UCDPA – John Lenehan

# GitHub URL

https://github.com/jlenehan/UCDPA\_JohnLenehan

# Abstract

An analysis is conducted on traffic collision data in Chicago from 2013 to present, following which a machine learning model is constructed to predict collision casualties (i.e. injuries or fatalities) based on factors such as [INSERT PROMINENT FEATURES HERE]. This analysis shows that [EXPAND]

# Introduction

For this project, the use case is defined as a machine learning tool for first responders in the Chicago municipal area to predict the likelihood of collisions requiring a callout. This tool would be beneficial in allocating resources to different parts of the city based on weather conditions and time of the week or year [OTHER FEATURES], to allow for faster response times to the scenes of traffic collisions.

# Dataset

The dataset used for this analysis is a merging of data from two sources; a live dataset of traffic collisions in Chicago from 2013 to present [1], and a static dataset of Chicago PD beats [2]. The beats data is used to determine which district the collision took place in, as this information wasn’t contained in the original dataset.

# Implementation Process

Step 0: Import Libraries

To begin, the necessary libraries for analysis must be imported; these are laid out below:

A screenshot of a computer code

Description automatically generated with low confidence

Additionally, functions from the machine learning module sci-kit learn (sklearn) are imported to build a machine learning engine; these functions are shown below:

A screen shot of a computer program

Description automatically generated with low confidence

Step 1: Import Data

Step 1 is to import the data; the collision data is imported from the web using the pd.read\_json() function. The data sources used in this project can be found at the below addresses:

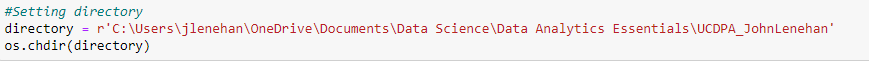
|  |  |  |
| --- | --- | --- |
| **Data name** | **Data Source (URL)** | **JSON link** |
| Collisions Data | https://data.cityofchicago.org/Transportation/Traffic-Crashes-Crashes/85ca-t3if [1] | https://data.cityofchicago.org/resource/85ca-t3if.json?$limit=99999999 |
| Beats Data | https://data.cityofchicago.org/Public-Safety/Boundaries-Police-Beats-current-/aerh-rz74 [2] | Data downloaded and imported as a csv file |

Note that the query “?$limit=100000000” is added to the json string of the collision dataset, to increase the json row limit from 1,000 to 100,000,000. This is to ensure that all data will be imported in the data pull.

A screen shot of a computer

Description automatically generated with low confidence

For the CPD beats data, the data is downloaded and stored as a csv file (included in the zip file for this assignment) and the data is read into the jupyter notebook using the .read\_csv() function. This method is used as the beats and district data is static, therefore the data wouldn’t benefit from being pulled live. The directory of the notebook is set to the assignment folder using the os.chdir() function:



From there the file can be read into the read\_csv() function locally:



Step 2: Merge Data

With all individual dataframes now cleaned and aggregated to show the desired data, these data must now be merged to produce the final amalgamated dataframe for analysis. Both joins used for this process are left joins, as the primary goal for this analysis is evictions; any data from the income or population datasets that doesn’t correspond with the can be dropped using the dropna() function.

Firstly the income dataframe is merged to the population data using a left join, on the borough column; this has the effect of attaching the 2020 population data to all rows in the income dataframe, irrespective of year:

A picture containing text, font, screenshot, line

Description automatically generated

This produces a dataframe of evictions across the 5 boroughs from 2017 to 2020, with income data accurate to each year and population data by borough from the 2020 census, assumed to be constant for this analysis. Some further cleaning of the new “NY\_Merged” dataframe is required; rows with null values are dropped using dropna(), redundant columns from the joins are filtered out, columns are renamed, and the year column is changed from a number entry to a string (a string entry is preferrable for the plotting step):

Chart, scatter chart

Description automatically generated with medium confidence

Subsequently the evictions data can be normalized to the population data by dividing the ‘Sum Evictions’ column by the ‘Population’ column. Furthermore the upper and lower margins of the household income can be calculated by either adding or subtracting the ‘Household Income MOE’ column from the ‘Household Income’ column.

Text

Description automatically generated

Lastly the correlation function .corr() is run to show how each column correlates against the others:

Table

Description automatically generated

Step 3: Describe Data

Once the dataset is merged to include the district data, the methods .columns, .shape, and the functions .info, and .describe are used to give an idea of what each dataset looks like.

A picture containing text, font, screenshot

Description automatically generated

Additionally the unique values of each dataset column is printed using a for loop, along with a count of unique values for each column:

A screen shot of a computer code

Description automatically generated with low confidence

The eviction data has a number of columns which could be useful for analysis; the marshal first and last names are given, along with latitude and longitude data of each eviction, and the council district in which the eviction was carried out. The data shown is for the past 6 years (2017 to 2022), across all 5 boroughs of New York city. For this analysis the scope is limited to the number of evictions by borough and year – for these columns the .info() function shows no null values, so it won’t be necessary to drop null values from this dataset. The year data is captured in the eviction\_date column but not in a form useful for analysis, so this needs to be extracted.

Step 3: Clean and Manipulate Data

1. Eviction Data

From looking at the eviction data, the executed\_date column is stored as an object datatype which makes datetime functions difficult – as such this needs to be converted to a datetime datatype. This is done using the pandas to\_datetime() function. From there the .year method is used on the DatetimeIndex() function to extract the year of each eviction.

A picture containing text, font, screenshot, line

Description automatically generated

Following this a count of evictions via the eviction\_zip column is executed, producing a pivot table grouped by year and borough using groupby(). For further analysis this is then ungrouped using reset\_index(), and the column is renamed from eviction\_zip to sum\_evictions. This gives the final evictions dataset of evictions by year and borough.

Text

Description automatically generated with medium confidence

Step 5: Plot Data

Seaborn lineplots are first deployed showing the evictions and evictions per 1,000, with hue set to borough to show the distinction between boroughs. This is done using the .lineplot() function:

Text

Description automatically generated

Next a scatter plot is produced showing the evictions per 1,000 against household income, grouped by borough:

Text

Description automatically generated

A custom function is defined to produce a scatter plot with a trendline using lnplot(), along with a correlation of the variables using .corr(); this function is named plot\_correlation():

Text

Description automatically generated

Lastly this new function is used to show the correlation of evictions per 1,000 against 2 income variables in the dataframe; household income, and household income MOE:

Text

Description automatically generated

# Results

Chart, line chart

Description automatically generated

Figure 1: Graph showing total eviction trends by borough from 2017 to 2020

Chart, line chart

Description automatically generated

Figure 2: Graph showing eviction trends per 1,000 population by borough from 2017 to 2020.

Table

Description automatically generated

Figure 3: Table of Evictions per 1,000 by borough and year, subsetted from the NY\_Merged dataframe.

Chart, line chart

Description automatically generated

Figure 4: Line plot of household income by year, grouped by borough.

Chart, scatter chart

Description automatically generated

Figure 5: Scatter plot of household income against evictions per 1,000, grouped by borough.

Chart, scatter chart

Description automatically generated

Figure 6: Correlation analysis of evictions per 1,000 against mean household income.

Chart, scatter chart

Description automatically generated

Figure 7: Correlation analysis of evictions per 1,000 against mean household income margin of error (standard deviation).

# Insights

* Insight 1
* Insight 2
* Insight 3
* Insight 4
* Insight 5

# References

[1] Levy, J. (n.d.). Traffic Crashes - Crashes [Dataset]. Retrieved from Chicago Data Portal: https://data.cityofchicago.org/Transportation/Traffic-Crashes-Crashes/85ca-t3if (Accessed: 14th May 2023).

[2] Chicago Police Department. (n.d.). Boundaries - Police Beats (current) [Data set]. Retrieved from Chicago Data Portal: https://data.cityofchicago.org/Public-Safety/Boundaries-Police-Beats-current-/aerh-rz74 (Accessed: 14th May 2023).