Predicting Crime Frequency in Prince George’s County, Maryland

Cole Bromfield, Edmund Park, Dennis Reyes

University of Maryland, College Park

DATA602 – PCS1, Fall 2023

Prof. Mohammad T. Hajiaghayi

**Predicting Crime Frequency in Prince George’s County, Maryland**

**Introduction**

Located in Maryland just outside of Washington, D.C., Prince George’s County offers comprehensive data on crime incidents within the county, going back to 2017. Using Python in conjunction with SQL, Json, and essential libraries such as requests, Pandas, and Prophet, this data can be put into a structured format that is conducive to easy analysis and visualization. Predictive methods such as regression analysis and Prophet’s additive models allow for effective forecasts regarding the aggregate quantity of total crime occurrences, as well as more specific subcategories of crime such as homicide, violent crime, etc.

**Background**

The relationship between crime and the calendar is a long-discussed topic, beyond simple long-term trends—it is common knowledge that murder rates increase in the summer, resulting in the famous correlation between ice cream sales and murder, oft-used by statistics professors in an attempt to explain the difference between correlation and causation. What other trends can we find when we examine comprehensive crime data that spans over six years? Can we effectively predict the number of total crime occurrences in Prince George’s county in the next month? Can we achieve greater precision and make the same prediction for the next week? With recent weeks utilized as testing data, the goal of this project is to provide an effective model that delivers to the user a reliable prediction for the total number of crime occurrences that will take place in the county across the ensuing seven days, accompanied by a helpful graphic and (admittedly less reliable) day-by-day breakdown

**Acquiring the Data**

The historical data was found at data.princegeorgescountymd.gov and imported using requests and Json. We converted the data to a list of dictionaries, each containing the details of an individual crime occurrence. From there, we were able to construct a Pandas DataFrame consisting of every crime occurrence from February of 2017 up until July of 2023. For each crime instance, this DataFrame showed the case ID, date, crime type (e.g. theft, homicide, etc.), reporting area, sector of the county (represented by a letter), beat (subsections of each sector), street at which the incident took place, latitude, longitude, and street number. The data pertaining to July of 2023 through the present was found at a separate location on the same website, and the same process for importation was used to create a DataFrame from this data before the two DataFrames were merged into one. Only minor cleanup was required at this point—using Python’s Datetime package, the entries in the “Date” column were converted to Datetime objects, allowing for easier date-related analysis. This allowed for the creation of an additional column displaying the day of the week for each crime occurrence, potentially to be used as an extra feature in making our prediction. The head for the cleaned DataFrame can be seen in figure 1 A screenshot of a computer

Description automatically generatedbelow.

Figure

**Regression Analysis**

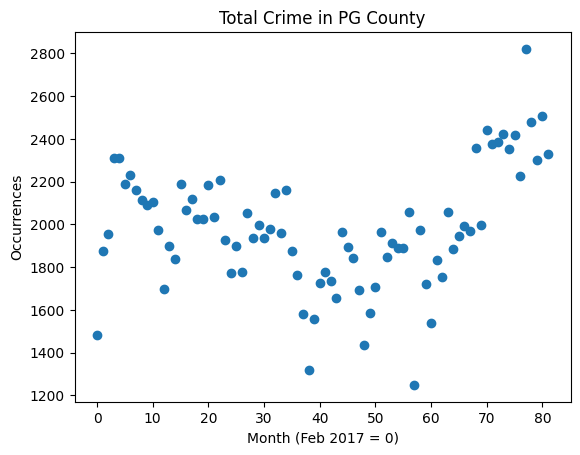
To generate predictive data about aggregate crime throughout the county, we turned to a logistic regression model, specifically polynomial regression. From the data we compiled for February of 2017 through November of 2023, we found the best method to be to sort all of the crime instances into one-month bins, displaying the total number of crimes occurring in any given month. This process required extensive effort to clean up and process the initial data, but after this initial processing we were able to put the monthly data into a scatter plot, providing a clear visual for the total number of instances of crime for each month in the specified period (Figure 2). To minimize complexity of regression analysis, the x-axis represents the number of months passed since February of 2017 (in other words, February of 2018 falls at , February of 2019 falls at , etc).

Figure 2

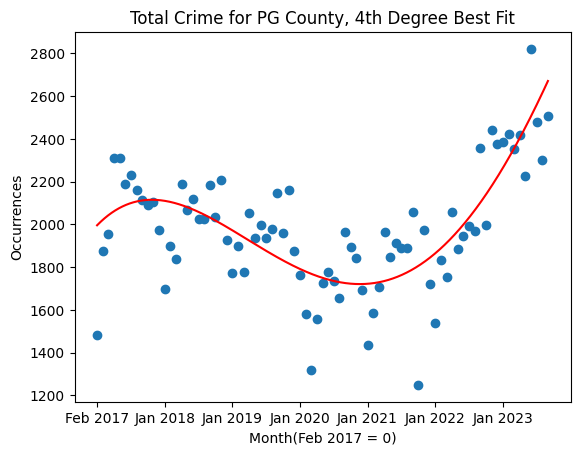
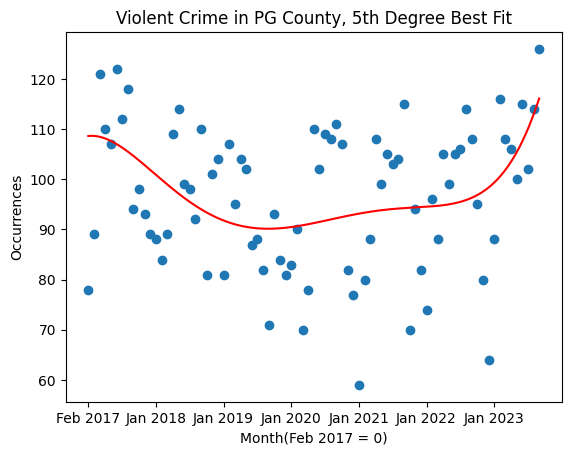
The subsequent task was to fit this data to a curve. To avoid a polynomial that would overfit or underfit the data, we ran the regression calculation using 80% of the data as training data for every possible degree of polynomial, from 1 (a line) to 5, to determine which generated the lowest mean absolute error with the remaining 20% of the data, used as testing data. Finally, we used this model to predict the amount of total crime in November of 2023 to test the accuracy before using the model to predict December of 2023. Ultimately, the degree 4 polynomial (see figure 3) was found to generate the lowest mean absolute error, and predicted 2,789 total crime instances for the month of December 2023. To determine the overall effectiveness of this model, we used a Mean Adjusted Percentage Error (MAPE) calculation with 20% of the datapoints, which were withheld as test data. The MSE for a model can be calculated as follows:

Figure 3

(Where represents the predicted/estimated value for , and N is the total number of data points in the testing data)

With the overall crime data calculation, we found a MAPE of 17.25%.

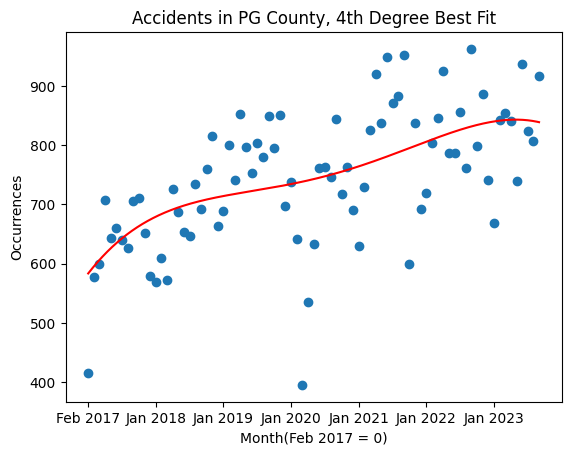
An identical process was done with various types of crime. After separating violent crime into its own DataFrame (using the categories “Homicide”, “Assault”, “Assault, Weapon”, and “Assault, Shooting”, we performed the same regression analysis, resulting in a 5th degree best fit polynomial (Figure 4). Under this model, we found a prediction of 124 violent crime occurrences for December of 2023 with a MAPE of 4.20.

Figure 4

Likewise, car accidents can be tracked in a similar fashion. As we can see in Figure 5, there exists a weakly positive correlation between time and monthly accidents, and the model predicts 833 total car accidents for December of 2023, with a MAPE of 4.06. However, there is a caveat when predicting car accidents—their distribution across any given year is not uniform. During the winter, a greater number of car accidents will naturally occur, due to the greater likelihood of hazardous driving conditions (e.g snow, ice, sleet). Therefore, we raised the question of whether a regression model that inputs two features may prove more useful in the case of car accidents (the same can be said for violent crime and its higher likelihood in warmer months), with the second feature being the integer value of the month (e.g. 10 for October), not including the year. Interestingly, this value actually comes out lower. However, this is likely due to the overall upward trend in the data more than cancelling out the seasonal differences in accident quantity. As a result, we determined that it was not sensible to use the integer value of the month as an additional feature, and instead stuck with the more common, single input value linear regression calculation to predict total crime for the ensuing month.

Figure 5

The next step was to narrow our prediction to an interval more compatible with a live prediction model—namely the next week as opposed to the next month. Although we attempted to use the same polynomial interpolation model as we had used for months, we found that it was considerably more computationally expensive, and opted instead to use Prophet, a Meta-created Python package intended to generate predictions based on time series data. Prophet’s prediction model carries the additional benefit of accounting for seasonality (e.g. month, season, day of the week, etc) and use of uncertainty intervals. Our new Prophet-generated predictive model can be seen in figure 6 on the following page. For a more accurate model, we removed outliers (figure 7). As an example projection, the A graph with blue lines and black dots

Description automatically generatedlatter model predicts 650 total crime incidents for December 11th through December 17th. Finally, we generated a third Prophet A graph with blue and black dots

Description automatically generatedmodel that incorporated holidays (figure 8). These charts include a scatterplot of the logs of all daily crime data as well as Prophet’s generated trend track and the calculated uncertainty intervals on either side of the calculated track. The prediction is based on the A graph with blue and black dots

Description automatically generatedcombined values found on this track for each of the next seven days.

Figure 6

Figure 7

Prophet also allows for the generation of more detailed componentwise trend graphs (see figure 9, next page). To analyze the effectiveness of each of these methods, we again used the MAPE.

Figure 8

A screenshot of a graph

Description automatically generatedThe results were as follows: 25.31 for the first model (including outliers, not counting holidays), 13.25 for the second (excluding outliers, not counting holidays), and 13.21 for the third model, which excluded outliers and counted holidays, giving the third model a very slight edge over the second.

Figure 9

**Frontend**

In developing the frontend, we harnessed the capabilities of React, Javascript, HTML, and a Restful API. We built a web application locally, there is a [README.md](https://github.com/dreyes53/data602-crime/blob/master/README.md) to set up the frontend and backend locally. Upon initializing the web application, a GET method API endpoint is triggered, establishing a connection to http://localhost:3000 with the specific endpoint /crime-predictions. The ensuing response encompasses the latest seven days' crime prediction data, the base64 encoded bytes of the projected plot, and the Mean Absolute Percentage Error (MAPE) calculations for our predictions. Utilizing this response, we dynamically present the dataset in a tabular format, decode the base64 encoded plot to showcase it as an image, and employ a Python script, [generate\_plot\_images.py](https://github.com/dreyes53/data602-crime/blob/master/backend/generate_plot_images.py), to create additional plot images (figure 10). These images find residence in the public directory as static images, enhancing their accessibility for rendering on the frontend.

A screenshot of a graph

Description automatically generated

Figure 10

**Backend**

For the backend, we employed Flask API, SQLite, Python, Restful API, and Docker. The aim was to establish a backend server for seamless Restful API interactions with the frontend. We devised a Dockerfile, enabling the building and running of our docker container locally, connecting to http://localhost:5000. Our initial step involved creating an SQLite database file, accomplished through a [Jupyter notebook](https://github.com/dreyes53/data602-crime/blob/master/ipynb_files/convert_historical_data_sqlite.ipynb) utilizing historical data from PG County. This transition to SQLite sought to minimize network latency when retrieving historical data. Subsequently, we implemented an endpoint that utilizes pagination to invoke the PG County API. Leveraging pandas, we read the SQLite database file and merged it with the retrieved dataset. This amalgamation facilitated our predictive modeling, enabling the generation of plots and datasets for crime predictions over the next seven days. Additionally, we utilized Github and established the repository https://github.com/dreyes53/data602-crime for collaborative code development.

**Future Directions:**

In our envisioned future state, we plan to deploy both the front-end and back-end, hosting servers on AWS rather than solely relying on local environments. Although we encountered challenges during testing with AWS deployment, we anticipate overcoming these hurdles. Furthermore, we aim to expand our predictive capabilities, not only forecasting for the next 7 days but extending to predictions for the next 30 days or even 365 days of crime trends.