Part A Deliverable

Data Summarization, Preprocessing, and Feature Selection

Deliverable Part A

Data Exploration

Data exploration is an important part of any data science project. This step provided insights into the characteristics of the data. During this stage, we also got a better understanding of many limitations. Understanding the limitations of the data you are working with allows you to make more informed decisions for the remainder of the process.

Through the exploration phase, a large class imbalance was discovered.

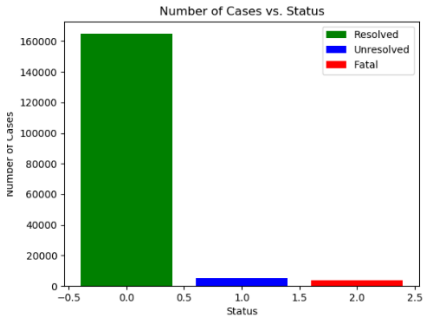

Figure 1: Class Imbalance

Statistical testing was completed on certain dimensions of the data. One of the tests completed and visualized was a boxplot of the case status. The distribution clearly showed that the fatal and unresolved cases were major outliers, while the resolved cases was the median.

Another visualization's purpose was to determine whether there was a difference in the case status, the gender, and the location. The scatter plot created used PHU testing locations. It did not show any indication that there was a difference in cases between genders.

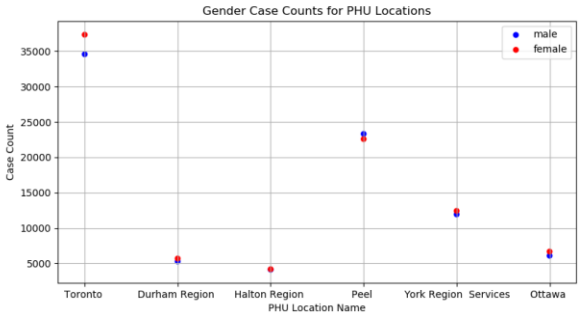

Figure 2: Gender Case Count vs. PHU Location

Data Preprocessing

Preprocessing was completed in 4 main stages. The stages included:

1. Handle missing data
2. Transform categorical attributes
3. Normalize data
4. Under sample

Handling Missing Data

A connection was made to the database using the psycopg2 module. Once it was established, several queries were made. During the first step of the pipeline, the queried data was converted into a pandas dataframe. Missing data was filled in a few different ways. The most common way was imputation. Some imputations were relatively easy since the data characteristics had labels such as “other”

Below is an example of handling missing special measure data.

```
# Replace 'None' special measure values to 'other'
data["Special Measure"].fillna("Other", inplace=True)
```

Another approach we took to handling missing data was imputing the mode on missing values. It is a common preprocessing technique to impute using the most observed value in the observed data. An example of this being done is below on the “age” column.

```
# Replace missing age values with the mode of ages seen
data["age"].fillna(data["age"].mode()[0], inplace=True)
```

However, filling in missing data for some attributes was challenging. For the binary attributes resolved, unresolved, and fatal, there were only two substitution options to consider (true or false). An assumption was made that it is more likely if the value were missing that the true value would be false. This is because in any situation, there can only be one true value between the three values. A sample cannot be both resolved and fatal at the same time. So the attribute has a probability of 2/3 to be false.

```
# Find any empty cells in resolved, unresolved, fatal and replace with False
data[["resolved", "unresolved", "fatal"]].fillna(False, inplace=True)
```

Converting to Numerical Attributes

One-hot encoding was used to convert categorical attributes to numeric values. This technique worked well for this dataset because the number of distinct values for the categorical attributes was low. Additionally, the data contained mostly nominal data. For example, when the “season” attribute was encoded:

season_fall	season_spring	season_summer	season_winter
0	0	0	1

Figure 3: One-hot Encoding

Some of the categorical attributes, including age, were ordinal. This made it easy to convert to numeric data. The process involved a string manipulation and data conversion in the code.

Normalization

For this step of the process, we used the standard score (z-score) standardize the data. This technique works well when the population is normally distributed. The function took advantage of pandas built-in mean and standard deviation functions.

```
def normalize_helper(feature):
    data[feature] = (data[feature] - data[feature].mean()) / data[feature].std()
```

Under Sample

Once there were no more empty data cells and it was all standardized, under sampling took place. The label to be predicted was the case status (resolved, unresolved, fatal).

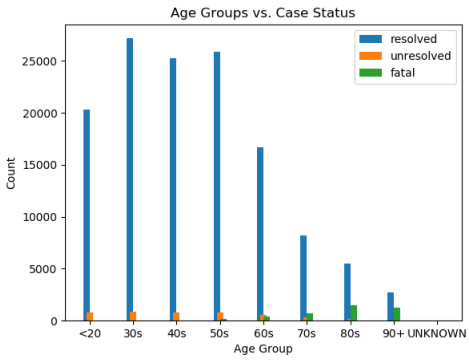

Figure 4: Case Status vs. Age Group

From the graph above, it is clear to see the resolved status is much higher than the other two. Thus, when training the model, it is important that the model has enough data of each label to properly predict. Random under sampling was used to fix the class imbalance. The process was essentially to pick a random subset of the resolved class and get rid of the rest. After under sampling was completed, the class labels were as follows.

- Resolved: 5000
- Unresolved: 5376
- Fatal: 3844

This is much more balanced going into training and should have a positive impact on the prediction outcomes.

Data Quality Issues

There were not any data quality issues we encountered. Everything we faced was common to deal with in a data science setting. Once the preprocessing was complete, a csv file of the data was generated.