Forecasting Fatalities in NYC

DSCT Capstone 1 - Data Wrangling
Jonathan D. Williams

Data Acquisition

The dataset for this project is the NYC EMS Incident Dispatch Data, which is made publicly available via NYC Open Data. This source contains data that is generated by the EMS Computer Aided Dispatch System, and spans from the time the incident is created in the system to the time the incident is closed in the system. It covers information about the incident as it pertains to the assignment of resources and the Fire Department's response to the emergency.

I opted to export the data as a CSV file and perform all analyses on a local machine. In doing so, however, it became apparent that there were some disadvantages from using this particular approach:

- **RAM restrictions.** At the time of this writing, the CSV file contained over 8.5 million records and occupied just over 2GB of hard disk space. The process of reading this information into a single DataFrame object--or several smaller DataFrame objects--was heavily taxing on the system memory and often resulted in task failures.
- **Lack of scalability.** This dataset is updated on a periodic basis to reflect new information extracted from the EMS Computer Aided Dispatch System. A static file does not capture these changes, and any future analysis will require the acquisition of a new (and larger) CSV file.

As an alternative, the data was obtained via the Socrata Open Data API (SODA) as batches of JSON files that were converted to multiple Pandas DataFrame objects. These DataFrame objects were then appended to a list.

Data Wrangling and Cleaning

Three functions were created to clean and transform the dataset as needed for this analysis: `drop_from_df`, `modify_dtype`, and `redesign_df`.

> **`drop_from_df`(*DataFrame object*)**
>
> The target variable for this analysis is based on the `incident_disposition_code` of the dataset. The `drop_from_df` function removes all rows where data is missing for this field. In addition, the function identifies and removes immaterial rows and columns from the dataset that either contain outliers or redundant geographic information that will confound analyses.

`modify_dtype`(*DataFrame object*)

This function converts the data types of all columns that contain ISO8601 information from `str` to `datetime`. Also, this function also ensures that the data types for columns that contain numeric data are set as `int` or `float` types in the event the correct data type is not inferred during import. Finally, select columns that contain categorical information are converted to category types in order to reduce the size of the DataFrame object in memory.

`redesign_df`(*DataFrame object*)

This function creates a boolean series called `fatality` that serves as the target variable for the binary classification models that will be used later in this project. In addition, it parses the values contained within the `incident_datetime` column and creates two new series that contain the corresponding year (`incident_year`) and month (`incident_month`) for each record. These new columns are then used to create a hierarchical index for the corresponding DataFrame object.

Data Formatting

The aforementioned functions were applied to each object contained within the list frames. All objects within frames were merged to form a single DataFrame object. Finally, this resulting DataFrame was exported to a CSV file using gzip compression in an effort to significantly reduce the file size from that of the original dataset.