I chose to use data from the Global Terrorism Database (GTD)1 for this analysis. This is an open-source database that includes information on terrorist attacks around the world from 1970 through 2017. The data were originally compiled by the University of Maryland2 and they defined terrorism as, “The threatened or actual use of illegal force and violence by a non-state actor to attain apolitical, economic, religious, or social goal through fear, coercion, or intimidation.” Their process for determining what attacks will ultimately be included into the database are as follows:

In order to maximize the efficiency, accuracy, and completeness of GTD collection, the GTD team combines automated and manual data collection strategies. The process begins with a universe of over one million media articles on any topic published daily worldwide in order to identify the relatively small subset of articles that describe terrorist attacks. This is accomplished by applying customized keyword filters to the “fire hose” of media articles available through a subscription to the Metabase Application Programming Interface (API) provided by Lexis Nexis. The English-language content from Metabase is supplemented with articles downloaded from the Open Source Enterprise (www.opensource.gov), which includes English-language translations of sources from over 160 countries in over 80 languages. This filter isolates an initial pool of potentially relevant articles, approximately 400,000 per month. These articles are then processed using more sophisticated natural language processing (NLP) and machine learning techniques to further refine the results, remove duplicate articles, and identify articles that are likely to be relevant. The GTD team manually reviews this second subset of articles to identify the unique events that satisfy the GTD inclusion criteria and are subsequently researched and coded according to the specifications of the GTD Codebook. Each month, GTD researchers at START review approximately 12,000 - 16,000 articles and identify attacks to be added to the GTD.2

My interests were to see if terrorist attacks are on the rise, what methods terrorists are using, locations of terrorist attacks, and preferred terrorist targets. Here are the steps I took to prepare the data and the graphs that I generated.

**PART 1**

1. **Data Preparation**—The original dataset contained 135 columns and was close to 200MB in size. Many of these columns were not needed and so I dropped many of them and ended up with 16 columns. I did not drop any rows and the data had 181,691 entries.

I also ended up changing many of the column names to something more meaningful. For example, the original column name of “iyear” was changed to the more descriptive “incident\_year”. Overall, I changed five column names.

I also wanted to see what types of variables were in my dataset. From the table below, it appears that we have a good representation of the data and there don’t appear to be a lot of missing or zero values.

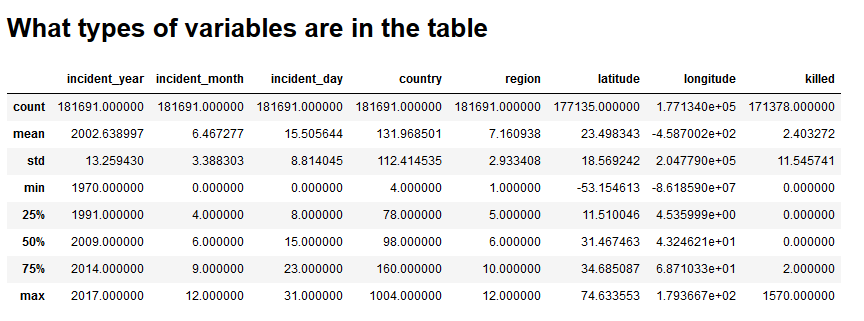


Table 1. Descriptive statistics that summarize the central tendency, dispersion, and shape of the terrorism dataset.

1. **Histograms**—I made four histogram plots that depicted the number of terrorist attacks versus the year, the month, the country, and the geographical region. The country and region charts only showed numbers and not actual country names and regions. One interesting point from these histograms is that the number of terrorist attacks seems to be on the rise. To further explore these two charts, I made some bar graphs.

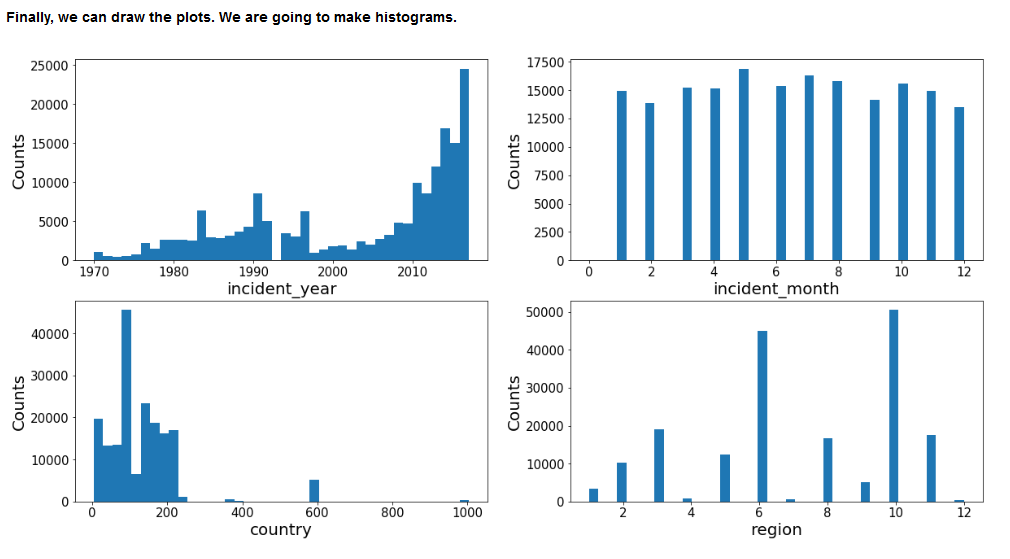


Figure 1. Histogram plots of incident\_year, incident\_month, country, and region for the terrorism dataset.

1. **Bar Charts**—The first bar chart showed the number of terrorist attacks per geographical region. After substituting the ordinal region designator for the region name, I discovered that the Middle East region had the most terrorist attacks. I repeated the exercise with the individual countries. There were way too many countries for a single bar chart, so I made two subset graphs that just showed the Top and Bottom 20 countries for terrorist attacks. The top country was Iraq and there were 10 countries that only had one attack each.

I made other bar charts indicating the preferred terrorist attack methods (bombing/explosion) and the favorite terrorist targets (private citizens and property).

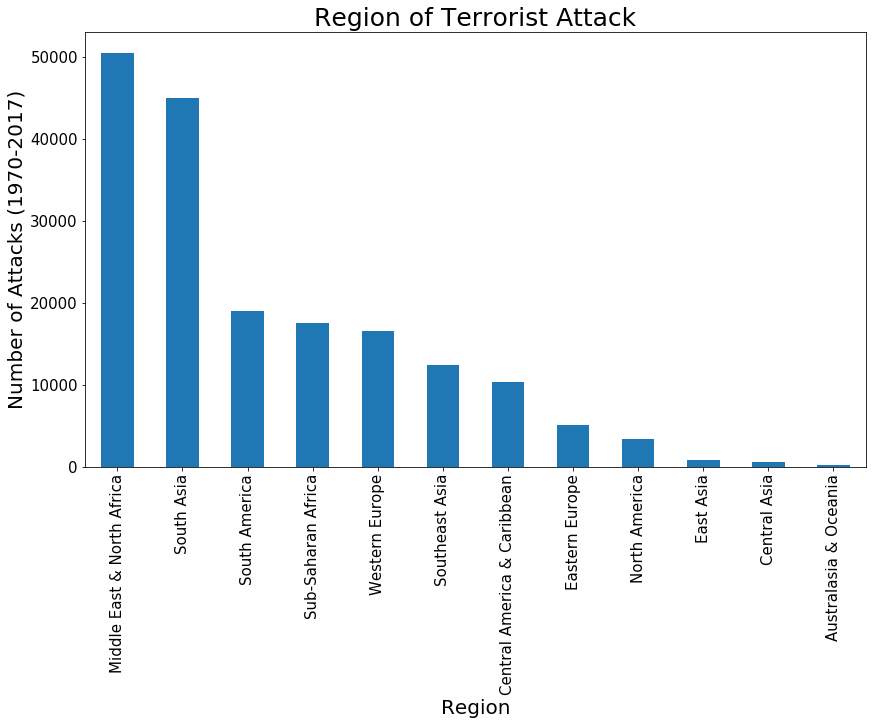


Figure 2. Bar chart of terrorist attack regions.

It is no surprise that the Middle East & North Africa regions have the most number of terrorist attacks.

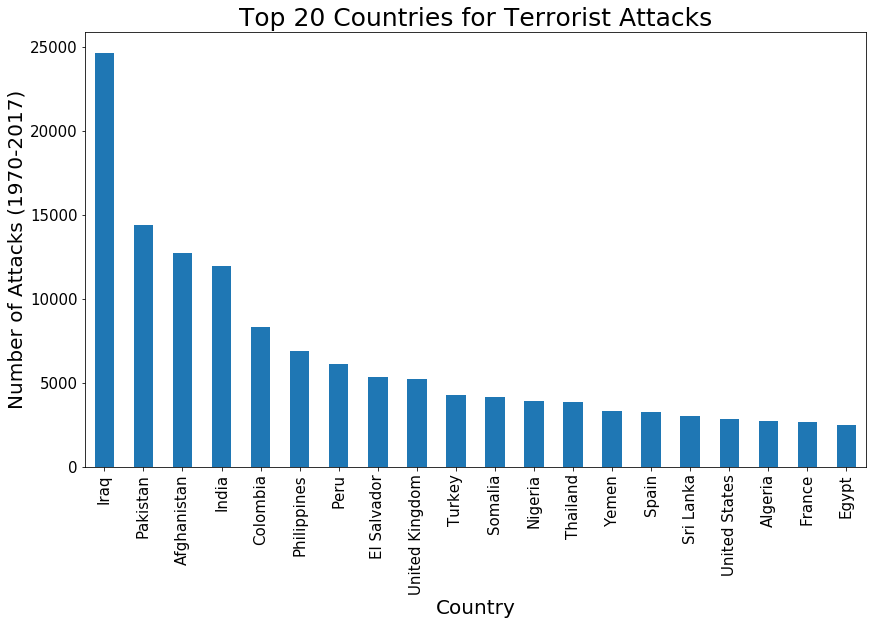


Figure 3. Bar chart of the top 20 countries for terrorist attacks.

Further drilling down from the region chart, we can see that Iraq is by far the leader when it comes to countries that have terrorist attacks.

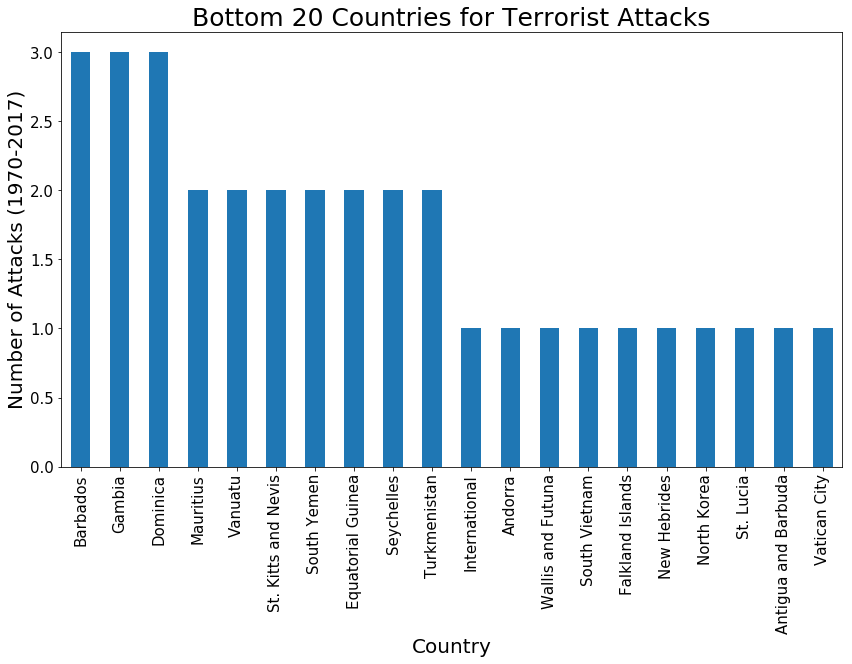


Figure 4. Bar chart of the bottom 20 countries for terrorist attacks.

This chart represents the “safest” countries with regard to terrorist attacks. I’m somewhat suspicious of North Korea only have 1 terrorist attack nearly 50 years. I don’t think they are completely forthcoming in their reporting.

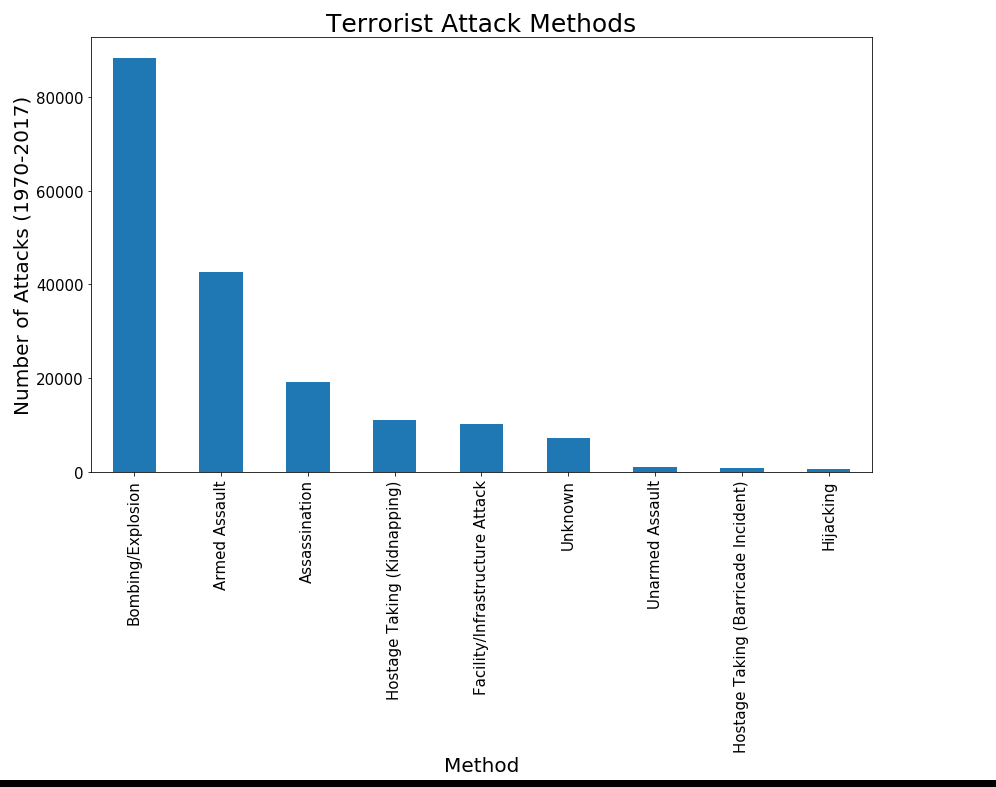


Figure 5. Bar chart indicating the most preferred terrorist attack methods.

We can see that the preferred terrorist attack method is bombing and explosions.

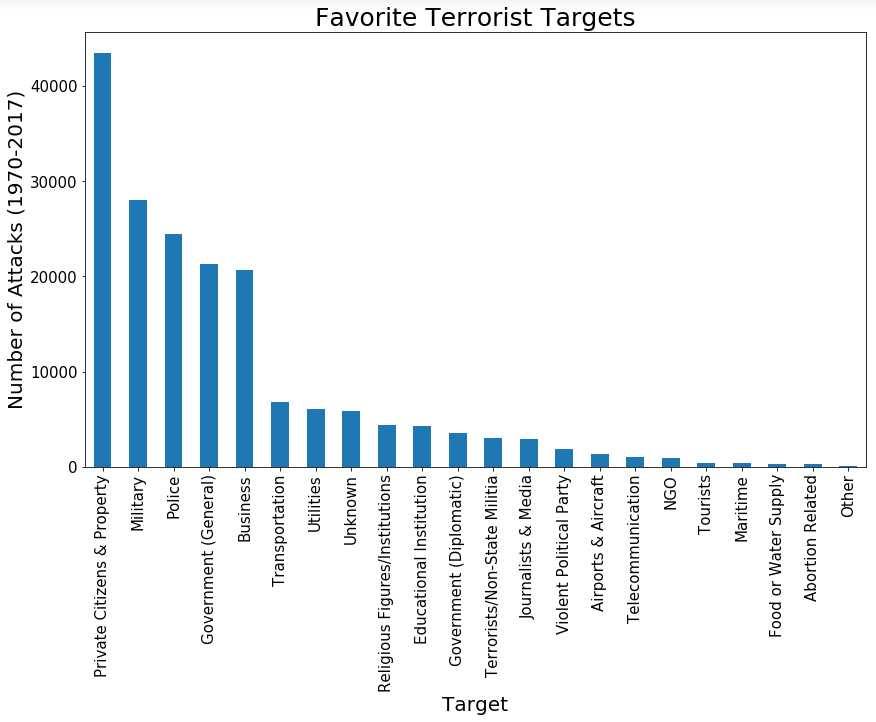


Figure 6. Bar chart indicating the favorite terrorist targets.

From a targeting perspective, terrorists seem to gravitate towards private citizens and property. I was surprised that the food and water supplies are not targeted very frequently. If a terrorist wanted to inflict the most casualties, all they would need to do is contaminate the water supply of a large city.

1. **Line Chart**—Since I had a time-series value (Year of Attack), I made a stacked line chart that showed the number of attacks per region from 1970 – 2017. This chart indicated a sharp uptake in attacks around 2004. As of 2017, however, the data suggest that overall terrorist attacks are on the decline.

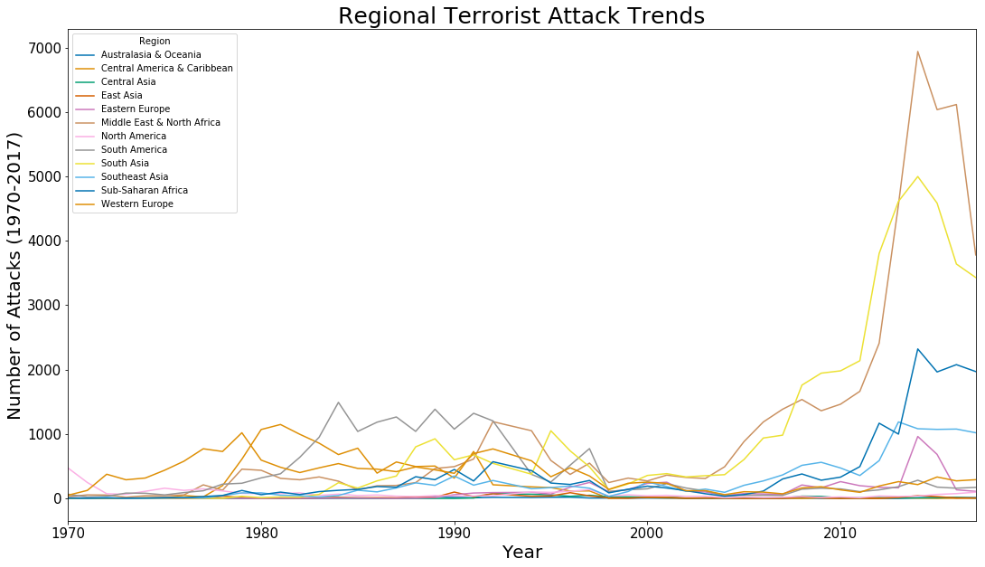


Figure 7. Line chart indicating the number of terrorist attacks between 1970 - 2017.

1. **Pearson’s Correlation Chart**—This chart was generated to see if there are any correlations between attack year, attack month, country, and region. The result indicated a fairly positive correlation (~0.50) between the year of a terrorist attack and the region in which it occurred.

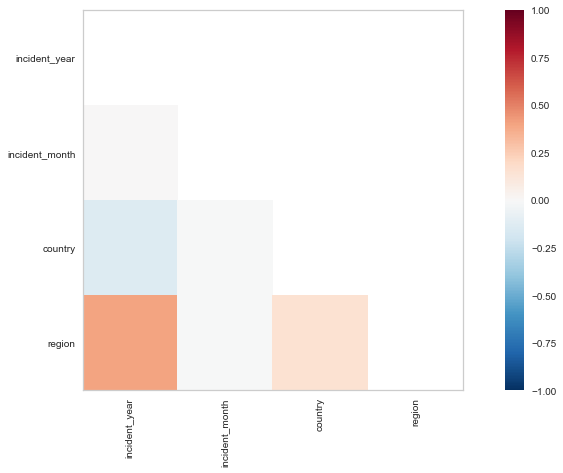


Figure 8. Pearson's correlation matrix for indicated variables.

1. **Geographical Chart**—My dataset came with latitude and longitude coordinates for each attack. I decided I would take a closer look at terrorist attacks in the United States. I used the GeoPandas2 module for this analysis and it worked very well. I was able to plot each location and I added a marker indicating whether the attack led to a fatality or not. As you might expect, California and New York are terrorist hotspots. Surprisingly, however, Puerto Rico is also a very active region for terrorists.

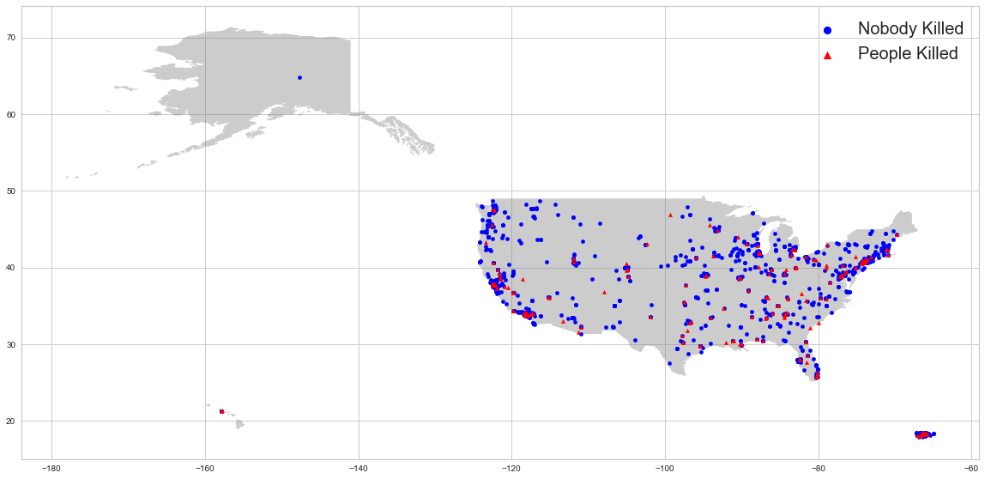


Figure 9. Map of terrorist attacks within the United States from 1970 - 2017. Data points indicate whether people died or not as a result of the attack.

1. **Stacked Bar Chart**—This graph was designed to show how likely a person is to survive a particular terrorist attack method. Each attack method contained a stacked bar chart indicating whether people survived that method or not. From the charts, we concluded that assassinations are the most successful way for terrorists to kill someone, but bombings/explosions have killed many more people.

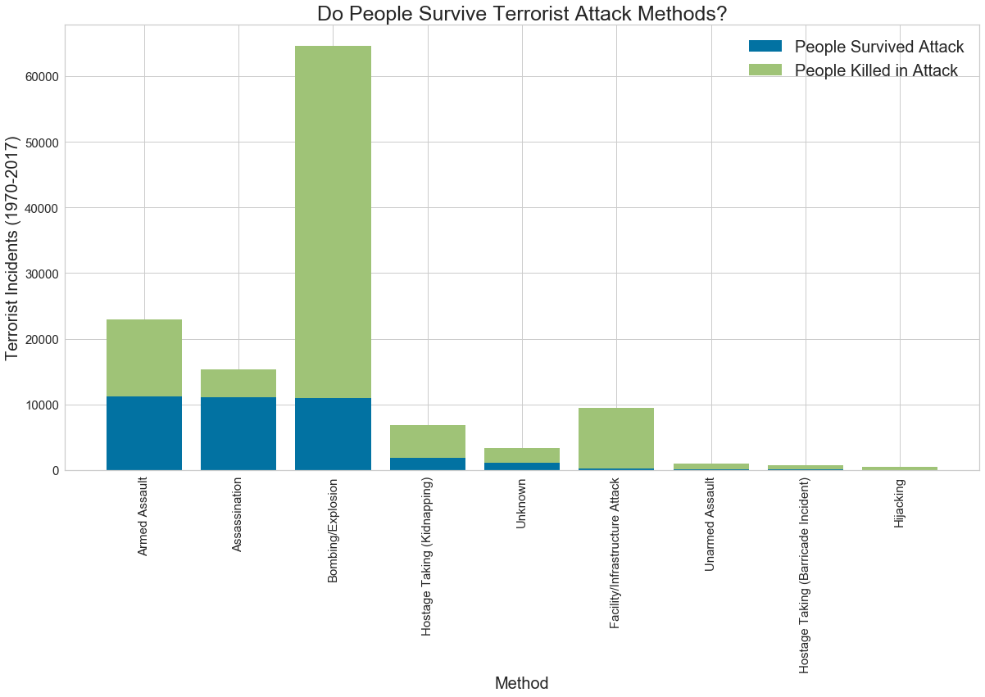


Figure 10. Stacked bar chart describing the effectiveness of particular terrorist attack methods.

1. **WordCloud Chart**—This dataset contained a column named “Summary” that provided a textual description of each attack. I used the WordCloud visualization module to produce a graphic that highlighted the most commonly used words when describing an attack. Some of the most used words include: assailants, perpetrators, unknown, attacked, and detonated. This technique is helpful when needing to comb through a lot of text in order to gain an overall sentiment.

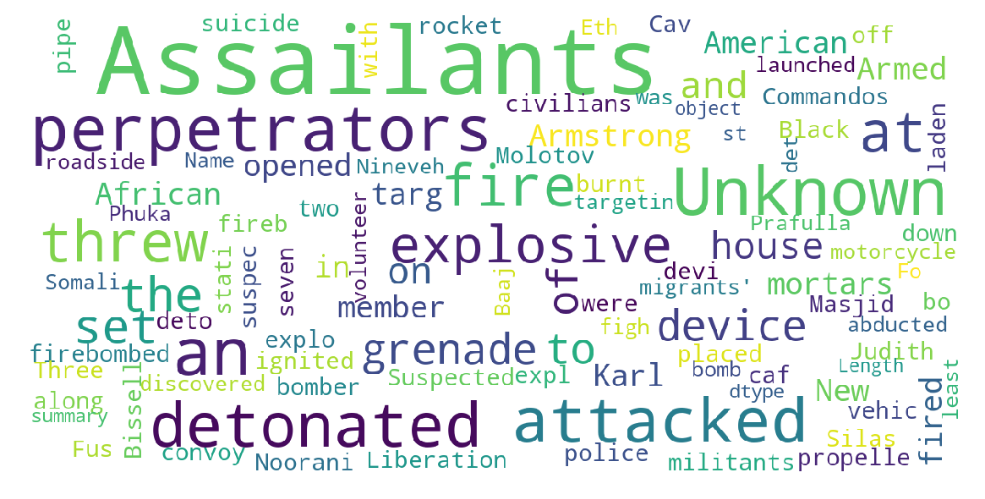


Figure 11. WordCloud graphic illustrating terms associated with terrorist attacks as described in the terrorist dataset.

I spent a lot of time on this project. I went through many different datasets before I decided on the terrorism data. The reason for this was that I wanted a dataset that lent itself to various types of charts. Many of the datasets I disregarded would have only allowed me to make a couple of different graphs. While this isn’t a bad thing, because sometimes our data will look like that, I figured that since this was a graph analysis project, I should try to make as many different graphs as I can.

Finally, I didn’t make all of the graphs that I could have from this dataset. There were a lot more variables I would have liked to look at, but I eventually needed to just stop making plots. I think the point of this project is to get us to use different types of graphs for different ways to interpret data. For each graph I produced in my code, I provided an explanation of why I chose that type of graph and the conclusions I was able to draw from it. Hopefully I was successful in my efforts.

**PART 2**

1. **Fill in missing values and eliminate features**

For the terrorism data set, there are two columns that are missing values, "incident\_month" and "incident\_day". I will follow the Titanic example and change these missing values to the median value for those columns.

I will use the Pandas *replace()* function to make the change.

Here is a data description for the “incident\_month” column before the replacement:

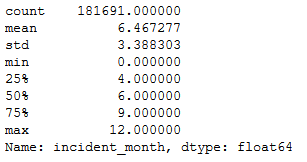


Table 2. Data description of incident\_month variable indicating the presence of a zero-value.

There are indeed some zero values.

Here is a data description for the “incident\_month” column after the replacement:

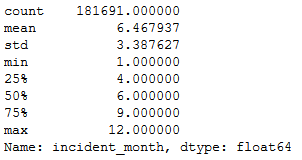


Table 3. Data description of incident\_month variable after zero-value had been changed to the median value.

We were successful in eliminating the zero values. We can repeat the process for the “incident\_day” column:

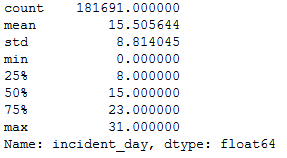
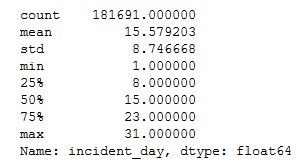
 

Table 4. Data description of incident\_month before (left) and after (right) data replacement of median value for all zero-values.

We have replaced the zero values for the “incident\_day” column with the median column value.

1. **Log transformation**

For this step, we will perform a log transformation on the “Country” data. These values range from 4 to 1004 and by performing a log transformation, we can compensate for the large skew that would result from using the regular country designation values.

First, we will import the *numpy* library and then define our log transformation function.

Our new column name will be “Country\_log1p”. Here is a description of the data after the log transformation has been performed. We can see that a new column has successfully been created.

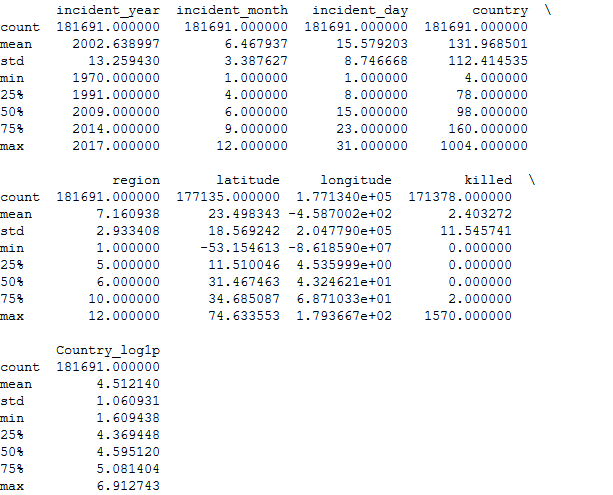


Table 5. Log transformation table of the country variable.

We should now make a histogram of this new log data to see if we have successfully compensated for our skew. As a reminder, the original histogram is presented first to show how skewed the data originally were.

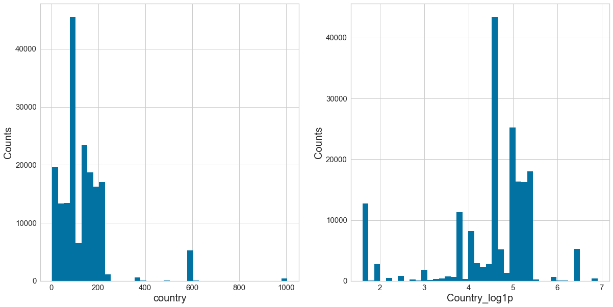


Figure 12. Comparison of original country histogram (left) versus the log transformation (right).

The log data are much better distributed than the original “Country” data. It looks like our log transformation worked well.

**\*\*UPDATE—CLASSMATE FEEDBACK\*\***

Based on classmate feedback, it was suggested that I do a log transform on the “killed” data column as well. That was a great idea and here is the result. The original data is on the left.

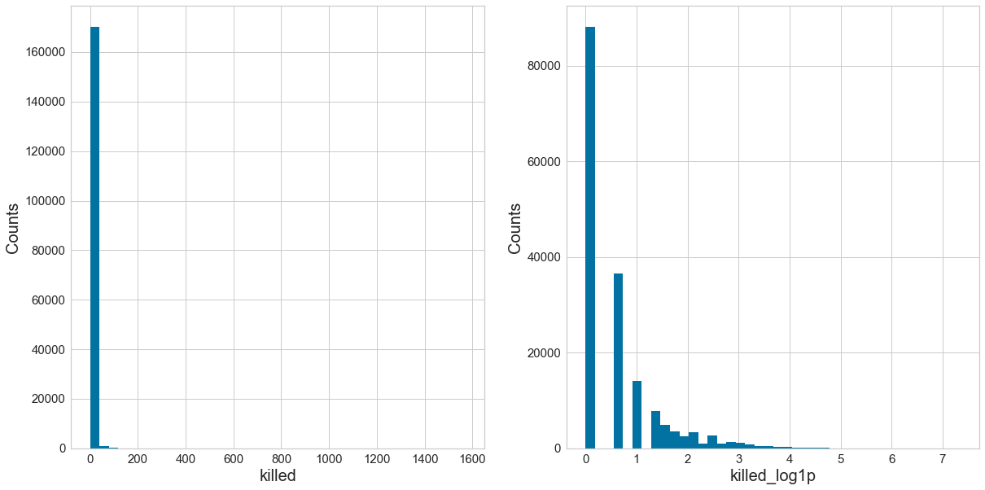


Figure 13. Comparison of original killed histogram (left) versus the log transformation (right).

This is a much better representation of the data that just that huge line at zero for the non-transformed “killed” data.

1. **One Hot Encoding**

The terrorism dataset contained a categorical data column called “attack\_type”. There are nine different categories within this column that represent the different terrorist attack methods. We are going to use the One Hot Encoding algorithm to convert these categories to either a value of 1 or 0. This is usually done prior to using machine learning algorithms that require numeric values and not string characters.3 After employing the algorithm, a snapshot of the output is shown below:

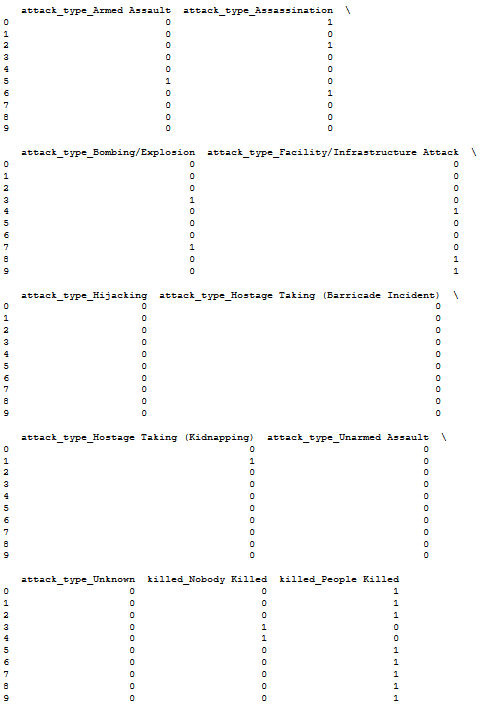


Table 6. Table of original One Hot Encoding output.

The results indicate that our One Hot Coding has been successful.

**\*\*UPDATE\*\***

I started on Part 3 and realized that I had made an incorrect classification in my One Hot Encoding. I have since adjusted that and the new One Hot Encoding is shown below. My error was including the “People Killed” and “Nobody Killed” categories. This was incorrect and I should have chosen another categorical variable. I ended up choosing the “region” column. So my new One Hot Encoding is considering the “attack\_type” and the “region”. These two will later be combined with the “killed” column to generate the final model.

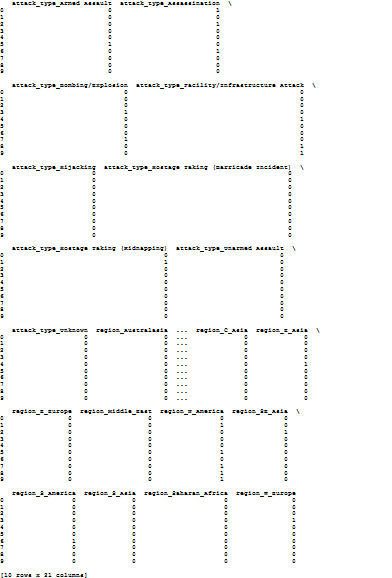


Table 7. Corrected One Hot Encoding table.

**PART 3**

The first thing we need to do is to create a features dataset that can be used to create our training and validation datasets. To do this, we will combine the numerical features and the dummy features we created for Part 2. The result is a dataset that contains the “incident\_year”, “incident\_month”, “Country\_log1p” and the One Hot Encoding results. Here is a sample of the result:

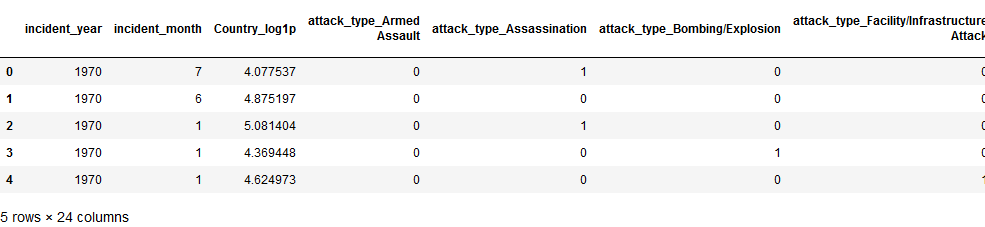


Table 8. Sample table of combined numerical and dummy features.

Now that we have successfully combined the two datasets, we can create a target dataset. This will consist of our “killed” column that has been reassigned values of “People Killed” and “Nobody Killed”. Here is the result:

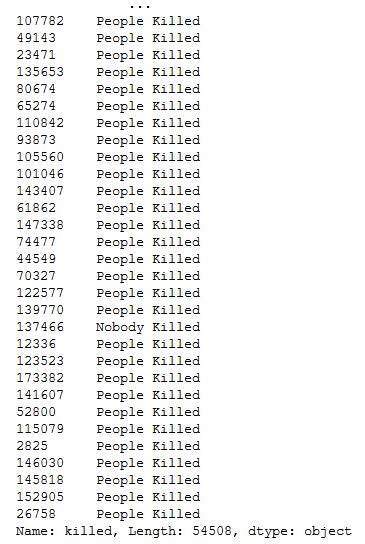


Table 9. Output table of target dataset.

After loading the sklearn library, we needed to separate our dataset into a training and validation set. Typically, we use a 70% to 30% split ratio of training to validation. After apply the code to our dataset, we ended up with the following values for our validation and training sets:



Table 10. Output of test and training data split ratios.

Our original dataset contained 181,691 entries. 70% of this number is 127,183. So it appears that our 70/30 split is correct.

Now we can print out how many “People Killed” and “Nobody Killed” totals from both our training and validation datasets.

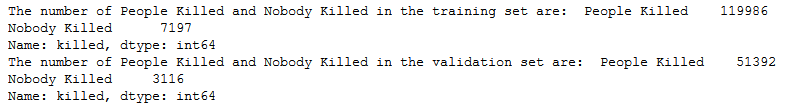


Table 11. Output of people killed and nobody killed values from the training and validation datasets.

We are now ready to display our model validation metrics. The first metric we can show is the confusion matrix score. To create the ConfusionMatrix, we need some test data. Score runs predict() on the data and then creates the confusion\_matrix from scikit-learn.4 Our cm.score() value was:



Table 12. Confusion matrix output score.

Now we can finally produce the actual ConfusionMatrix.

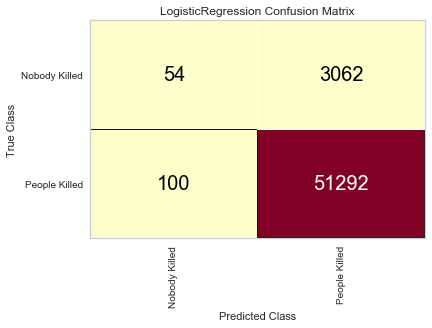


Figure 14. Confusion Matrix for the model.

Our results look really good. Our model was able to successfully predict 51,292 people killed out of a possible 51,392.

To further look at our model’s performance, we can generate a classification report. A classification report is a text summary of the main metrics for assessing the success of a classifier:5

* Precision—the ability not to label an instance positive that is actually negative
* Recall—the ability to find all positive instances
* F1-score—a weighted harmonic mean of precision and recall

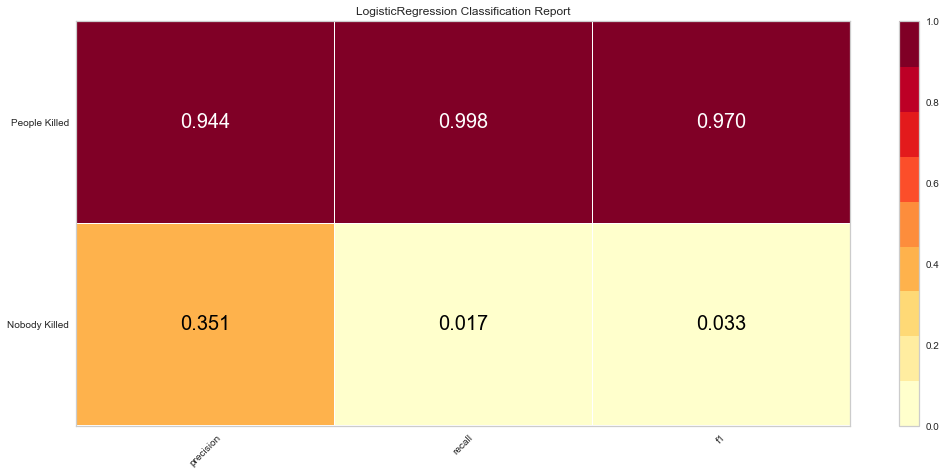


Figure 15. Logistic model classification report.

The darker zones in the above Classification Report show the model’s highest areas of performance. In this example, the model was very good at predicting the number of people killed based on location and attack method.

Finally, we can look at the Receiving Operating Characteristic (ROC) curve. The ROC curve is a common method for evaluating the quality of a binary classifier. By plotting the ROC curve, we can see how the model performs. Additionally, it is also common to calculate the area under the ROC curve (AUCROC) to judge the overall equality of a model at all possible thresholds.6 Below are the AUCROC and ROC curves for our model.

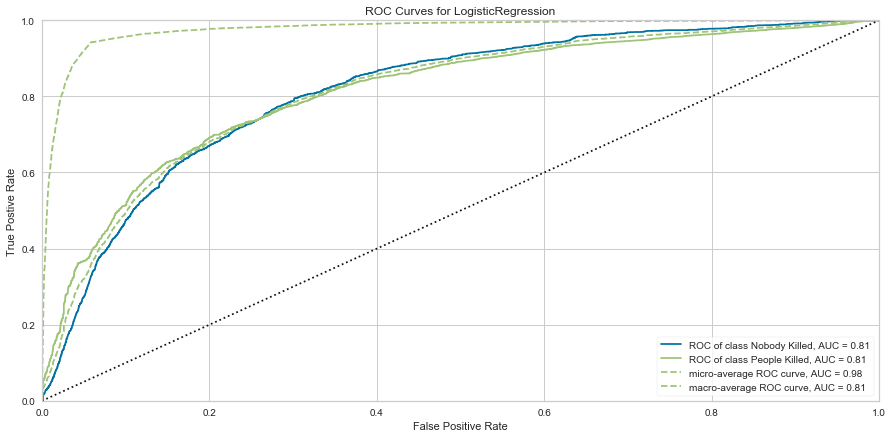


Figure 16. Receiving Operating Characteristic (ROC) curve plot for the logistic model.

Our ROC curve looks very good. A perfect discrimination (no overlap in the two distributions) has a ROC curve that passes through the upper left corner (100% sensitivity, 100% specificity). Therefore, the closer the ROC curve is to the upper left corner, the higher overall accuracy of the model.7 The AUC value is a measure of how well a parameter can distinguish between two groups (Nobody Killed/People Killed). The higher the number, the better the ability of the model to make the distinction. Our results are quite high indicating our model’s ability to correctly make the distinction.

**REFERENCES:**

1Global Terrorism Database. (n.d.). Retrieved from https://www.kaggle.com/START-UMD/gtd.

2 Global Terrorism Database. (2019). Retrieved from https://start.umd.edu/gtd/.

3 GeoPandas 0.6.0. (n.d.). Retrieved from http://geopandas.org/.

4 Vasudev, R. (2017, August 2). What is One Hot Encoding? Why And When do you have to use it? Retrieved from https://hackernoon.com/what-is-one-hot-encoding-why-and-when-do-you-have-to-use-it-e3c6186d008f.

5 Confusion Matrix. (2019). Retrieved from https://www.scikit-yb.org/en/latest/api/classifier/confusion\_matrix.html.

6 Bengfort, B., Bilbro, R., & Ojeda, T. (2018). *Applied text analysis with Python: enabling language-aware data products with machine learning*. Sebastopol, CA: OReilly Media, Inc.

7 Albon, C. (2018). *Machine learning with Python cookbook: practical solutions from preprocessing to deep learning*. Sebastopol, CA: OReilly Media.

8 Schoonjans, F. (2018, November 9). ROC curve analysis with MedCalc. Retrieved from https://www.medcalc.org/manual/roc-curves.php.