Fatal Injury and Violence Analytics

Proof of Concept

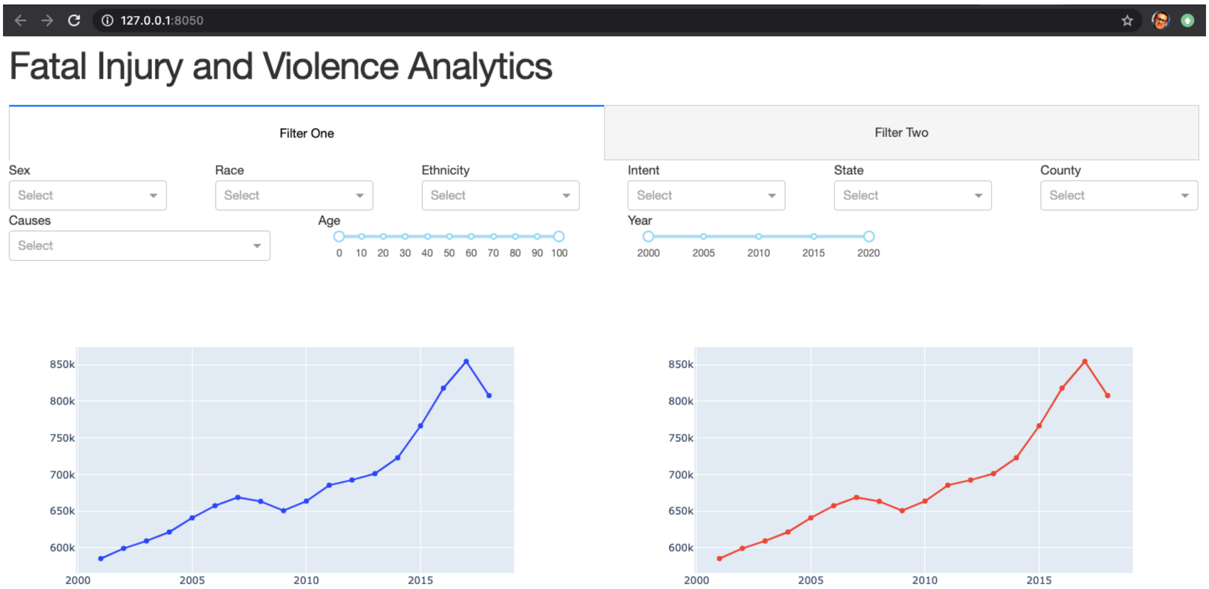
The proposed software solution is based on Dash, which is a Python framework broadly used to implement data science web dashboards.

Dash is the preferred solution for the following reasons:

- Dash has a free license
- Dash can be easily integrated with Python data analytics since it is based on Python
- Dash allows the implementation of advanced web dashboards with less efforts with respect to other web frameworks

User Interface

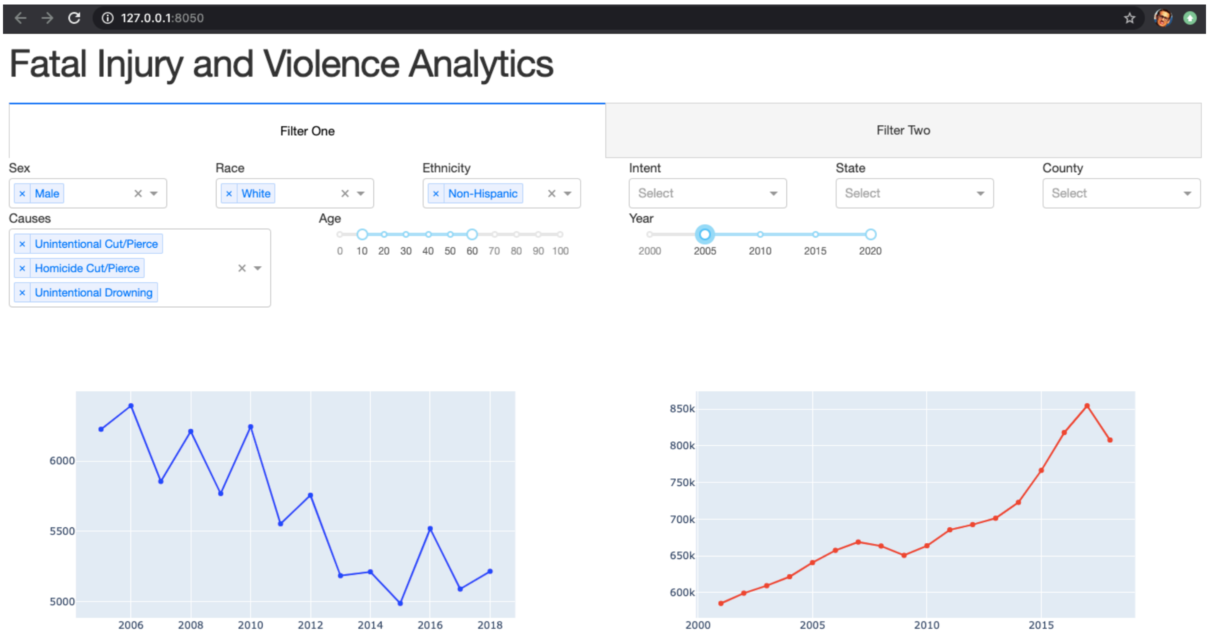
The user interface of the dashboard is depicted in the following picture:



Fatal Injury and Violence Analytics

At the top of the page the user can apply various filters to select different subsets of data. It is worth noticing that there are two tabs at the top that allows to specify two different filters in order to perform comparative analysis between different subsets.

The line plots show the total amount of deaths aggregated per year. The plots are automatically update once a new filter is added or removed by the user. The following picture shows an example with multiple filters and the corresponding output.



127.0.0.1:8050

Fatal Injury and Violence Analytics

Filter One | Filter Two

Sex: × Male

Race: × White

Ethnicity: × Non-Hispanic

Intent: Select

State: Select

County: Select

Causes: × Unintentional Cut/Pierce, × Homicide Cut/Pierce, × Unintentional Drowning

Age: 10 – 60

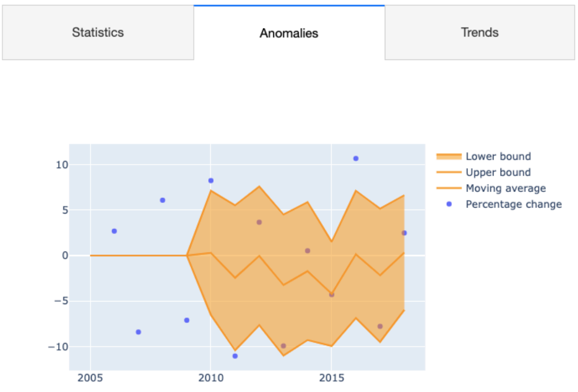
Year: 2005 – 2020

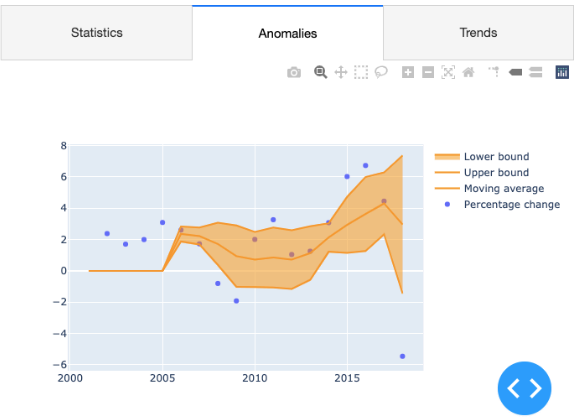
At the bottom of the dashboard there are two sections that allows to perform more advanced analytics. The two sections report the information derived from the two filters explained above.

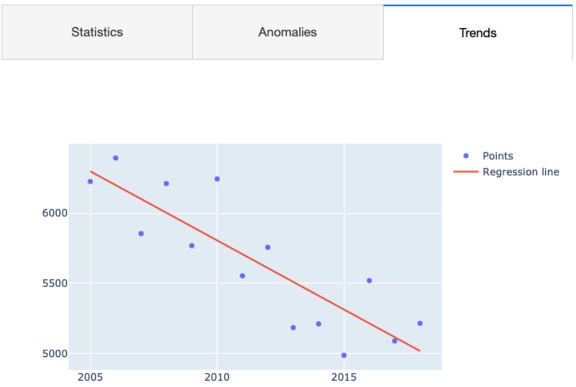
There are three tabs for each section:

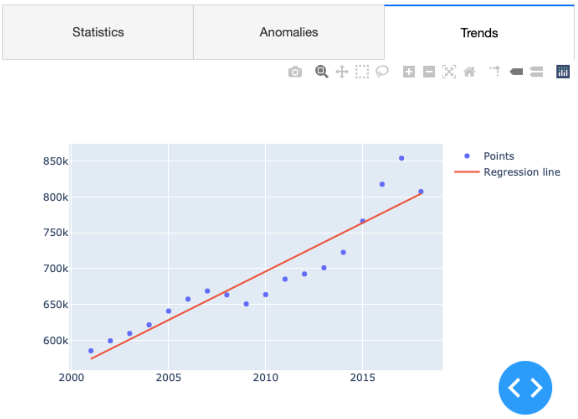
- Statistics shows the distribution of a specific feature once the filters have been applied. A dropdown menu allows to select the feature's distribution to be analyzed. The statistics tab should be integrated with additional indicators such as the mean, the standard deviation, the median and other basic statistics.
- Anomalies shows the percentage change of deaths for each year. The information about percentage changes can be used to detect trend changes, drops, anomalies and spikes. Indeed, we can compute the moving average of the past years in order to recognize strong deviations from past changes. The approach is explained in more details in the following Stack Overflow thread: https://stackoverflow.com/questions/22583391/peak-signal-detection-in-realtime-timeseries-data?noredirect=1&lq=1. The algorithm implementation is quite simple. The hyperparameters of the algorithms such as the number of past years for the mean and the number of standard deviations should be specified by the user.
- Trends shows the regression line obtained by fitting a linear regression model to the filtered subset of data. The regression line provides a visual representation of the relevant trends present during the specified period. Furthermore, we can also provide a quantitative measure of the trends by using the slope of the regression line.

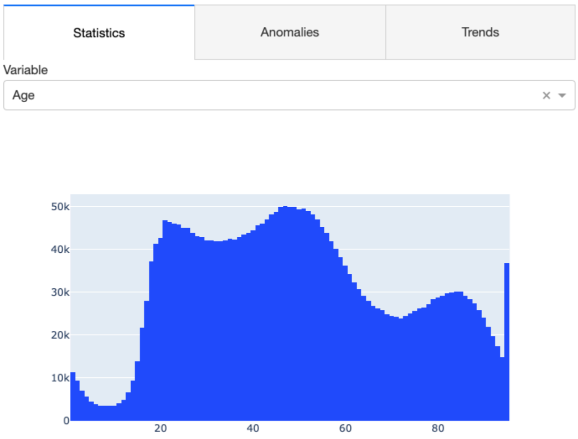
The following pictures provide some examples of the outputs of the three tabs.





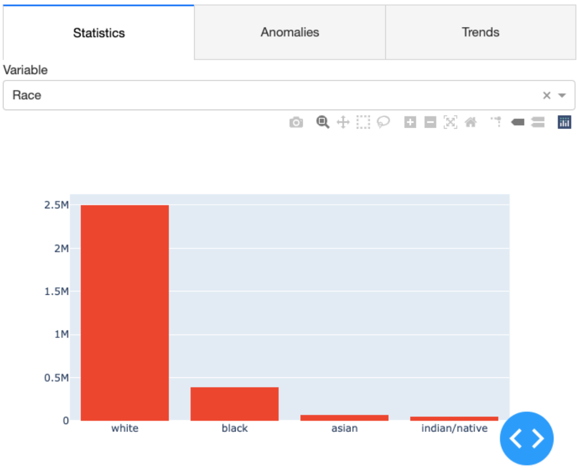






Variable

Age



Variable

Race

Deployment

The deployment of solution is the easiest part because it is a well standardized process with a lot of documentation. The best solution is to deploy a Docker container hosted on AWS with the Python application running on it.

Performance

The performance of the software solution is very good for data visualization. However, the application of filters to Pandas data frames does not show great performance and seems to require too much time (5 seconds). The best solution is to use a Database Management System (DBMS) such as MySql or MongoDB to store the mortality database and apply filters on DB before running analytics. Indeed, DBMS offers great performance for this kind of operations due to the support of indices and query optimizations. Once the filtered data are obtained from the DB, the analytics can be run entirely on Python without any performance issues.

Implementation

The implementation of the solution is attached to this file. Once the required Python modules are installed, the application can be launched by using the following command:

```
> python3 app.py
```

The implementation is very basic and does not consider the performance issues outlined above. Furthermore, the statistics provided are still very basic and not fully configurable by the user. Anyway, the provided implementation represents a good starting point to obtain a complete working solution based on Dash.