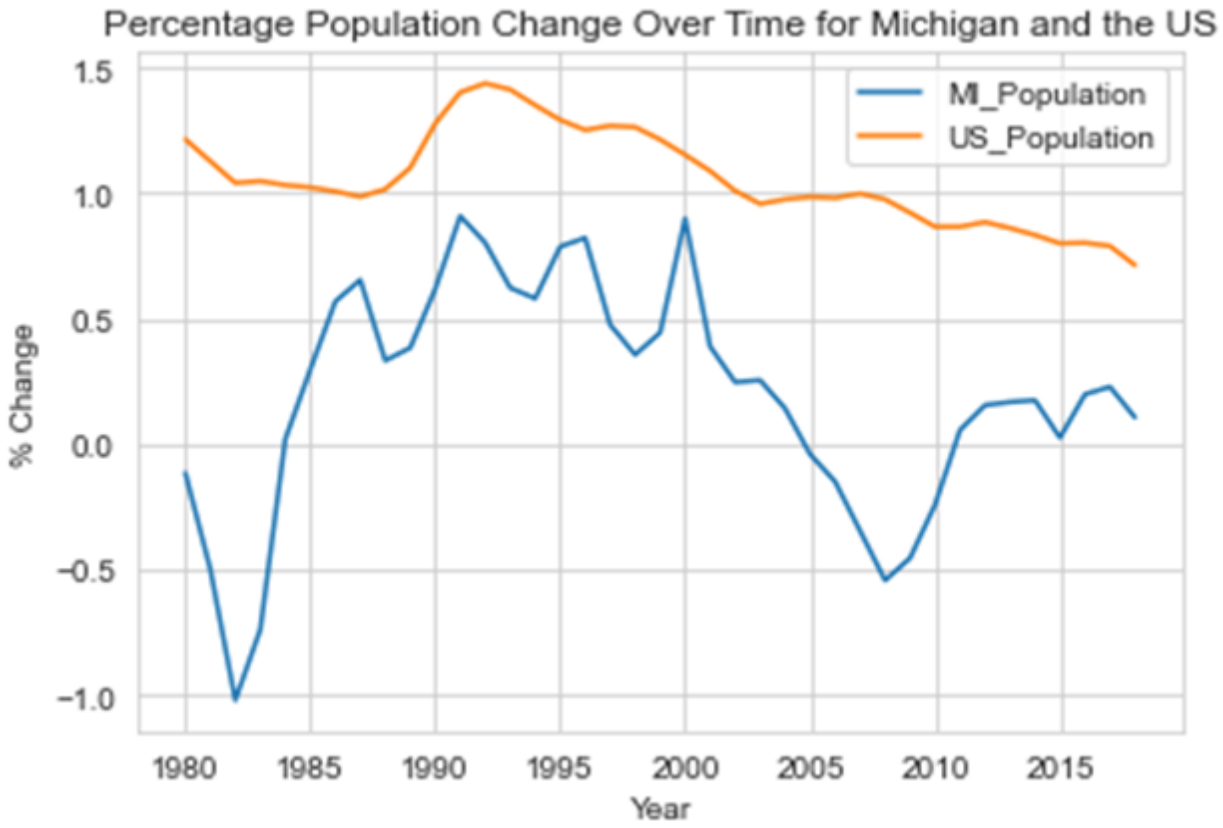
Based on the plot above, we see that the plot is almost completely flat. This suggests that there is little to no relationship between the US unemployment rate and Michigan's crime rate after controlling for the effects of other variables in the model. Moreover, from our regression

analysis, we found a coefficient of -0.0016 for the predictor variable US unemployment rate. This seems counterintuitive to what one may expect since the model is saying an increase in US unemployment rate leads to a very slight decrease in crime rate in Michigan. However, we also found a p-value of 0.960 which reveals that the model is statistically insignificant at an alpha level of 5%. Our next step of our analysis would be to look at the variables at a deeper level such as looking at a specific city in Michigan and analyzing the city's crime and unemployment rates. Likewise, we would want to explore other types of models that better fits the data, such as a polynomial regression.

The fourth question we analyzed was how has Michigan's population changed over the years compared to the US population? This is an important question to answer because if Michigan's population is declining or at a much slower growth rate compared to the US, this may suggest that Michigan is less attractive to potential residents. Likewise, trends in the state's population are important for policies such as in regards to resource allocation and infrastructure planning.

To answer this question, we decided to calculate the average annual percentage change in population for Michigan and the US over the years. To do this, we looked at each year and took a sum of Michigan's and US' population. Afterwards, we found the percentage change of the population trends. Lastly, for ease of comparison, we took the average of every year's percentage population change for both MI and US.

Below is a time series line plot showing how the population of Michigan and the US has changed over the years (1979-2018).

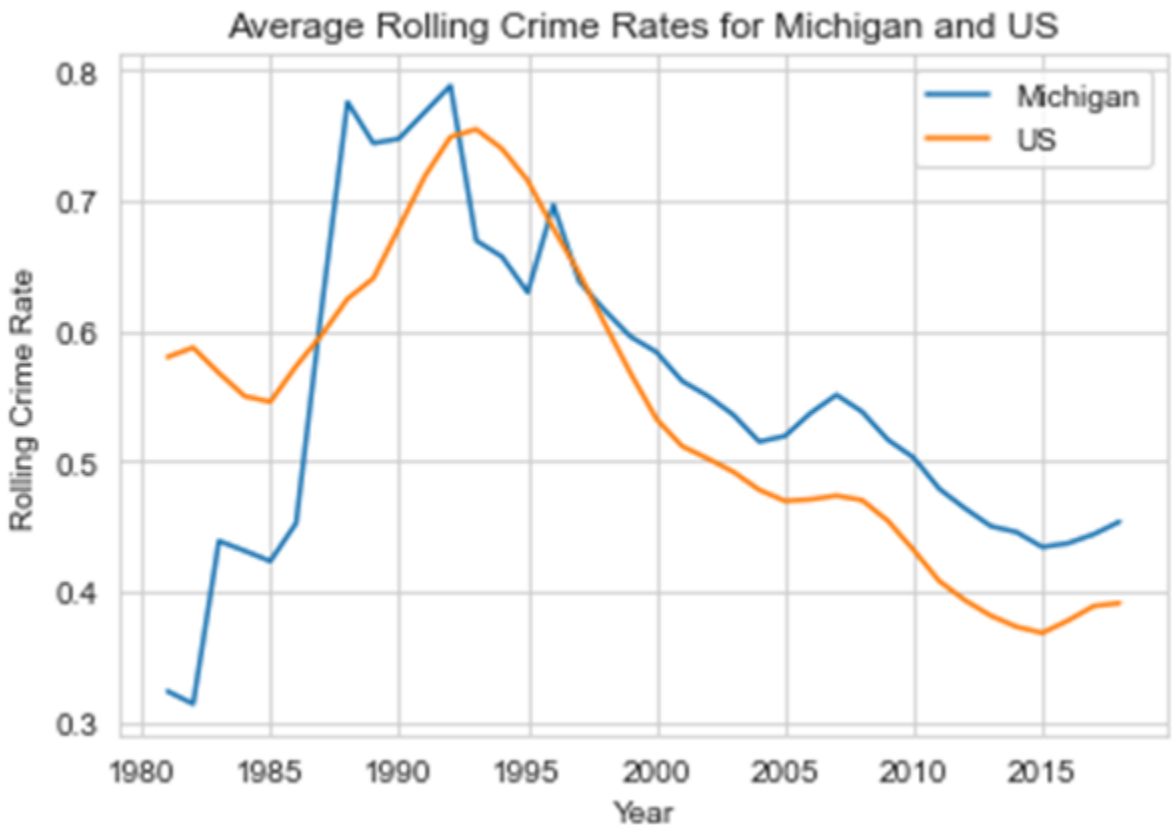


Based on the plot above, we see that Michigan has overall been at a much slower growth rate for its population compared to the United States. There were a few periods where MI's growth rate grew at a faster rate compared to the US such as in the early periods after 1983 where the slope of the line is much steeper than the US, or for a short period of time from 1997-2000. However, we see multiple sharp declines in Michigan's population compared to the US. These declines could suggest that MI is dealing with a lack of job opportunities or other opportunities making it less attractive for residents to be in MI, or the state contains a large aging population compared to other states. Moreover, we found the average annual percentage change for MI's population to be 0.19% and US to be 1.06%, revealing that even though Michigan is growing, it is at a much slower rate compared to the nation.

These results were also similarity found in US 2020 Census Data. In 2021, Michigan Daily released an article describing that although Michignas population is growing, the state " has grown very slowly compared to the rest of the nation" (Michigan Daily, 2021). One stated factor for this slow growth was that about "25% of Michigan's population is older than 60"(Michigan Daily, 2021), leading to a lower birth rate. Thus, we would want to dig deeper into these results to get to a better conclusion by looking at how other state's overall growth rates are, and by how much is Michigan behind. Likewise, we want to explore additional factors that lead

to Michigan's population growth such as the state's updated demographics in 2023 and even immigration levels in the state compared to other states.

The fifth question we analyzed was is there a relationship between Michigan crime rates and US crime rates? Understanding the relationship between Michigan and US crime rates is important for policymakers to effectively reduce crime rates both at the state and national level. If there is a strong, identifiable relationship between these variables, law enforcement can better forecast crime rates at the state and national level. This can help policymakers with implementing solutions to tackle the problem ahead of time before crime rates begin to peak. To answer these questions, our approach was to smooth the data and reduce noise to identify patterns in the crime rates in MI and the US. We did this by calculating the rolling average of crime rates in Michigan and the US over a window of three years. Afterwards, we examined the average rolling means for the US and Michigan for each year. Likewise, we also found the correlation coefficient of both rolling crime rates to obtain further evidence on the strength of the relationship between the two variables.



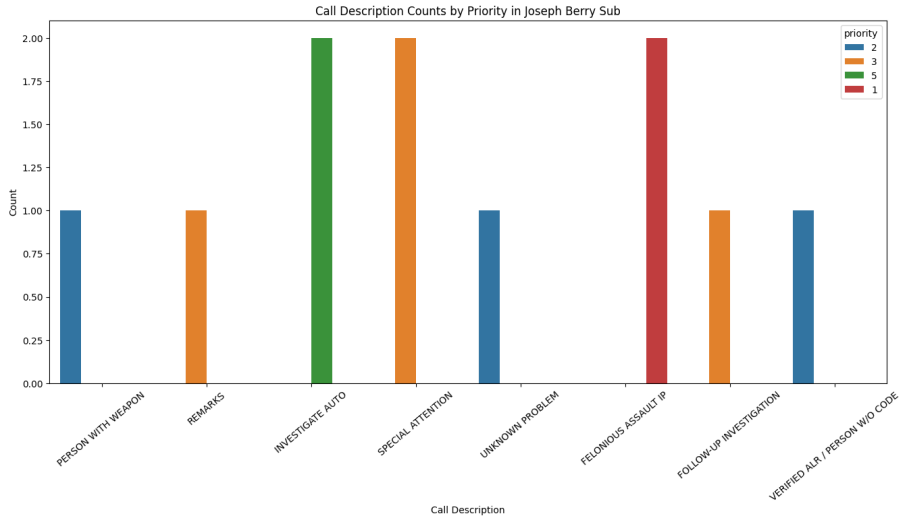
Based on the plot above, we see that the crime rates for both Michigan and the US were generally on an increasing trend from the mid-1980s to around 1993. Afterwards, both crime rates have been on an overall declining trend. From the plot, there seems to be a moderately strong positive relationship between Michigan's and US' rolling crime rates, specifically as we

look from 2000 and onwards. This is further supported by the correlation coefficient we obtained of 0.686. Regarding our next steps, we would want to dig deeper by examining any trends in different types of crime in Michigan and comparing them to the US to see if there are any patterns or differences. Additionally, we would like to analyze the crime rates in different regions of Michigan in order to obtain more insights into the drivers of crime in the state..

Detroit 911 Calls Dataset:

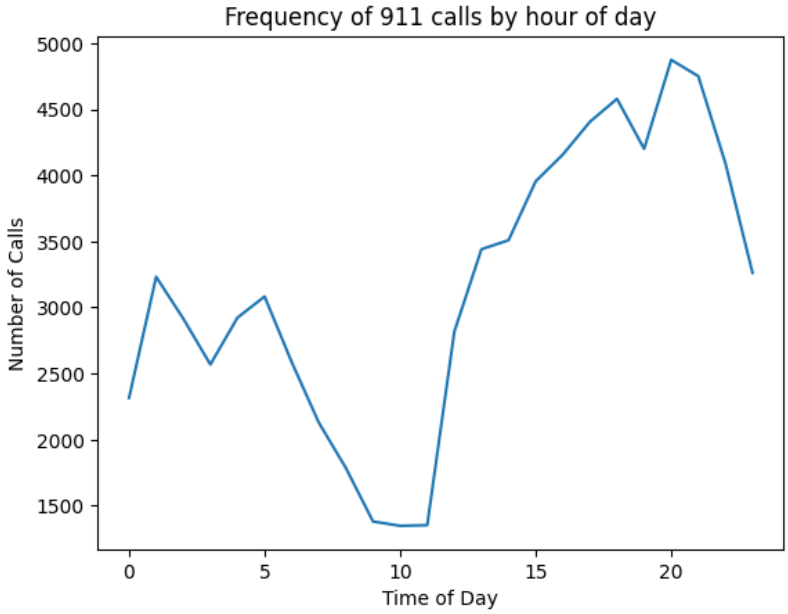
The first question we wanted to explore in this dataset is which neighborhood has the highest mean total time spent by officers. Further, what is the makeup of crimes in that area in terms of call descriptions and priority of calls? This matters because it can give us an idea of why a neighborhood may experience 911 calls that take longer to resolve and if there is a specific call description specific to the neighborhood, we will be able to isolate this description. For community members and law enforcement, this can be helpful to understand when thinking of solutions to potential causes of 911 calls, for example gun violence. Breaking down the types of calls in the highest mean time spent neighborhood could also help us understand the relationship between priority and time spent and ask further questions. We broke this question into two parts. First we grouped the dataset by neighborhoods and performed the average on the "totaltime" column to get the neighborhood with the highest average which was Joseph Berry Sub. Then, we filtered the dataframe to only include instances in this neighborhood and plotted the count of each description of call in the neighborhood by hue of priority.

The plot shows us the breakdown of calls from Joseph Berry Sub. It looks like Felonious assault, car theft, and special attention makeup the most calls. It makes sense that this neighborhood has the largest mean of time spent because its most frequent category of call is a priority one call. It can be assumed that priority one calls require the most time on scene because of their severity. If we wanted to dig deeper, we might look at more data from the city of Detroit to see if Joseph Berry Sub has a pattern of experiencing felonious assault and auto theft and seeing if there is a remedy or intervention that could help this problem. It might also be useful to see the category of call that requires the highest total time for each neighborhood to better contextualize these results.



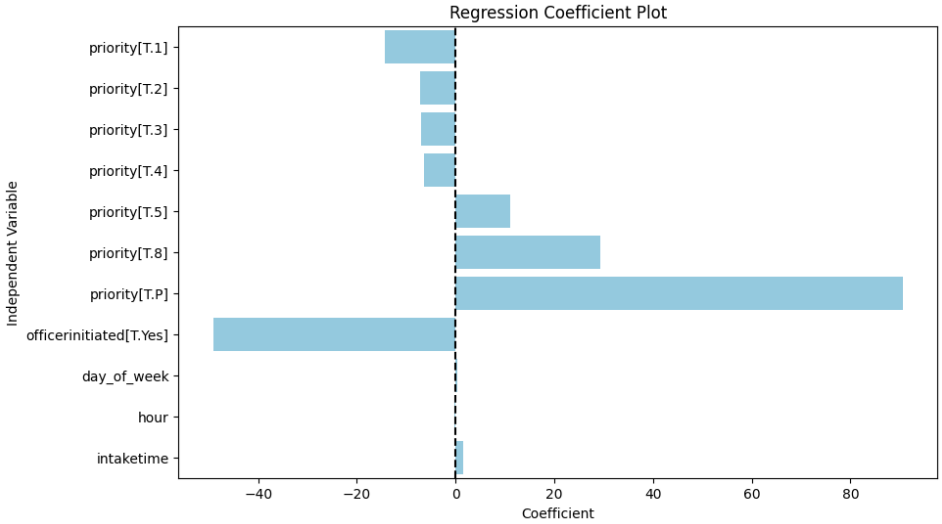
The second question we aimed to answer regarding this dataset was whether there were any patterns in times of day that calls were made over the last month and if there was a way we could visualize the times that calls are made. This question matters because it can help police and first responders to 911 calls be staffed and prepared at the right times of day. If a higher volume of calls come from the hours of 2-3am then we can allocate enough staff and manpower to respond to these issues at those times. Given that law enforcement is tasked with protecting the whole city at all times, it could be helpful to understand time patterns that can inform when they have more officers available to address 911 calls. We approached this question by first figuring out how to manipulate the "call_timestamp" column in the dataset. We had to make this column compatible with the datetime library and then create a new column with just the hour of day the call was reported in. Next, we had to group a new dataframe by this column and perform a count on it. After that we plotted it using a basic line plot that ended up making the answer very clear.

As anticipated, the rate of calls steadily increases at the beginning of the day (12 AM) and then suddenly drops until the evening and climbs until the end of the day. This line plot shows us the count of calls from the last month at each of these hours. Most of the calls, around 5000 of the overall calls are made around 19 hours which is 7pm. This was an interesting result as we assumed many emergencies, like gunfire, theft, and assault, would occur later in the night. The number of calls actually drops around 10-11pm, and slightly increases between 12am-2am. If we wanted to dig deeper, we might plot each priority of call or category of call against this line to see how specific categories and descriptions of calls differ in terms of hour of day they are called in.



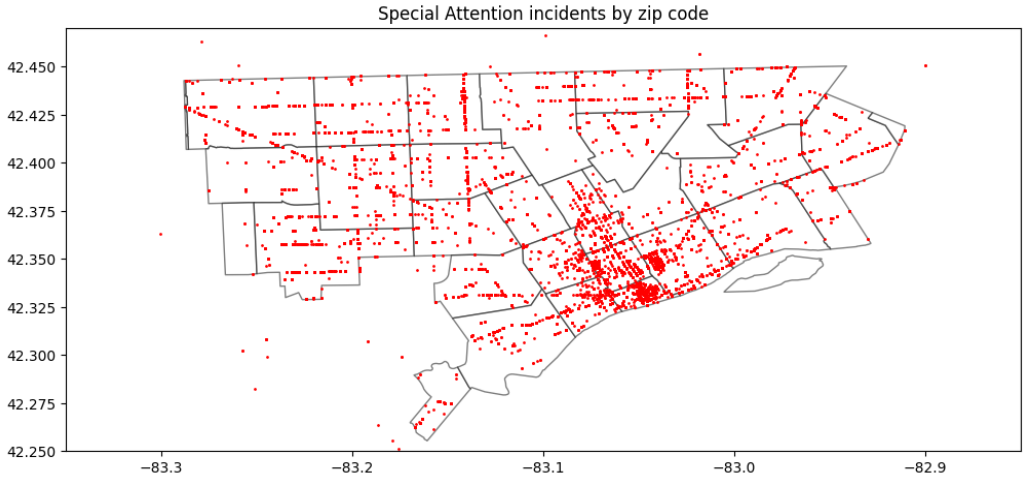
The next question we asked about the 911 calls dataset was which independent variables, specifically day of week, priority of call, hour of day, intake time, and whether an officer was initiated most affected the total time spent on each call. We decided to ask this question because it can be visualized efficiently through linear regression. This question is important because there are many assumptions that can be made about what contributes to a 911 call that takes longer than others. Many are inclined to assume a priority 1 call could contribute to an incident with a higher total time, but there is also reason to believe having an officer initiated could result in more time taken. At the end of the day, it is important to isolate one factor that contributes to longer incidents, so the emergency call center and emergency services dispatched in response to 911 calls can increase efficiency if they are responsible for the factor. For example, if longer intake time directly increased total time relative to other variables, we could assume that those taking the calls are contributing to longer total times. Emergency services are meant to be rapid, so identifying this fault is vital for the city of Detroit. We approached this question by conducting a linear regression among the variables mentioned and analyzed the results by adding the coefficients and p values from those results into their own dataframe and then plotting that new dataframe.

The independent variable with the highest coefficient was “priority[T.P]”. A higher coefficient in a regression model tells us that there is a stronger relationship between the independent variable and the dependent variable. Specifically, a one-unit increase in the independent variable is associated with a larger increase (or decrease, depending on the sign of the coefficient) in the dependent variable. However, it is important to keep in mind that a higher coefficient does not necessarily imply causality or a strong predictive power of the model. The coefficient should always be interpreted in the context of the other variables and the model as a whole. Relative to these other variables, though, it is clear that the priority of P has the biggest impact on the total time taken to address the incident. We are unsure of what a priority of P means in context of this dataset and we reached out to the city of detroit to clarify, but did not hear back from them. If we wanted to dig deeper, we should find out what this category means and also look into why having an officer initiated decreases the total time. We can ask questions such as what kinds of incidents do officers initiate and let it guide further analysis.



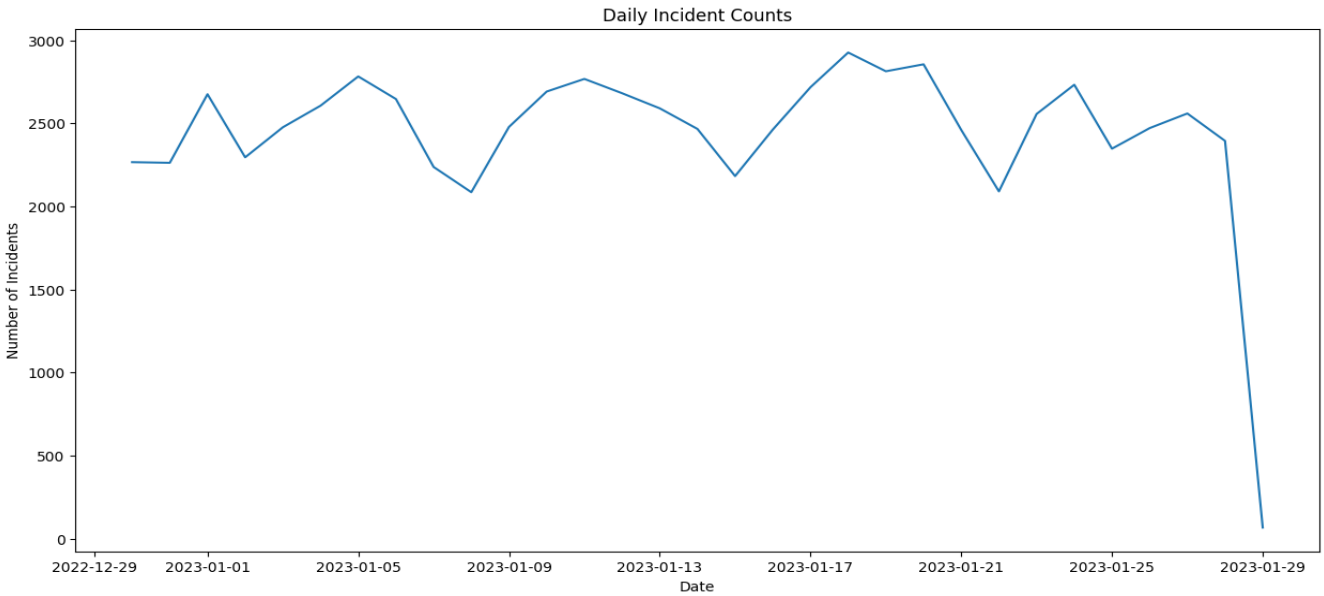
The fourth question we asked about the 911 calls dataset is the most frequent call description and whether there were spatial patterns within that category. We found initially that the highest category of calls is “SPECIAL ATTENTION”, which we were not able to find out a definition for as the DPD did not respond to our email inquiry, however we can assume this category refers to areas that Police believe to be hotspots for 911 calls. It could even represent Project Greenlight businesses that pay the police department to be monitored 24/7 and are given priority one status for all 911 calls. However, a closer look determined all these calls are priority 3, so it is unlikely to be that. To answer the question about patterns within this category however we filtered the dataframe for only call descriptions matching “SPECIAL ATTENTION” then used a zip code tract for the city of detroit to plot each instance of incident.

The plot shows us that many of these calls are clustered in Downtown and Midtown. This is a useful result because it can prove to us where most calls come from. For any category of call, it would make sense that they cluster in downtown as this is a lively part of the city, but it can also result in over policing. For example, if the police pay special attention to these places, they might be adding to the crime instead of deterring it. If we wanted to dig deeper into this question, we need to find out what exactly this category of call means and how police deal with the situation. It would also be interesting to plot police presence against occurrence of 911 calls, including special attention but also more violent crimes to see if there is any relationship between those factors.



The fifth question we investigated was how does the number of total incidents change over the course of the month? We have seen how incident volume changes over the course of a day, but it may be useful to also visualize how the numbers change over the course of the entire dataset. This question matters as we can link a significant spike or dip in numbers to a current event. For example, did a drop in numbers mid month have anything to do with the city wide police initiative to reduce crime? Understanding these patterns can help authorities allocate resources more effectively and proactively respond to potential incidents. Additionally, identifying unusual changes in incident counts can help identify potential issues or emergencies that may require immediate attention. Since we already have a datetime column in our dataset, we solved this question by creating a new dataframe called daily_counts, and used the timestamp column to create a column representing the number of incidents a day. We then used shift to calculate the change in number of incidents from day to day.

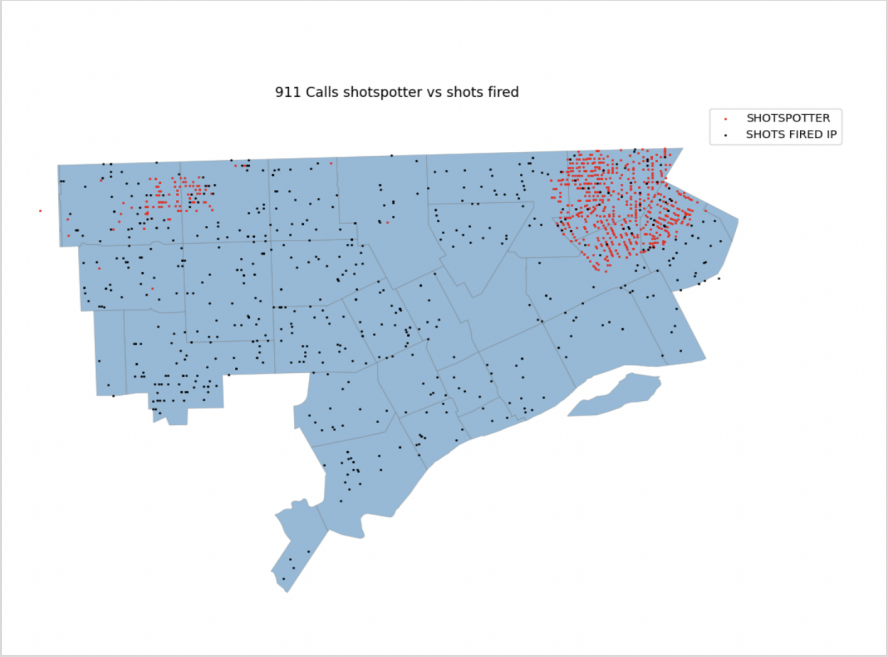
As the line plot shows below, 911 calls are generally steady throughout the month with a spike here and there. The data January 17th, 2023 shows the highest number of calls and deviates from the days prior. We did research regarding this day in Detroit to see if something had occurred or if there was a city wide event with high attendance. We could not find anything significant to explain the results shown. However, had there been a larger outlier, we might have been able to uncover its origins through a visualization like this one. If we wanted to dig deeper into this, we might also track the weather in Detroit throughout this month. If we could compare this line plot to one that represents the weather, for example, we might know if there were especially low temperatures on January 17th that resulted in increased calls. We could also compare the number of calls from our dataset to the number of police reports on each day per month to gauge any correlation over time with that information.



The sixth question we asked was what is the difference between the category of calls titled “SHOTSPOTTER” and “SHOTS FIRED IP”. We noticed that both categories of calls refer to gun violence related calls, however upon further look and research we found that Shotspotter is a device installed in certain neighborhoods in Detroit that senses gunfire and automatically alerts police and emergency services. We wanted to find out if this data can help us derive any conclusions about the effectiveness of Shotspotter. This question matters as Shotspotter is a nationally controversial topic. According to the Detroit Free Press, the “Detroit City Council approved a $7 million contract to expand the controversial audio surveillance technology ShotSpotter into more neighborhoods”. Detroit residents have been vocal about their protest of the devices and question “whether ShotSpotter had sufficient data to prove its efficacy in preventing gun crime” (Afana, 2022).We approached the question initially by filtering our dataframe to isolate call descriptions that are “Shotspotter” or “Shots Fired”. Then, we mapped each instance of the categories on a map of Detroit using Geopandas. Next, we used the map to guide mathematical functions that supplement the map and give us a clearer idea of how the call categories differ.

The plot does not reveal much until a closer look at the numbers. As you can see below, Shotspotter calls are clustered in a few neighborhoods while “Shots Fired” are spaced evenly throughout the city. Doing some simple calculations we find that 857 calls are from Shotspotter while 617 are Shots Fired. Given the distribution of each category, these numbers are alarming as it means 11% of the dataset deals with gunfire from the same cluster of neighborhoods. Further, we can assume that the police spend the same amount of time on scene for all cases of gunfire, but when we averaged the time on scene for both categories, the mean time police spend on the

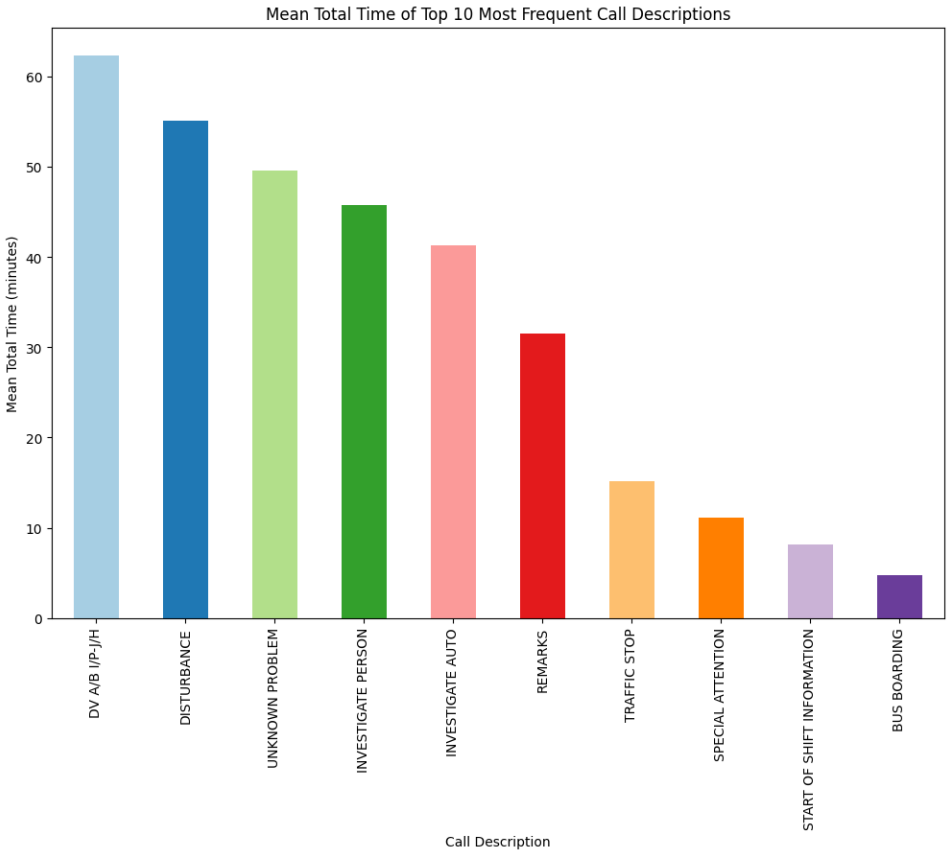
scene for “shots fired” is 1.46x higher than “Shotspotter”. This dramatic difference could prove that Shotspotter is going off more times than it should in these areas and is an ineffective device. If we had more resources and time to prove this hypothesis, we could use 911 data from the last year versus years prior to Shotspotter and track the differences in 911 calls across the city. Further, we could investigate how police report false alarms from Shotspotter and incorporate this data if it exists.



The seventh question we asked about the 911 calls dataset was which categories have the highest total time. We asked about total time since response time has many missing values and would make an inaccurate variable to isolate. Since there are also many call descriptions, we chose to only plot the 10 most frequent descriptions and compare the total time across just these categories. This question matters because it can give the police and public an idea of what kind of calls take the longest and can also show if there is any relationship between the most frequent calls and the total time it takes to resolve the call. If there is a particular category that is frequent and takes the police or emergency services longer on average to resolve, it gives them a weakness to address and a way to potentially make 911 calls smoother for those experiencing difficulty. We approached this question by creating a list of the top 10 most frequent categories and making a dataframe out of incidents that fall into one of those categories. Next, we grouped the data by call description and total time, sorted the data and plotted it using different colors.

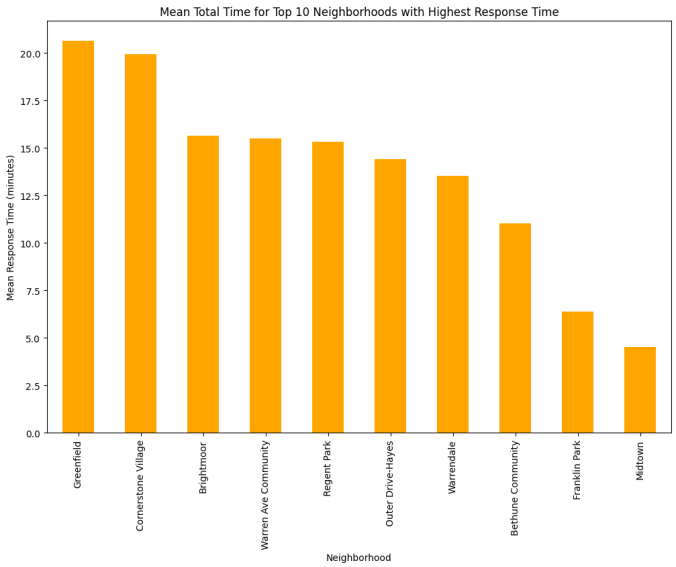
The result is that out of the 10 most frequent categories, DV A/B, Disturbance, and unknown problem take the longest on average. A limitation to our analysis is not having the details about each category, so we are not able to assess how reasonable these results are. However, for anyone familiar with emergency services in Detroit, this breakdown could be helpful in allocating extra resources and operations to the categories that require it most . If we

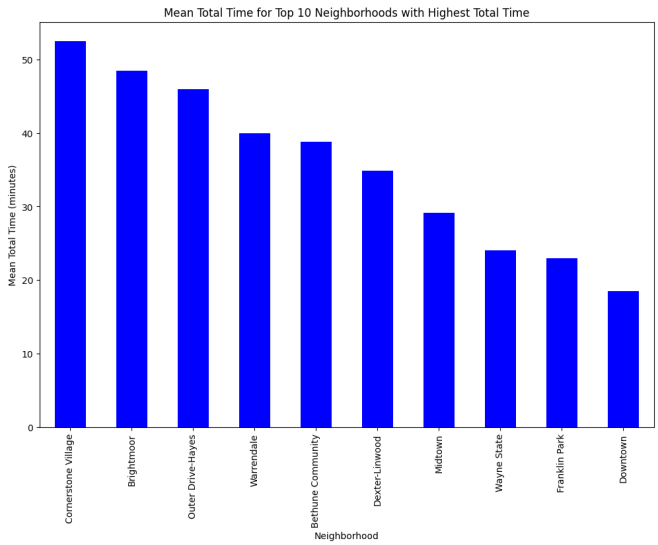
wanted to dig deeper we could do a larger analysis of connection between the total time and the frequency of a call description.We could also consider analyzing the data by different time periods or geographic locations. For example, we could look at the mean total time for each call description by hour of day or day of week, or compare the mean total time across different neighborhoods or police precincts.



Going off the last question, lastly, we would like to find out the neighborhood with the highest response time and total time. We understand that the “totalresponsetime” category within the data has many missing values, but it still provides a starting point to gauging which neighborhoods may inefficiently served by the 911 call response. Knowing which neighborhoods have lower response times can help groups advocate for them and hopefully lead to more resources allotted to these locations. We approached this question by grouping the data by neighborhoods and calculating response times for each neighborhood, sorting neighborhoods by time and selecting the top 10. Then we calculated the mean total time for these neighborhoods and plotted them. We repeated this process for the ‘totaltime’ column so we can compare the two charts.

The results are not as different as we anticipated, Cornerstone Village and Brightmoor have the highest total times and response times. A Deadline Detroit report from 2019 claimed that the slowest response times in Detroit are in poor neighborhoods, “including Warrendale, Brightmoor, Jefferson-Chalmers, Chandler Park and Ravendale” (Fisher 2019) . This claim continues to be proved by our results which contain a few of these neighborhoods. This is very valuable for people who live in these areas to know along with police. Emergency responders can improve their services by figuring out how to reduce response times in these neighborhoods. Investigating the mean total time across neighborhoods is important because it can help identify areas that may require additional resources to improve emergency response times. By identifying neighborhoods with longer response times, emergency services can work to address any potential issues that may be causing delays in response times.To dig deeper into this question, we can further investigate factors that may be contributing to longer response times in certain neighborhoods. For example, we can look at the distance between emergency services and the neighborhoods in question, as well as the overall population density and demographics of the neighborhoods. We can also investigate any potential infrastructure or logistical issues that may be causing delays in emergency response times.





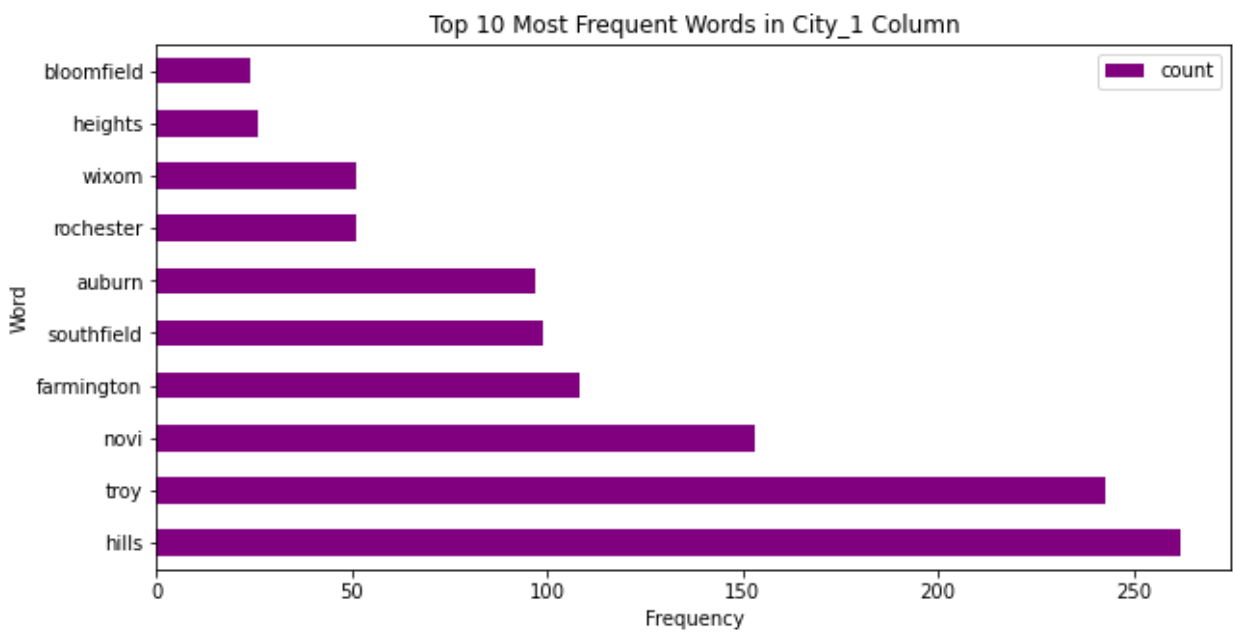
Ann Arbor Planning and Development Projects DataSet

The first question we used to analyze the data given in this dataset was finding the top 10 cities in Michigan with the most foreign owned companies. We wanted to see the trends about what cities are most popular when it comes to foreign owned companies and understand what factors they might consider before picking a city. This question is important to ask because it can help policymakers and economic planners understand the patterns of foreign investment in the United States and identify areas where there may be opportunities for growth and development.

The presence of foreign companies in the US can help to increase the country's global economic influence and strengthen its ties with other countries. Additionally, foreign owned companies can bring significant job opportunities to a city or region. By analyzing the data and identifying the top cities for foreign investment, policymakers and planners can identify potential areas for job creation and work to attract even more foreign investment to those areas.

Our approach to answering this question is performing a text analysis on our data frame to determine the top 10 most frequently occurring words in the column. We converted all the strings in the City_1 column to lowercase and replaced any NaN values with empty strings. Then removed stopwords from each string in the City_1 column using a lambda function and the apply method. We then created a counter object that counts the frequency of each word in the City_1 column by joining all the strings in the column into one long string, splitting the string into words, and then counting the frequency of each word.

Plot:

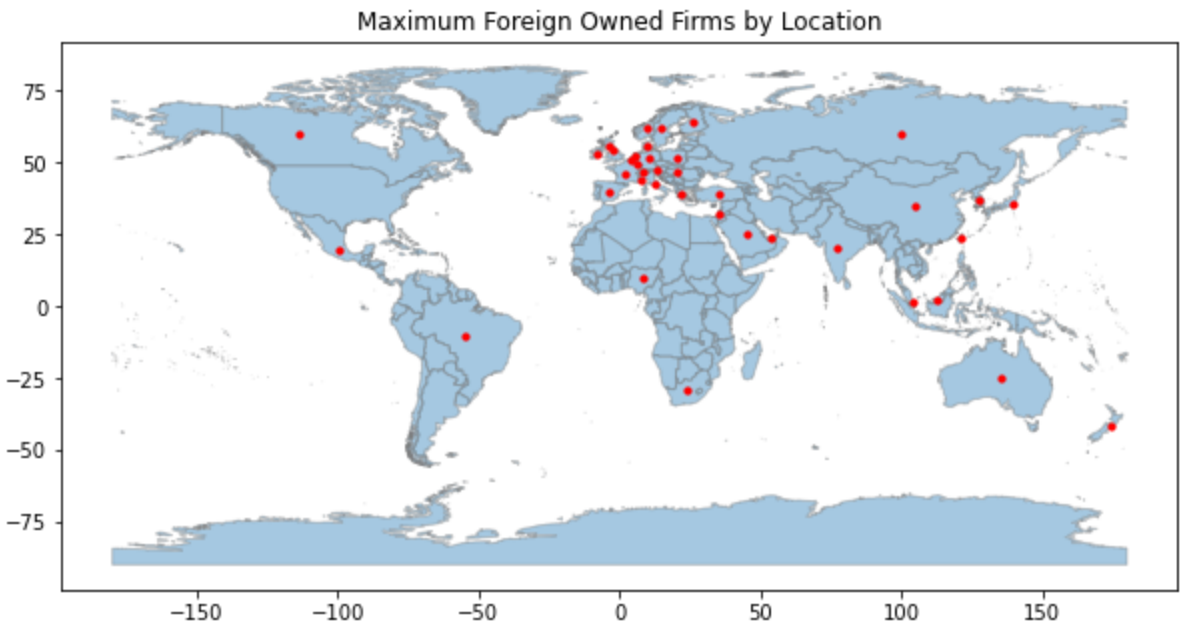


From this plot, we can see that the word that has occurred most frequently is ‘Hills’. But Hills is not a city in Michigan, so this is a cumulative of multiple cities that have the word “Hills” in it. If we go down the list, Troy is the second most frequently occurring word. This tells us that Troy has the most number of foreign owned companies with around 240 companies in the area. We can also see that Bloomfield Hills is the city with the least amount of foreign owned companies. There are about 25 foreign owned companies in Bloomfield Hills which is about 10% of the number of companies in Troy.

If we wanted to dig deeper, we might consider exploring other columns in the dataset to see if there are any correlations or relationships between the top words identified in the City_1 column and other features of the data. Additionally, we might try using different preprocessing techniques or machine learning algorithms to gain further insights into the patterns and trends in the data.

The second question we wanted to think about to further analyze this data was where did these foreign companies originate from. Knowing the countries of origin of foreign-owned companies can provide insights into the specific industries or sectors that are most likely to invest in Michigan. For example, if a large percentage of foreign-owned companies are from countries known for their technology industries, it may suggest that Michigan has a competitive advantage in attracting technology companies. Understanding the factors that attract foreign investment from different countries can also inform future economic development and investment strategies. For example, if a significant number of foreign-owned companies are from a particular region or country, policymakers and economic planners can develop targeted strategies to attract more investment from that region or country.

We wanted to answer this question by conducting a spatial analysis of the foreign-owned companies using sjoin. We first grouped the data by X and Y coordinates and found the maximum value of foreign-owned firms in each group. We then performed a spatial join to match the countries in the shapefile to the countries of origin in the company data. The result is a GeoDataFrame that includes the maximum value of foreign-owned firms for each point. Finally, the code plots the points on a world map, with the size of each point corresponding to the number of foreign-owned firms in that location.



This GeoDataFrame provides insight into the countries of origin for foreign-owned firms in Michigan. The data indicates that a majority of these firms are from Europe, while there are relatively fewer firms from the Americas. However, the data also shows a significant number of firms from Australia and Asia, indicating a diverse range of origin countries. In summary, foreign-owned firms in Michigan come from various regions, but the concentration is mostly from Europe.

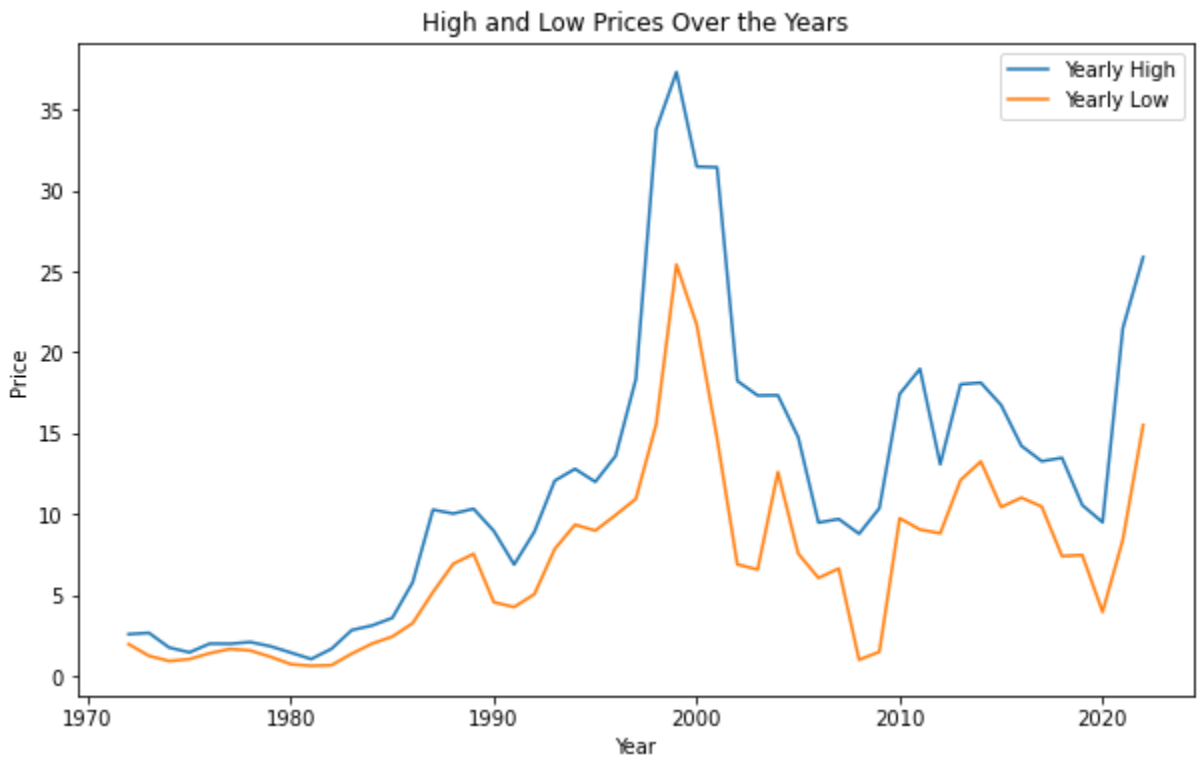
If we wanted to dig deeper into this, we could explore additional variables and factors that may be influencing the distribution of foreign investment. For example, we could analyze the types of industries or sectors in which foreign-owned companies are most prevalent, and identify any patterns or trends that may be contributing to their location choices.

Ford Stock Prices Data

The first question we used to analyze the data given in this dataset was to understand how the stock prices varied over the years and when it has been the highest and when it has been the lowest. Analyzing the variation of stock prices over the years is an important question because it provides insight into the performance of the company in the stock market. By understanding the trends and patterns of stock prices, investors and financial analysts can make informed decisions about whether to buy or sell the company's stock. Additionally, stock prices can provide a valuable measure of the company's overall financial health and stability, which is essential for long-term investment planning and strategy. Therefore, understanding how stock prices vary

over the years and when they have been highest or lowest can help investors and analysts make more informed decisions and potentially maximize their returns.

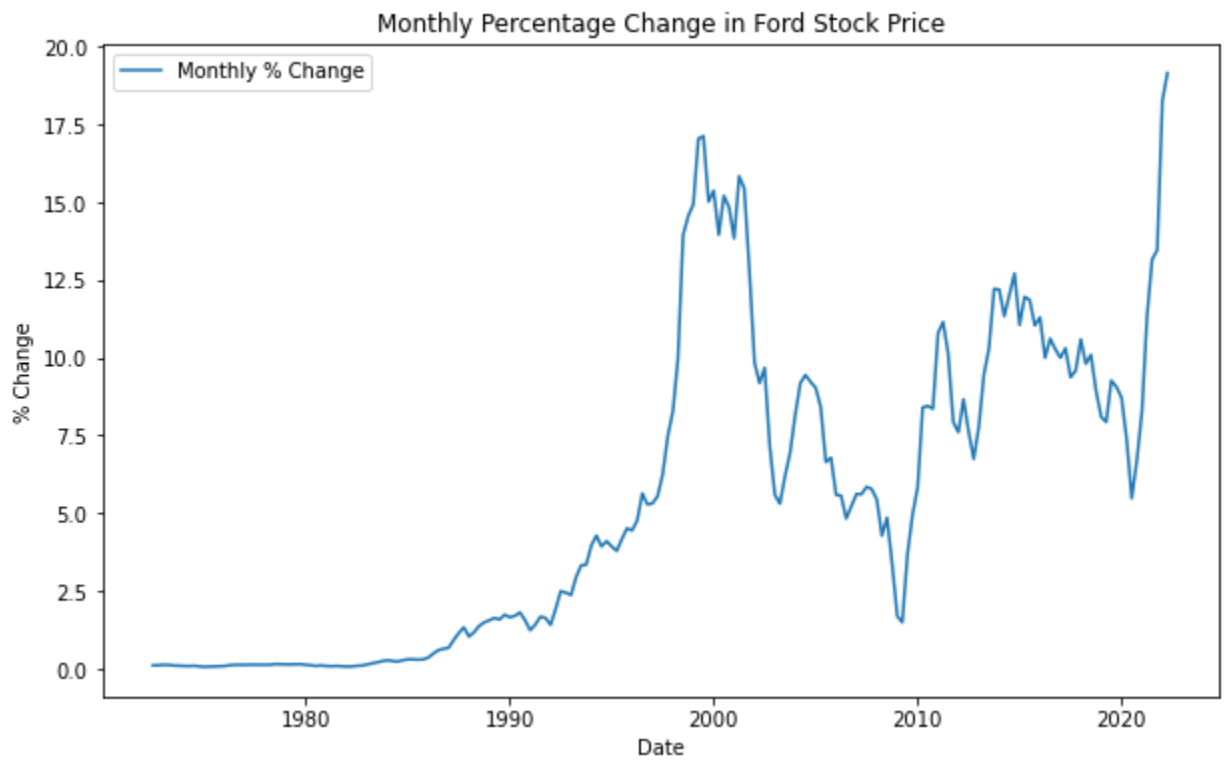
Our approach to answering this question was to first convert the 'Date' column to a datetime format and set it as the index. We then resampled the data by year and got the maximum and minimum 'High' and 'Low' values for each year. We then plot the yearly high and low prices as two separate lines on a plot. We then find the highest and lowest 'High' and 'Low' prices and their corresponding dates.



In this plot we see that the blue line represents the higher stock prices and the orange line represents the lower stock prices. We can also tell that the highest stock price was about $37 in the year 1999-2000, which then was verified when we calculated it manually – $37.30 in 1999. And the lowest stock price was about $0.50 in the year 1980-1981, which then was verified when we calculated it manually – $0.64 in 1981. It is very interesting to think about how the stock price went up by almost 40 times in the span of 20 years.

To dig deeper, we could perform a more detailed analysis of the factors that influenced the stock prices in the years with the highest and lowest prices. For example, we could examine the financial reports, news articles, and industry trends during those years to identify the specific events or conditions that led to these prices. Additionally, we could analyze the correlation between the stock prices and other variables such as interest rates, inflation, and GDP growth to understand the macroeconomic factors that affect the stock prices.

Another analysis we conducted on this dataset was understanding the monthly percentage change in Ford stock price. Understanding this data is important because it can provide insight into the performance of the company over time. If the monthly percentage change in the stock price is consistently positive, it can indicate that the company is growing and performing well. On the other hand, if the monthly percentage change is consistently negative, it can suggest that the company is struggling or facing challenges.



In our code, we compute the quarterly average of the adjusted closing price of Ford stock and plot it as a line chart. The resample() method is used to group the data by quarter and calculate the mean value of each group. The result is stored in the variable monthly_pct_change.

The next part of the code creates a new figure using the subplots() function. The plot() method is used to plot the monthly percentage change on the y-axis and the date on the x-axis, using the index of the monthly_pct_change variable for the x-axis values and the values of the variable for the y-axis values.

We can see that there is a long period of consistent growth in the stock prices from around 1985 to 1998, with the percentage change consistently positive during this period. The peak is reached around 1998-1999, where we see the highest percentage change in the stock price over the entire period of analysis. This suggests that this period was a particularly profitable time for Ford, and investors who bought into the company at this time may have seen significant returns on their investments. After this peak, however, the percentage change

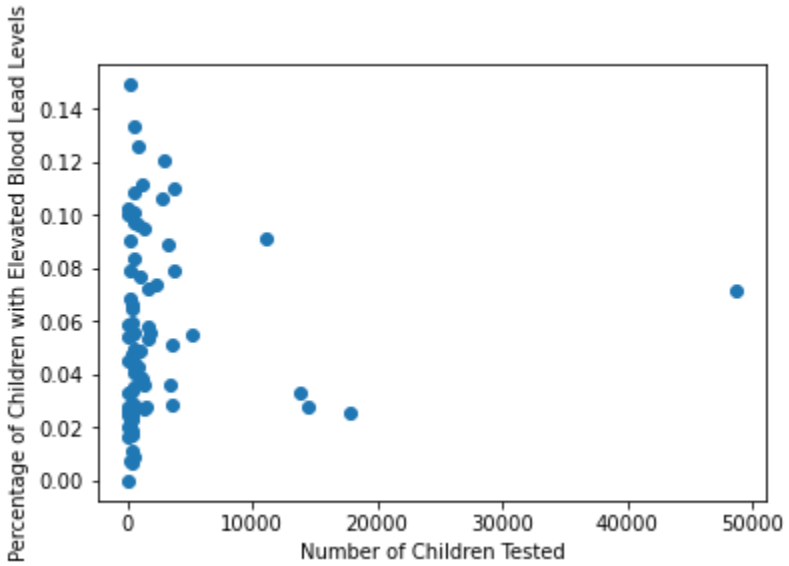
becomes much more variable, with frequent fluctuations that can be both positive and negative. This indicates a much less predictable pattern in the stock prices, which may make investing in Ford less attractive during this period.

If we wanted to dig deeper, we could compare Ford's stock price to its competitors' stock prices to see how Ford is performing relative to other companies in the same industry.

Lead Blood Levels Data

In our analysis for this dataset, we are analyzing the relationship between the number of children tested and the percentage of children with elevated blood lead levels by calculating the correlation coefficient. It is important to conduct this analysis because lead exposure is a serious public health concern, particularly for children, who are more vulnerable to its harmful effects. Elevated blood lead levels in children can cause a range of health problems, including developmental delays, learning difficulties, and behavioral problems. Therefore, it is important to identify and prevent lead exposure in children.

By analyzing the relationship between the number of children tested and the percentage of children with elevated blood lead levels, we can assess the effectiveness of efforts to identify and prevent lead exposure.



Our code first calculates the percentage of children with elevated blood lead levels and stores the result in a new column. And then we calculate the percentage of children under 6 with

elevated blood lead levels and stores the result in a new column. Next, we calculated the correlation coefficient between the number of children tested and the percentage of children with elevated blood lead levels using the .corr() method. The correlation coefficient measures the strength and direction of the linear relationship between two variables, ranging from -1 (perfect negative correlation) to 1 (perfect positive correlation). And then we created a scatter plot of the number of children tested versus the percentage of children with elevated blood lead levels.

From this graph we can tell that our correlation coefficient is 0.0491.

To dig deeper, we could perform a regression analysis to examine the relationship between different variables and the prevalence of elevated blood lead levels. For example, we could use multiple regression to examine the effects of factors such as poverty rates, housing conditions, and environmental pollution on the prevalence of elevated blood lead levels.

Unemployment rates in Michigan and US Data

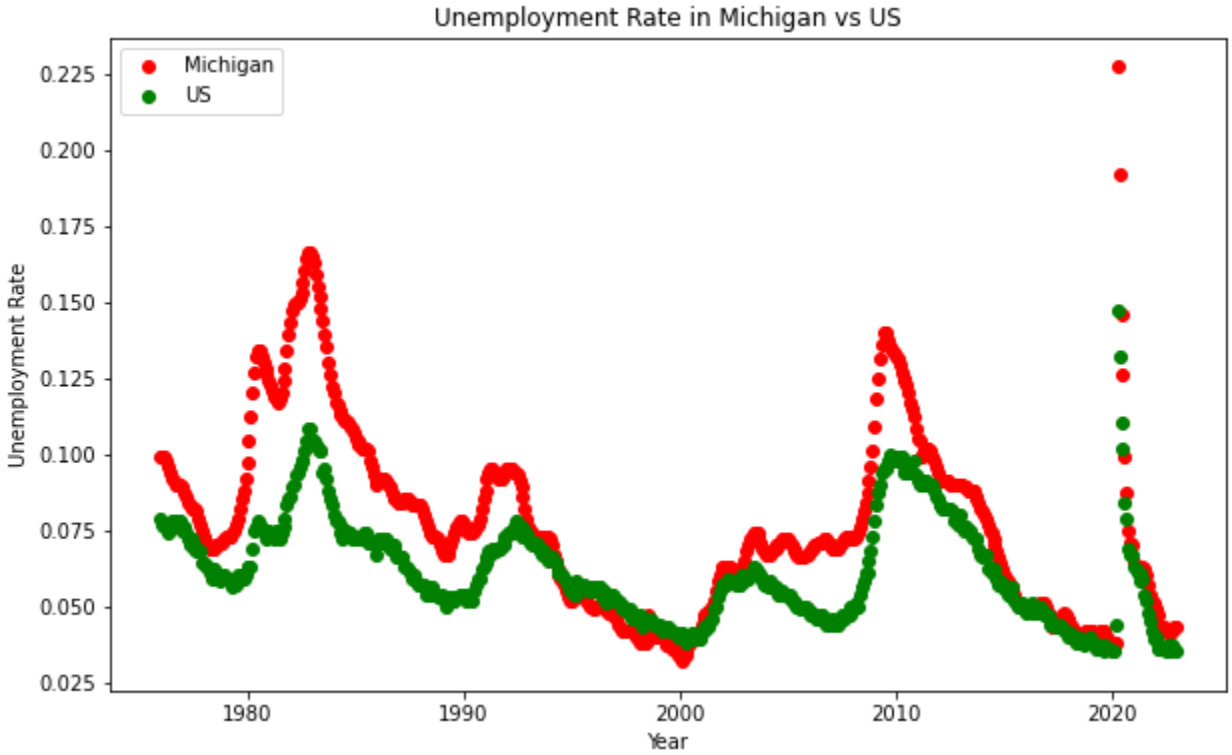
To analyze this dataset, we wanted to first understand what the highest and lowest unemployment rates have been in Michigan. Analyzing the highest and lowest unemployment rates in Michigan can provide valuable insights into the economic trends of the state. Understanding the fluctuations in unemployment rates can help policymakers and analysts identify the factors that have contributed to these trends and develop strategies to mitigate negative impacts or take advantage of positive trends. It can also provide important information for individuals and businesses to make informed decisions regarding employment, investment, and financial planning.

Our approach to coding this was creating a figure and an axis object using the subplots() function, and then calling boxplot() on the axis object. We also removed the x-tick labels for debugging purposes, so only the boxplot is displayed.

From this graph, we can see that the highest unemployment rate in Michigan is around 0.225 and the lowest is around 0.030. The mean is about 0.075.

If we wanted to dig deeper, we would consider looking at demographic factors that could impact unemployment rates, such as age, gender, race, education level, and industry. This could involve conducting more granular analyses of the data to see if there are any notable differences or disparities in unemployment rates among different demographic groups.

Another way of analyzing this data was to compare the unemployment rates in Michigan and the US overall. Comparing the unemployment rates between Michigan and the US is important because it can provide insights into the economic performance of Michigan relative to the overall US economy. High unemployment rates in Michigan relative to the US average may indicate that the state's economy is underperforming or facing unique challenges. On the other hand, lower unemployment rates in Michigan relative to the US may indicate that the state's economy is performing well and could be a useful benchmark for policymakers and economists.



Our code converts the 'Calendar Date' column of the unemployment_df DataFrame into a datetime format using the pandas to_datetime() method. It then sets the 'Calendar Date' column

as the index of the DataFrame using the set_index() method. Finally, it creates a scatter plot using the matplotlib library, with the unemployment rates of Michigan and the US on the y-axis and the dates on the x-axis.

From the scatter plot of the unemployment rates in Michigan and the US, we can see that the unemployment rate in Michigan is consistently higher than the overall unemployment rate in the US. The red dots representing the unemployment rates in Michigan are generally above the green dots representing the US unemployment rates. This suggests that the state of Michigan has a higher unemployment rate compared to the US on average. We can also observe that the unemployment rates in Michigan and the US have similar trends over time, with both rates peaking during the recession in 2008-2009, and then gradually declining in the following years. Overall, this graph highlights the importance of comparing state-level data with national data to gain a better understanding of regional economic trends.

To dig deeper into this dataset, we can use time series techniques to analyze the patterns and trends in the unemployment rates over time. This could involve decomposing the series into its trend, seasonal, and residual components, performing forecasting, or identifying any significant events or changes that might have affected the unemployment rates.

References

Fisher, S. (2019, August 1). Part 4: Police response times are slowest in Detroit's poorest neighborhoods. Deadline Detroit. https://www.deadlinedetroit.com/articles/21515/part_4_police_response_times_are_slowest_in_detroit_s_poorest_neighborhoods.

Michigan Daily. (2021, August 18). Michigan's updated census data has arrived. Here's what you need to know. https://www.michigandaily.com/government/michigans-updated-census-data-has-arrived-heres-what-you-need-to-know/.

National Criminal Justice Reference Service. (1983). Unemployment and Crime [Abstract]. Retrieved April 18, 2023, from https://www.ojp.gov/ncjrs/virtual-library/abstracts/unemployment-and-crime-1#additional-details-0.

Nelson, K. (2022, October 11). Detroit City Council approves $7M for ShotSpotter. Detroit Free Press. https://www.freep.com/story/news/local/michigan/detroit/2022/10/11/detroit-city-council-approves-7m-for-shotspotter/69556058007/

Appendix

911 Calls Data Code:

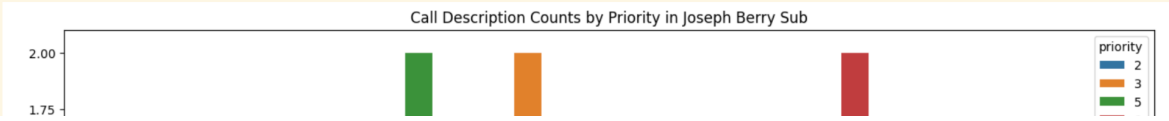
```
#1: In what neighborhood is the mean total time spent the highest and lowest? What plot can be used to best represent these values?
total_time=df.groupby('neighborhood')['totaltime'].mean().sort_values(ascending=True)
highest=total_time.idxmax()
lowest=total_time.idxmin()
total_time
print(highest, "neighborhood has the highest total time spent and ",lowest,'neighborhood has the lowest total time spent')
```

[3] Python

```
Joseph Berry Sub neighborhood has the highest total time spent and  Douglass neighborhood has the lowest total time spent
```

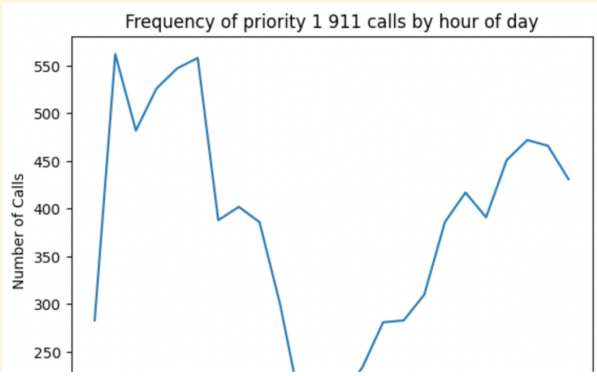
```
#plot
plt.figure(figsize=(16, 7))
highest_df=df[df.neighborhood==highest]

sns.countplot(data=highest_df, x='calldescription', hue='priority')

# Set title and axis labels
plt.title('Call Description Counts by Priority in Joseph Berry Sub')
plt.xlabel('Call Description')
plt.xticks(rotation=40)
plt.ylabel('Count')

plt.show()
```

[4] Python



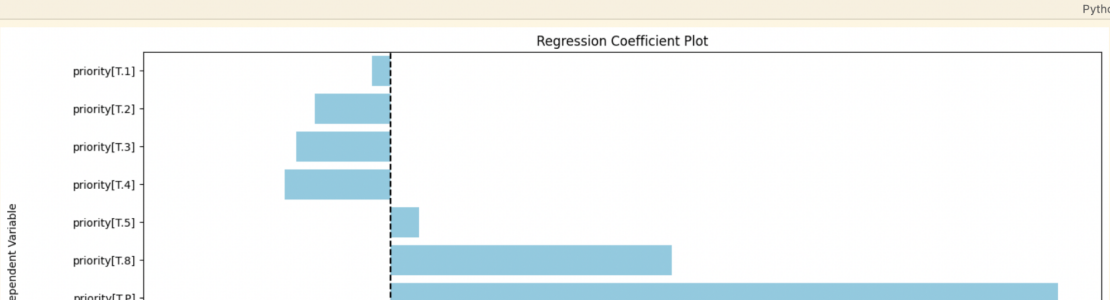
```python
df_p1 = df[df['priority'] == '1']


p1_counts = df_p1.groupby('hour')['call_timestamp'].count()

fig, ax = plt.subplots()
ax.plot(p1_counts.index, p1_counts.values)




ax.set_xlabel('Time of Day')
ax.set_ylabel('Number of Calls')
ax.set_title('Frequency of priority 1 911 calls by hour of day')

plt.show()
```

[8]




```python
model = smf.ols('totaltime ~ dispatchtime + priority + hour + intaketime + officerinitiated', data=df).fit()

import seaborn as sns

# Extract the coefficients and their p-values from the model
coef_df = pd.DataFrame({'coef': model.params.values, 'pval': model.pvalues.values}, index=model.params.index)

# Filter out the intercept coefficient
coef_df = coef_df.drop('Intercept')

# Create the coefficient plot
plt.figure(figsize=(16, 7))
sns.barplot(x='coef', y=coef_df.index, data=coef_df, color='skyblue', orient='h')
plt.axvline(x=0, color='black', linestyle='--')
plt.title('Regression Coefficient Plot')
plt.xlabel('Coefficient')
plt.ylabel('Independent Variable')
plt.show()
```

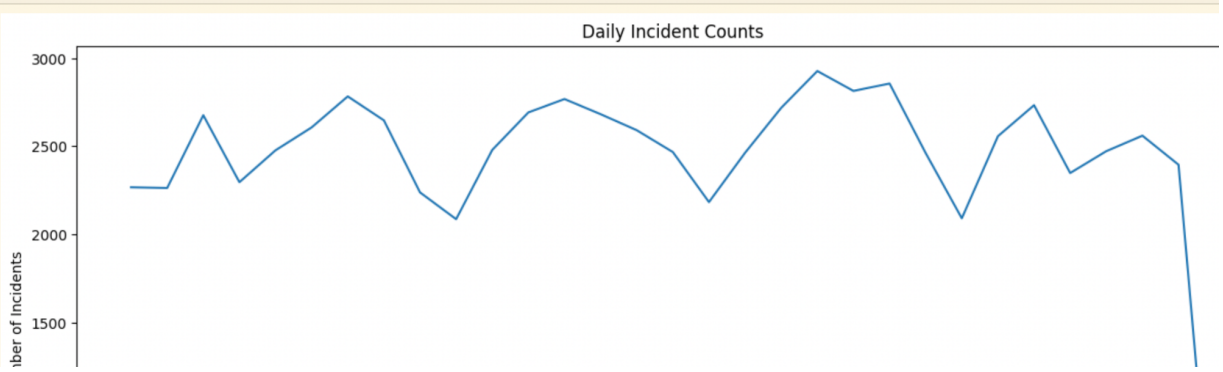
[21]



```python
import matplotlib.pyplot as plt
import geopandas as gpd

# Load the zip code map
zip_map = gpd.read_file('City_of_Detroit_Zip_Codes/zip_codes.shp')

# Filter the data to only include special attention calls
df2 = df[df.calldescription == 'SPECIAL ATTENTION']


points_gdf = gpd.GeoDataFrame(df2, geometry=gpd.points_from_xy(df2.longitude, df2.latitude))

# Plot the points on top of the zip_map
# Set the y-axis limits to focus on Detroit
ax = zip_map.plot(color='white', edgecolor='black', alpha=0.5, figsize=(12,8))

ax.set_xlim([-83.35, -82.85]) # Set the x-axis limits to focus on Detroit
ax.set_ylim([42.25, 42.47])
points_gdf.plot(ax=ax, color='red', markersize=1)

ax.set_title('Special Attention incidents by zip code')

#priority_colors = {'1': 'red', '2': 'yellow', '3': 'green'}
#for priority, color in priority_colors.items():
    #df[df['priority'] == priority].plot(ax=ax, markersize=1, color=color, label=priority)


#plt.legend(loc='upper right')

plt.show()
```

[150]

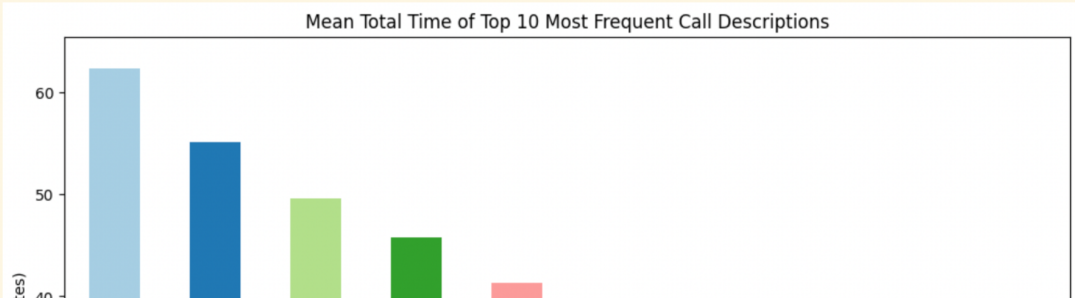
```python
import matplotlib.pyplot as plt

# Plot the daily incident counts as a line chart

plt.figure(figsize=(15, 7))
plt.plot(daily_counts["call_timestamp"], daily_counts["count"])

# Add axis labels and a title
plt.xlabel("Date")
plt.ylabel("Number of Incidents")
plt.title("Daily Incident Counts")

# Show the plot
plt.show()
```



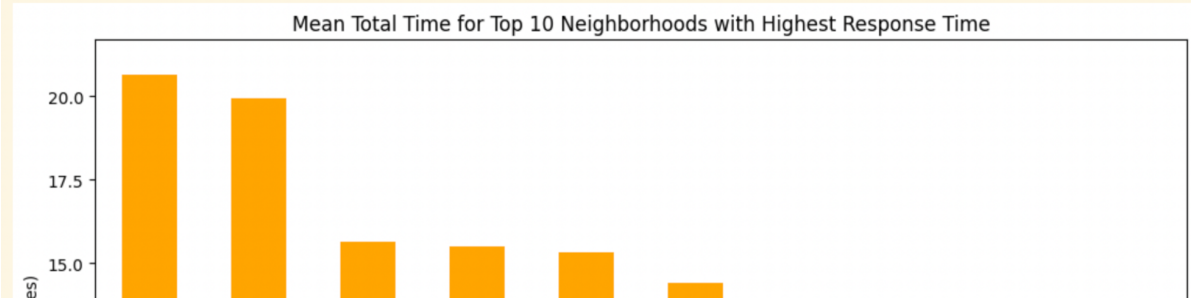
```python
df_shot2 = df[(df.calldescription == 'SHOTSPOTTER') | (df.calldescription == 'SHOTS FIRED IP')]



# Convert the latitudes and longitudes in the DataFrame to the same CRS as the city_map data
#dfs= gpd.GeoDataFrame(df_shot2, geometry=gpd.points_from_xy(df_shot2['longitude'], df_shot2['latitude']), crs="EPSG:4326")
#dfs = df_shot2.to_crs(n_map.crs)

#dfs= gpd.GeoDataFrame(df_shot2, geometry=gpd.points_from_xy(df_shot2['longitude'], df_shot2['latitude']))
#dfs = df_shot2.to_crs(n_map.crs)

# Create the plot
fig, ax = plt.subplots(figsize=(14,10))
ax.set_title('911 Calls shotspotter vs shots fired')
ax.set_facecolor('lightblue') # Set the background color
ax.set_xlim([-83.35, -82.85]) # Set the x-axis limits to focus on Detroit
ax.set_ylim([42.25, 42.47]) # Set the y-axis limits to focus on Detroit
ax.axis('off')
city_map.plot(ax=ax, alpha=0.5, edgecolor='gray', linewidth=0.5)
#dfs.plot(ax=ax, markersize=1, color='red')
colors = {'SHOTSPOTTER': 'red', 'SHOTS FIRED IP': 'black'}
for cat, color in colors.items():
    df_shot2 [df_shot2 ['calldescription'] == cat].plot(ax=ax, markersize=1, color=color, label=cat)


plt.legend(loc='upper right')


plt.show()
```

[17] Python

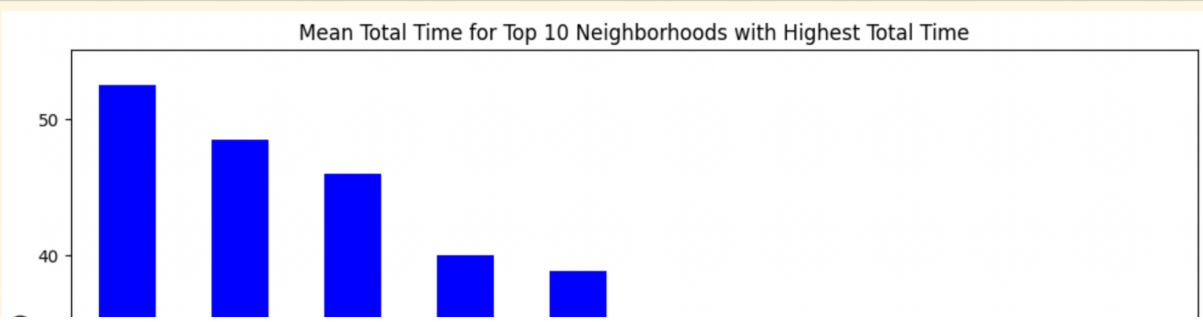
```python
import numpy as np

top_10_call_desc = df['calldescription'].value_counts().nlargest(10).index
df_top_10 = df[df['calldescription'].isin(top_10_call_desc)]

# Calculate the mean total time for each call description and sort in descending order
mean_total_time = df_top_10.groupby('calldescription')['totaltime'].mean().sort_values(ascending=False)

# Plot the mean total time for each call description
mean_total_time.plot(kind='bar', figsize=(12,8), color=plt.cm.Paired(np.arange(len(mean_total_time))))
plt.title('Mean Total Time of Top 10 Most Frequent Call Descriptions')
plt.xlabel('Call Description')
plt.ylabel('Mean Total Time (minutes)')
plt.show()
```



```python
# Group the data by neighborhood and calculate the total time for each neighborhood
# Sort the neighborhoods by total time in descending order and select the top ten neighborhoods
# Calculate the mean total time for each of the top ten neighborhoods
# Plot the mean total time for each of the top ten neighborhoods using a bar chart
neighborhood_total_time = df.groupby('neighborhood')['totalresponsetime'].sum()

# Sort the neighborhoods by total time in descending order and select the top ten neighborhoods
top_10_neighborhoods = neighborhood_total_time.sort_values(ascending=False)[:10]

# Calculate the mean total time for each of the top ten neighborhoods
mean_total_time = df[df['neighborhood'].isin(top_10_neighborhoods.index)].groupby('neighborhood')['totalresponsetime'].mean()

# Plot the mean total time for each of the top ten neighborhoods using a bar chart
plt.figure(figsize=(12,8))
mean_total_time.sort_values(ascending=False).plot(kind='bar', color='orange')
plt.title('Mean Total Time for Top 10 Neighborhoods with Highest Response Time')
plt.xlabel('Neighborhood')
plt.ylabel('Mean Response Time (minutes)')
plt.show()
```




```python
# Group the data by neighborhood and calculate the total time for each neighborhood
# Sort the neighborhoods by total time in descending order and select the top ten neighborhoods
# Calculate the mean total time for each of the top ten neighborhoods
# Plot the mean total time for each of the top ten neighborhoods using a bar chart
neighborhood_total_time = df.groupby('neighborhood')['totaltime'].sum()

# Sort the neighborhoods by total time in descending order and select the top ten neighborhoods
top_10_neighborhoods = neighborhood_total_time.sort_values(ascending=False)[:10]

# Calculate the mean total time for each of the top ten neighborhoods
mean_total_time = df[df['neighborhood'].isin(top_10_neighborhoods.index)].groupby('neighborhood')['totaltime'].mean()

# Plot the mean total time for each of the top ten neighborhoods using a bar chart
plt.figure(figsize=(12,8))
mean_total_time.sort_values(ascending=False).plot(kind='bar', color='blue')
plt.title('Mean Total Time for Top 10 Neighborhoods with Highest Total Time')
plt.xlabel('Neighborhood')
plt.ylabel('Mean Total Time (minutes)')
plt.show()
```



*All other datasets codes are merged with this report

In [ ]:
```
import numpy as np
import pandas as pd
import statsmodels.api as sm
import statsmodels.formula.api as smf
from matplotlib import pyplot as plt
import seaborn as sns
```

In [ ]:
```
#1: How does unemployment rates affect crime rates in MI?
```

In [ ]:
```
crime_df = pd.read_csv('michigan_crime.csv')
crime_df=crime_df.dropna()
crime_df["year"]= pd.DatetimeIndex(crime_df['date']).year
```

In [ ]:
```
crime_df
population_df=pd.read_csv('michigan-population.csv')
population_df["year"]= pd.DatetimeIndex(population_df['date']).year
population_df=population_df[population_df["year"]>=1979]
population_df=population_df[["year", "Population"]]
population_df
crime_df_real=pd.merge(crime_df,population_df, on="year")
```

In [ ]:
```
def fn(x):
    MI= x["MI"]


    MI_pop=x["Population"]
    crimes=(MI*MI_pop)/100000  #since the MI column gives us the amount of crimes per
    MI_percent_crime=(crimes/MI_pop)*100
    return MI_percent_crime

crime_df_real["MI_crime_percent"]=crime_df_real.apply(fn, axis=1)
crime_df_real.rename(columns={'Population': 'MI_Population'}, inplace=True)
crime_df_real
```

Out[ ]:

	date	MI	US	year	MI_Population	MI_crime_percent
0	12/31/1979	323.6076	548.8576	1979	9266268	0.323608
1	12/31/1980	341.9483	596.6383	1980	9255553	0.341948
2	12/31/1981	304.5850	593.4743	1981	9209287	0.304585
3	12/31/1982	294.5904	570.8213	1982	9115198	0.294590
4	12/31/1983	716.6501	538.1224	1983	9047754	0.716650
5	12/31/1984	281.9861	539.9269	1984	9049452	0.281986
6	12/31/1985	270.9147	558.0640	1985	9076293	0.270915
7	12/31/1986	803.9038	620.1437	1986	9127775	0.803904
8	12/31/1987	780.1304	612.4915	1987	9187481	0.780130
9	12/31/1988	741.7204	640.5839	1988	9217998	0.741720
10	12/31/1989	709.1556	666.8998	1989	9253295	0.709156
11	12/31/1990	790.3782	729.6139	1990	9310462	0.790378
12	12/31/1991	803.0743	758.1771	1991	9395022	0.803074
13	12/31/1992	770.0752	757.6663	1992	9470323	0.770075
14	12/31/1993	433.8630	747.1478	1993	9529240	0.433863
15	12/31/1994	766.1226	713.5909	1994	9584481	0.766123
16	12/31/1995	687.8207	684.4633	1995	9659871	0.687821
17	12/31/1996	635.3033	636.6358	1996	9739184	0.635303
18	12/31/1997	589.9632	610.9780	1997	9785450	0.589963
19	12/31/1998	620.8312	567.5850	1998	9820231	0.620831
20	12/31/1999	574.9219	522.9527	1999	9863775	0.574922
21	12/31/2000	555.0064	506.5299	2000	9952450	0.555006
22	12/31/2001	553.8929	504.5185	2001	9991120	0.553893
23	12/31/2002	540.7229	494.3771	2002	10015710	0.540723
24	12/31/2003	511.2888	475.8351	2003	10041152	0.511289
25	12/31/2004	492.2406	463.1556	2004	10055315	0.492241
26	12/31/2005	553.7761	469.0428	2005	10051137	0.553776
27	12/31/2006	564.4118	479.3354	2006	10036081	0.564412
28	12/31/2007	533.8160	471.7739	2007	10001284	0.533816
29	12/31/2008	514.9438	458.6142	2008	9946889	0.514944
30	12/31/2009	499.7629	431.8787	2009	9901591	0.499763
31	12/31/2010	492.9867	404.5023	2010	9877597	0.492987
32	12/31/2011	442.7648	387.0597	2011	9883053	0.442765
33	12/31/2012	454.9650	387.7538	2012	9898289	0.454965

	date	MI	US	year	MI_Population	MI_crime_percent
34	12/31/2013	452.1734	369.1334	2013	9914802	0.452173
35	12/31/2014	429.1417	361.5539	2014	9932033	0.429142
36	12/31/2015	420.5807	373.7372	2015	9934483	0.420581
37	12/31/2016	460.8874	397.5208	2016	9954117	0.460887
38	12/31/2017	449.8896	394.8597	2017	9976752	0.449890
39	12/31/2018	449.3636	380.5590	2018	9987286	0.449364

In [ ]:

```
unempl_df=pd.read_csv('Unemployment_Rate.csv')
unempl_df

unempl_df['year'] = pd.to_datetime(unempl_df['Calendar Date']).dt.year
unemployed_df=unempl_df.groupby("year")["Unemployment_Rate_MI", "Unemployment_Rate_US'
unemployed_df=unemployed_df[unemployed_df["year"]>=1979]
unemployed_df
```

```
C:\Users\anjal\AppData\Local\Temp\ipykernel_42548\1321907894.py:5: FutureWarning: Ind
exing with multiple keys (implicitly converted to a tuple of keys) will be deprecate
d, use a list instead.
  unemployed_df=unempl_df.groupby("year")["Unemployment_Rate_MI", "Unemployment_Rate_
US"].mean().reset_index()
```

Out[ ]:

	year	Unemployment_Rate_MI	Unemployment_Rate_US
3	1979	0.078667	0.058500
4	1980	0.123000	0.071750
5	1981	0.123750	0.076167
6	1982	0.154583	0.097083
7	1983	0.144833	0.096000
8	1984	0.112333	0.075083
9	1985	0.100000	0.071917
10	1986	0.089083	0.070000
11	1987	0.083917	0.061750
12	1988	0.075667	0.054917
13	1989	0.071833	0.052583
14	1990	0.077000	0.056167
15	1991	0.092833	0.068500
16	1992	0.090667	0.074917
17	1993	0.073500	0.069083
18	1994	0.060833	0.061000
19	1995	0.053167	0.055917
20	1996	0.048917	0.054083
21	1997	0.042917	0.049417
22	1998	0.039750	0.045000
23	1999	0.037417	0.042167
24	2000	0.036917	0.039667
25	2001	0.051583	0.047417
26	2002	0.061917	0.057833
27	2003	0.070917	0.059917
28	2004	0.069500	0.055417
29	2005	0.067917	0.050833
30	2006	0.070333	0.046083
31	2007	0.070833	0.046167
32	2008	0.082167	0.058000
33	2009	0.131333	0.092833
34	2010	0.121917	0.096083
35	2011	0.100083	0.089333
36	2012	0.090583	0.080750

	year	Unemployment_Rate_MI	Unemployment_Rate_US
37	2013	0.087250	0.073583
38	2014	0.072333	0.061583
39	2015	0.054250	0.052750
40	2016	0.050083	0.048750
41	2017	0.045750	0.043583
42	2018	0.041833	0.038917
43	2019	0.040917	0.036833
44	2020	0.100583	0.080917
45	2021	0.059333	0.053667
46	2022	0.043417	0.036417

In [ ]:
```
unemployed_df["Mi_percent_unemp"]=unemployed_df["Unemployment_Rate_MI"].apply(lambda x
unemployed_df["US_percent_unemp"]=unemployed_df["Unemployment_Rate_US"].apply(lambda x
unemployed_df
```

Out[ ]:

	year	Unemployment_Rate_MI	Unemployment_Rate_US	Mi_percent_unemp	US_percent_unemp
3	1979	0.078667	0.058500	7.866667	5.850000
4	1980	0.123000	0.071750	12.300000	7.175000
5	1981	0.123750	0.076167	12.375000	7.616667
6	1982	0.154583	0.097083	15.458333	9.708333
7	1983	0.144833	0.096000	14.483333	9.600000
8	1984	0.112333	0.075083	11.233333	7.508333
9	1985	0.100000	0.071917	10.000000	7.191667
10	1986	0.089083	0.070000	8.908333	7.000000
11	1987	0.083917	0.061750	8.391667	6.175000
12	1988	0.075667	0.054917	7.566667	5.491667
13	1989	0.071833	0.052583	7.183333	5.258333
14	1990	0.077000	0.056167	7.700000	5.616667
15	1991	0.092833	0.068500	9.283333	6.850000
16	1992	0.090667	0.074917	9.066667	7.491667
17	1993	0.073500	0.069083	7.350000	6.908333
18	1994	0.060833	0.061000	6.083333	6.100000
19	1995	0.053167	0.055917	5.316667	5.591667
20	1996	0.048917	0.054083	4.891667	5.408333
21	1997	0.042917	0.049417	4.291667	4.941667
22	1998	0.039750	0.045000	3.975000	4.500000
23	1999	0.037417	0.042167	3.741667	4.216667
24	2000	0.036917	0.039667	3.691667	3.966667
25	2001	0.051583	0.047417	5.158333	4.741667
26	2002	0.061917	0.057833	6.191667	5.783333
27	2003	0.070917	0.059917	7.091667	5.991667
28	2004	0.069500	0.055417	6.950000	5.541667
29	2005	0.067917	0.050833	6.791667	5.083333
30	2006	0.070333	0.046083	7.033333	4.608333
31	2007	0.070833	0.046167	7.083333	4.616667
32	2008	0.082167	0.058000	8.216667	5.800000
33	2009	0.131333	0.092833	13.133333	9.283333
34	2010	0.121917	0.096083	12.191667	9.608333
35	2011	0.100083	0.089333	10.008333	8.933333
36	2012	0.090583	0.080750	9.058333	8.075000

	year	Unemployment_Rate_MI	Unemployment_Rate_US	Mi_percent_unemp	US_percent_unemp
37	2013	0.087250	0.073583	8.725000	7.358333
38	2014	0.072333	0.061583	7.233333	6.158333
39	2015	0.054250	0.052750	5.425000	5.275000
40	2016	0.050083	0.048750	5.008333	4.875000
41	2017	0.045750	0.043583	4.575000	4.358333
42	2018	0.041833	0.038917	4.183333	3.891667
43	2019	0.040917	0.036833	4.091667	3.683333
44	2020	0.100583	0.080917	10.058333	8.091667
45	2021	0.059333	0.053667	5.933333	5.366667
46	2022	0.043417	0.036417	4.341667	3.641667

In [ ]:

In [ ]:

```
merged_df = pd.merge(crime_df_real, unemployed_df, on='year')
merged_df
#sns.regplot(data=merged_df, x='x', y='y')
model = smf.ols('MI_crime_percent ~ Mi_percent_unemp', data=merged_df)
results=model.fit().summary()
results
```

Out[ ]:

OLS Regression Results

Dep. Variable:	MI_crime_percent	**R-squared:**	0.063
Model:	OLS	**Adj. R-squared:**	0.038
Method:	Least Squares	**F-statistic:**	2.549
Date:	Tue, 18 Apr 2023	**Prob (F-statistic):**	0.119
Time:	23:57:00	**Log-Likelihood:**	19.572
No. Observations:	40	**AIC:**	-35.14
Df Residuals:	38	**BIC:**	-31.77
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	0.6427	0.068	9.387	0.000	0.504	0.781
Mi_percent_unemp	-0.0130	0.008	-1.597	0.119	-0.029	0.003

Omnibus:	3.396	**Durbin-Watson:**	1.082
Prob(Omnibus):	0.183	**Jarque-Bera (JB):**	2.733
Skew:	0.516	**Prob(JB):**	0.255
Kurtosis:	2.241	**Cond. No.**	24.2

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [ ]:

```
corr_coef = merged_df['Mi_percent_unemp'].corr(merged_df['MI_crime_percent'])
corr_coef
```

Out[ ]:

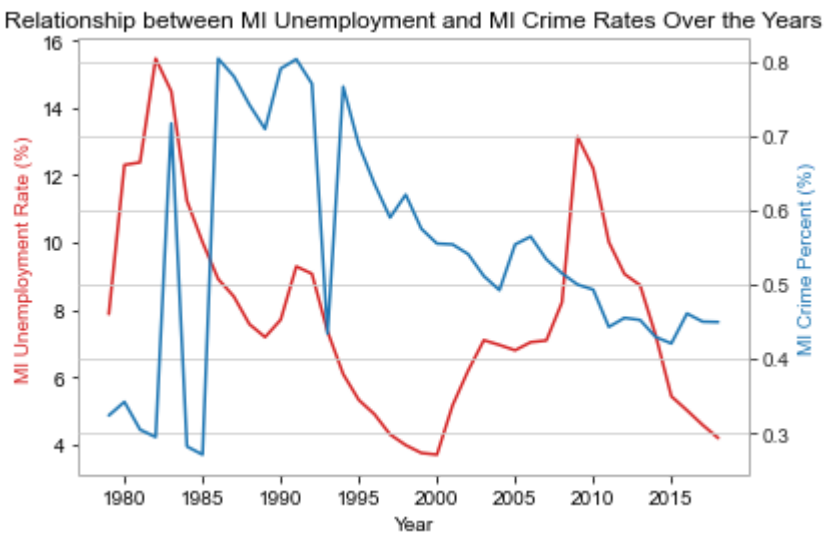
```
-0.25074313148274807
```

In [ ]:

In [ ]:

```
import matplotlib.pyplot as plt



year = merged_df["year"]
unemployment = merged_df["Mi_percent_unemp"]
crime_percent = merged_df["MI_crime_percent"]

fig, ax1 = plt.subplots()

sns.set_style("whitegrid")
ax1.set_title("Relationship between MI Unemployment and MI Crime Rates Over the Years'
```

```
ax1.set_xlabel("Year")

ax1.set_ylabel("MI Unemployment Rate (%)", color="tab:red")

ax1.plot(year, unemployment, color="tab:red")


ax2 = ax1.twinx()

ax2.set_ylabel("MI Crime Percent (%)", color="tab:blue")


ax2.plot(year, crime_percent, color="tab:blue")

plt.show()
```



In [ ]:

```
#question 2

#How has the relationship between Michigan's crime rate and MI unemployment rate chang
```

In [ ]:

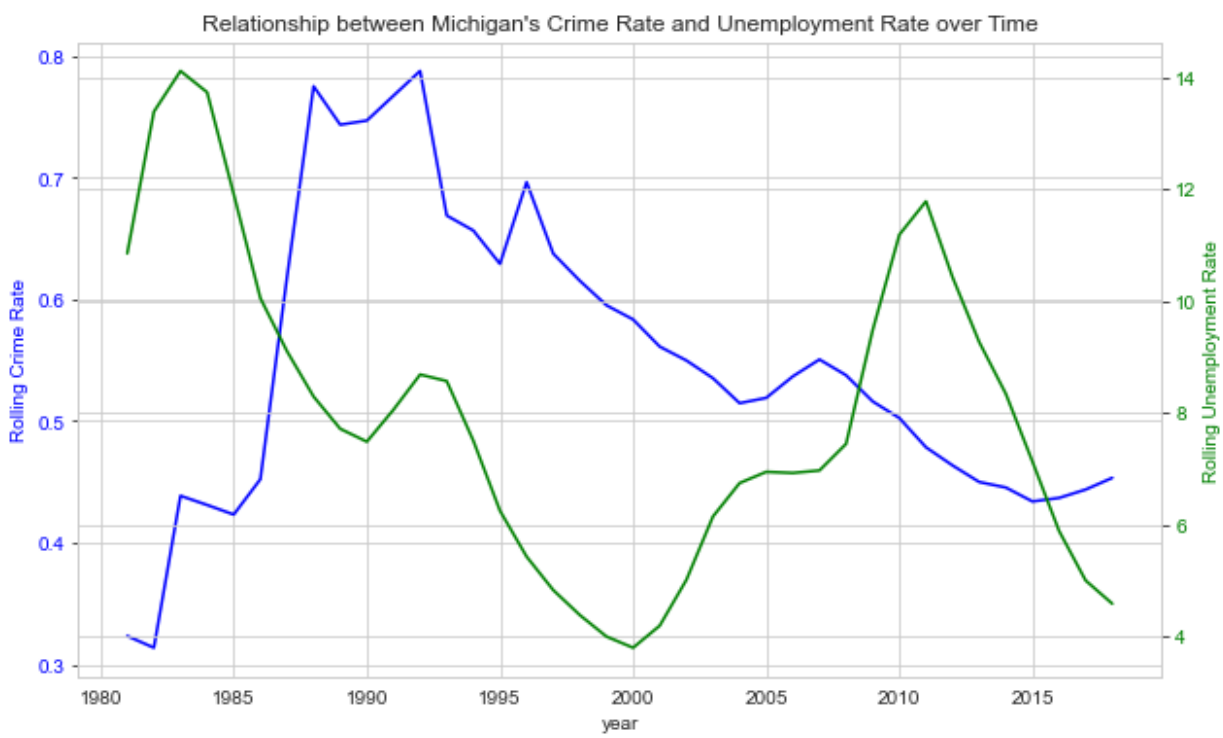
```
df=merged_df.set_index('year')
df
df['rolling_crime'] = df['MI_crime_percent'].rolling(window=3).mean()
df['rolling_unemp'] = df['Mi_percent_unemp'].rolling(window=3).mean()


df_grouped = df.groupby('year')['rolling_crime', 'rolling_unemp'].mean()


df_grouped


corr1 = df_grouped['rolling_crime'].corr(df_grouped['rolling_unemp'])
corr1
```

```
C:\Users\anjal\AppData\Local\Temp\ipykernel_42548\140592349.py:7: FutureWarning: Inde
xing with multiple keys (implicitly converted to a tuple of keys) will be deprecated,
use a list instead.
  df_grouped = df.groupby('year')['rolling_crime', 'rolling_unemp'].mean()
```

Out[ ]:

```
-0.39193022964923774
```

In [ ]:
```
fig, ax1 = plt.subplots(figsize=(10,6))
ax2 = ax1.twinx()
sns.set_style("whitegrid")
# plot rolling crime rate on first y-axis
sns.lineplot(data=df_grouped, x='year', y='rolling_crime', color='blue', ax=ax1)
ax1.set_ylabel('Rolling Crime Rate', color='blue')
ax1.tick_params(axis='y', labelcolor='blue')

# plot rolling unemployment rate on second y-axis
sns.lineplot(data=df_grouped, x='year', y='rolling_unemp', color='green', ax=ax2)
ax2.set_ylabel('Rolling Unemployment Rate', color='green')
ax2.tick_params(axis='y', labelcolor='green')

# add title and labels
plt.title("Relationship between Michigan's Crime Rate and Unemployment Rate over Time"
plt.xlabel('Year')

# show the plot
plt.show()
```



In [ ]:
```
#Question 3
#How much does the US unemployment rate affect Michigan's crime rate, while controllir
#variables such as Michigan's unemployment rate, the population, and the US crime rate
```

In [ ]:
```
merged_df
US_pop=pd.read_csv("united-states-population.csv")

US_pop=US_pop[["date", " Population"]]
US_pop.rename(columns={' Population': 'US_Population'},inplace=True)
US_pop
```

Out[ ]:

	date	US_Population
0	12/31/1950	148281550
1	12/31/1951	150598453
2	12/31/1952	152941727
3	12/31/1953	155451199
4	12/31/1954	158205873
...	...	...
146	12/31/2096	393602753
147	12/31/2097	393728779
148	12/31/2098	393843026
149	12/31/2099	393946323
150	12/31/2100	394041155

151 rows × 2 columns

In [ ]:

```
US_pop["year"]= pd.DatetimeIndex(US_pop['date']).year
US_pop=US_pop[US_pop["year"]>=1979]
US_pop=US_pop[["year", "US_Population"]]
US_pop
crime_df_real2=pd.merge(crime_df_real,US_pop, on="year")
```

In [ ]:

```
crime_df_real2.columns
```

Out[ ]:

```
Index(['date', 'MI', 'US', 'year', 'MI_Population', 'MI_crime_percent',
       'US_Population'],
      dtype='object')
```

In [ ]:

```
def fn2(x):
    US= x["US"]
    US_pop=x["US_Population"]
    crimes=(US*US_pop)/100000  #since the MI column gives us the amount of crimes per
    US_percent_crime=(crimes/US_pop)*100
    return US_percent_crime

crime_df_real2["US_crime_percent"]=crime_df_real2.apply(fn2, axis=1)
crime_df_real2

merged_df2 = pd.merge(crime_df_real2, unemployed_df, on='year')
```

In [ ]:

```
merged_df2.head()
```

Out[ ]:

	date	MI	US	year	MI_Population	MI_crime_percent	US_Population	US_crime_p
0	12/31/1979	323.6076	548.8576	1979	9266268	0.323608	220463115	0.!
1	12/31/1980	341.9483	596.6383	1980	9255553	0.341948	223140018	0.!
2	12/31/1981	304.5850	593.4743	1981	9209287	0.304585	225654008	0.!
3	12/31/1982	294.5904	570.8213	1982	9115198	0.294590	228001418	0.!
4	12/31/1983	716.6501	538.1224	1983	9047754	0.716650	230389964	0.!

In [ ]:

In [ ]:

```
model = smf.ols('MI_crime_percent ~ US_percent_unemp+Mi_percent_unemp+year+MI_Populati
results=model.fit().summary()
results
```

Out[ ]:

OLS Regression Results

Dep. Variable:	MI_crime_percent	**R-squared:**	0.462
Model:	OLS	**Adj. R-squared:**	0.383
Method:	Least Squares	**F-statistic:**	5.839
Date:	Tue, 18 Apr 2023	**Prob (F-statistic):**	0.000537
Time:	23:57:01	**Log-Likelihood:**	30.671
No. Observations:	40	**AIC:**	-49.34
Df Residuals:	34	**BIC:**	-39.21
Df Model:	5		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	-17.9135	8.502	-2.107	0.043	-35.191	-0.636
US_percent_unemp	-0.0016	0.032	-0.050	0.960	-0.067	0.064
Mi_percent_unemp	-0.0046	0.020	-0.232	0.818	-0.045	0.036
year	0.0093	0.005	2.019	0.051	-5.92e-05	0.019
MI_Population	-7.957e-08	1.32e-07	-0.602	0.551	-3.48e-07	1.89e-07
US_crime_percent	1.2442	0.292	4.260	0.000	0.651	1.838

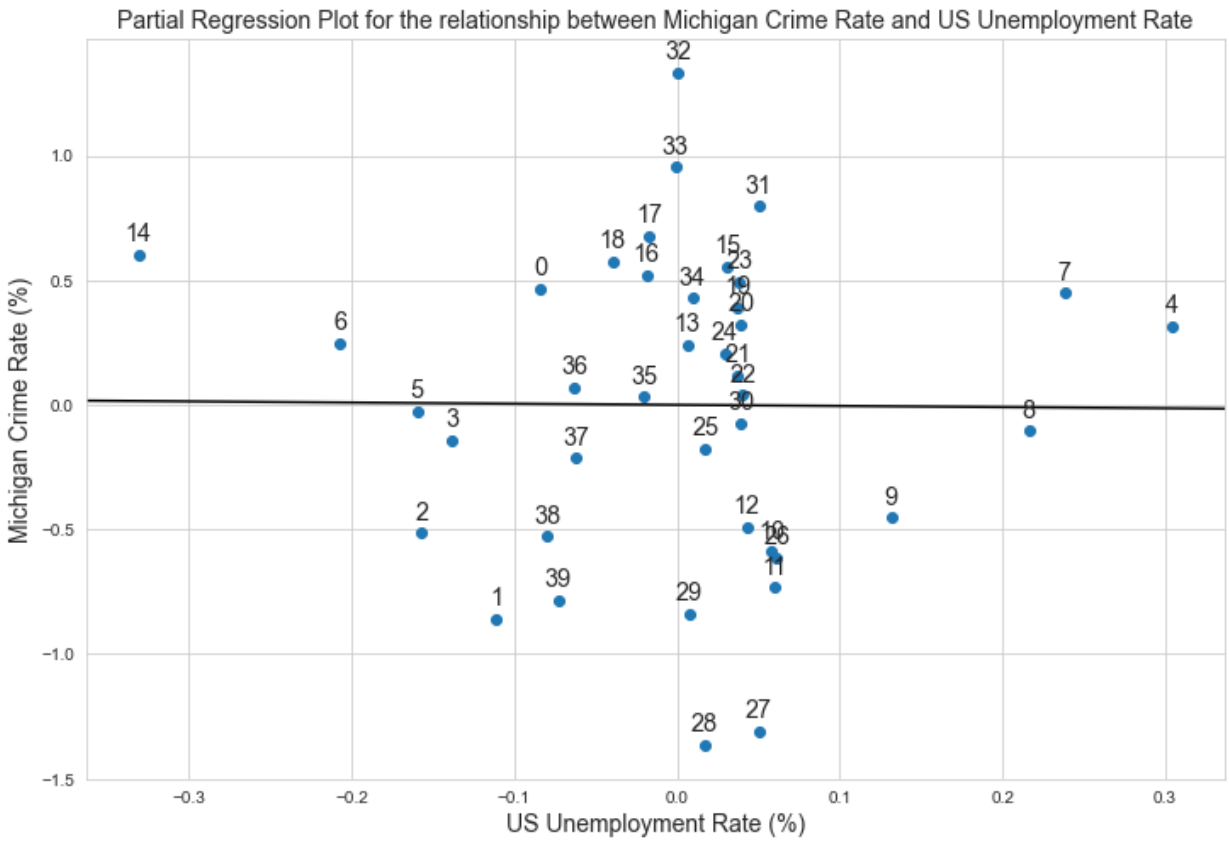
Omnibus:	4.556	**Durbin-Watson:**	1.769
Prob(Omnibus):	0.102	**Jarque-Bera (JB):**	4.878
Skew:	-0.059	**Prob(JB):**	0.0872
Kurtosis:	4.707	**Cond. No.**	4.26e+09

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 4.26e+09. This might indicate that there are
strong multicollinearity or other numerical problems.

In [ ]:

```
from statsmodels.graphics.regressionplots import plot_partregress

fig = plt.figure(figsize=(12, 8))
ax = fig.add_subplot(111)
plot_partregress('US_percent_unemp', 'MI_crime_percent', ['Mi_percent_unemp', 'year',

# Set plot labels and title
ax.set_xlabel("US Unemployment Rate (%)",  fontsize=14)
ax.set_ylabel("Michigan Crime Rate (%)", fontsize=14)
ax.set_title("Partial Regression Plot for the relationship between Michigan Crime Rate

# Show the plot
plt.show()
```

```
eval_env: 1
```



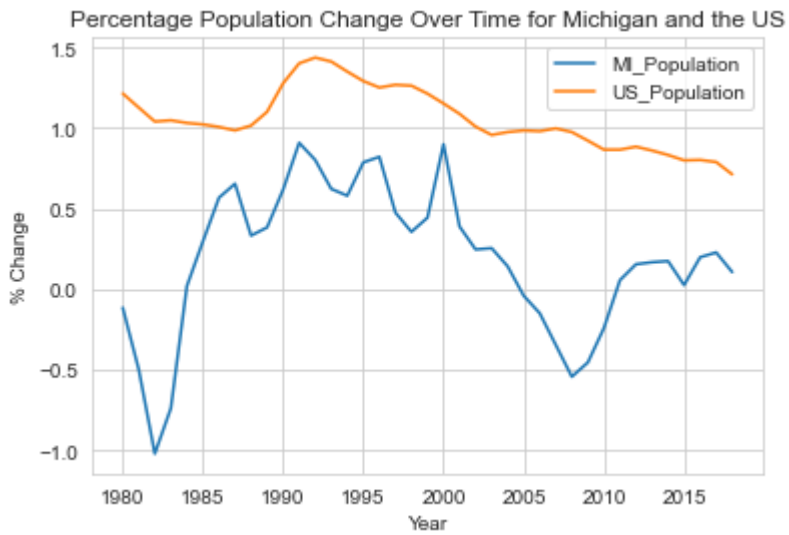
In [ ]:
```
#question 4
#How has MI's population changed compared to the US population?
```

In [ ]:
```
merged_df2
pop_trend = merged_df2.groupby("year")[["MI_Population", "US_Population"]].sum()
pop_trend

percentage_change= pop_trend.pct_change() * 100
Us_change=percentage_change["US_Population"].mean()
MI_change=percentage_change["MI_Population"].mean()
Us_change



#percentage_change

percentage_change
MI_change
```

Out[ ]:
```
0.1933315687582627
```

In [ ]:
```
sns.set_style("whitegrid")
sns.lineplot(data=percentage_change, dashes=False)


plt.title("Percentage Population Change Over Time for Michigan and the US")
plt.xlabel("Year")
plt.ylabel("% Change")
```

```
plt.show()
```



In [ ]:
```
#q5: What's the relationship between MI crime and US crime rates?
```

In [ ]:
```
merged_df2
q5_df = merged_df2.set_index('date')
q5_df.columns


q5_df['MI_rolling_crime'] = q5_df['MI_crime_percent'].rolling(window=3).mean()
q5_df['US_rolling_crime'] = q5_df['US_crime_percent'].rolling(window=3).mean()


q5df_grouped = q5_df.groupby('year')['MI_rolling_crime', 'US_rolling_crime'].mean()


q5df_grouped



corr2 = q5df_grouped['MI_rolling_crime'].corr(q5df_grouped['US_rolling_crime'])
corr2
```

```
C:\Users\anjal\AppData\Local\Temp\ipykernel_42548\887109767.py:10: FutureWarning: Ind
exing with multiple keys (implicitly converted to a tuple of keys) will be deprecate
d, use a list instead.
  q5df_grouped = q5_df.groupby('year')['MI_rolling_crime', 'US_rolling_crime'].mean()
```

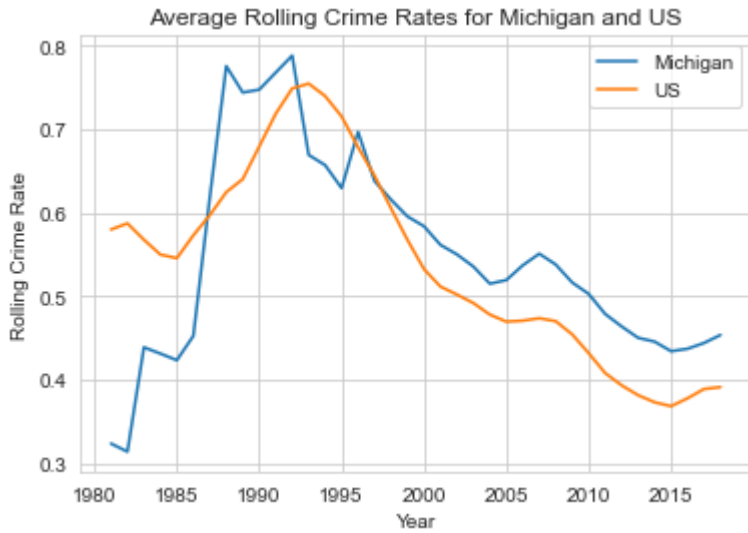
Out[ ]:
```
0.6857590644755148
```

In [ ]:
```
corr_coef = q5df_grouped['MI_rolling_crime'].corr(q5df_grouped['US_rolling_crime'])
corr_coef
```

Out[ ]:
```
0.6857590644755148
```

In [ ]:
```
import matplotlib.pyplot as plt

plt.plot(q5df_grouped.index, q5df_grouped ['MI_rolling_crime'], label='Michigan')
plt.plot(q5df_grouped.index, q5df_grouped ['US_rolling_crime'], label='US')
plt.title('Average Rolling Crime Rates for Michigan and US')
```

```
plt.xlabel('Year')
plt.ylabel('Rolling Crime Rate')
plt.legend()
plt.show()
```



In [ ]:

In [ ]:
```
import numpy as np
import pandas as pd
import statsmodels.api as sm
import statsmodels.formula.api as smf
from matplotlib import pyplot as plt
import seaborn as sns
import requests
from bs4 import BeautifulSoup
import json
import string
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.tokenize import RegexpTokenizer
from nltk.tokenize import word_tokenize
from collections import Counter
import geopandas as gpd
```

```
[nltk_data] Downloading package stopwords to /Users/admin/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [ ]:
```
pip install geopandas
```

```
Collecting geopandas
  Downloading geopandas-0.12.2-py3-none-any.whl (1.1 MB)
     |████████████████████████████████| 1.1 MB 1.8 MB/s eta 0:00:01
Collecting pyproj>=2.6.1.post1
  Downloading pyproj-3.5.0-cp39-cp39-macosx_10_9_x86_64.whl (8.5 MB)
     |████████████████████████████████| 8.5 MB 41.0 MB/s eta 0:00:01
Collecting fiona>=1.8
  Downloading Fiona-1.9.3-cp39-cp39-macosx_10_15_x86_64.whl (18.3 MB)
     |████████████████████████████████| 18.3 MB 928 kB/s eta 0:00:01
Requirement already satisfied: packaging in /Users/admin/opt/anaconda3/lib/p
ython3.9/site-packages (from geopandas) (21.3)
Collecting shapely>=1.7
  Downloading shapely-2.0.1-cp39-cp39-macosx_10_9_x86_64.whl (1.4 MB)
     |████████████████████████████████| 1.4 MB 20.7 MB/s eta 0:00:01
Requirement already satisfied: pandas>=1.0.0 in /Users/admin/opt/anaconda3/l
ib/python3.9/site-packages (from geopandas) (1.4.2)
Requirement already satisfied: click~=8.0 in /Users/admin/opt/anaconda3/lib/
python3.9/site-packages (from fiona>=1.8->geopandas) (8.0.4)
Requirement already satisfied: attrs>=19.2.0 in /Users/admin/opt/anaconda3/l
ib/python3.9/site-packages (from fiona>=1.8->geopandas) (21.4.0)
Requirement already satisfied: certifi in /Users/admin/opt/anaconda3/lib/pyt
hon3.9/site-packages (from fiona>=1.8->geopandas) (2021.10.8)
Collecting munch>=2.3.2
  Downloading munch-2.5.0-py2.py3-none-any.whl (10 kB)
Requirement already satisfied: importlib-metadata in /Users/admin/opt/anacon
da3/lib/python3.9/site-packages (from fiona>=1.8->geopandas) (4.11.3)
Collecting cligj>=0.5
  Downloading cligj-0.7.2-py3-none-any.whl (7.1 kB)
Collecting click-plugins>=1.0
  Downloading click_plugins-1.1.1-py2.py3-none-any.whl (7.5 kB)
Requirement already satisfied: six in /Users/admin/opt/anaconda3/lib/python
3.9/site-packages (from munch>=2.3.2->fiona>=1.8->geopandas) (1.16.0)
Requirement already satisfied: python-dateutil>=2.8.1 in /Users/admin/opt/an
aconda3/lib/python3.9/site-packages (from pandas>=1.0.0->geopandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /Users/admin/opt/anaconda3/li
b/python3.9/site-packages (from pandas>=1.0.0->geopandas) (2021.3)
Requirement already satisfied: numpy>=1.18.5 in /Users/admin/opt/anaconda3/l
ib/python3.9/site-packages (from pandas>=1.0.0->geopandas) (1.21.5)
Requirement already satisfied: zipp>=0.5 in /Users/admin/opt/anaconda3/lib/p
ython3.9/site-packages (from importlib-metadata->fiona>=1.8->geopandas) (3.
7.0)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /Users/admin/opt/
anaconda3/lib/python3.9/site-packages (from packaging->geopandas) (3.0.4)
Installing collected packages: munch, cligj, click-plugins, shapely, pyproj,
fiona, geopandas
Successfully installed click-plugins-1.1.1 cligj-0.7.2 fiona-1.9.3 geopandas
-0.12.2 munch-2.5.0 pyproj-3.5.0 shapely-2.0.1
Note: you may need to restart the kernel to use updated packages.
```

Ann Arbor Planning and Development Projects

Spatial and Temporal data of construction permits in Ann Arbor 2022-2023. Spatial data is reported as an address

In [ ]:
```
df = pd.read_csv('firms.csv')
df.head()
df
```

Out[ ]:

	ObjectID	X	Y	DisplayX	DisplayY	Active_or_Not	Foreign_Ow
0	1	134.999996	-24.999997	134.999996	-24.999997	Active	
1	2	134.999996	-24.999997	134.999996	-24.999997	Active	
2	3	134.999996	-24.999997	134.999996	-24.999997	Active	
3	4	134.999996	-24.999997	134.999996	-24.999997	Active	
4	5	134.999996	-24.999997	134.999996	-24.999997	Active	
...	...	...	...	...	...	...	
968	970	-2.695310	54.758440	-2.695310	54.758440	Active	Unit
969	971	-2.695310	54.758440	-2.695310	54.758440	Active	Unit
970	972	-2.695310	54.758440	-2.695310	54.758440	Active	Unit
971	973	-2.695310	54.758440	-2.695310	54.758440	Active	Unit
972	974	134.999996	-24.999997	134.999996	-24.999997	Active	

973 rows × 12 columns

In [ ]:
```
df.columns
```

Out[ ]:
```
Index(['ObjectID', 'X', 'Y', 'DisplayX', 'DisplayY', 'Active_or_Not',
       'Foreign_Owned_Firms', 'Company_Name', 'City_1', 'State_Province',
       'FirmLoc', 'Region', 'geometry'],
      dtype='object')
```

1. What are the top 10 cities with the most foreign owned companies?

</font>

The presence of foreign companies in the US can help to increase the country's global economic influence and strengthen its ties with other countries. Knowing this data is important because it can help policymakers and economic planners understand the patterns of foreign investment in the United States and identify areas where there may be opportunities for growth and development.

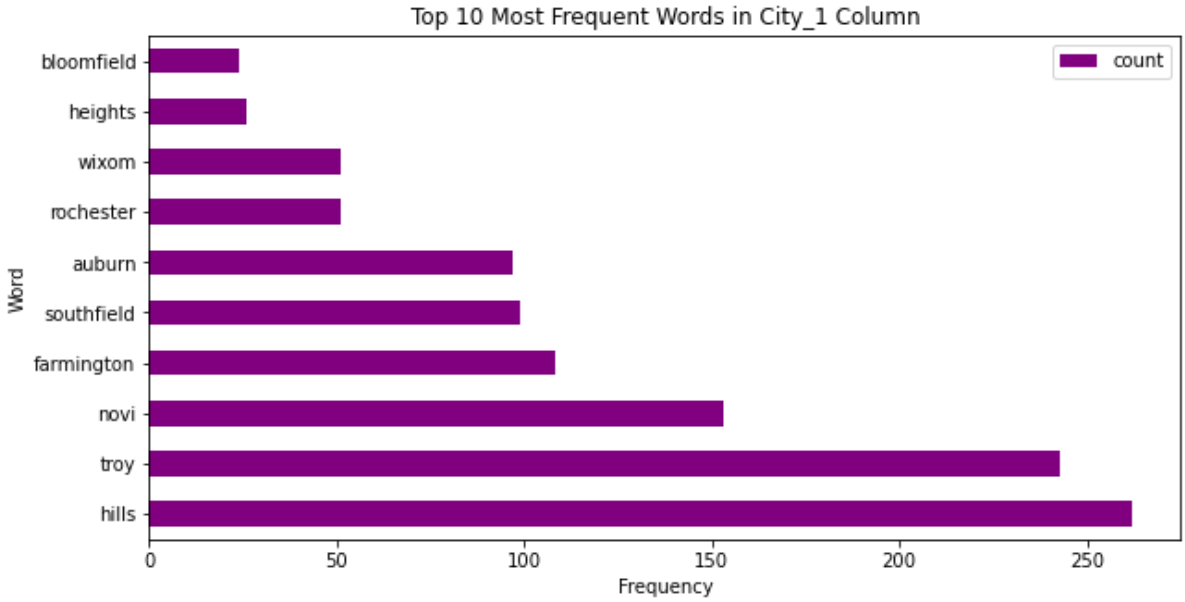
In [ ]:
```
# Convert the City_1 column to lowercase
df['City_1'] = df['City_1'].str.lower()

# Replace NaN values with empty strings
df['City_1'] = df['City_1'].fillna('')

# Remove stopwords
stop_words = set(stopwords.words('english'))
df['City_1'] = df['City_1'].apply(lambda x: ' '.join([word for word in x.spl
```

```
# Count the frequency of words in the City_1 column
word_count = Counter(' '.join(df['City_1']).split())

# Get the top 10 most frequent words
top_10_words = word_count.most_common(10)

# Print the results
for word, count in top_10_words:
    print(f"{word}: {count}")
```

```
hills: 262
troy: 243
novi: 153
farmington: 108
southfield: 99
auburn: 97
wixom: 51
rochester: 51
heights: 26
bloomfield: 24
```

In [ ]:

```
word_freq_df = pd.DataFrame.from_dict(word_count, orient='index', columns=['

# Sort the dataframe by count in descending order
word_freq_df = word_freq_df.sort_values(by=['count'], ascending=False)

# Plot a horizontal bar graph of the top 10 most frequent words
word_freq_df[:10].plot(kind='barh', figsize=(10,5), color='purple')
plt.title('Top 10 Most Frequent Words in City_1 Column')
plt.xlabel('Frequency')
plt.ylabel('Word')
plt.show()




'''# Create a list of the top 10 words and their counts
labels = [word for word, count in top_10_words]
values = [count for word, count in top_10_words]

# Create the pie chart
plt.pie(values, labels=labels, autopct='%1.1f%%')
plt.title('Top 10 Most Frequent Words in City Names')
plt.show()'''
```



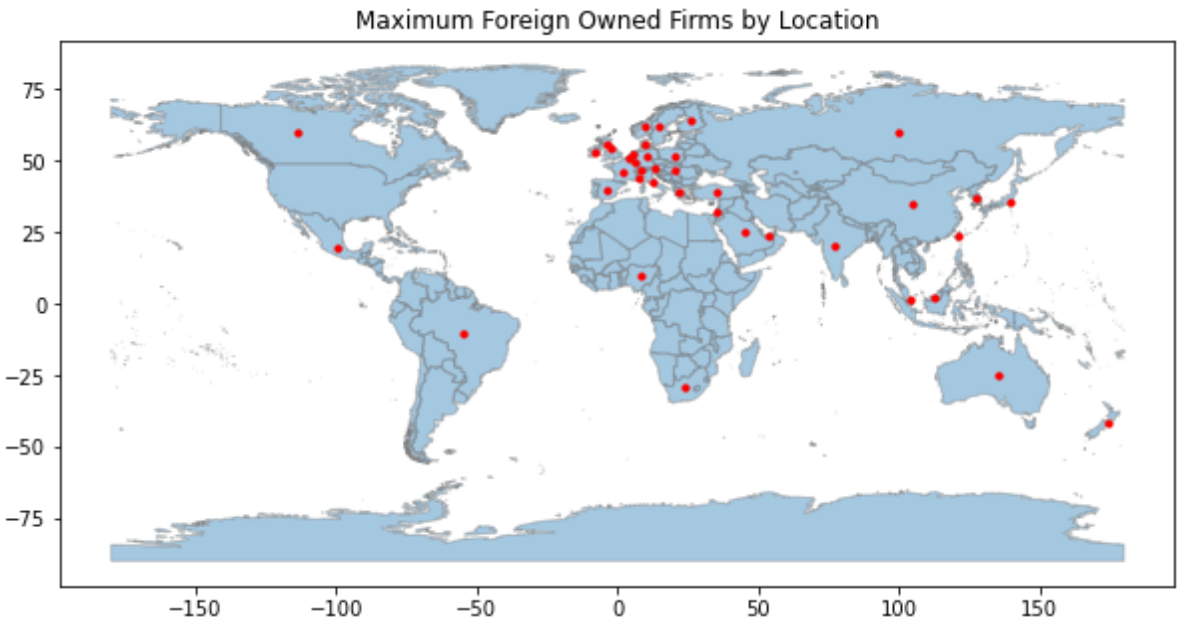
Out[ ]:
```
"# Create a list of the top 10 words and their counts\nlabels = [word for wo
rd, count in top_10_words]\nvalues = [count for word, count in top_10_words]
\n\n# Create the pie chart\nplt.pie(values, labels=labels, autopct='%1.1f%
%')\nplt.title('Top 10 Most Frequent Words in City Names')\nplt.show()"
```

2. Plot the countries where these firms originate from?

In [ ]:
```
def max(s):
    x = s.sort_values(by='Foreign_Owned_Firms', ascending=False)
    return x[:1]
new = df.groupby(['X', 'Y'])
new = new.apply(max)
```

In [ ]:
```
# Load the shapefile of world countries
world = gpd.read_file('worldhope.shp')

# Perform a spatial join to match the countries in the shapefile to the coun
companies_geo = gpd.GeoDataFrame(df, geometry=gpd.points_from_xy(df.X, df.Y)
companies_geo.crs = world.crs
companies_by_country = gpd.sjoin(world, companies_geo, how='right', op='inte

# Keep only the maximum value for each point
new = companies_by_country.groupby(['X', 'Y'])['Foreign_Owned_Firms'].max().
new = gpd.GeoDataFrame(new, geometry=gpd.points_from_xy(new.X, new.Y))
new.crs = world.crs

# Plot the points on a world map
fig, ax = plt.subplots(figsize=(10, 6))
ax.set_aspect('equal')
world.plot(ax=ax, alpha=0.4, edgecolor='grey')
new.plot(ax=ax, markersize=10, color='red')
ax.set_title('Maximum Foreign Owned Firms by Location')
plt.show()
```

```
/Users/admin/opt/anaconda3/lib/python3.9/site-packages/IPython/core/interact
iveshell.py:3309: FutureWarning: The `op` parameter is deprecated and will b
e removed in a future release. Please use the `predicate` parameter instead.
  if await self.run_code(code, result, async_=asy):
```



Ford Stock Prices

Now train a logistic regression model on covid ~ age + risk. And Interpret it.

In [ ]:
```
ford_df = pd.read_csv('Ford-Stock-Prices.csv')
ford_df.head()
ford_df
```

Out[ ]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	1972-06-01	0.000000	2.173495	2.149165	2.153220	0.115848	1091238
1	1972-06-02	2.153220	2.173495	2.141055	2.149165	0.115630	1174468
2	1972-06-05	2.149165	2.169440	2.141055	2.149165	0.115630	5209582
3	1972-06-06	2.149165	2.157275	2.116725	2.124835	0.114321	1424158
4	1972-06-07	2.124835	2.137000	2.112670	2.112670	0.113666	675088
...	...	...	...	...	...	...	...
12558	2022-03-18	16.420000	16.900000	16.280001	16.860001	16.860001	89050700
12559	2022-03-21	16.870001	17.110001	16.330000	16.480000	16.480000	72583600
12560	2022-03-22	16.719999	17.309999	16.700001	17.090000	17.090000	77647100
12561	2022-03-23	16.820000	17.049999	16.570000	16.680000	16.680000	63296800
12562	2022-03-24	16.740000	16.980000	16.639999	16.830000	16.830000	56119200

12563 rows × 7 columns

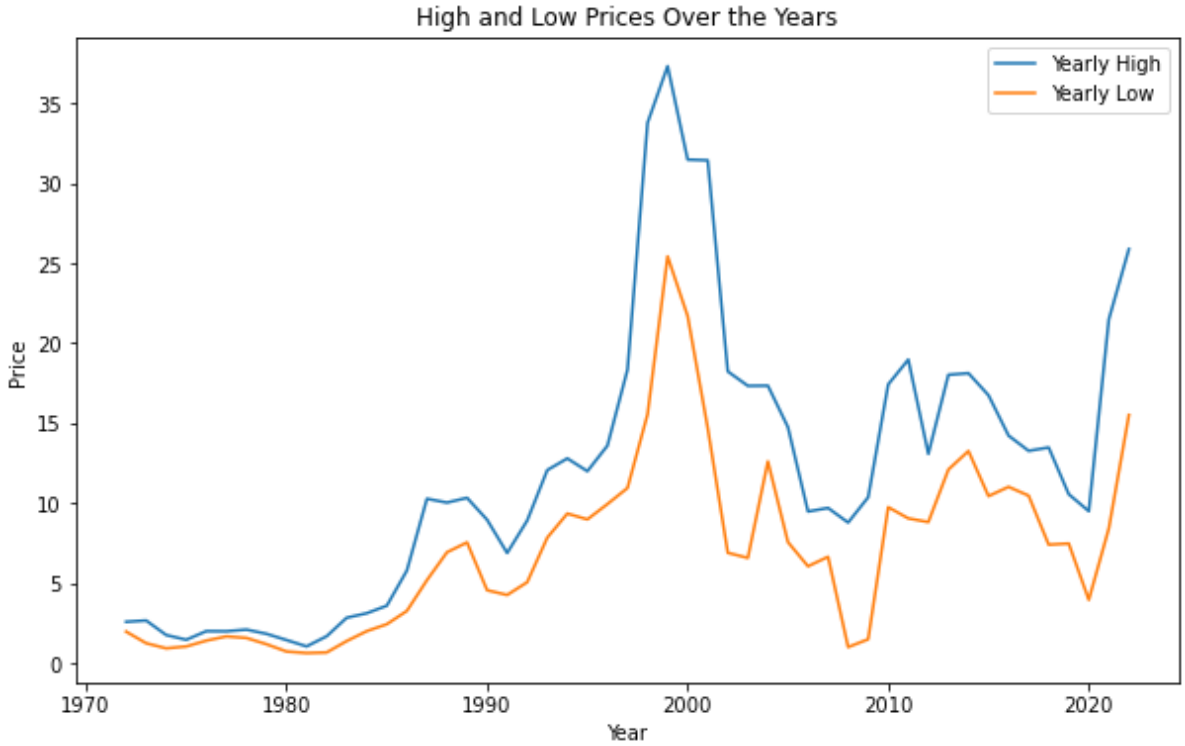
In [ ]:
```
ford_df.columns
```

Out[ ]:
```
Index(['Date', 'Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume'], dtype
='object')
```

3. How has the stock prices varied over the years? When has it been the highest and when has it been the lowest?

In [ ]:
```
ford_df['Date'] = pd.to_datetime(ford_df['Date'])
ford_df.set_index('Date', inplace=True)

# Resample the data by year and get the maximum and minimum 'High' and 'Low'
yearly_high = ford_df['High'].resample('Y').max()
yearly_low = ford_df['Low'].resample('Y').min()

# Create a new figure
fig, ax = plt.subplots(figsize=(10, 6))

# Plot the yearly high and low prices as two separate lines
ax.plot(yearly_high.index.year, yearly_high.values, label='Yearly High')
ax.plot(yearly_low.index.year, yearly_low.values, label='Yearly Low')

# Set the title, x-axis label, y-axis label, and legend
ax.set_title('High and Low Prices Over the Years')
ax.set_xlabel('Year')
ax.set_ylabel('Price')
ax.legend()
```

```
# Display the plot
plt.show()
```



Did seeing the results of this analysis help you come up with any ideas/speculations about why customers may be leaving the bank? How may you test whether these ideas are correct (e.g., what data do you need, what analysis would you run)? Often regression is useful for generating hypotheses that you can then go out and test!

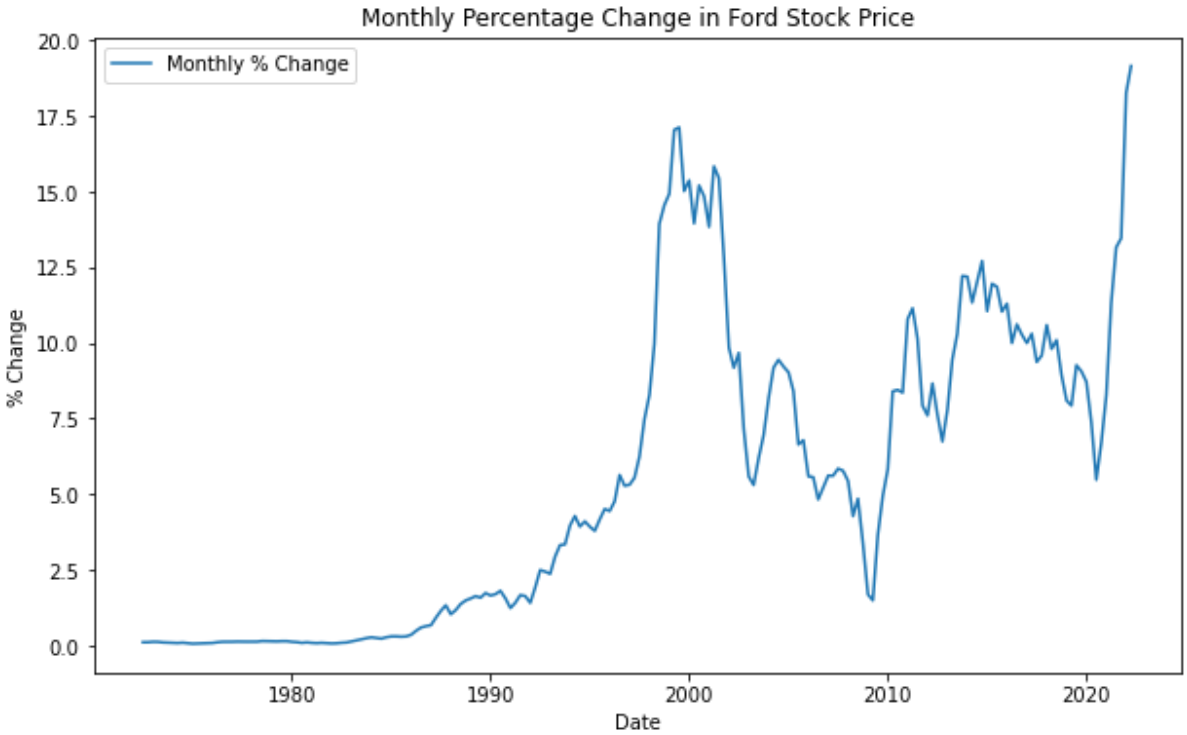
In [ ]:
```
highest_high = ford_df['High'].max()
lowest_low = ford_df['Low'].min()

# Find the dates where these prices occur
highest_high_date = ford_df.loc[ford_df['High'] == highest_high, 'High'].ind
lowest_low_date = ford_df.loc[ford_df['Low'] == lowest_low, 'Low'].index[0]

# Print the results
print("Highest 'High' price: ${:.2f} on {}".format(highest_high, highest_hig
print("Lowest 'Low' price: ${:.2f} on {}".format(lowest_low, lowest_low_date
```

```
Highest 'High' price: $37.30 on 1999-04-19
Lowest 'Low' price: $0.64 on 1981-10-29
```

4. Are there any seasonal patterns in the stock price?

In [ ]:
```
# Compute the monthly percentage change in the adjusted closing price
monthly_pct_change = ford_df['Adj Close'].resample('Q').mean()

# Create a new figure
fig, ax = plt.subplots(figsize=(10, 6))

# Plot the monthly percentage change as a line chart
ax.plot(monthly_pct_change.index, monthly_pct_change.values, label='Monthly

# Set the title, x-axis label, y-axis label, and legend
ax.set_title('Monthly Percentage Change in Ford Stock Price')
ax.set_xlabel('Date')
ax.set_ylabel('% Change')
```

```
ax.legend()

# Display the plot
plt.show()
```



Lead Blood Levels

In [ ]:
```
lead_df = pd.read_csv('blood.csv')
lead_df.head()
lead_df
```

Out[ ]:

	OBJECTID	GEOID10	CntTested	EBLL	Under6CntTested	Under6EBLL	Under18CntTeste
0	1	26123	569	25	499	24	53
1	2	26157	997	39	862	31	90
2	3	26141	115	3	96	3	10
3	4	26025	2902	351	2711	336	277
4	5	26111	546	5	370	3	40
...	...	...	...	...	...	...	
78	79	26105	464	39	429	39	44
79	80	26035	514	50	432	15	44
80	81	26079	291	2	211	1	22
81	82	26101	368	22	320	20	33
82	83	26161	3563	103	2951	75	314

83 rows × 13 columns

In [ ]:
```
lead_df.columns
```

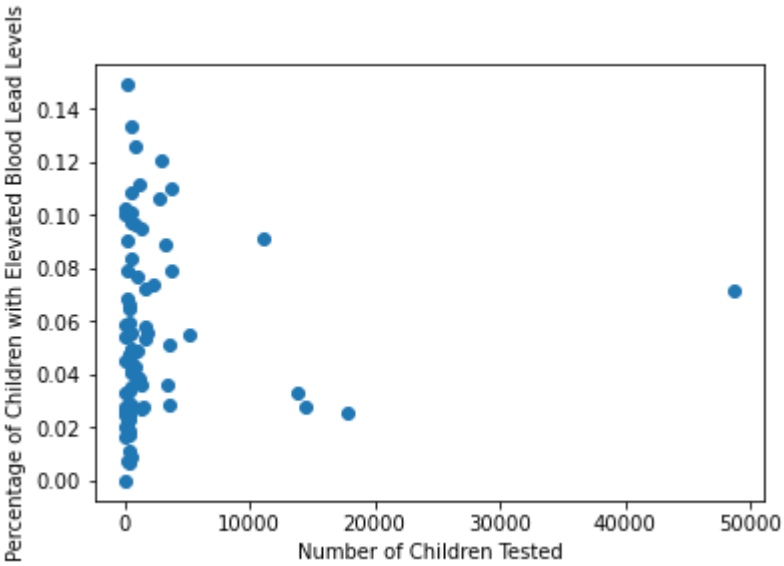
Out[ ]:
```
Index(['OBJECTID', 'GEOID10', 'CntTested', 'EBLL', 'Under6CntTested',
       'Under6EBLL', 'Under18CntTested', 'Under18EBLL', 'PctEBLL',
       'PctUnder6EBLL', 'PctUnder18EBLL', 'Shape__Area', 'Shape__Length'],
      dtype='object')
```

5. What is correlation coefficient between the number of children tested and the percentage with elevated blood levels?

Now, can you say something about what makes for a *high* quality wine? That is, use the `quality` column to label some wines as high quality and some as low quality. Then do some exploratory analysis and run a regression.

In [ ]:
```
# Calculate the percentage of children with elevated blood lead levels
lead_df['PctEBLL'] = lead_df['EBLL'] / lead_df['CntTested']

# Calculate the percentage of children under 6 with elevated blood lead leve
lead_df['PctUnder6EBLL'] = lead_df['Under6EBLL'] / lead_df['Under6CntTested'

# Calculate the correlation coefficient between the number of children teste
corr_coef = lead_df['CntTested'].corr(lead_df['PctEBLL'])

# Print the correlation coefficient
print(f"Correlation coefficient: {corr_coef}")

# Create a scatter plot of the number of children tested vs. the percentage
plt.scatter(lead_df['CntTested'], lead_df['PctEBLL'])
plt.xlabel('Number of Children Tested')
plt.ylabel('Percentage of Children with Elevated Blood Lead Levels')
plt.show()
```

```
Correlation coefficient: 0.049192335194929465
```



Unemployment rates in US

In [ ]:
```
unemployment_df = pd.read_csv('unemployment.csv')
unemployment_df.head()
unemployment_df
```

Out[ ]:

	Calendar Date	Date	Unemployment_Rate_MI	Unemployment_Rate_US	Rank	Highlights
0	01/1976	Jan-76	0.099	0.079	NaN	NaN
1	02/1976	Feb-76	0.099	0.077	NaN	NaN
2	03/1976	Mar-76	0.099	0.076	NaN	NaN
3	04/1976	Apr-76	0.098	0.077	NaN	NaN
4	05/1976	May-76	0.096	0.074	NaN	NaN
...	...	...	...	...	...	...
559	08/2022	Aug-22	0.041	0.037	38.0	Michigan's seasonally adjusted unemployment ra...
560	09/2022	Sep-22	0.041	0.035	43.0	Michigan's seasonally adjusted unemployment ra...
561	10/2022	Oct-22	0.042	0.037	41.0	Michigan's seasonally adjusted unemployment ra...
562	11/2022	Nov-22	0.043	0.036	43.0	Michigan's seasonally adjusted unemployment ra...
563	12/2022	Dec-22	0.043	0.035	44.0	Michigan's seasonally adjusted unemployment ra...

564 rows × 6 columns

6. What is the highest and lowest unemployment rate in Michigan?

In [ ]:

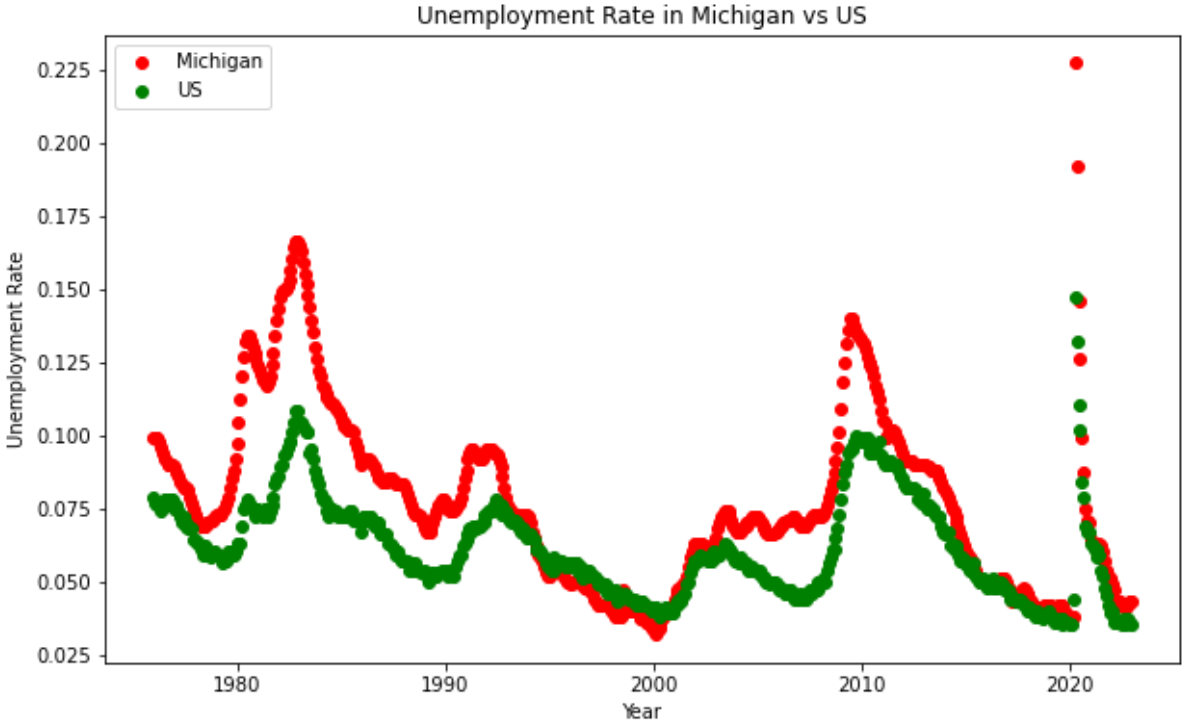
```
fig, ax = plt.subplots()
ax.boxplot(unemployment_df['Unemployment_Rate_MI'])
ax.set_xlabel('Michigan')
ax.set_ylabel('Unemployment Rate')

# set the tick locations and labels
ax.set_xticks([1])
ax.set_xticklabels([''])
ax.set_title('Distribution of Unemployment Rates in Michigan')
```

```
# show the plot
plt.show()
```

7. Compare the unemployment rates in Michigan and the United States over time.

In [ ]:

```
# Convert 'Calendar Date' to datetime format
unemployment_df['Calendar Date'] = pd.to_datetime(unemployment_df['Calendar

# Set 'Calendar Date' as the index
unemployment_df.set_index('Calendar Date', inplace=True)

# Create a scatter plot of the unemployment rates for Michigan and the US
fig, ax = plt.subplots(figsize=(10, 6))

ax.scatter(unemployment_df.index, unemployment_df['Unemployment_Rate_MI'], l
ax.scatter(unemployment_df.index, unemployment_df['Unemployment_Rate_US'], l

ax.set_title('Unemployment Rate in Michigan vs US')
ax.set_xlabel('Year')
ax.set_ylabel('Unemployment Rate')
ax.legend()

plt.show()
```


Unemployment Rate in Michigan vs US
Michigan
US
Unemployment Rate
0.225
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
1980
1990
2000
2010
2020
Year