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 and concatenated to form the core dataframe.

Data Wrangling and Cleaning

Four functions were created to clean and transform the dataset as needed for this analysis: `drop_errors`, `drop_immaterial`, `reduce_memory`, and `format_df`.

`drop_errors`(*DataFrame object*)

This function removes any observation with a missing value for incident_disposition_code since the target variable is derived from this feature of the dataset. In addition, the function identifies and removes all rows that contain invalid duration metrics as given by valid_dispatch_rspns_time_indc and valid_incident_rspns_time_indc.

`drop_immaterial`(*DataFrame object*)

This function removes all observations that contain outliers and columns with immaterial information such as:

- incidents created to transport a patient from one facility to another
- incidents where units were assigned to stand by in case they were needed
- incidents that pertain to special events
- incidents that were once closed but later reopened
- features that contain redundant geographic information for incident

`reduce_memory`(*DataFrame object*)

The purpose of this function is to reduce the size of the DataFrame object in memory. It converts the data types of all columns that contain ISO8601 information from `str` to `datetime`. 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 `object` to `category` types.

`format_df`(*DataFrame object*)

This function creates a boolean series called fatality that serves as the target variable for 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 combined with cad_incident_id to create a hierarchical MultiIndex.

The aforementioned functions were each applied to the core dataframe. Finally, the resulting clean 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.