## Tentative Methodology (Project Approach)

### Figure 1: Graph of overall Methodology

### Data Collection:

Retrieve the 4 desired datasets from Canada’s Open Government Portal: <https://open.canada.ca/data/en/dataset/cc5ea139-c628-46dc-ac55-a5b3351b7fdf>

### Data Processing:

This phase has been started by dealing with duplicate “Incident Number” observations in each of the 4 datasets. In the “Animals” dataset, I created a “UniqueID” column based on the “Incident Number” value and a running count of occurrences of that “Incident Number” so each observation had a unique identifier. In the “Incident” dataset, each duplicate occurrence of the “Incident Number” held and NA value as the “Incident Type” (and held no other new information) so I simple dropped those rows. In the “Activities” and “Responses” datasets, the “Activity Type” and “Response Type” attributes respectively, were encoded using one-hot encoding so that each distinct category has a column with binary values of “0” for no and “1” for yes. I decided on this approach after discussing the options with Professor Abdou and I decided encoding these two variables was the best way to maintain the information in the dataset and be able to merge the 4 datasets together.

I then merged the four datasets together using the “UniqueID” value generated in the “Animals” dataset to ensure there is unique identifier for each observation in the newly combined dataset, named Complete\_HWC\_Data.

Next, I will do some data cleaning to deal with typos or mistaken entries. To find these errors I will compare unique values for “Activity Type”, “Response Type”, and other categorical attribute values with the values that are listed in Data Dictionary which was provided with the Open Data record to ensure valid entries.

### Exploratory Data Analysis:

Generate an Exploratory Data Analysis Report using the Panda's ProfileReport. Use report to observe variance within variables, correlation between variables, missing values, distinct values per variable, etc.

Use histograms and various other plots to subset data and explore patterns in incident types by human activity, animals involved, location, and time of year.

Use statsmodels Python module and Time Series Analysis (tsa) to plot and visualize trends over time. <https://towardsdatascience.com/analyzing-time-series-data-in-pandas-be3887fdd621>

### Modeling:

Before deciding on which models to use, there are some preprocessing steps that needed to be looked into and addressed. Because these steps will be dealt with in the creation of the model, I’ve included this information as part of the Modelling phase of the analysis instead of part of the Preprocessing phase.

#### Splitting Data

Splitting data into training and test data for the purpose of predictive modelling. I will use TimeSeriesSplit function of the scikit-learn package to split and cross-validate my training and test data sets. As described by Howell (2023), doing a TimeSeriesSplit “means our test [data] is always in the future compared to the data our model is fitted on.”

#### Dimensionality Reduction

When researching dimensionality reduction techniques looking for the best fit for my dataset containing mixed data (both numeric and categorical data). Mahmood (2021) indicates that “[w]hen dealing with mixed data, FAMD is a recommended way to handle the unnecessary factor and reduce the dimensions of the data”. Mahmood (2021) goes on to describe that Factor Analysis of Mixed Data (FAMD) analyzes the dataset using a combination of Principal Component Analysis (PCA) and Multiple Correspondence Analysis (MCA) techniques to determine which attributes are the most critical components for modeling the data.

#### Dealing with Missing Values

The dataset has a few variables with a few (between 3-40) missing values. These variables are “Incident Type”, “Species Common Name”, “Sum of Number of Animals”, “Activity Type” variables, “Latitude Public”, “Longitude Public”, and “Within Park”. Because there are only a few missing values, I will impute this data using the KNN Imputation method.

The “Response Type” variables are missing values for around 1500 observations. This is still only about 2% of the data so I will also impute this data using the KNN Imputation method.

Lastly, there are some variables with several missing values (over 30,000). These include “Animal Health Status”, “Cause of Animal Health Status”, “Animal Behaviour”, “Reason for Animal Behaviour”, “Animal Attractant”, “Deterrents Used”, “Animal response to Deterrents”. These attributes have too many missing values to consider any kind of imputation. I don’t want to completely discount this data so I will still use it for pattern mining by sub-setting the data to all rows that have complete values for these variables and generate some graphs and correlation measures (using EDA report) to view it separately from the rest of the data. When it comes time to do modelling, these attributes will be dropped from the training data.

#### Dealing with Imbalanced Data

My dependent variable, Incident Type, is very imbalanced. Boyle (2019) succinctly addresses the problem with imbalanced classes as “Most machine learning algorithms work best when the number of samples in each class are about equal. This is because most algorithms are designed to maximize accuracy and reduce error.”

In my research into the pros and cons of the options for dealing with imbalanced classes, both under sampling and over sampling options have some downsides. Under sampling means you are not using some of the data from the majority class in order to balance the classes. In this dataset, the majority class contains about 66% percent of the observations, with the next highest class consisting of about 18% of the observations, so there could be a lot of valuable predictive information in the majority class that would be eliminated with under sampling. Over sampling adds data points (either randomly or synthetically generated) to the minority classes to balance the classes. Stewart (2020) points out one of the concerns with using over sampling is the risk of adding false data points could lead to overfitting. In this dataset, we would need to add very large amounts of datapoints to the minority classes in order for them to balance with the majority. That many “false” data points would cause concern that the data is no longer providing an accurate depiction of the actual data.

I also researched a third method of dealing with imbalanced the cost-sensitive learning method. Brownlee (2020) describes that “while most machine learning algorithms assume that all classes have an equal number of examples… [cost-sensitive learning] modifie[s] [algorithms] to change the way learning is performed to bias towards those classes that have fewer examples in the training dataset.” Stewart (2020), Brownlee (2020), and Selvan (2020) all claim that using cost-sensitive learning to deal with imbalanced datasets - where class weights are taken into account in the machine learning algorithm – perform more effectively than under or over sampling. I’ve decided to use this method to address the imbalance in my Incident Type variable.

#### Dealing with Categorical Variables

Most decision tree classifiers (including the commonly used scikit-learn decision tree classifier and random forest classifier) cannot handle categorical data without having it encoded to numeric form. When the categorical data is ordinal, this process is quite straight forward; however, with nominal data (where there is no inherit order), the process of converting categories to numeric involves creating “a new dummy variable for each level of the original variable” (Dingwall and Potts, n.d.). There are two main methods for encoding nominal categorical data: one-hot encoding and get\_dummies. Kumar (2021) explains that the get\_dummies method can be problematic if your test data set has a different set of columns depending on whether or not some categories exist in the test set that do or do not exist in the training set. The one-hot encoding method does not have this issue. Dingwall and Potts (n.d.) describe the following two potential problems that one-hot encoding can have in tree-based models:

1. The resulting sparsity virtually ensures that continuous variables are assigned higher feature importance.
2. A single level of a categorical variable must meet a very high bar in order to be selected for splitting early in the tree building. This can degrade predictive performance.

Even with those stated potential problems with using one-hot encoding, it is still a very popular and commonly used method. The alternative to encoding categorical data is of course leaving your categorical data as-is and using a tool like H2O for Python which can build a decision tree or random forest model using categorical data.

Based on my findings, I’ve decided to use one-hot-encoding on two of the three models I am generating using scikit-learn, and will maintain the categorical data for the last model using H2O. More specifics about these three models is included in the section below.

#### Models

I will be building three different models to compare the results. All three models will be using tree classifiers, but will have differences in the type of model or treatment of categorical variables. They are summarized below:

* + - 1. Decision tree classifier: using scikit-learn in Python and Gini index as the selection measure. For this model, one hot encoding will be applied to all categorical variables and I will use the sklearn.utils.class\_weight function to balance the class weights (how: Stewart (2020) and Brownlee (2020)).
      2. Random Forest Classifier using scikit-learn: For this model, one hot encoding will be applied to all categorical variables and I will use sklearn.utils.class\_weight function to balance the class weights. (How: Brownlee (2020))
      3. Random Forest Classifier using H20: For this model, the categorical variables will remain as they are (with string values) and I will use the balance\_classes option in H20 to balance the class weights. (how: <https://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/drf.html>).

#### Evaluation

Evaluate and compare the results of each model. The common evaluation metrics are accuracy, precision, recall, and F1 score. Accuracy is often not the best metric to use when evaluating imbalanced datasets. Precision is a good metric when your main aim is to minimize false positives and recall is a good metric when you want to maximize the true positive rate. In this situation, I see it as more important to maximize the true positive rate (successfully predict incident types) than it is to minimize false positive rates so while I will be considering all the performance metrics in my evaluation, I will use recall as my main performance metric.

### Visualization And Reporting

I will present the results of the project using various plots and graphs. Based on the research I’ve conducted, I will make recommendations for the mitigation of incident by type with intention of promoting health and safety of humans and animals in these 35 National Parks.