MSc Data Analytics

Final Year Project

Geospatial Analysis of Explosive Data for Strategic Planning and Decision Making

By

TITO THOMAS DAVID

Project Unit: U22444

Supervisor: Dr. ALEXANDER GEGOV

September 2023

ACKNOWLDEGMENTS

I express my gratitude to all those individuals who contributed to the development of this work, for the invaluable support they provided throughout the process. I am particularly appreciative of NATO for granting me the opportunity to work with them and for their insightful suggestions. I am especially grateful to Dr. Alexander Gegov, my advisor, for his guidance and help throughout the course of this project.

Lastly, I extend my heartfelt thanks to my family, including my parents who instilled in me the perseverance to overcome challenges, and my friends who offered their unwavering support. I am sincerely grateful to each and every one of you.

Thank you all.

ABSTRACT

The project "Geospatial analysis of explosive data for strategic planning and decision making" conducted in collaboration with NATO, helps us take a significant step ahead in using geospatial data and machine learning techniques to enhance strategic planning and resource allocation. We will also classify the severity of attack locations which will further aid NATO to increase their preparedness and planning for future attacks.

The selection of new features and further preprocessing helps us lay the groundwork upon which we implement our machine learning model. The project highlights the use of spatial data i.e., latitude and longitude to perform geospatial analysis for strategic planning and resource allocation for our client NATO. Python libraries offer a multitude of maps which enable the users to perform such type of complex analysis.

Further the classification of the severity of attack locations using machine learning algorithms help NATO in their effort to plan for the future. The novel machine learning model is chosen based on extensive research and analysis and the results are compared to other benchmark models using different classifiers in order to validate and measure the novel model's robustness.

With the number of explosive attacks on the rise since the turn of the century, NATO and other organisations are always on high alert to adopt new plans to help them control and mitigate such threats. Our project certainly aims to be part of this phase wherein we provide them a tool for better analysis of the geographic data based on the attack locations. The severity factor is a further addition to this tool.

CONTENTS

List of Tables

List of Figures

Glossary

NATO	: North Atlantic Treaty Organization
KNN	: K-nearest neighbour
MLP	: Multi Layer Perceptron
SVM	: Support Vector Machine
TP	: True Positive
TN	: True Negative
FP	: False Positive
FN	: False Negative
GTD	: Global Terrorism Dataset
AI	: Artificial Intelligence
SPS	: Science for Peace and Security
PS	: Public Security

Chapter 1

1.1 Introduction

In today's world, safety is a wide and broad aspect and one if the first area of concerns are the almost unpredictable terrorist activities. Explosive incidents or rather the term "Bombings" are nowadays a general topic of discussion across the world. There is a substantial increase in the count of bombing events over the past few years which pose a threat to the general society and also the organisations or the local bodies protecting people from such attacks such as my client NATO. If there could be some way of predicting the areas which are more prone to such attacks with high severity, such models or methodologies could be helpful for NATO for strategic planning and allocation of their resources more effectively.

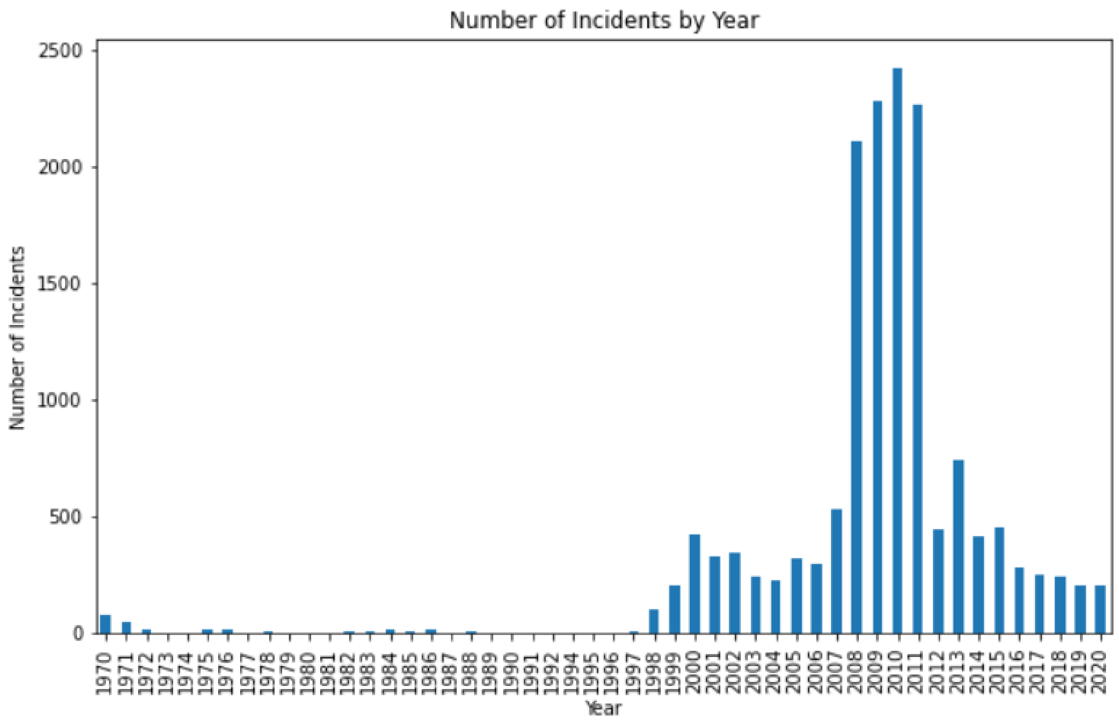


Figure 1: Number of incidents per year

The above figure shows the number of bombings over all these years starting from the year 1970. There is rapid increase in the cases starting from 1998 whereas the period 2008-2011 show the greatest number of cases worldwide. Even in recent times the number of cases is significant and if this trend continues will pose a major threat to world peace and security.

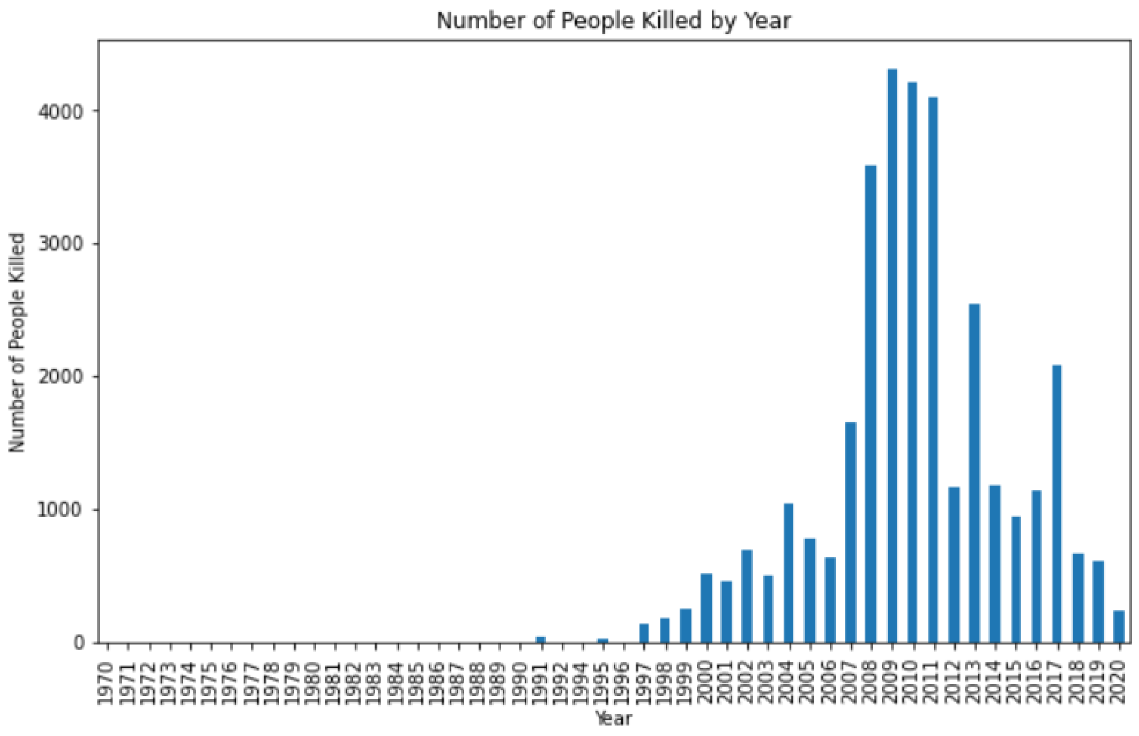


Figure 2: Number of people killed by year

From the above figure we can see the deaths over all these years. Since as stated above year 2008-2011 was the most affected in terms of incidents, we can see the high number of deaths too from the above plot. There are significant number of people killed in recent times and if the trend continues it will pose a threat the objective and the idea of NATO.

When we talk about NATO it is their duty to protect their associated nations and provide them security. The North Atlantic Treaty Organization Allied Command Transformation (NATO ACT) can use this model or technology to understand the area of attacks, the severity and see the trend over all these years (1970-2020) and make better decisions which can in some way restore peace and a sense of security among the people in the future as the Public Security (PS) of its members and partner countries is their priority under one of its programs; Science for Peace and Security (SPS).

In this research we will try to use the tools given by python and other libraries to perform the geospatial analysis and further use machine learning algorithms to classify the severity level of the regions over all these years which will further enhance the analysis phase of the project.

1.2 Aim of Project

The aim of the NATO project is to enable or help the NATO to gain insights into attacks caused by explosive materials, identifying the high-risk areas so that the resources can be allocated accordingly and steps can be taken to be prepared in advance. This project will also help in analysing the attack patterns and trends. Further predictive models can help several organisations to forecast the severity of such attacks in the future. Such measures will help NATO to enhance their preparedness for such attacks.

1.3 Objective

The purpose or the objective of this project or study will be to perform geospatial analysis of the explosive data and help the organisation make better and informed decisions in terms of resource allocation which will help NATO be better prepared for such incidents in the future. Further we will classify the severity of the attacks using geospatial data and further enhance the quality of analysis.

1.4 Outcome

The outcome from this NATO project are as follows:

1.Collect data from single or multiple sources and link them together if necessary.

2.Improve the quality of data by some pre-processing techniques

3.Use python libraries to perform geospatial analysis to provide insight into the spatial data.

4.Develop a machine learning model using random forest for classification.

5.Compare the new model with the existing benchmark models using comparison metrics.

1.4.1 Potential Benefit

This technology will be useful to the NATO as we have already stated that such types of analysis are always a blessing to any organisation which is using it. NATO can use this geospatial analysis and the severity factor to better understand the trend over the years and further make decisions which will shape up the future in terms of security and world peace. This will contribute towards NATO's strategic objectives and have a clear link to security.

1.5 Project Client

The client for the project is NATO's Science for Peace and Security (SPS) Programme which plays a crucial role in promoting progress in civil science and technology to address emerging security challenges and their impact on global security. This initiative promotes cooperation among scientists, experts, and officials from NATO and partner nations, encouraging joint efforts to address these challenges. It accomplishes this by offering support for security-related initiatives through several grant mechanisms.

1.6 Project Limitation

The project's primary limitation lies in the constraint of time. An extended duration would have facilitated a more comprehensive exploration and analysis of spatial relationships.

1.7 Project Organization

The project will develop and perform Geospatial analysis of explosive data using AI algorithms for strategic planning and decision making. The chapters defining each part of the project are as follows:

Chapter 1: In this chapter we will analyse the basic problem of our project, aim, objective, client information and the limitations of the project cycle. We will also investigate the ethics involved in our project.

Chapter 2: In this section we will go deeper into the project background. The literature review which is the assessment of the existing literature will be investigated and analysed.

Chapter 3: We take a basic look into the project cycle of a machine learning based project. Also, the several algorithms which were used in the benchmark classification models are analysed in detail. Finally, the project environment is also discussed.

Chapter 4: In this section, we investigate the data pre-processing and the data preparation phase of the project. Feature engineering follows data pre-processing in which we examine and describe the features selected for the project. We also examine some data visualisation techniques on the input data.

Chapter 5: The requirements and design section describe the algorithm and any pre-requisites used in the project.

Chapter 6: In this chapter we solely describe the geospatial analysis phase of the project using different python libraries and built in maps.

Chapter 7: This chapter is specifically for the implementation of the machine learning model and its pre-requisites.

Chapter 8: In this section we discuss the results of the novel model and the benchmark models. Also, we further analyse the performance of each classifier.

Chapter 9: In this chapter we provide our final conclusions based on the analysis of the project.

1.8 Summary

In this chapter, the project idea was presented along with its intended purpose, goals, and anticipated results. We identified the project client which is NATO along with the limitations which we faced in the project's development. Finally, a preview of the organisational structure of the coming chapters were described in brief.

Chapter 2

2.1 Background

In the contemporary world, the escalation of terrorist attacks or incidents has given rise to the emergence and necessity of methodologies for comprehending these issues and examining the patterns or trends over time. By integrating geospatial analysis with various machine learning algorithms, a potent tool is produced that empowers the user to scrutinize and analyse data in a more innovative and informed manner, thereby aiding organizations in making superior decisions and devising appropriate plans for the future. Consequently, this review of literature delves into the extant research in the domains of geospatial analysis and machine learning principles to acquire valuable insights for NATO's strategic planning endeavours.

2.2 Literature Review

2.2.1 Regulation

Geospatial analysis refers to the examination of data in relation to its geographic location, specifically focusing on latitude and longitude. Each place on the world map has a corresponding location in terms of these coordinates. In order to analyse trends, geographic data serves as the foundation of the project. This approach has gained considerable importance due to its capacity to reveal patterns, hotspots, and, most importantly, identify high-risk areas based on the severity factor. Such identification is crucial for future developments in predicting and preventing similar attacks.

Yoshizumi, A. et al.(2020) highlights the importance of geospatial data and its role in several publications in the recent years. The trend of using spatial data is on the rise as such type of analysis help in addressing the common problems faced by the people in today’s world.

In the study by Thakur, N. et al.(2022) the authors presented a comprehensive data mining framework aimed at uncovering meaningful insights from vast datasets related to global terrorism. Their work involved data preprocessing, data cleaning, and the extraction of valuable information using the pandas library. Additionally, geospatial and location data analysis was conducted using Folium, alongside various data visualization tools such as Matplotlib, Seaborn, and Plotly. The paper showcased the integration of Foursquare API for accessing venue data, contributing to the understanding of trending venues around terrorist attack locations. This research demonstrates the application of data mining techniques to counterterrorism efforts and highlights the role of Python-based frameworks in addressing complex real-world challenges.

Deng, H. et al. (2023) presented an innovative geospatial data hub designed to facilitate online geospatial analysis. The platform leverages contemporary cloud service technologies, adheres to the latest OGC standard API interfaces, and adopts the GeoCube organizational model to seamlessly integrate and manage geospatial data and models. While this work primarily focuses on a broader geospatial context, it offers valuable insights into the development and utilization of geospatial data platforms.

Lee, Y. et al. (2019) performed geospatial analysis for uncovering tourist patterns in the Los Angeles area. Although the scope of the dataset is not same as our project but the paper extensively uses and highlights the importance of hot spot analysis using maps. Such type of analysis was informative and helpful for knowing the concentration of people and trend analysis. The findings of this paper have been helpful to note the significance of hotspot analysis in geospatial analysis.

One of the key references in our research is the work by Kanevski, M. et al. (2008) where the authors underscore the significance of machine learning including algorithms such as SVM in the analysis, processing, and visualisation of geographic and environmental data.

In the field of geospatial analysis for strategic planning and decision making, the classification of explosive incident data holds great importance. This stage involves the utilization of machine learning algorithms to categorize incidents based on their severity which can be

Kanevski, M. et al. (2008) suggest the use of different classifiers for geospatial analysis and highlighted the remarkable potential of machine learning algorithms in the domain of geospatial and environmental applications which are vital for the different phases of environmental data analysis, ranging from the exploration of spatial data to the identification and modeling of spatio-temporal patterns.

In conclusion, the literature review reveals the advancements made in geospatial analysis and machine learning, specifically in their application to the analysis of explosive data within the domain of counterterrorism. These methodologies offer NATO valuable resources to enhance planning, allocate resources effectively, and improve preparedness. By integrating analysis and machine learning, NATO can adopt an evidence-based approach to effectively address terrorist risks.

NATO consistently strives to promote peace and instil a sense of security among its member nations. Through the utilization of geospatial analysis and machine learning tools, NATO can gain a deeper understanding of the impact of such incidents and respond accordingly. It is important to acknowledge that NATO's resources are limited yet crucial. Therefore, it is imperative for NATO to identify the areas that require focused attention in the case of bombing or explosive incidents. Resources must be allocated strategically to enable future planning and minimize the impact of such incidents on both human casualties and the resulting destruction, encompassing property and the economy.

2.2 Summary

The literature review highlights the significance of geospatial analysis in various domains and emphasizes the role of machine learning algorithms in such type of analysis. Existing studies sets the foundation for our project idea for performing geospatial analysis and classification using machine learning.

Chapter 3

3.0 Methodology

3.1 Introduction

In this chapter we will analyse the life cycle of machine learning projects in detail along with the algorithms which will be used for our study. The objective is to provide a basic understanding of all the processes involved in development and deployment of a machine learning model.

3.2 Project Life Cycle

When undertaking any project or study that employs machine learning principles, it is imperative to adhere to and implement fundamental steps to achieve favourable outcomes. This is particularly relevant in our case, where we aim to perform classification on input data and geospatial analysis. The machine learning life cycle encompasses a series of steps that provide structure to the project and effectively allocate the company's resources. Adhering to these steps enables companies to develop sustainable, cost-effective, and high-quality AI products.

In this section, we will utilize the Cross-Industry Standard Process for the development of Machine Learning applications with Quality assurance methodology (CRISP-ML(Q)) to elucidate each step in the machine learning life cycle. The CRISP-ML(Q) is an industrial standard that facilitates the creation of sustainable machine learning applications.

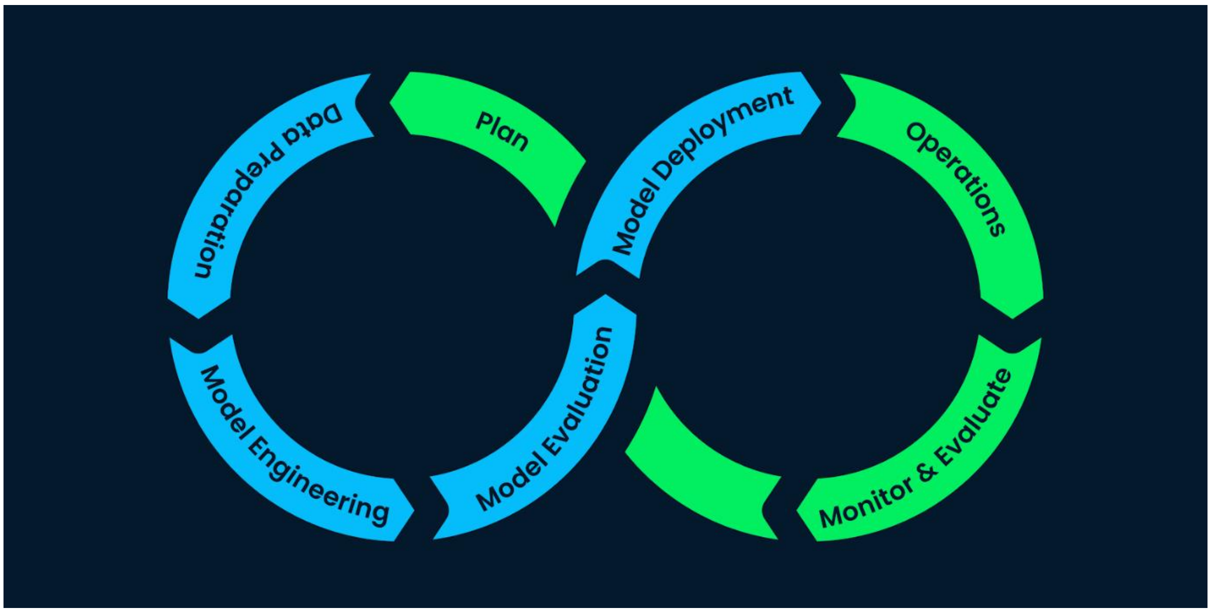


Figure 3: Life cycle of project

The 6 steps in a standard machine learning life cycle:

- Planning

- Data Preparation
- Model Engineering
- Model Evaluation
- Model Deployment
- Monitoring and Maintenance

Planning:

The planning phase encompasses the evaluation of the ML application's scope, success metric, and feasibility. It is imperative to comprehend the intricacies of the business and ascertain how machine learning can be employed to enhance the existing process. For instance, it is crucial to determine whether machine learning is necessary or if comparable outcomes can be achieved through straightforward programming techniques.

Data Preparation:

The section on data preparation is subdivided into four components: data acquisition and labelling, cleansing, management, and processing.

- Data acquisition and labelling: The initial step is to determine the method of data collection, which may involve gathering internal data, utilizing open-source resources, purchasing data from vendors, or generating synthetic data. Each approach has its advantages and disadvantages, and in some instances, data may be obtained through all four methods. Following data collection, labelling is necessary.
- Data cleansing: Subsequently, the data will undergo a cleansing process that involves imputing missing values, analysing incorrectly labelled data, eliminating outliers, and reducing noise.
- Data processing: The data processing phase entails feature selection, addressing imbalanced classes, feature engineering, data augmentation, and normalizing and scaling the data.
- Data management: Finally, data storage solutions, data versioning for reproducibility, metadata storage, and ETL pipeline creation will be established. This component ensures a consistent data flow for model training.

Model Engineering:

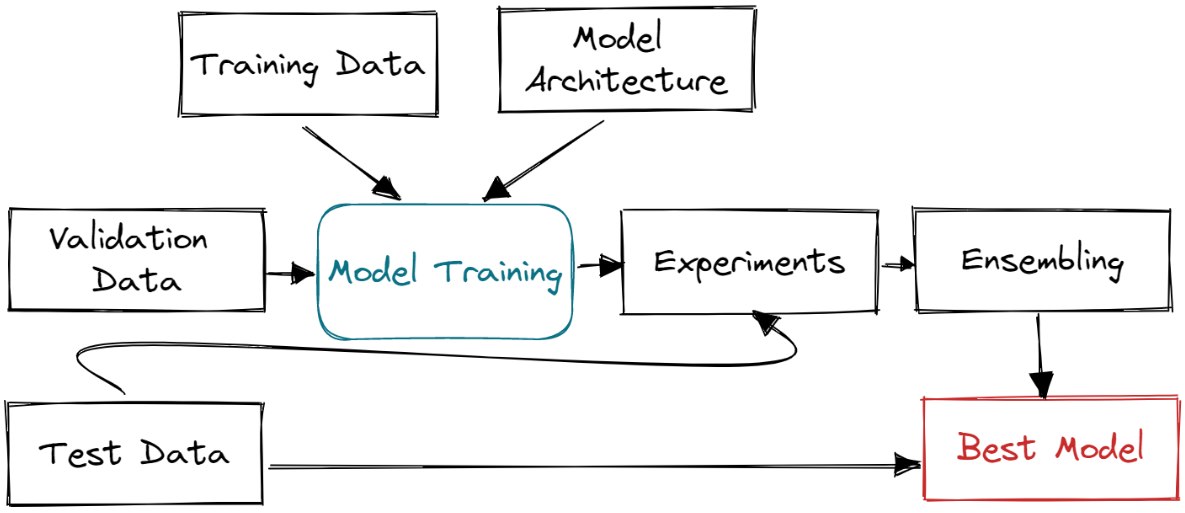


Figure 4: Model engineering

1. Constructing a proficient model architecture through thorough research.
2. Establishing model metrics.
3. Conducting training and validation of the model using the designated training and validation datasets.
4. Monitoring and documenting experiments, metadata, features, code modifications, and machine learning pipelines.
5. Implementing model compression and ensembling techniques.
6. Analysing and interpreting the outcomes by collaborating with domain knowledge experts.

Model Evaluation:

Initially, our model will undergo testing using a designated test dataset, with the active participation of subject matter experts to identify any errors in the predictions. Additionally, it is imperative that we adhere to established industrial, ethical, and legal frameworks when developing AI solutions.

Moreover, we will assess the robustness of our model by subjecting it to both random and real-world data, ensuring that it can provide prompt inferences that deliver value.

Ultimately, we will compare the obtained results with the predetermined success metrics and make a decision regarding the deployment of the model. Throughout this phase, meticulous documentation and versioning of every process will be maintained to uphold the standards of quality and reproducibility.

Model Deployment:

During this phase, we shall proceed with the deployment of machine learning models into the existing system. One such example is the implementation of automated warehouse labelling based on the product's shape. To achieve this, we shall integrate a computer vision model into the current system, which will utilize camera images to generate the labels.

Typically, these models can be deployed on various platforms such as cloud and local servers, web browsers, packaged software, and edge devices. Subsequently, one can access the predictions through APIs, web applications, plugins, or dashboards.

Monitoring and Maintenance:

Upon the deployment of the model to the production environment, it is imperative to maintain a continuous monitoring and enhancement of the system. This entails the monitoring of model metrics, hardware, and software performance, as well as customer satisfaction.

3.3 Research Methodology

In order to forecast or classify the severity factor which is essential for the NATO in their planning agenda for the future, we will be using the random forest classifier for classification. Although we use the random forest classifier, the algorithms used in the benchmark models will be explained in this section namely SVM, Gradient Boosting, Logistic Regression, Naïve Baiyes, KNN, Decision Tree, AdaBoost and MLP. A detailed explanation of the classifiers used in this study will follow and the we will be discussing their accuracy towards the end of the report.

3.3.1 Decision Tree

An internal node symbolizes a characteristic (or attribute), a branch symbolizes a rule for making decisions, and each leaf node signifies the outcome in a decision tree, which bears resemblance to a flowchart. The initial node at the top of the decision tree is referred to as the root node. It possesses the capability to segregate data based on attribute values. Recursive partitioning entails the iterative division of a tree. This framework, akin to a flowchart, facilitates the process of decision-making. It is a representation that accurately mirrors human thought processes. Decision trees are easily comprehensible and interpretable due to this characteristic.

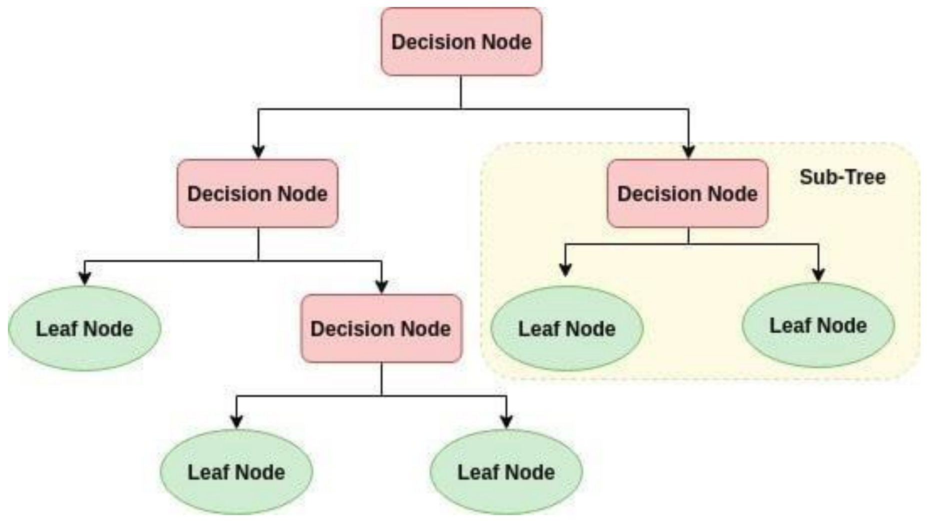


Figure 5: Structure of decision tree

The decision tree is a form of machine learning algorithm that can be characterized as a white box. It possesses an internal decision-making logic that is openly shared, unlike black box algorithms such as neural networks which lack this capability. In comparison to the neural network algorithm, the decision tree algorithm exhibits a faster training process. The temporal complexity of decision trees is determined by the quantity of records and the number of attributes present in the provided data. Notably, the decision tree strategy is non-parametric and distribution-free, as it does not depend on the assumptions of a probability distribution.

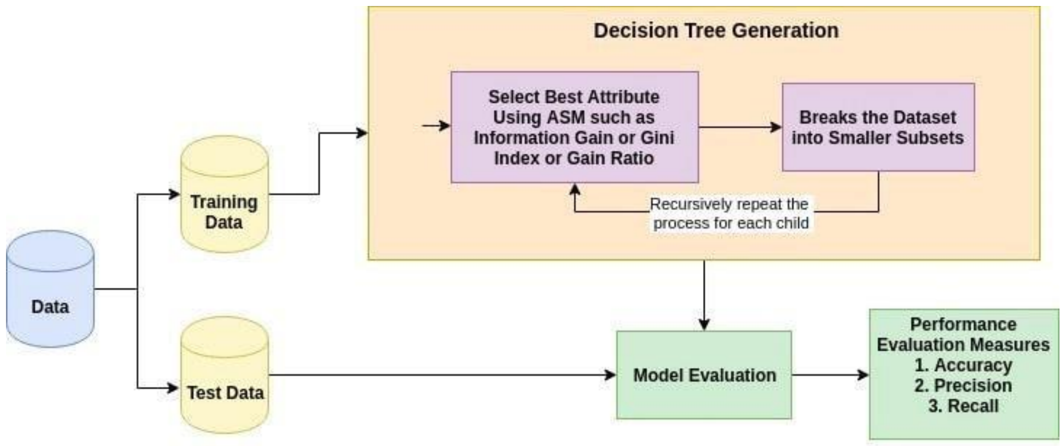

Decision Tree Generation
Select Best Attribute Using ASM such as Information Gain or Gini Index or Gain Ratio
Breaks the Dataset into Smaller Subsets
Recursively repeat the process for each child
Training Data
Data
Test Data
Model Evaluation
Performance Evaluation Measures
1. Accuracy
2. Precision
3. Recall

Figure 6: Steps of decision tree classification

The working of a decision tree algorithm is simple and easy to understand. For any decision tree algorithm, the fundamental principle is specified as follows:

1. The first step is to divide the records and using that we choose the best attribute using the ASM or Attribute Selection Measures.
2. Secondly, we divide our dataset into small sections, and make the attribute obtained in step 1 as the decision node.
3. We repeat this method in a recursive way of the decision tree until one of the conditions are satisfied:
    - The identical property value applies to each and every tuple.
    - There are no more characteristics left
    - There are no more occurrences.

3.3.2 K-Nearest Neigbhour:

The KNN algorithm utilizes the concept of "feature similarity" to predict the values of novel data points. Specifically, the value assigned to a new point is contingent upon its resemblance to the points contained within the training set.

The following are the steps involved in the algorithm:

1. Initially, the computation of the distance between each training point and the new point is carried out.
2. Subsequently, the k data points that exhibit the closest proximity (as per the distance metric) are selected.
3. The ultimate prediction for the new point is determined by computing the average of these selected data points.

The initial step involves the computation of the distance between each training point and the new point. Various methods exist to determine this distance, with the most commonly employed ones being the Euclidean, Manhattan (for continuous variables), and Hamming distances (for categorical variables).

- The Euclidean distance (y) is determined by taking the square root of the sum of the squared differences between a new point (x) and an existing point.
- The Manhattan distance is obtained by summing the absolute differences of two real vectors.

Distance functions

Euclidean $$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

Manhattan $$\sum_{i=1}^{k}|x_i - y_i|$$

- When conducting an analysis of categorical variables, it is recommended to employ the Hamming distance. This distance metric yields a value of zero when the values of (x) and (y) are identical, and a value of one when they differ.

$$D_H = \sum_{i=1}^{k}|x_i - y_i|$$

$$x = y \Rightarrow D = 0$$

$$x \neq y \Rightarrow D = 1$$

Once the distance between a new observation and the points in our training set has been measured, the subsequent step involves selecting the closest points. The number of points to be considered is determined by the value of k.

The following step entails choosing the appropriate k value. This value determines the number of neighbours that are considered when assigning a value to any new observation.

For a very low value of k (let us say k=1), the model tends to overfit on the training data, resulting in a significant error rate on the validation set. Conversely, a high value of k leads to poor performance on both the training set and the validation set. By carefully observing the validation error curve, it becomes evident that the curve reaches its minimum at k = 9. The model performs optimally at this value of k, although it may vary for different datasets. Due to its resemblance to an elbow, this curve is commonly referred to as an "elbow curve" and is typically utilized to determine the appropriate k value.

3.3.3 Naive Bayes Classifier

A Naive Bayes classifier is a probabilistic machine learning model utilized for classification tasks. The fundamental principle of the classifier is founded on the Bayes theorem:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

By utilizing Bayes' theorem, it is possible to determine the likelihood of the occurrence of hypothesis A, given that evidence B has transpired. In this context, B serves as the evidence while A represents the hypothesis. The underlying premise is that the predictors or features are independent, meaning that the presence of a specific feature does not impact the others. This characteristic is commonly referred to as "naive."

3.3.4 SVM or Support Vector Machine

The Support Vector Machine (SVM) is widely recognized as a prominent algorithm in Supervised Learning, serving as a valuable tool for both classification and regression tasks. Nonetheless, its predominant application lies in the field of classification within the field of Machine Learning.

The fundamental objective of the SVM algorithm is to establish an optimal line or decision boundary, referred to as a hyperplane, capable of effectively partitioning an n-dimensional space into distinct classes. This enables the seamless categorization of future data points into their appropriate categories.

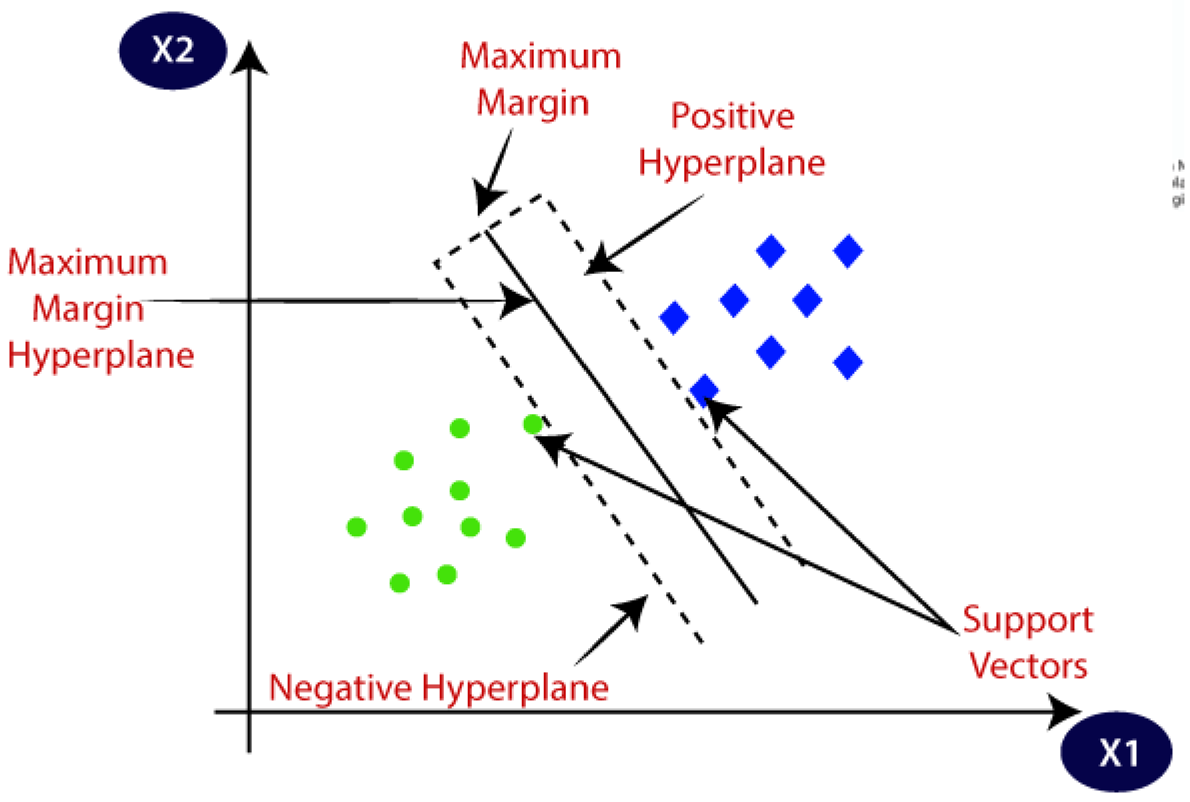


Figure 7: SVM

The Support Vector Machine (SVM) algorithm selects the extreme points or vectors that contribute to the construction of the hyperplane. These exceptional instances are referred to as support vectors, thus giving rise to the name of the algorithm as Support Vector Machine. We will examine the diagram provided, depicting two distinct categories that are distinguished by means of a decision boundary or hyperplane.

Working of SVM:

1. The functionality of the Support Vector Machine (SVM) algorithm can be comprehended through the utilization of an illustrative example. Let us consider a dataset comprising two distinct tags, namely "green" and "blue," with two corresponding features denoted as x1 and x2. Our objective is to develop a classifier capable of categorizing coordinate pairs (x1, x2) into either the "green" or "blue" class. Given that the dataset exists in a two-dimensional space, it is feasible to separate these two classes by employing a straight line. However, it is important to note that there may exist multiple lines that can effectively segregate these classes.

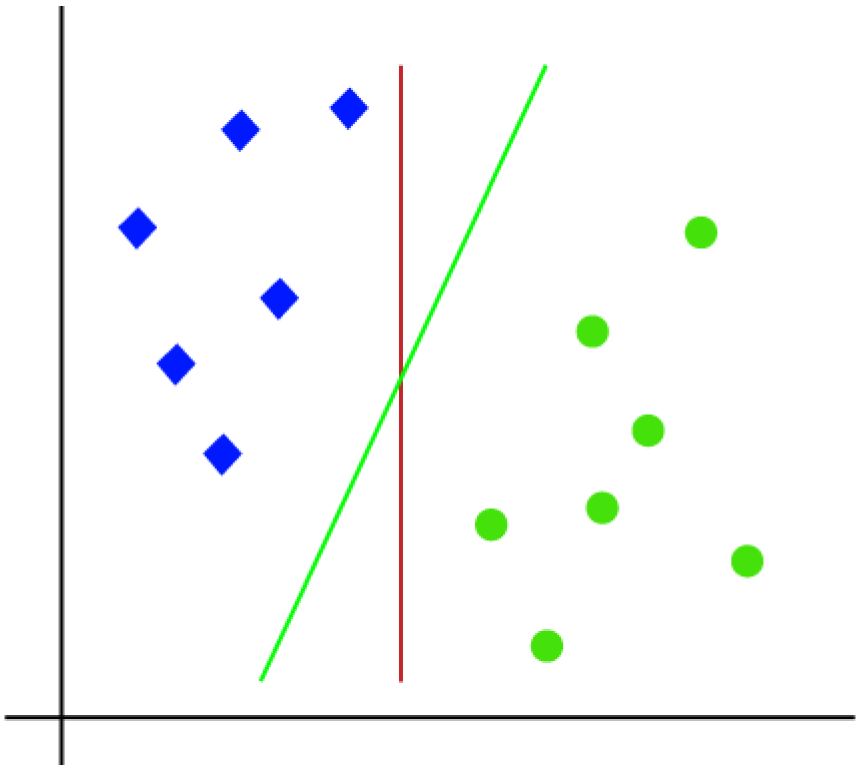


Figure 8: Separation of hyperplane in SVM

2. Therefore, the Support Vector Machine (SVM) algorithm facilitates the identification of the optimal line or decision boundary, which is referred to as a hyperplane. The SVM algorithm determines the nearest points of the lines from both classes, which are known as support vectors. The distance between the vectors and the hyperplane is defined as the margin, and the objective of the SVM is to maximize this margin. The hyperplane that possesses the maximum margin is recognized as the optimal hyperplane.

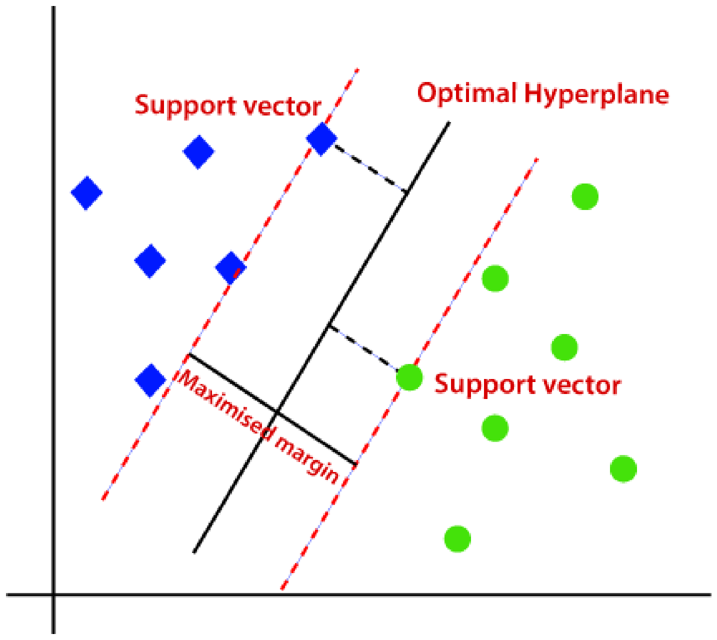


Figure 9: Support Vector

3.3.5 Gradient Boost Classifier:

Gradient Boosting is an algorithm that operates based on functional gradients, repeatedly selecting a function that leads towards a weak hypothesis or negative gradient. This iterative process aims to minimize a loss function. The Gradient Boosting classifier combines multiple weak learning models to create a robust predictive model.

The Gradient Boosting classification can be divided into three fundamental components:

- Loss Function:

  The primary objective of the loss function is to evaluate the model's predictive performance using the available data.

- Weak Learner:

  A weak learner is responsible for classifying the data, although it tends to make numerous errors in the process. Typically, decision trees are utilized as weak learners.

- Additive Model:

  The additive model describes how the trees are incrementally, iteratively, and sequentially added. With each iteration, we approach the final model.

The process of implementing a gradient boosting classifier entails the following steps:

1. Firstly, the model needs to be fitted.
2. Subsequently, the hyperparameters and parameters of the model should be adjusted accordingly.
3. Once the model is properly configured, forecasts can be generated.
4. Lastly, the obtained results can be interpreted and analysed.

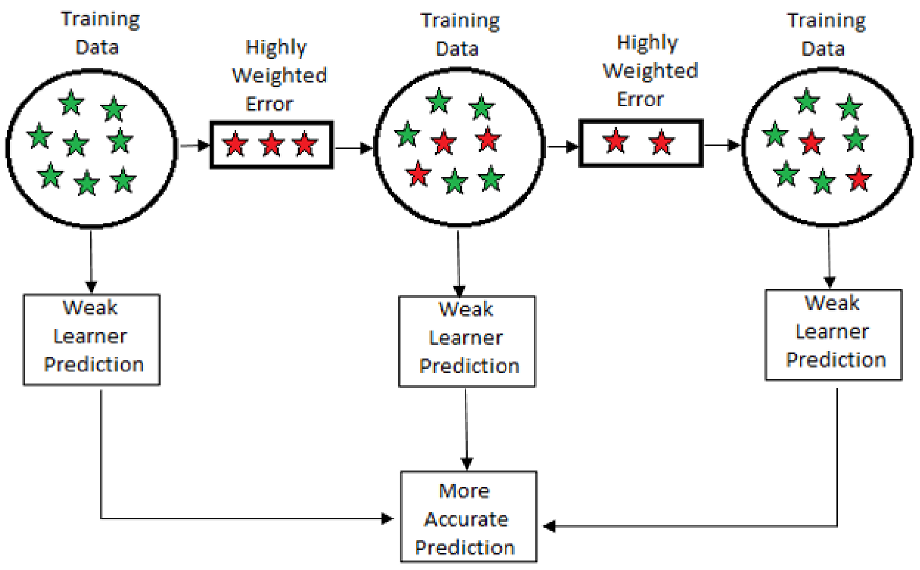


Figure 10: Gradient Boost Classifier

Advantages and Disadvantages of Gradient Boosting:

Advantages:

- Gradient Boosting often exhibits exceptional predictive accuracy.
- It offers a wide range of options for hyperparameter tuning and the ability to optimize various loss functions.
- Gradient Boosting frequently performs well with both numerical and categorical data without requiring pre-processing of the input.
- It can handle missing data without the need for imputation.

Disadvantages:

- The Gradient Boosting classifier may continue to improve to reduce all inaccuracies, potentially leading to overfitting and an overemphasis on outliers.
- Computational costs can be high, as Gradient Boosting often requires many trees (>1000), which can be memory and time-intensive.
- Due to its high degree of flexibility, numerous variables interact and significantly influence the behaviour of the technique.
- Gradient Boosting is less interpretable, although this can be remedied with various tools.

3.3.6 Logistic Regression:

Logistic regression is a supervised machine learning algorithm primarily utilized for classification tasks, wherein the objective is to forecast the probability of an instance belonging to a specific class. It is employed for classification algorithms and is referred to as logistic regression. The term "regression" is used because it takes the output of the linear regression function as input and employs a sigmoid function to estimate the probability for the given class.

The distinction between linear regression and logistic regression lies in the fact that linear regression produces a continuous value that can encompass any range, whereas logistic regression predicts the probability of an instance belonging to a particular class or not.

In logistic regression, a logit transformation is applied to the odds, which represents the probability of success divided by the probability of failure. This transformation is commonly known as the log odds or the natural logarithm of odds. The logistic function is represented by the following formulas:

Logit(pi) = 1/(1+ exp(-pi))

ln(pi/(1-pi)) = Beta_0 + Beta_1*X_1 + … + B_k*K_k

In the logistic regression equation presented, the dependent or response variable is logit(pi), while the independent variable is denoted as x. The beta parameter, also known as the coefficient, is commonly estimated using maximum likelihood estimation (MLE) in this model. MLE involves testing various values of beta through multiple iterations to optimize the fit of log odds. These iterations generate the log likelihood function, and the goal of logistic regression is to maximize this function in order to obtain the best estimate for the parameter.

Once the optimal coefficient (or coefficients, in the case of multiple independent variables) is determined, the conditional probabilities for each observation can be calculated, logged, and summed to yield a predicted probability. In binary classification, a probability less than 0.5 is indicative of predicting 0, while a probability greater than 0 predicts 1.

After the model has been computed, it is considered good practice to evaluate how well it predicts the dependent variable, which is referred to as goodness of fit. The Hosmer-Lemeshow test is a widely used method for assessing the fit of the model.

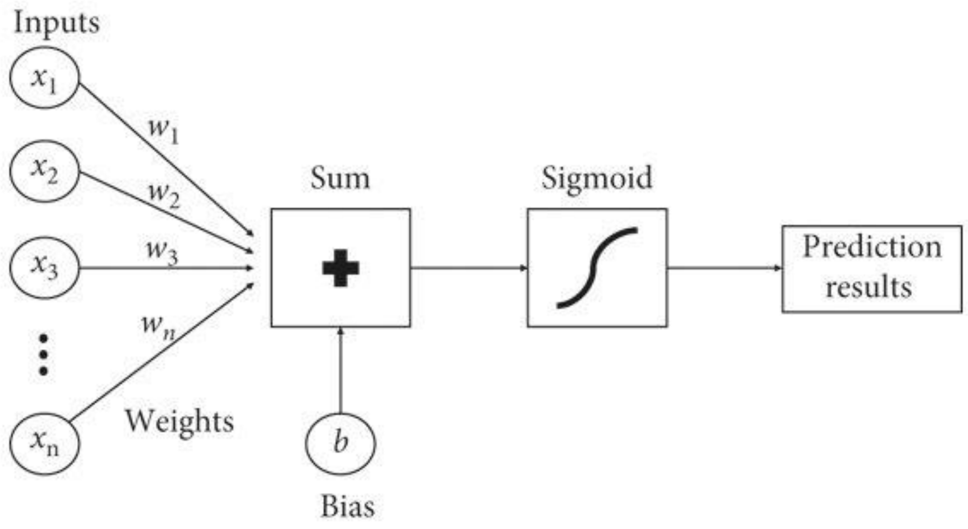


Figure 11: Logistic Regression

3.3.7 AdaBoost Classifier:

Ada-boost, also known as Adaptive Boosting, is an ensemble boosting classifier that was proposed by Yoav Freund and Robert Schapire in 1996. Its purpose is to increase the accuracy of classifiers by combining multiple classifiers. AdaBoost is an iterative ensemble method that builds a strong classifier by combining multiple poorly performing classifiers, resulting in a high accuracy strong classifier. The fundamental concept behind Adaboost is to set the weights of classifiers and train the data sample in each iteration to ensure accurate predictions of unusual observations. Any machine learning algorithm can be used as a base classifier if it accepts weights on the training set.

Adaboost must meet two conditions:

- The classifier should be trained interactively on various weighed training examples, and in each iteration,
- It should try to provide an excellent fit for these examples by minimizing training error.

The algorithm works in the following steps:

1. Initially, the Adaboost algorithm randomly selects a subset of training data.
2. The AdaBoost machine learning model is then iteratively trained by selecting the training set based on the accurate prediction of the previous training.
3. It assigns higher weights to incorrectly classified observations, increasing their probability of being correctly classified in the next iteration.
4. Additionally, it assigns weights to the trained classifiers in each iteration based on their accuracy. The more accurate a classifier is, the higher weight it receives.
5. This process continues until the complete training data fits without any errors or until the specified maximum number of estimators is reached.
6. To classify, a "vote" is performed across all the learning algorithms that have been built.

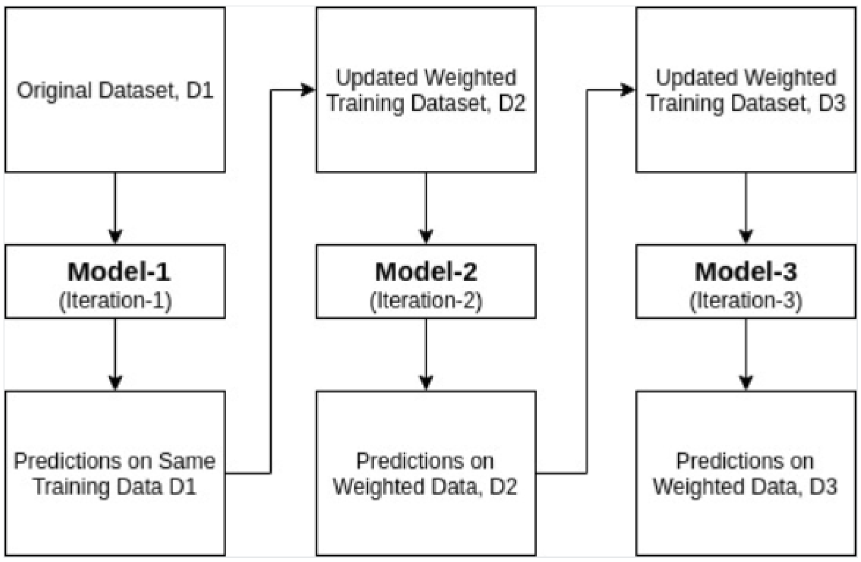

Original Dataset, D1
Updated Weighted Training Dataset, D2
Updated Weighted Training Dataset, D3
Model-1 (Iteration-1)
Model-2 (Iteration-2)
Model-3 (Iteration-3)
Predictions on Same Training Data D1
Predictions on Weighted Data, D2
Predictions on Weighted Data, D3

Figure 12: Classification using AdaBoost

3.3.8 MLP or Multi-Layer Perceptron:

The Multilayer Perceptron Classifier (MLPC) is a classification model that is founded on the feedforward artificial neural network. The MLPC is comprised of numerous layers of nodes, with each layer being fully connected to the subsequent layer in the network. The input layer of nodes represents the input data, while all other nodes map inputs to outputs through a linear combination of the inputs with the node's weights (w) and bias (b), and by applying an activation function. This can be expressed in matrix form for MLPC with K+1 layers as follows:

$$\mathrm{y}(\mathbf{x}) = \mathrm{f}_K(\ldots \mathrm{f}_2(\mathbf{w}_2^T \mathrm{f}_1(\mathbf{w}_1^T \mathbf{x} + b_1) + b_2) \ldots + b_K)$$

Nodes in intermediate layers use sigmoid function:

$$\mathrm{f}(z_i) = \frac{1}{1 + e^{-z_i}}$$

Nodes in the output layer use softmax function:

$$\mathrm{f}(z_i) = \frac{e^{z_i}}{\sum_{k=1}^{N} e^{z_k}}$$

The quantity of nodes N in the output layer is directly proportional to the quantity of classes.

MLPC utilizes backpropagation for the acquisition of knowledge by the model. The logistic loss function is employed for optimization purposes, and L-BFGS is utilized as an optimization routine.

The Advantages of MLP Classifiers:

MLP classifiers possess several benefits that make them a valuable tool in machine learning. Firstly, they are capable of modelling complex non-linear relationships between input features and target labels. This attribute renders them particularly useful for tasks that involve non-linear decision boundaries or intricate patterns.

Secondly, MLP classifiers exhibit a high degree of flexibility in handling various types of data, including numerical, categorical, and text data. Additionally, they can accommodate multiple output classes, making them suitable for multi-class classification problems.

Lastly, MLP classifiers possess the ability to automatically learn relevant features from raw input data through the hidden layers. This feature eliminates the need for manual feature engineering, thereby saving time and effort in the preprocessing stage.

3.4 Environment

3.4.1 Anaconda/Jupiter Notebook

Anaconda is a software, or more precisely, a freely available open-source platform that facilitates users in writing and executing code in Python, one of the most widely utilized programming languages. Developed by continuum.io, a company specializing in Python development, Anaconda serves as a fundamental tool and currently stands as the most prevalent means for acquiring proficiency in Python for domains such as scientific computing, data science, and machine learning. Additionally, it aids in establishing an environment accommodating various Python versions and package iterations. Furthermore, Anaconda enables users not only to create an environment but also to install, uninstall, and update packages within the respective environments.

There are several tools included in the anaconda navigator but for the sake of this project we will be using the most popular Jupyter Notebook which allows to run and edit documents or code and view the output and results. Some of the most noted advantages of using Jupyter Notebook are as follows:

- Easy-to-use
- Interactive data science environment
- Flexibility a key factor.

3.4.2 Python

Python is one of the most used programming languages in the world and the most relevant. Each and every programming language has its advantages and disadvantages, but when we talk about doing complex programs especially in the field of artificial intelligence, simplicity has to be a decisive factor to run and maintain such tasks. It has been proven over the years that python for machine learning is the best tool to work on. It helps programmers and developers to write reliable systems as due to its simplistic nature they can focus more on the problem at hand rather than the technical details of the programming language.

It is easy to learn and can be easily read and understood by humans which makes easier to build on the existing models. Another main feature of Python is that its extensive collection of libraries and frameworks. Amany programmers or developers can use the wide variety of built-in libraries to execute common pieces of code which are general in nature in order to reduce the development time. So, in order to sum up the following are some of the benefits of using python as the programming language:

- Simple and consistent
- Extensive collection of libraries and frameworks
- Platform Independence

3.4.3 Excel

Microsoft Excel is one of the most used software's around the world and not just for data analysis or machine learning capabilities. Although it does not have built in AI or machine learning capabilities, but is used in conjunction with other tools having machine learning capabilities. It is often used for preparing and organizing data which has to further used by machine learning models and has proven to be a vital tool for analysis.

Three of the most used software's/platforms were listed and explained in brief for the purpose of this project. Even though there are other platforms available, the features offered by the above listed were decisive and relevant to the project.

3.5 Summary

In this chapter we described the project life cycle which is necessary for any project based on machine learning principles. We analysed several classifiers in detail which were used in the benchmark models. Finally, the environment used for the project development such as the platforms used were discussed and described.

Chapter 4

4.0 Data Cleaning and Exploratory Data Analysis (EDA)

4.1 Introduction

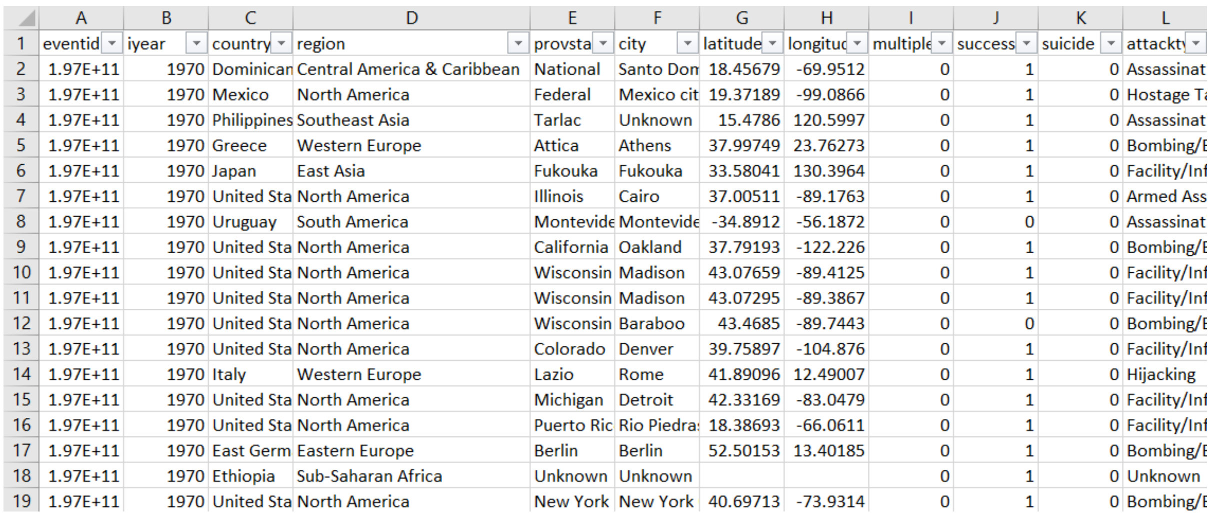
In this chapter we will have the first look on the dataset used in this study which is the GTD or Global Terrorism Dataset. Further we will explore the data, observe, and analyse the patterns and most importantly looks for any irregularities in the initial data.

For this section we will be using Microsoft Excel and the Jupyter Notebook (Anaconda) for executing the python code and viewing the results.

Using excel, we will have the first glance on the data and having a closer look at the attributes of the data. after which we execute the python code on Jupyter to perform data visualisation and data cleaning. Each step of the exploratory phase will be explained in detail as we progress through this chapter.

4.2 Data structure

The dataset is downloaded from the GTD website and comes with the .csv (coma separated values) extension. We can use Microsoft Excel to have a first glance at the data as shown below:



	A	B	C	D	E	F	G	H	I	J	K	L
1	eventid	iyear	country	region	provsta	city	latitude	longituc	multiple	success	suicide	attackty
2	1.97E+11	1970	Dominican	Central America & Caribbean	National	Santo Don	18.45679	-69.9512	0	1	0	Assassinat
3	1.97E+11	1970	Mexico	North America	Federal	Mexico cit	19.37189	-99.0866	0	1	0	Hostage Ta
4	1.97E+11	1970	Philippines	Southeast Asia	Tarlac	Unknown	15.4786	120.5997	0	1	0	Assassinat
5	1.97E+11	1970	Greece	Western Europe	Attica	Athens	37.99749	23.76273	0	1	0	Bombing/E
6	1.97E+11	1970	Japan	East Asia	Fukouka	Fukouka	33.58041	130.3964	0	1	0	Facility/Inf
7	1.97E+11	1970	United Sta	North America	Illinois	Cairo	37.00511	-89.1763	0	1	0	Armed Ass
8	1.97E+11	1970	Uruguay	South America	Montevide	Montevide	-34.8912	-56.1872	0	0	0	Assassinat
9	1.97E+11	1970	United Sta	North America	California	Oakland	37.79193	-122.226	0	1	0	Bombing/E
10	1.97E+11	1970	United Sta	North America	Wisconsin	Madison	43.07659	-89.4125	0	1	0	Facility/Inf
11	1.97E+11	1970	United Sta	North America	Wisconsin	Madison	43.07295	-89.3867	0	1	0	Facility/Inf
12	1.97E+11	1970	United Sta	North America	Wisconsin	Baraboo	43.4685	-89.7443	0	0	0	Bombing/E
13	1.97E+11	1970	United Sta	North America	Colorado	Denver	39.75897	-104.876	0	1	0	Facility/Inf
14	1.97E+11	1970	Italy	Western Europe	Lazio	Rome	41.89096	12.49007	0	1	0	Hijacking
15	1.97E+11	1970	United Sta	North America	Michigan	Detroit	42.33169	-83.0479	0	1	0	Facility/Inf
16	1.97E+11	1970	United Sta	North America	Puerto Ric	Rio Piedra:	18.38693	-66.0611	0	1	0	Facility/Inf
17	1.97E+11	1970	East Germ	Eastern Europe	Berlin	Berlin	52.50153	13.40185	0	1	0	Bombing/E
18	1.97E+11	1970	Ethiopia	Sub-Saharan Africa	Unknown	Unknown			0	1	0	Unknown
19	1.97E+11	1970	United Sta	North America	New York	New York	40.69713	-73.9314	0	1	0	Bombing/E

Figure 13: Preview of GTD Data

The datasets used in this project: GTD or Global Terrorism Dataset

4.2.1 About the dataset:

The START. (n.d.). specifies that the Global Terrorism Database (GTD) is a publicly accessible database that encompasses data on terrorist incidents worldwide spanning from 1970 to 2020, with future annual updates anticipated. Distinguishing itself from numerous other event databases, the GTD provides comprehensive information on both domestic and transnational/international terrorist activities that have taken place within this timeframe, currently comprising over 200,000 cases. Pertaining to each incident recorded in the GTD, details such as the date and location of the event, the weaponry employed, the target's nature, the number of casualties, and, when ascertainable, the accountable group or individual are made available.

4.2.2 Characteristics of the GTD

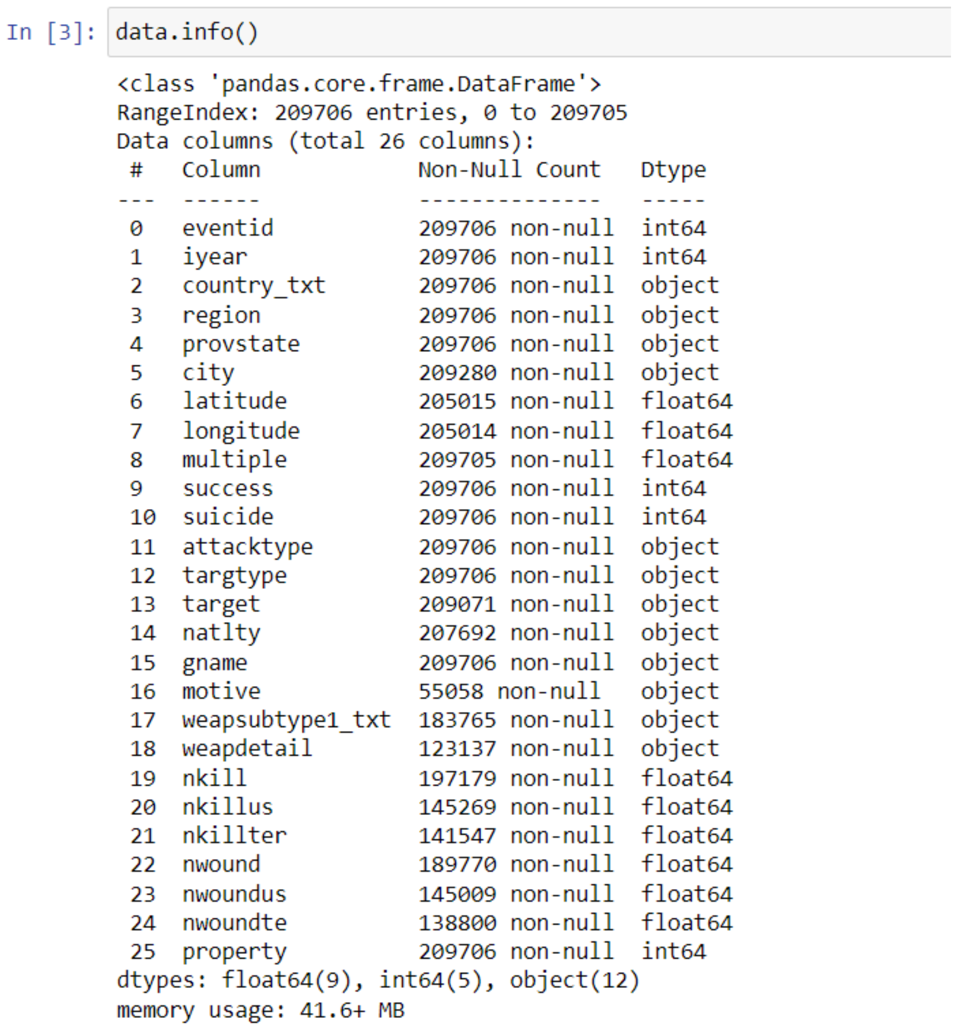
According to the GTD website (START, n.d.), the following are the main features of the dataset:

- Contains information on over 200,000 terrorist attacks
- Currently the most comprehensive unclassified database on terrorist attacks in the world
- Includes information on more than 88,000 bombings, 19,000 assassinations, and 11,000 kidnappings since 1970
- Includes information on at least 45 variables for each case, with more recent incidents including information on more than 120 variables
- More than 4,000,000 news articles and 25,000 news sources were reviewed to collect incident data from 1998 to 2017 alone

From the above characteristics it is vital to note that this dataset has more than 88,000 bombings data which is very relevant to the subject topic which is the geospatial analysis of explosive data.

4.2.3 More on the data:

After reading the dataset and storing into a data frame, we can see the various attributes of the GTD dataset:



```
In [3]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209706 entries, 0 to 209705
Data columns (total 26 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   eventid           209706 non-null  int64  
 1   iyear             209706 non-null  int64  
 2   country_txt       209706 non-null  object 
 3   region            209706 non-null  object 
 4   provstate         209706 non-null  object 
 5   city              209280 non-null  object 
 6   latitude          205015 non-null  float64
 7   longitude         205014 non-null  float64
 8   multiple          209705 non-null  float64
 9   success           209706 non-null  int64  
 10  suicide           209706 non-null  int64  
 11  attacktype        209706 non-null  object 
 12  targtype          209706 non-null  object 
 13  target            209071 non-null  object 
 14  natlty            207692 non-null  object 
 15  gname             209706 non-null  object 
 16  motive            55058 non-null   object 
 17  weapsubtype1_txt  183765 non-null  object 
 18  weapdetail        123137 non-null  object 
 19  nkill             197179 non-null  float64
 20  nkillus           145269 non-null  float64
 21  nkillter          141547 non-null  float64
 22  nwound            189770 non-null  float64
 23  nwoundus          145009 non-null  float64
 24  nwoundte          138800 non-null  float64
 25  property          209706 non-null  int64  
dtypes: float64(9), int64(5), object(12)
memory usage: 41.6+ MB
```

From the above figure, there are a variety of attributes available for the machine learning model and further analysis. Let us take a closer look into some of the important features given in the dataset:

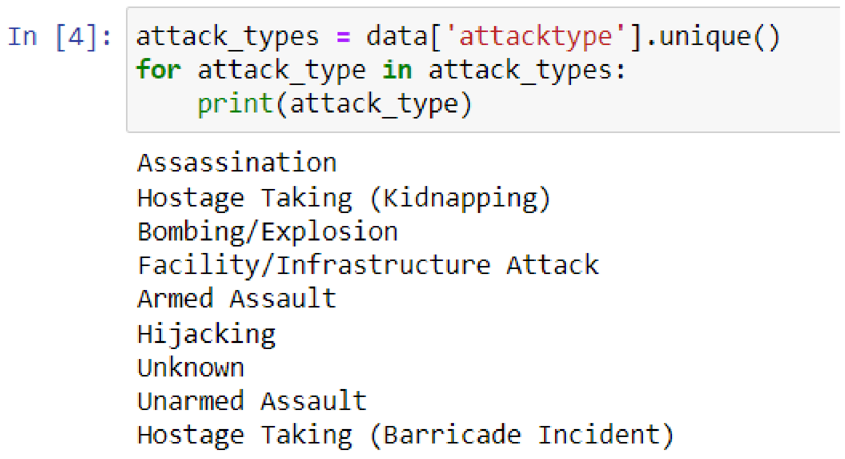
S.NO	ATTRIBUTES	EXPLANATION	EXAMPLE DATA
1.	Eventid	each attack or incident has a unique id	197000000001
2.	Iyear	the year of the incident	1970
3.	country_txt	the country where the attack happened	Dominican Republic
4.	Region	the region of the attack	Western Europe

5.	City	the city where the attack took place	Santo Domingo
6.	Latitude	the latitude position of attack	18.456792
7.	Longitude	the longitude position of attack	-69.951164
8.	Attacktype	what kind of attack took place	Assassination
9.	Targtype	what kind of personnel or institution was affected	Private Citizens & Property
10.	Target	the name of the target	Julio Guzman
11.	Natlty	the nationality of the affected	Dominican Republic
12.	Gname	the name of the group responsible	Black Nationalists
13.	Motive	the reason for the attack if any	To protest the Cairo Illinois Police Department
14.	weapsubtype1_txt	the exact type of weapon used	Automatic or Semi-Automatic Rifle
15.	Weapdetail	more detailed explanation of the attack weapon	Explosive
16.	Nkill	the number of people killed	7
17.	Nwound	the number of people wounded	51

From the above table we can see the various columns or features in the input dataset. For the purpose of this study, we will not be dealing with all the data due to its huge size as here we are dealing with data from the 1970 to 2020. Also, we will be refining this base dataset further according to our needs and perform data cleaning and pre-processing steps which are essential in order to make the input dataset into a suitable form for the geospatial analysis and machine learning models.

4.3 Data Preprocessing

The GTD dataset has a huge variety of "attacktype" in their dataset which can be seen as follows:



```
In [4]: attack_types = data['attacktype'].unique()
        for attack_type in attack_types:
            print(attack_type)

        Assassination
        Hostage Taking (Kidnapping)
        Bombing/Explosion
        Facility/Infrastructure Attack
        Armed Assault
        Hijacking
        Unknown
        Unarmed Assault
        Hostage Taking (Barricade Incident)
```

Since we are doing a geospatial analysis of explosive data, it is only relevant to filter the GTD dataset using the code:

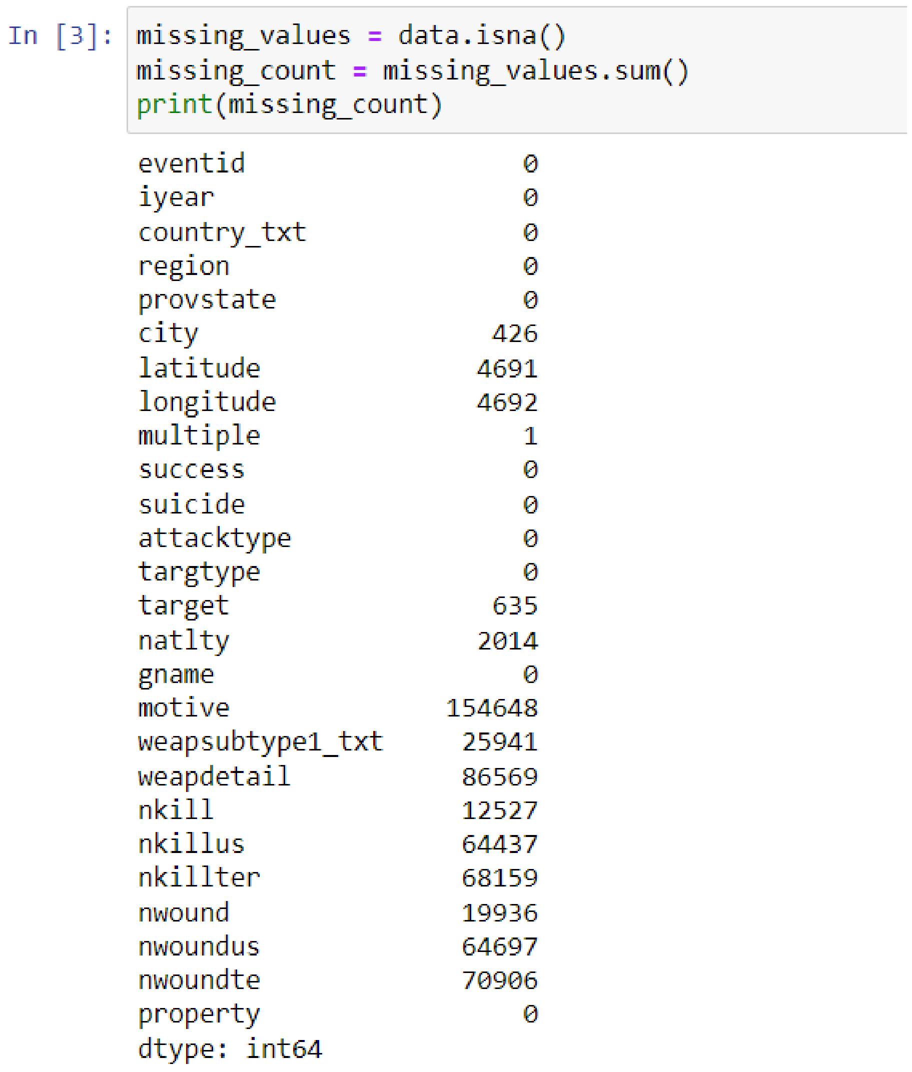
data = data[data['attacktype'] == 'Bombing/Explosion']

The resulting dataset will now only hold records for explosive data which is the primary focus of this project.

4.4 Data Cleaning

Data cleaning refers to the systematic process of rectifying and eliminating erroneous, corrupted, replicated, or deficient data within a given dataset. Since we are not the author/owner of the dataset, we cannot assume that it will be a prefect data since it is a very large dataset. Failure to treat the data before feeding it into models and conducting further analysis may result in unreliable outcomes and even errors.

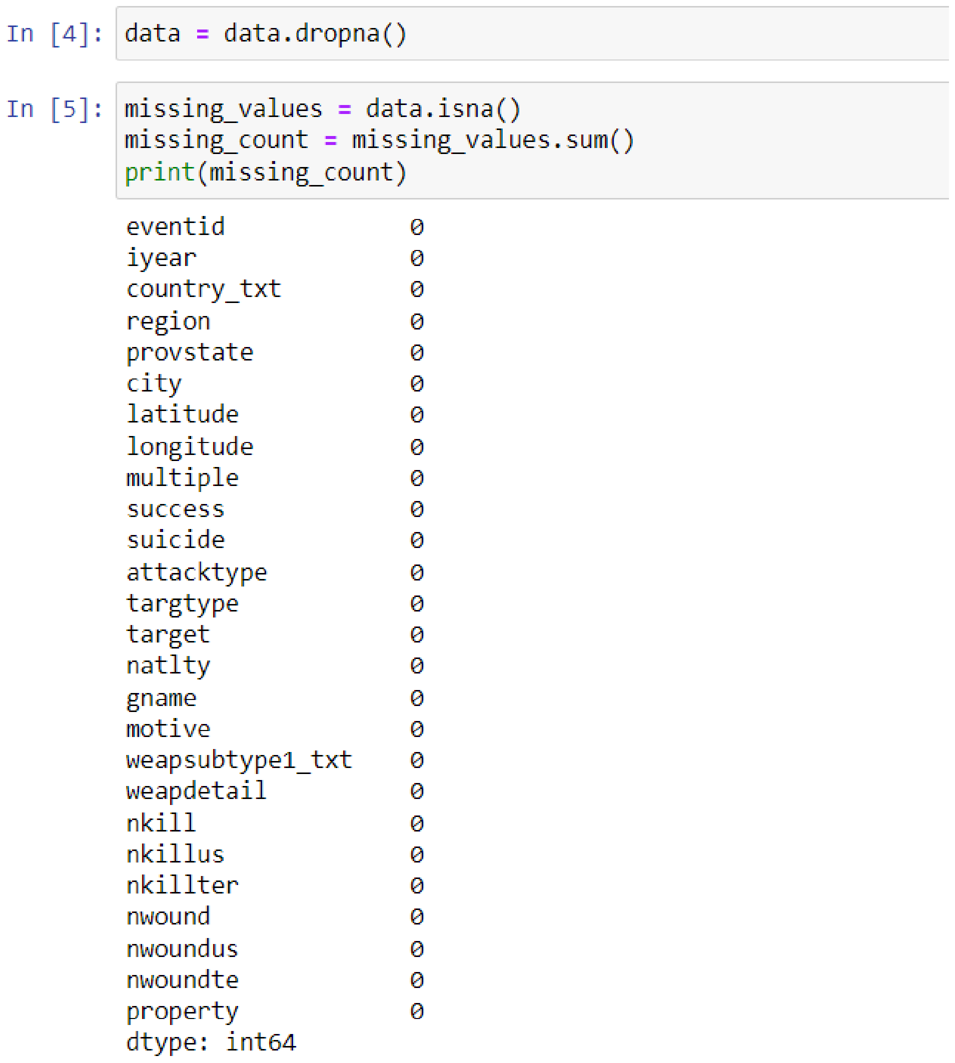
Firstly, the first thing to do is to handle the missing values. When we look in the dataset, we can see that almost every column has rows which hold no value or is 'blank' whose count can be seen below:



```
In [3]: missing_values = data.isna()
        missing_count = missing_values.sum()
        print(missing_count)
```

```
eventid                  0
iyear                    0
country_txt              0
region                   0
provstate                0
city                   426
latitude              4691
longitude             4692
multiple                 1
success                  0
suicide                  0
attacktype               0
targtype                 0
target                 635
natlty                2014
gname                    0
motive              154648
weapsubtype1_txt     25941
weapdetail           86569
nkill                12527
nkillus              64437
nkillter             68159
nwound               19936
nwoundus             64697
nwoundte             70906
property                 0
dtype: int64
```

Having huge of missing data is not beneficial in most cases and for the sake of our project we will be removing rows with missing data:



```
In [4]: data = data.dropna()
```

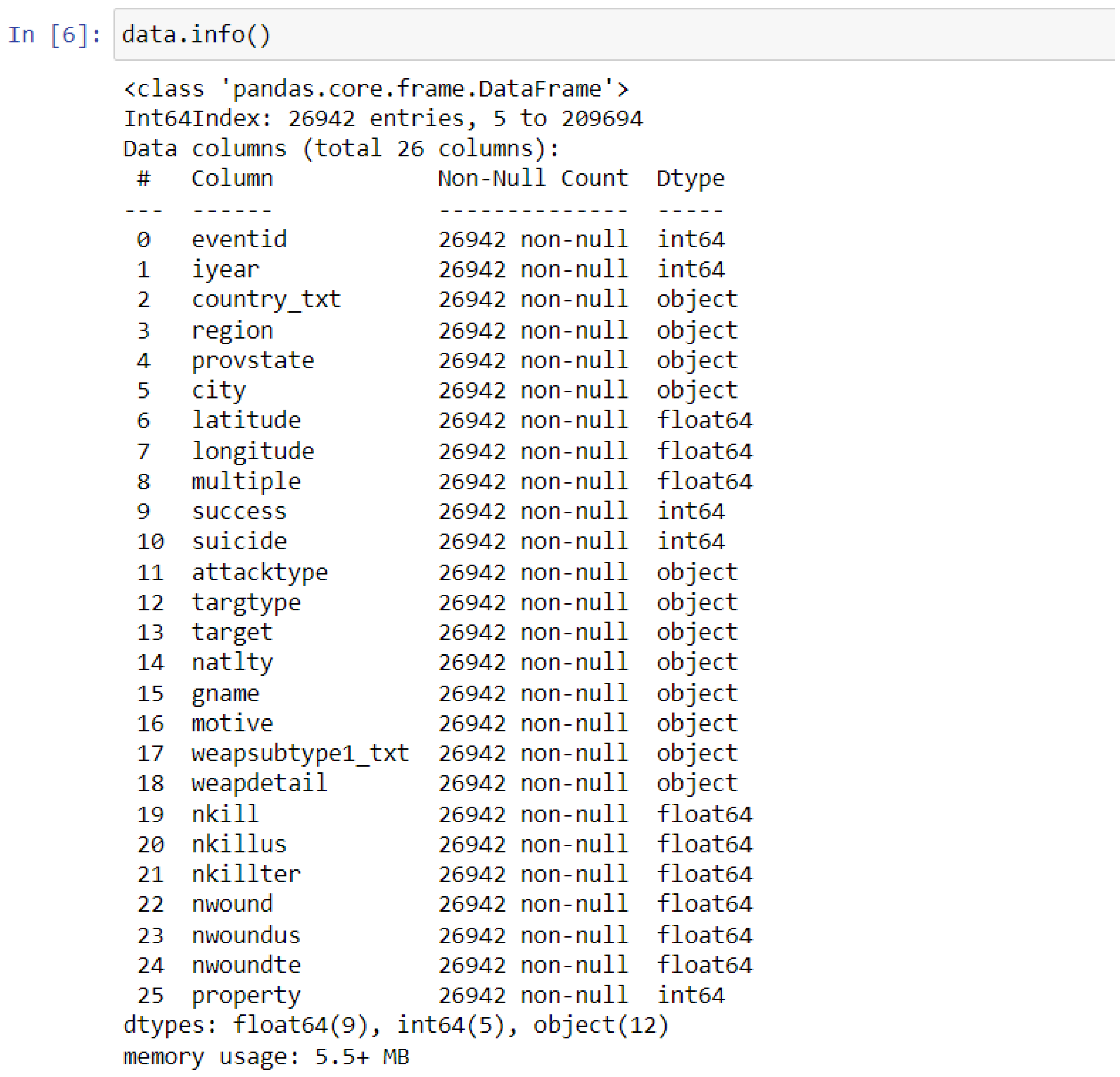
```
In [5]: missing_values = data.isna()
        missing_count = missing_values.sum()
        print(missing_count)
```

```
eventid            0
iyear              0
country_txt        0
region             0
provstate          0
city               0
latitude           0
longitude          0
multiple           0
success            0
suicide            0
attacktype         0
targtype           0
target             0
natlty             0
gname              0
motive             0
weapsubtype1_txt   0
weapdetail         0
nkill              0
nkillus            0
nkillter           0
nwound             0
nwoundus           0
nwoundte           0
property           0
dtype: int64
```

Now as we can from the above code structure, using the dropna () we remove the rows with missing values and then we check again to see the missing count which as you can now see show 0 for all columns. It must be noted that we still have sufficient data to perform geospatial analysis.

Secondly, there are around 26 columns in the input dataset of which not all are relevant to the project topic. One always must remove unwanted columns as the more precise the set of features are for the machine learning model and geospatial analysis, the easier it will be for the model to deliver reliable results.

Using the data.info () we can see the list of currently present columns in the dataset:

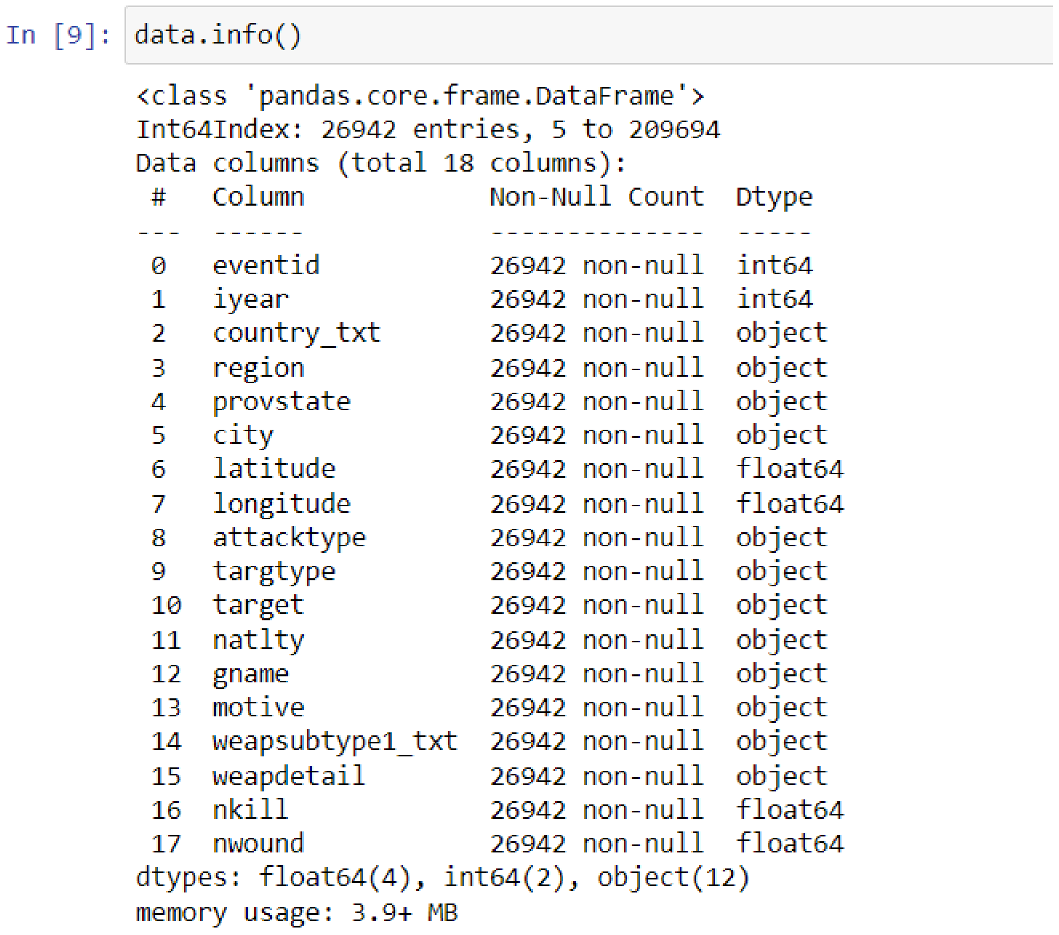


```
In [6]: data.info()

        <class 'pandas.core.frame.DataFrame'>
        Int64Index: 26942 entries, 5 to 209694
        Data columns (total 26 columns):
         #   Column            Non-Null Count  Dtype  
        ---  ------            --------------  -----  
         0   eventid           26942 non-null  int64  
         1   iyear             26942 non-null  int64  
         2   country_txt       26942 non-null  object 
         3   region            26942 non-null  object 
         4   provstate         26942 non-null  object 
         5   city              26942 non-null  object 
         6   latitude          26942 non-null  float64
         7   longitude         26942 non-null  float64
         8   multiple          26942 non-null  float64
         9   success           26942 non-null  int64  
         10  suicide           26942 non-null  int64  
         11  attacktype        26942 non-null  object 
         12  targtype          26942 non-null  object 
         13  target            26942 non-null  object 
         14  natlty            26942 non-null  object 
         15  gname             26942 non-null  object 
         16  motive            26942 non-null  object 
         17  weapsubtype1_txt  26942 non-null  object 
         18  weapdetail        26942 non-null  object 
         19  nkill             26942 non-null  float64
         20  nkillus           26942 non-null  float64
         21  nkillter          26942 non-null  float64
         22  nwound            26942 non-null  float64
         23  nwoundus          26942 non-null  float64
         24  nwoundte          26942 non-null  float64
         25  property          26942 non-null  int64  
        dtypes: float64(9), int64(5), object(12)
        memory usage: 5.5+ MB
```

Now we remove 8 columns which are not relevant to the project using the data.drop () command. The list of columns namely:

- multiple
- success
- suicide
- property
- nkillus
- kilter
- nwoundus
- nwoundte

In order to check that the above-mentioned columns have indeed be removed, we again use the data.info () command:



```
In [9]: data.info()

        <class 'pandas.core.frame.DataFrame'>
        Int64Index: 26942 entries, 5 to 209694
        Data columns (total 18 columns):
         #   Column            Non-Null Count  Dtype
        ---  ------            --------------  -----
         0   eventid           26942 non-null  int64
         1   iyear             26942 non-null  int64
         2   country_txt       26942 non-null  object
         3   region            26942 non-null  object
         4   provstate         26942 non-null  object
         5   city              26942 non-null  object
         6   latitude          26942 non-null  float64
         7   longitude         26942 non-null  float64
         8   attacktype        26942 non-null  object
         9   targtype          26942 non-null  object
         10  target            26942 non-null  object
         11  natlty            26942 non-null  object
         12  gname             26942 non-null  object
         13  motive            26942 non-null  object
         14  weapsubtype1_txt  26942 non-null  object
         15  weapdetail        26942 non-null  object
         16  nkill             26942 non-null  float64
         17  nwound            26942 non-null  float64
        dtypes: float64(4), int64(2), object(12)
        memory usage: 3.9+ MB
```

Therefore, after removing the unwanted columns from the dataset, the result is a much smaller dataset which is smaller but more relevant for the analysis phase. From the initial 26 set of features or columns, we have chosen only 17.

4.5 Feature Engineering

4.5.1 Introduction:

Feature engineering is an essential component in the development of any machine learning model or project. It entails the deliberate selection or alteration of features within our dataset, with the primary objective of enhancing the model's performance and overall accuracy using the available data. The significance of feature engineering lies in the fact that the choice of features greatly influences the overall success of the project.

4.5.2 Feature Selection:

When it comes to deciding the set of features for our model, it is necessary to understand what our aim of the project is after all. The focus of the project "Geospatial analysis of explosive data for strategic planning and decision making" is the spatial data along with its associated features. Therefore, we select both latitude and longitude in our set of features. Both latitude and longitude are the geographical coordinates that specify a location's position on the surface of the earth in relation to the equator(latitude) and prime meridian (longitude). These points provide us a precise way to identify any point on earth.

The next feature to be selected is the year of the attack/incident. The GTD dataset offers users with a time series dataset which allows us to analyse and further capture trends, patterns, or seasonality in our data. In further chapters we will see how the year plays vital role in geospatial analysis phase and other visualisations.

The next feature to be included is the targtype or the target type which indicate the different variety of targets attacked by explosive incidents. Different target types have different levels of security and by including it as a feature, the model will be able to identify vulnerable areas that require enhanced security measures.

We also include target as a feature which offers more granular level of detail to the exact and specific target that was attacked if any.

gname or Group Name is the next feature to be added to the set of features. It is relevant to the aim of the project and specifies the group responsible for the attack for e.g., Black Nationalists. It must be noted that not always we know the groups which organised and conducted the attack, hence the dataset holds the value ‘Unknown’ for those set of attacks and they have been also taken into consideration for our project.

Finally, the nationality of the attack grp is also taken a feature for the project which can be vital to understand the international dimensions of the attacks in general.

The target variable in this case is severity which will be created using the existing set of features in the original dataset, which in turn will be explained in the next section.

4.5.3 Feature Creation and Transformation:

In this section we will see the creation of a new feature i.e., ‘severity’ and we will see the relevance of the feature to our project. As we know we are dealing with a terrorism dataset with several features but there is no such feature which describes how intense the impact has been on that area due to the attack or in common sense how severe it has been.

If we can quantify the severity of explosive incidents, we can assess the level of risk associated with different events. High severe attacks/incidents will require more attention and planning. They also help in resource allocation as incidents with higher scores must be allocated more resources which will help them mitigate their impact in the future such as security forces, medical facilities etc.

Also in geospatial analysis, we will be able to identify regions with severity as the key factor which will help NATO to study the hotspots of such attacks in the past years and observe the trends and patterns and act accordingly. It will help in long term preparedness for NATO to develop proactive strategies and contingency plans

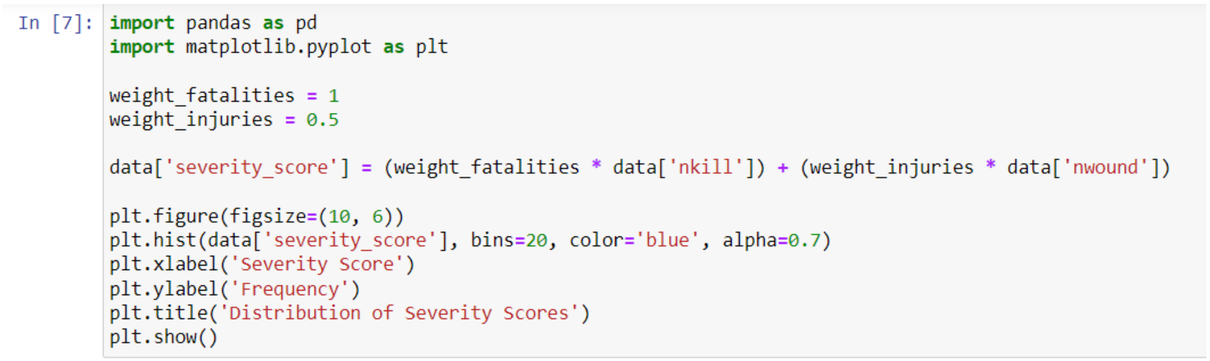
In order to calculate the severity value for each incident we will add the number of people killed with the number of people wounded in the attack. To both the values we multiple a certain weight to each of them by assuming that both have different levels of importance.

Fatalities and injuries have different levels of impact as one leads to loss of life which is more severe than the other. Hence to show this level of detail in our analysis we specify the following weights:

weight_fatalities = 1

weight_injuries = 0.5

Now we can run the python code for the whole segment:



```
In [7]: import pandas as pd
        import matplotlib.pyplot as plt

        weight_fatalities = 1
        weight_injuries = 0.5

        data['severity_score'] = (weight_fatalities * data['nkill']) + (weight_injuries * data['nwound'])

        plt.figure(figsize=(10, 6))
        plt.hist(data['severity_score'], bins=20, color='blue', alpha=0.7)
        plt.xlabel('Severity Score')
        plt.ylabel('Frequency')
        plt.title('Distribution of Severity Scores')
        plt.show()
```

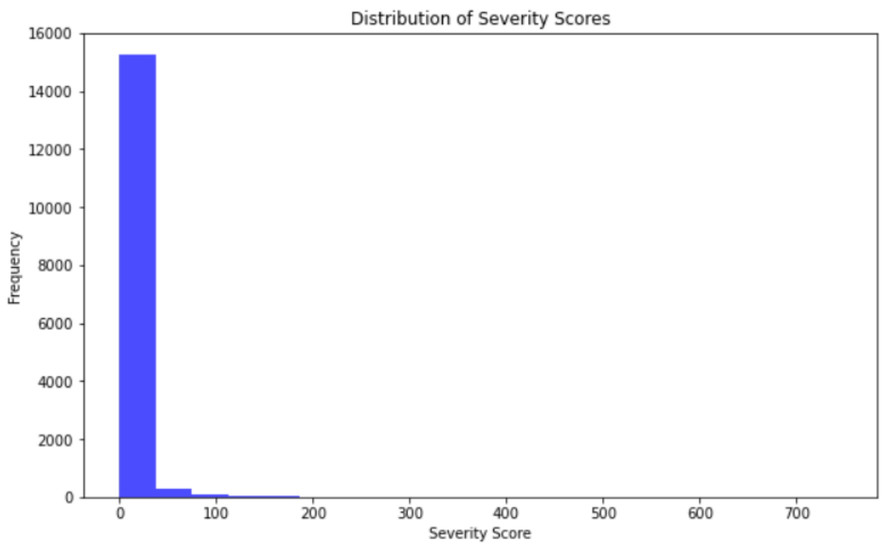


Figure 14: Distribution of Severity Score

The bar graph shows the distribution of severity scores generated after executing the code.

Now that we have created a new column to the dataset ie ‘severity_score’, it in itself is not sufficient for the purpose of this project. Using the severity score we will group the data into 3 categories i.e.

Category(Severity)	Value
Low severity	0
Medium	1
High severity	2

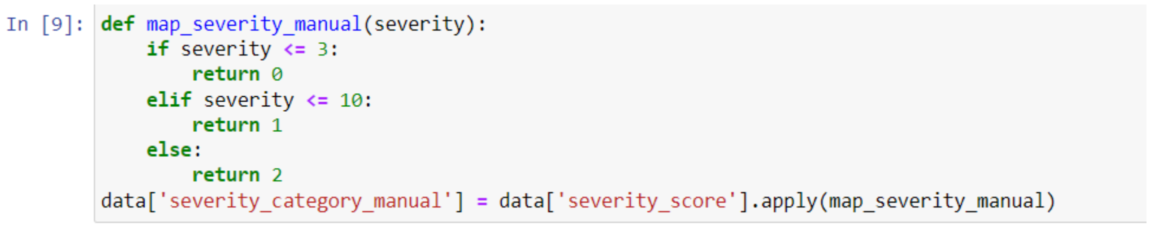
The severity scores ranges from 0 to 746 which is being divided into the three categories. The cutoff value is as follows:

if severity <= 3 : then less severe

if severity <= 10: then moderately severe

else it will be highly severe

Code executed for the above phase:



```
In [9]: def map_severity_manual(severity):
            if severity <= 3:
                return 0
            elif severity <= 10:
                return 1
            else:
                return 2
        data['severity_category_manual'] = data['severity_score'].apply(map_severity_manual)
```

The cutoff value has been decided and selected so as to have sufficient data in all 3 classes which can be seen in figure 15:

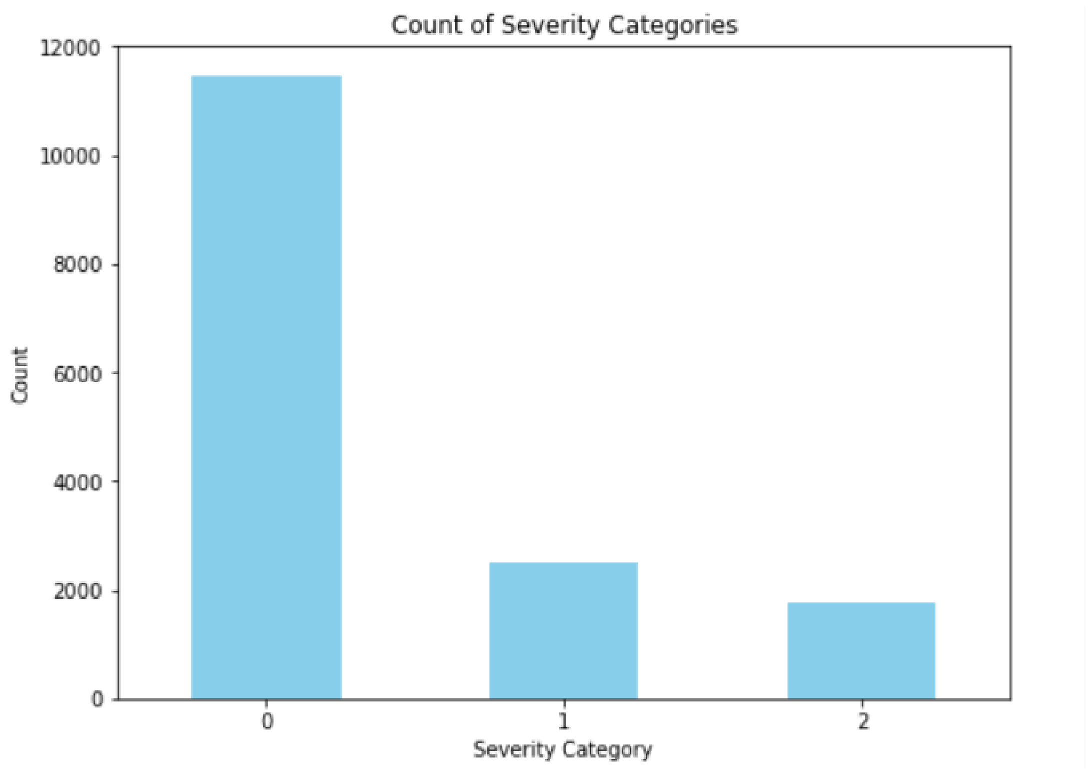
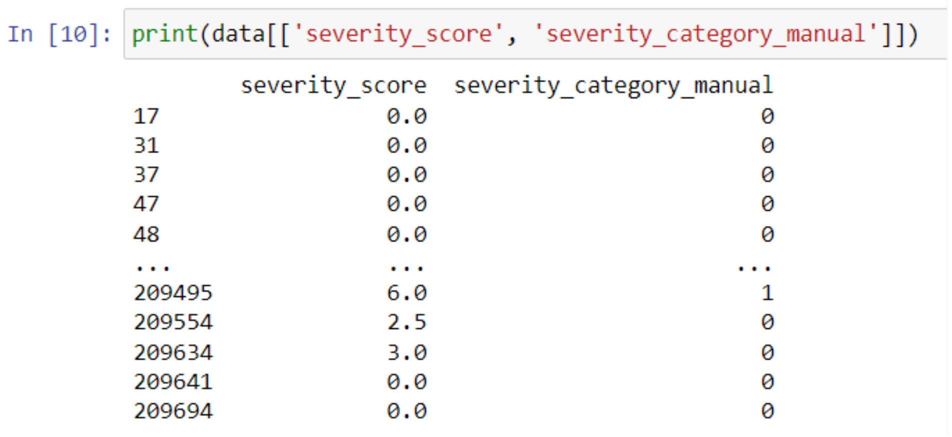


Figure 15 : Count of severity categories

Therefore for the classification phase of the project, we have created a new feature ‘severity_category_manual ’ having 3 classes namely 0,1,2 which each indicate their respective levels of severity. A small preview of the new data is shown below:



```
In [10]: print(data[['severity_score', 'severity_category_manual']])

         severity_score  severity_category_manual
17                  0.0                         0
31                  0.0                         0
37                  0.0                         0
47                  0.0                         0
48                  0.0                         0
...                 ...                       ...
209495              6.0                         1
209554              2.5                         0
209634              3.0                         0
209641              0.0                         0
209694              0.0                         0
```

4.5.4 Conclusion:

In conclusion, feature engineering is very essential for any project which uses machine learning capabilities and played its part in enhancing the effectiveness of our geospatial analysis project of explosive data. The set of features selected from the input dataset were in line with the requirements and goal of the project. We also created a new feature ‘severity’ by doing some geometric calculations on a set of available features from the input dataset.

There were also some challenges or limitations encountered while feature engineering like the subjectivity of score weights and the cutoff thresholds. In order to further improve our analysis, we can always explore advanced feature selection techniques in the future.

4.6 Description of features

The below table shows us the set of features which have been taken into consideration for the machine learning phase of the project. Each feature has been explained in detail in the previous section:

S.NO	FEATURE	DETAIL	EXAMPLE DATA
1.	Iyear	the year of the incident	1970
2.	Latitude	the latitude position of attack	18.456792
3.	Longitude	the longitude position of attack	-69.951164
4.	Targtype	what kind of personnel or institution was affected	Private Citizens & Property
5.	Gname	the name of the group responsible	Black Nationalists
6.	Natlty	the nationality of the affected	Dominican Republic
7.	Target	the name of the target	Julio Guzman

4.7 Data Splitting

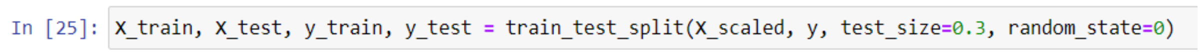
Data splitting in the context of machine learning and data analysis is the process of dividing the dataset into two or more subsets for specific purposes. basically, we divide the dataset into the training set and the testing set. The quantity of training data should be higher than the other two data. Also, it should be unbiased to any class or category, so that model can adequately learn from the data. Splitting is done in such a manner that it ensures that the distribution of data across different categories or classes in both the training and testing sets closely follows the distribution in the original dataset. By maintaining this consistency, it reduces the chances of the trained models making unreliable or inaccurate predictions.

There are several methods available to split the data but we will be using one of the most common splitting methods which is the “Train-Test Split" or "Holdout Validation."

4.7.1 Train-Test Split

A train test split refers to the process of dividing data into two distinct sets, namely a training set and a testing set. The training set is utilized to train the model, while the testing set is employed to evaluate the model's performance. This approach enables the model to be trained on the training set and subsequently tested for accuracy on the unseen testing set. There are several methods of conducting such type of split, but the most common method and followed practice is to have one third of the data for training and the rest for testing which ensures that both sets are representative of the entire dataset and provides a reliable means of measuring the accuracy of the models.

For the purpose of this project, we selected test_size=0.3 which means that 70% of the data is used for training and 30% will be used for testing. This is achieved by executing the code as shown below:



```
In [25]: X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=0)
```

4.8 Data Visualization

In this section, we will see some of the visualizations of the input dataset (GTD). It is implemented using Python libraries and tools which were executed in Jupyter Notebook. We will be only analysing basic visualizations in this section as we know that the aim of the project is geospatial analysis of explosive data which includes analysing different kinds of maps using spatial data, which will be explained in detail in the coming chapters.

4.8.1 Bar graph 1:

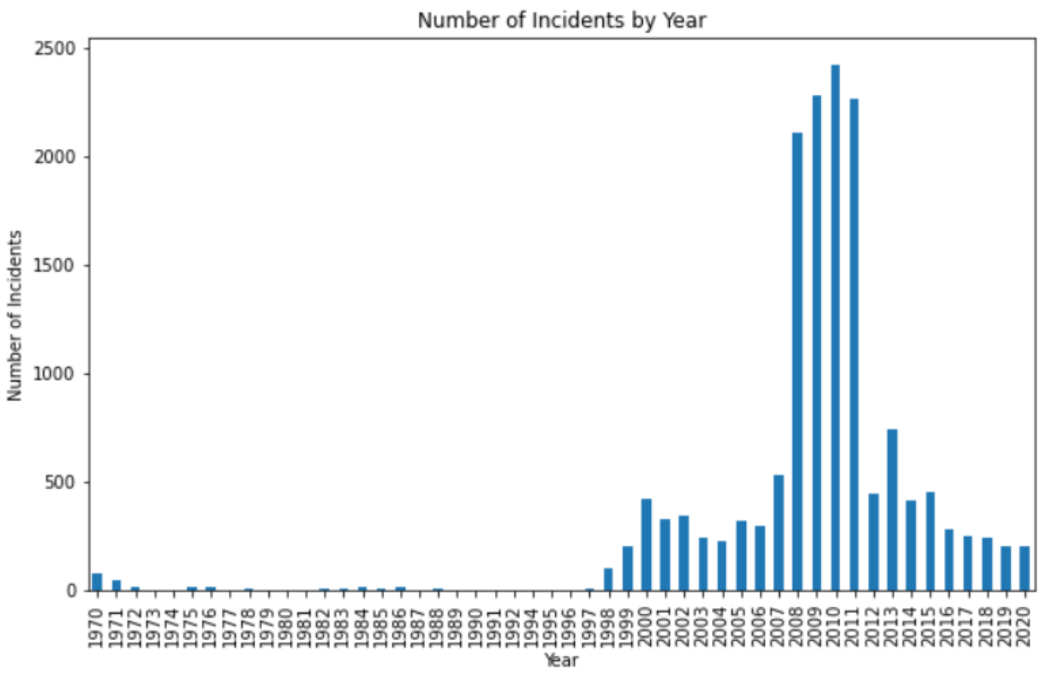


Figure 16: Number of incidents per year

Inference: From the above graph we can see the distribution of explosive attacks/incidents over the years (1970-2020). We can see the trend over time which is that the although the number of attacks were a bare minimum in 90s but there has been a significant increase in cases since then.

4.8.2 Bar graph 2:

```
In [6]: deaths_by_year = data.groupby('iyear')['nkill'].sum()
        plt.figure(figsize=(10, 6))
        deaths_by_year.plot(kind='bar')
        plt.title('Number of People Killed by Year')
        plt.xlabel('Year')
        plt.ylabel('Number of People Killed')
        plt.show()
```

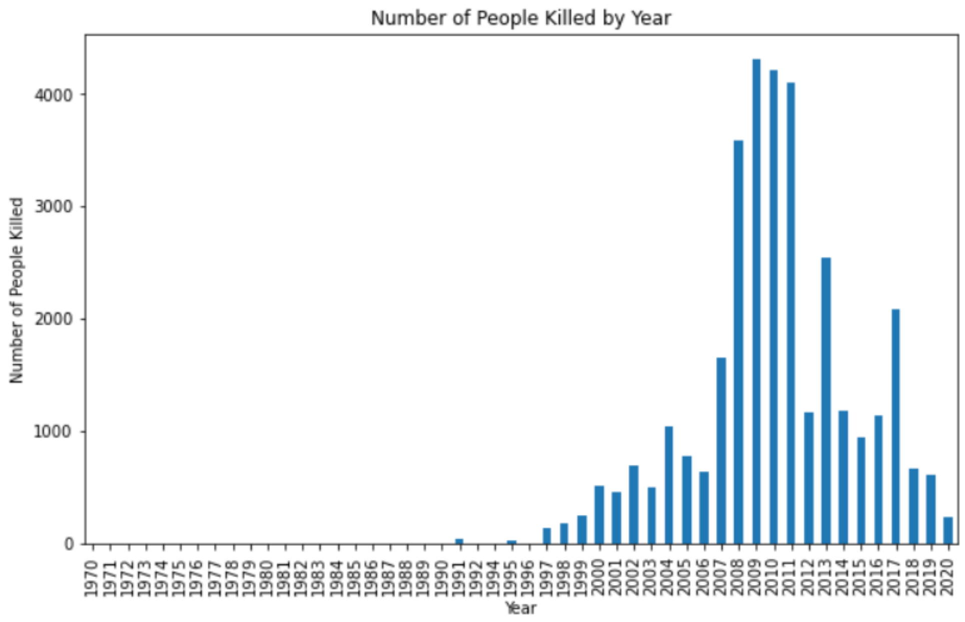


Figure 17: Number of people killed by year

Inference: From the above graph we can see the annual trend in the number of people killed as a result of explosive incidents (1970-2020). We can see that from the year 1997 there has been a significant increase in deaths. The years 2008-2010 recorded the highest fatalities and organisations such as NATO can gain insights and take necessary actions for the future.

4.8.3 Pie Chart:

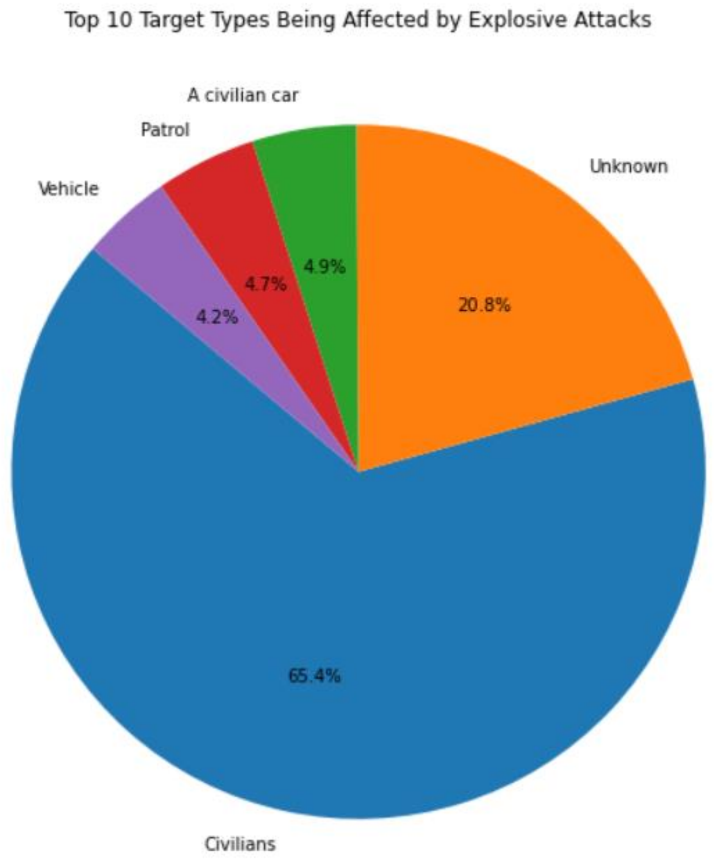


Figure 18: Top 10 target types affected by explosive attacks

Inference: From the above pie chart we can see the distribution of explosive attacks across various target types and their associated percentages. A significant portion of explosive attacks is directed at civilian targets. (65.4%). NATO can use this information to prioritize its efforts and allocate resources strategically.

4.8.4 Horizontal Bar Chart:

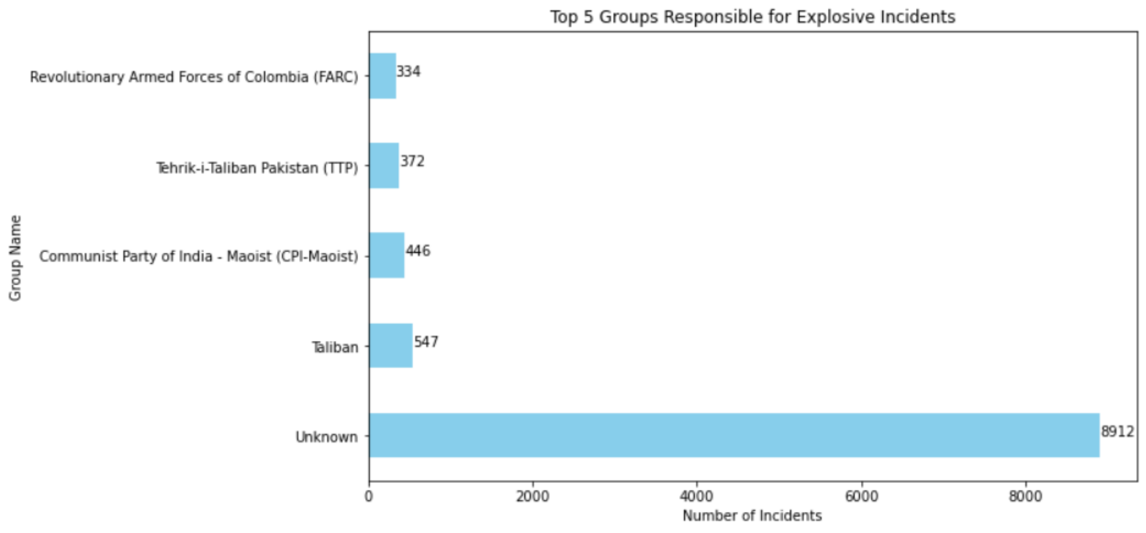


Figure 19: Top 5 groups responsible for explosive attacks

Inference: From the above bar chart we can see the top 5 groups responsible for explosive attacks over the entire time period (1970-2020). As we can see that mostly we do not know the people responsible for the attacks and this can be clearly seen in the plot as 8912 attacks are labelled as 'unknown' making it the majority. However, Taliban are in the second place with significant number of attacks executed around the world.

4.8.5 Treemap:

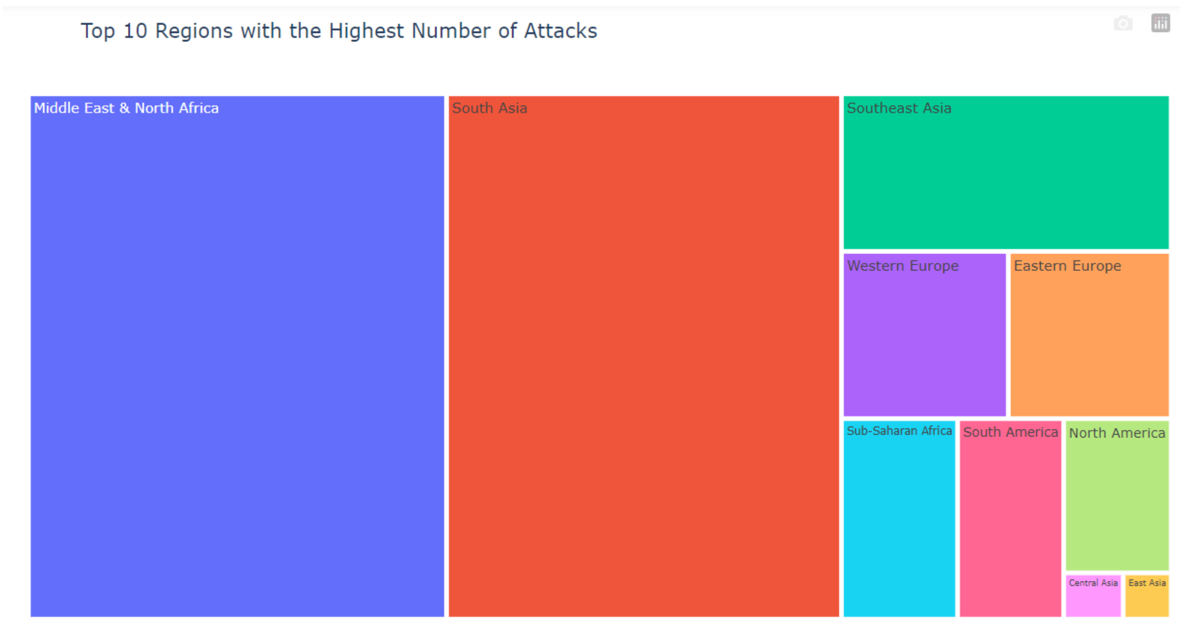


Figure 20: Top 10 regions with highest number of explosive attacks

Inference: From the treemap we can see the top 10 regions affected by explosive attacks over the years (1970-2020). Middle East & East Africa are the most affected closely followed by the South Asian Region

Chapter 5

5.0 Requirements and Design

5.1 Introduction

In this chapter, we will discuss the machine learning algorithms used along with the prerequisites. We will outline the methodology, algorithms and techniques which were utilized to carry out geospatial analysis of explosive data ensuring that our approach aligns with the project's objectives. An in-depth explanation of each technique used will be given in this section

Requirements

1. Perform geospatial analysis on explosive incident data using python libraries and tools such as geopandas using the Global Terrorism Database (GTD) as the primary dataset.
2. Identify high-risk areas, analyse attack patterns, and create predictive models to classify regions based on attack severity, with the aim of aiding NATO in strategic planning and resource allocation for enhanced preparedness

5.2 Design

5.2.1 Machine Learning Algorithm

Machine learning (ML) is a subfield of artificial intelligence (AI) that facilitates computers to acquire knowledge and improve their performance over time without explicit programming. ML algorithms have the capability to recognize patterns in data and learn from them in order to make predictions. In essence, machine learning algorithms and models acquire knowledge through experience.

In traditional programming, a computer engineer writes a set of instructions that guide a computer on how to transform input data into a desired output. These instructions are typically based on an IF-THEN structure, where specific actions are executed when certain conditions are met.

On the other hand, machine learning is an automated process that allows machines to solve problems with minimal or no human intervention and make decisions based on past observations.

Although artificial intelligence and machine learning are often used interchangeably, they are distinct concepts. AI encompasses a broader scope, involving machines making decisions, acquiring new skills, and solving problems in a manner similar to humans. Machine learning, on the other hand, is a subset of AI that enables intelligent systems to autonomously learn new information from data.

Instead of explicitly programming machine learning algorithms to perform tasks, one can provide them with labeled examples of data (known as training data). This enables the algorithms to automatically calculate, process, and identify patterns.

In simple terms, the Chief Decision Scientist at Google characterizes machine learning as an advanced labeling mechanism. By instructing machines to label objects such as apples and pears through the use of fruit examples, they will eventually be able to label these objects without any assistance, provided they have been trained with appropriate and accurate examples.

Machine learning has the capability to handle vast amounts of data and can achieve higher levels of accuracy compared to humans. It can aid in saving time and money by performing tasks and analyses such as resolving customer issues to enhance customer satisfaction, automating support ticket processes, and extracting data from internal and external sources through data mining.

Machine learning can be classified into three fundamental types based on the learning method employed, namely supervised learning, reinforcement learning, and unsupervised learning. For this particular project, we will be utilizing supervised learning.

Supervised Learning:

Supervised learning algorithms and models make predictions based on labeled training data. Each training sample consists of an input and an expected output. A supervised learning algorithm examines this sample data and makes an inference, essentially an educated estimation, when determining the labels for unseen data.

This approach to machine learning is the most prevalent and widely used. It is referred to as "supervised" because these models require manually tagged sample data to learn from. The data is labeled to inform the machine about the patterns (such as similar words and images, data categories, etc.) it should be searching for and establishing connections with.

5.2.1.1 Principal Component Analysis or PCA

When dealing with high-dimensional data, machine learning models frequently exhibit signs of overfitting, resulting in a diminished capacity to generalize beyond the examples in the training set. Therefore, it is crucial to employ dimensionality reduction techniques prior to model creation. One widely used unsupervised learning technique for reducing data dimensionality is Principal Component Analysis (PCA). PCA enhances interpretability while simultaneously minimizing information loss. It aids in identifying the most influential features within a dataset and facilitates the visualization of data in 2D and 3D. PCA assists in identifying a series of linear combinations of variables.

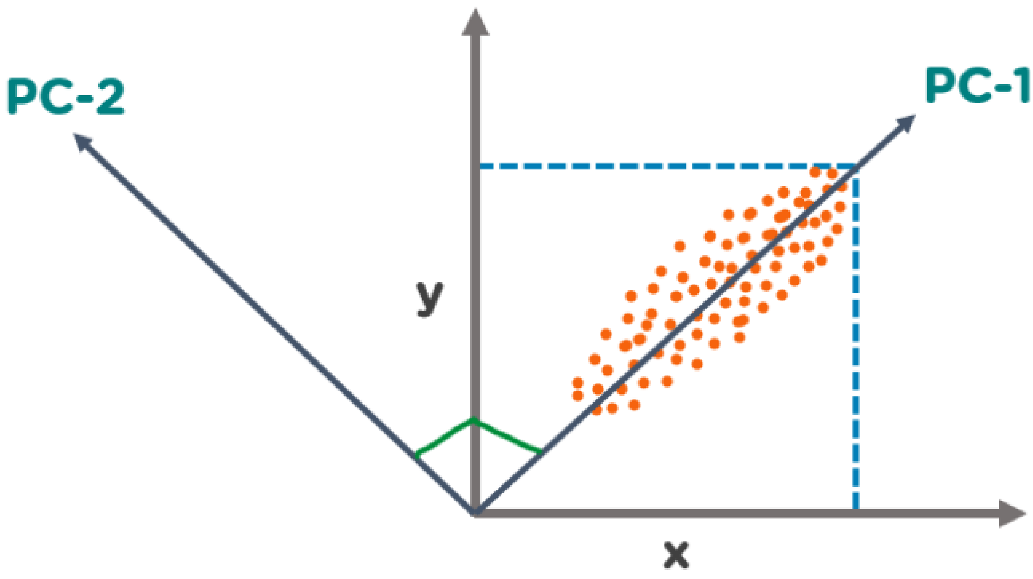


Figure 21: PCA

In the above figure, multiple points have been plotted on a two-dimensional plane. Within this context, two principal components can be identified. PC1 represents the primary principal component, responsible for elucidating the highest degree of variance within the data. Conversely, PC2 denotes an additional principal component that is orthogonal to PC1. The Principal Components, being a straight line, effectively capture a significant portion of the data's variance, possessing both direction and magnitude. It is important to note that principal components serve as orthogonal projections, meaning they are perpendicular, projecting the data onto a lower-dimensional space.

5.2.1.2 Random forest Classification

Random Forest Classification is a highly potent and extensively utilized machine learning technique specifically designed for classification tasks. At its core, it functions as an ensemble learning method, leveraging the predictive capabilities of multiple decision trees to yield resilient and precise classification outcomes.

The distinctive characteristic of Random Forest lies in its ensemble of decision trees. Each tree is constructed independently, and their predictions are amalgamated to form the ultimate classification decision. This ensemble approach significantly enhances the performance and generalizability of the model, rendering it particularly effective in addressing intricate classification challenges.

Random Forest introduces diversity within the ensemble through bootstrapped sampling. Prior to the creation of each decision tree, the algorithm randomly selects a subset of the training data. Consequently, each tree is trained on a slightly different dataset, thereby introducing variability and mitigating the risk of overfitting.

Moreover, Random Forest employs randomized feature selection. At each decision node in a tree, the algorithm considers only a random subset of the available features. This randomization of features ensures that individual trees do not excessively rely on specific attributes, thereby augmenting the robustness of the model and reducing sensitivity to noise.

The construction of each decision tree adheres to established principles of decision tree algorithms, utilizing criteria such as Gini impurity for classification. The ensemble of trees collaborates in making predictions. In classification tasks, the prediction of each tree is tallied, and the class with the majority vote is designated as the final prediction. In regression, the predictions of individual trees are averaged to derive the ultimate result.

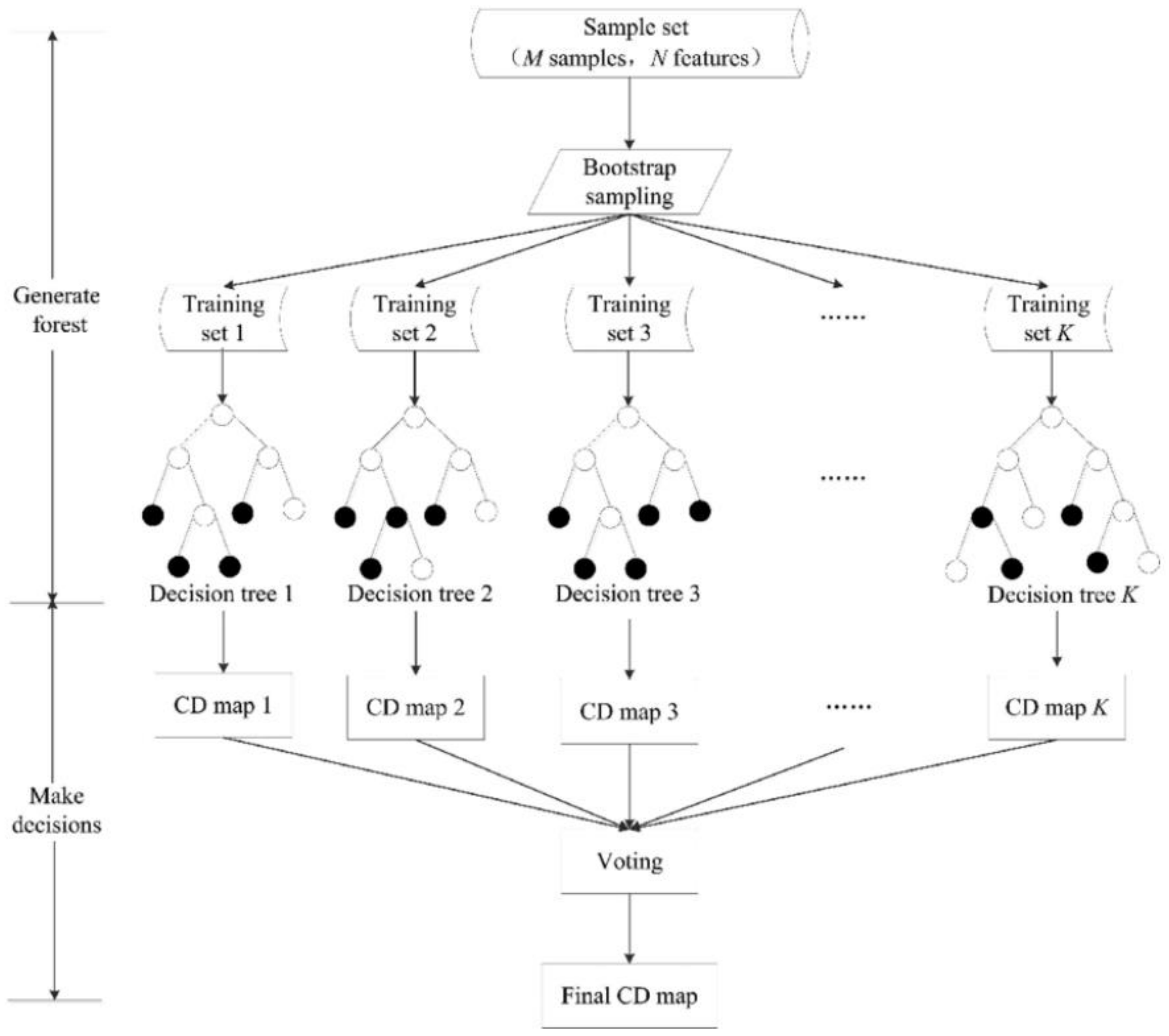


Figure 22: Random Forest Classification

In conclusion, Random Forest Classification combines the advantages of ensemble learning, bootstrapped sampling, and feature randomness to establish a robust and accurate classifier. Its capability to manage complex data relationships and mitigate overfitting makes it an essential tool for diverse classification tasks in the field of machine learning.

5.2.1.6 Python Libraries used for Geospatial Analysis

We used several built-in libraries available in python to analyse the spatial data for analysis such as folium, geopandas and plotly. Each of the libraries are explained in detail in this section:

Folium: it is a python library which enable users to create interactive maps. It helps in visualising geospatial data like the distribution of explosive incidents. Folium's integration with

Leaflet.js allows for the easy generation of maps with various markers, overlays, and plugins.in our project we have used the folium library to create interactive maps displaying incident locations, severity levels, and hotspots which are very vital for our study and also other machine learning projects. It is very intuitive to use and offers a high level of interactivity.

Geopandas: It is a library which is built on top of Pandas which adds geospatial capabilities. As stated by geopandas.org, the goal of GeoPandas is to make working with geospatial data in python easier. It combines the capabilities of pandas and shapely, providing geospatial operations in pandas and a high-level interface to multiple geometries to shapely. GeoPandas enables you to easily do operations in python that would otherwise require a spatial database such as PostGIS.It enables the integration of geographic boundaries such as country with the incident data. Hence it is a very powerful tool for geospatial analysis.

GeoPandas

Figure 23: Logo of Geopandas

Matplotlib and Plotly: According to matplotlib.org, Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible. Some of its key features are as follows:

- Create publication quality plots.
- Make interactive figures that can zoom, pan, update.
- Customize visual style and layout.

Matplotlib can be described as a versatile data visualisation library whereas plotly provides interactive and web-based visualisation capabilities, both of which are essential for creating plots and graphs to visualize the geospatial data and gain meaningful insights from its analysis. We have already seen the usage of matplotlib in the basic visualisations such as the bar graphs described in the previous chapter whereas we will see the use of plotly in the next chapter for the creation of choropleth maps and time series variation maps. Both libraries helps us visualize and understand the trends and pattern in data.

In summary, these libraries provided tools for spatial data analysis and interactive map creation which contributed to the overall effectiveness of the project aim for better and informed decision making and strategic planning for the future.

5.3 Summary

To summarize, we took a broader view on what machine learning is and its relevance and importance. Also, the machine learning algorithms along with the pre-requisites were explained in detail. Finally, the python libraries which were used to analyse the spatial data were also looked into and their importance in this project was discussed.

Chapter 6

6.0 Geospatial Analysis

6.1 Introduction

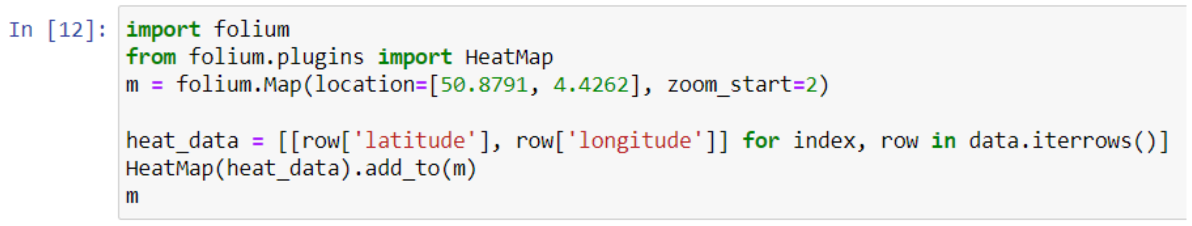
For geospatial analysis, we use location-based data to gain valuable insights to analyse explosive incidents. With the help of specialized python libraries such as Folium, Geopandas and Plotly we will be able to transform the input data into informative visualisations which will help us pinpoint high risk areas and trends of explosive incidents. The different maps used are described in detail in the following sections.

6.2 Implementation of different maps using python libraries

6.2.1 Heatmap using Folium

Using the folium library, we generate a heatmap showing us the attack density or concentrations over the different regions in a world map. In a heatmap, values in the dataset or matrix are represented as colours. Higher values or intensity are represented by warmer colours for e.g., red and smaller values are represented by coolers colours like blue.

We use the following code in Jupyter Notebook to generate a heatmap:



```
In [12]: import folium
         from folium.plugins import HeatMap
         m = folium.Map(location=[50.8791, 4.4262], zoom_start=2)

         heat_data = [[row['latitude'], row['longitude']] for index, row in data.iterrows()]
         HeatMap(heat_data).add_to(m)
         m
```

We use the folium library in which we import Heatmap. In such a map we can specify a centre for point of reference which acts as the point of focus. Since out client is NATO, we specify the location as the headquarters of NATO which is Brussels. Therefore, we input the values of latitude and longitude accordingly as shown below:

location=[50.8791, 4.4262]

Output:

Out[12]:

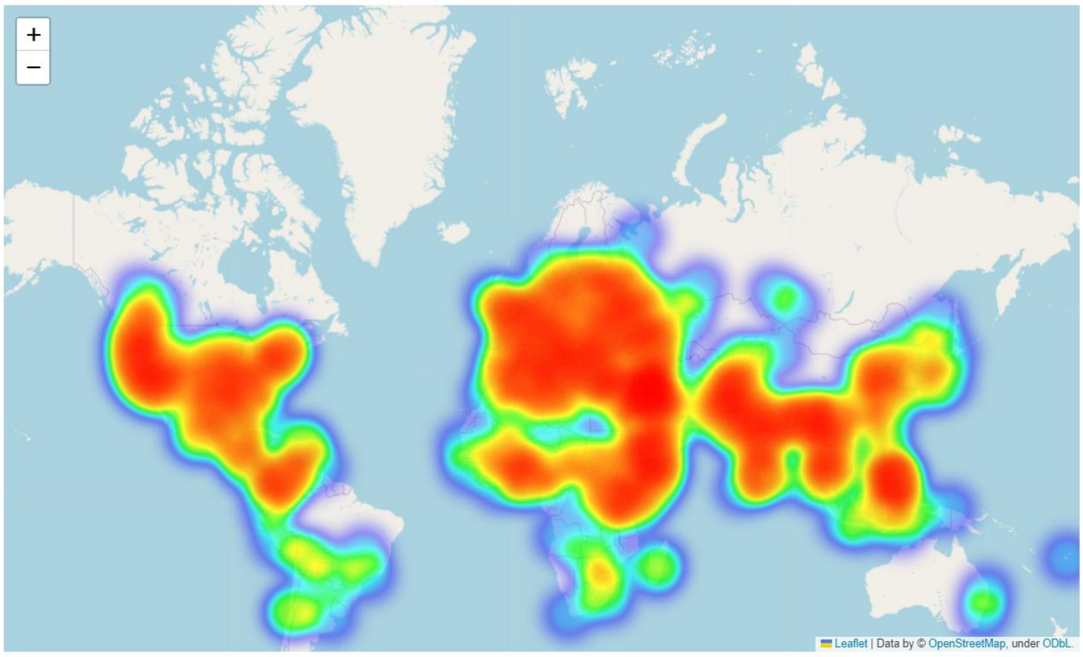


Figure 24: Heatmap of attack locations

The above is the output of the executed code for the folium map. It displays a world map showing the regions with hotspots of explosive attacks over the years. The warmer the colour i.e. red indicates highly affected areas whereas blue indicates least affected. From the map it is clearly visible that in the time period (1970-2020), most of Europe, North America and parts of Asia had the larger concentration of explosive attacks.

The output map is an interactive map i.e., one can zoom in and out to observe more specifically the intense regions of a particular region.

For e.g. if we want to focus on the Europe region:

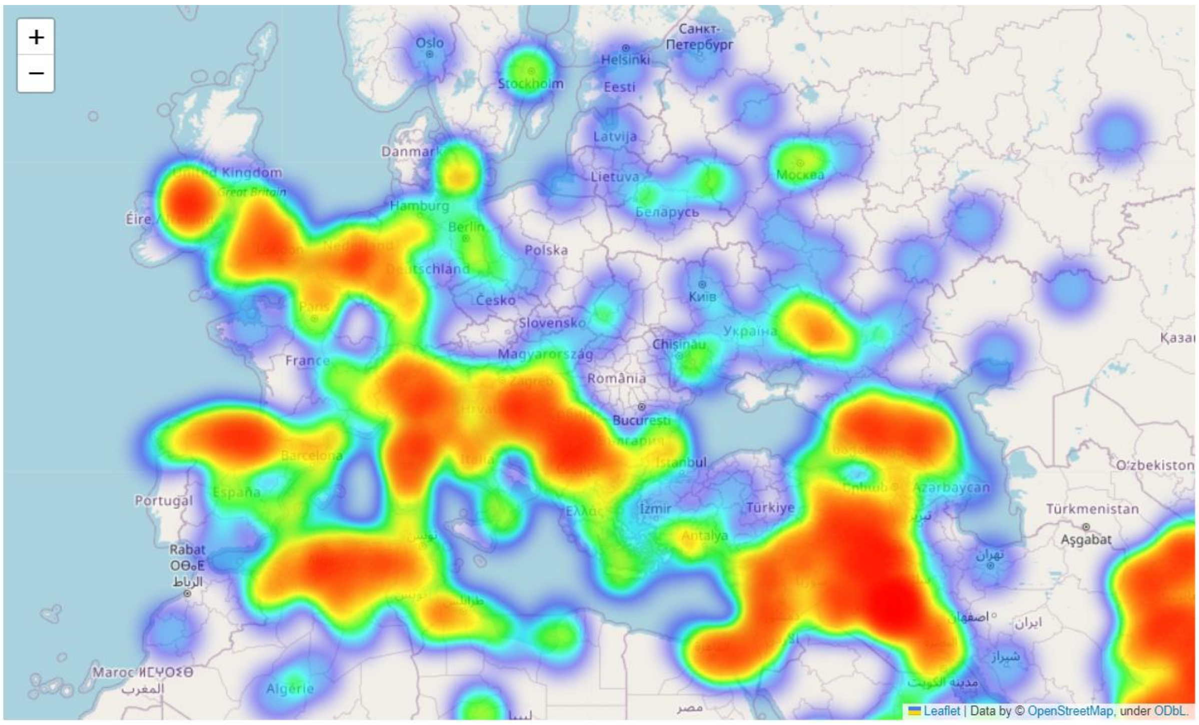


Figure 25: A more zoomed in heatmap

If we only want to focus just one country, we can zoom in accordingly like in the case for United Kingdom:

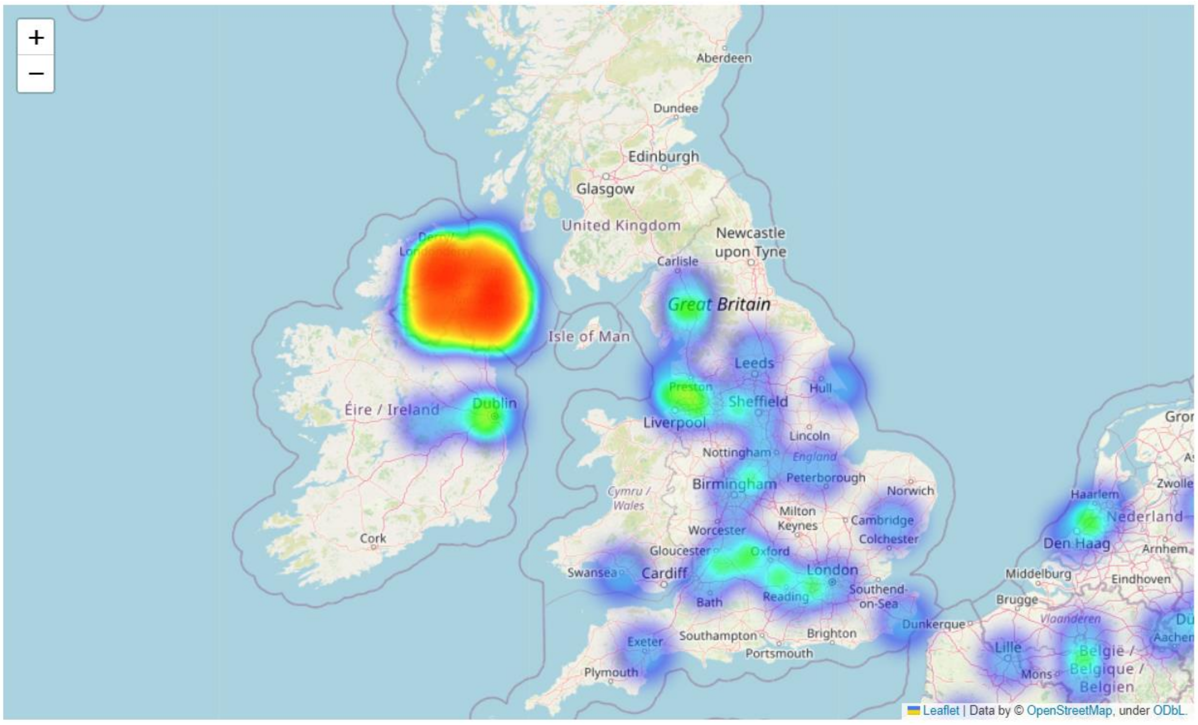


Figure 26: Further applying zoom

We can also focus one city like London and view and analyse the exact area or location of attack as shown below:

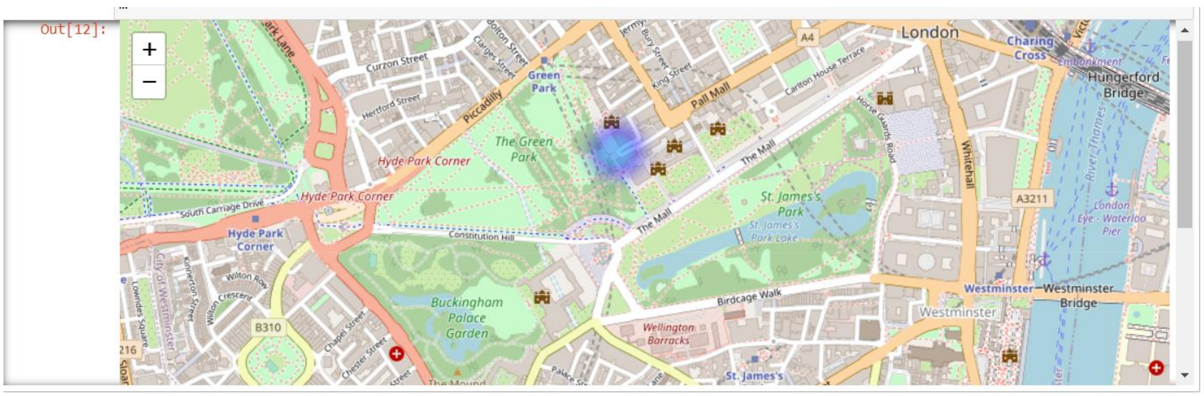


Figure 27: More application of zoom

The blue spot indicates the area of attack which can be further zoomed in to its granular level.Areas with higher density of incidents can be seen as hotspots on the map. NATO can therefore identify such regions with high frequency of attacks. By analysing where the concentration of attacks, they can identify the regions which require heightened security measures and resource allocation. They can take steps such as deploying more forces or coordinating with the local government to make necessary decisions.

The heatmaps can also act as communication tools too. NATO can use them to raise public awareness about the global nature of terrorism and the challenges it faces. Publicly sharing such maps will help in fostering international collaboration in addressing threats in the future. Heatmaps therefore are a vital tool in aiding geospatial analysis for explosive data which in turn helps NATO to take necessary actions and help them in strategic planning for the future.

6.2.2 Severity map using Folium:

Using the folium library, we can visualize the severity of explosive incidents at different geographic locations. In previous chapters, we have seen the creation of the severity factor by using the following mathematical formula in the python code:

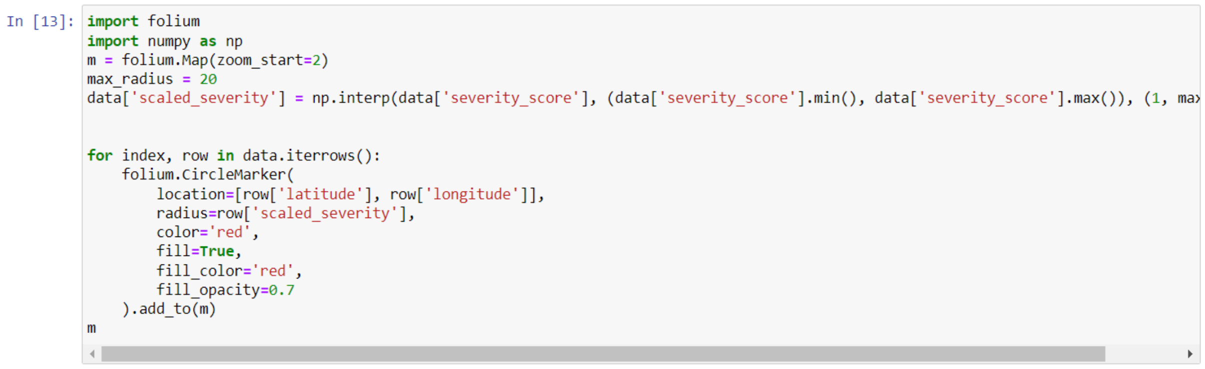
weight_fatalities = 1

weight_injuries = 0.5

data['severity_score'] = (weight_fatalities * data['nkill']) + (weight_injuries * data['nwound'])

Therefore, each geographical location which was under attack in the past years has a severity value which is a numerical value which signifies the intensity of the attack (no of people killed +no of wounded). Higher the value, higher is the severity.

The python code for the map is executed in Jupyter notebook:



```
In [13]: import folium
         import numpy as np
         m = folium.Map(zoom_start=2)
         max_radius = 20
         data['scaled_severity'] = np.interp(data['severity_score'], (data['severity_score'].min(), data['severity_score'].max()), (1, max


         for index, row in data.iterrows():
             folium.CircleMarker(
                 location=[row['latitude'], row['longitude']],
                 radius=row['scaled_severity'],
                 color='red',
                 fill=True,
                 fill_color='red',
                 fill_opacity=0.7
             ).add_to(m)
         m
```

In the above map we set the radius=20, which is the size of the largest circle that will appear on the map. We specify the zoom value as 2 as indicated by:

m = folium.Map(zoom_start=2)

which determines the closeness of the initial map which will be first displayed. The value 2 gives us a broader view of the entire map. Also, when we do not scale the original values, we obtained a map with huge sized data points. Therefore in order to avoid such a problem, using the np.interp we scale the original values to a consistent range.

Output:

Out[14]:

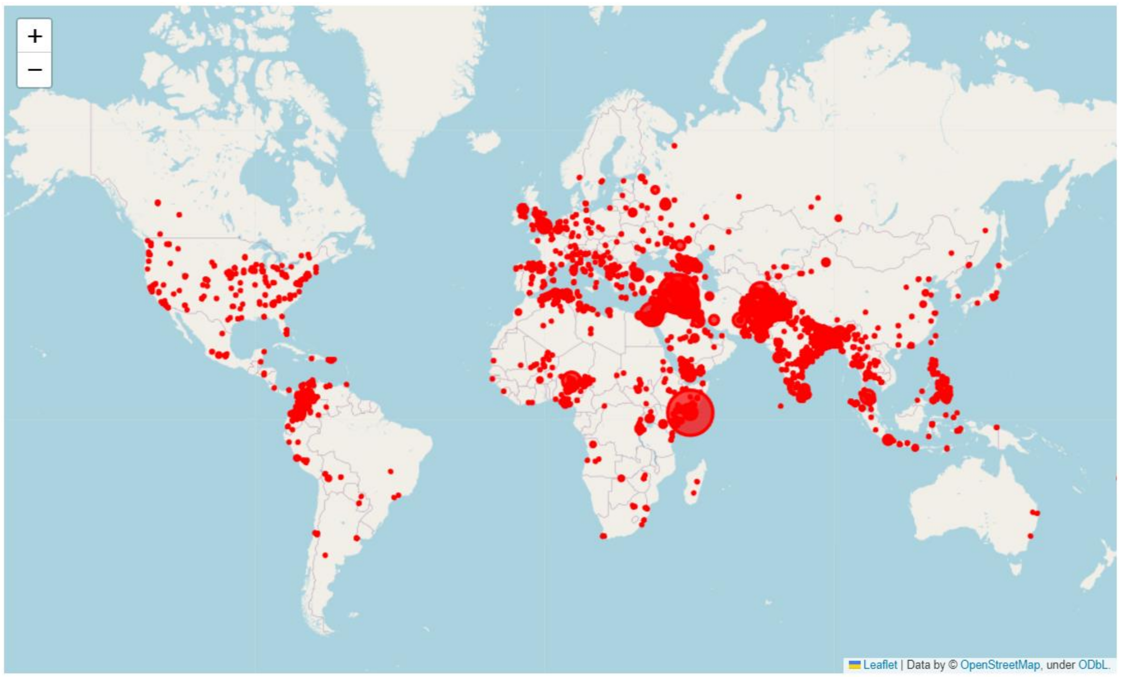


Figure 28: Severity map using folium

The above is the first output we obtain after executing the code for the map.As we can see, the bigger red markers or circles are the regions which have higher severity ie more no of deaths and people wounded as and when compared to those of the smaller circles. In the previous map, we could see the areas which has more no of attacks but now we also have the region wise segregation based on the severity factor.

If we want to turn to our attention to any specific continent like in the case of Africa which is shown below:

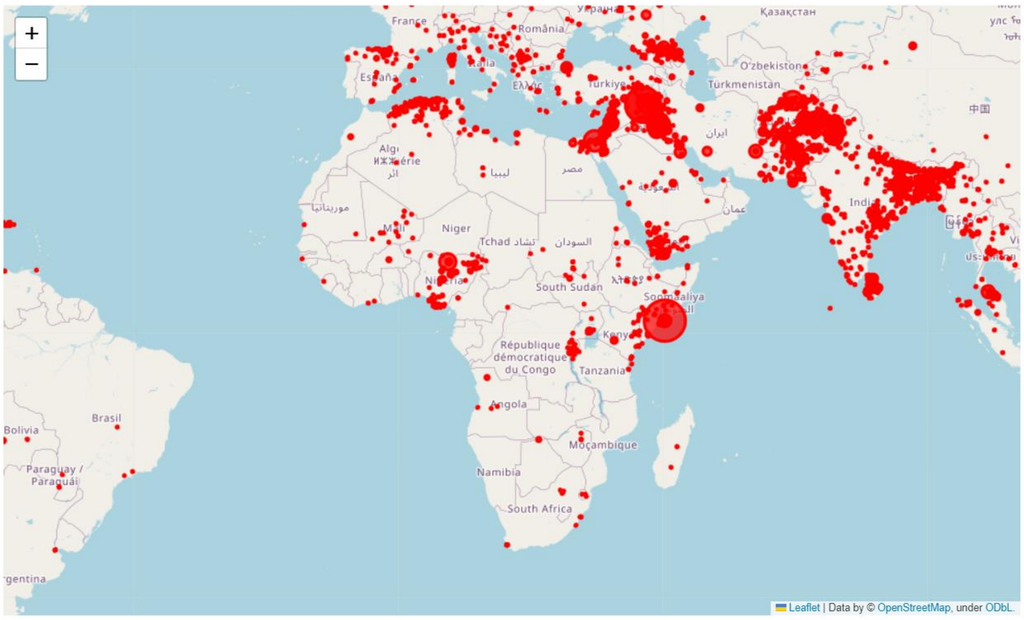


Figure 29: A more zoomed in map focussing on Africa

We can clearly see that over the years, the eastern part of Africa was the worst affected in terms of the causalities.

Focussing more on that area and following the big red marker, we can gain further insights:

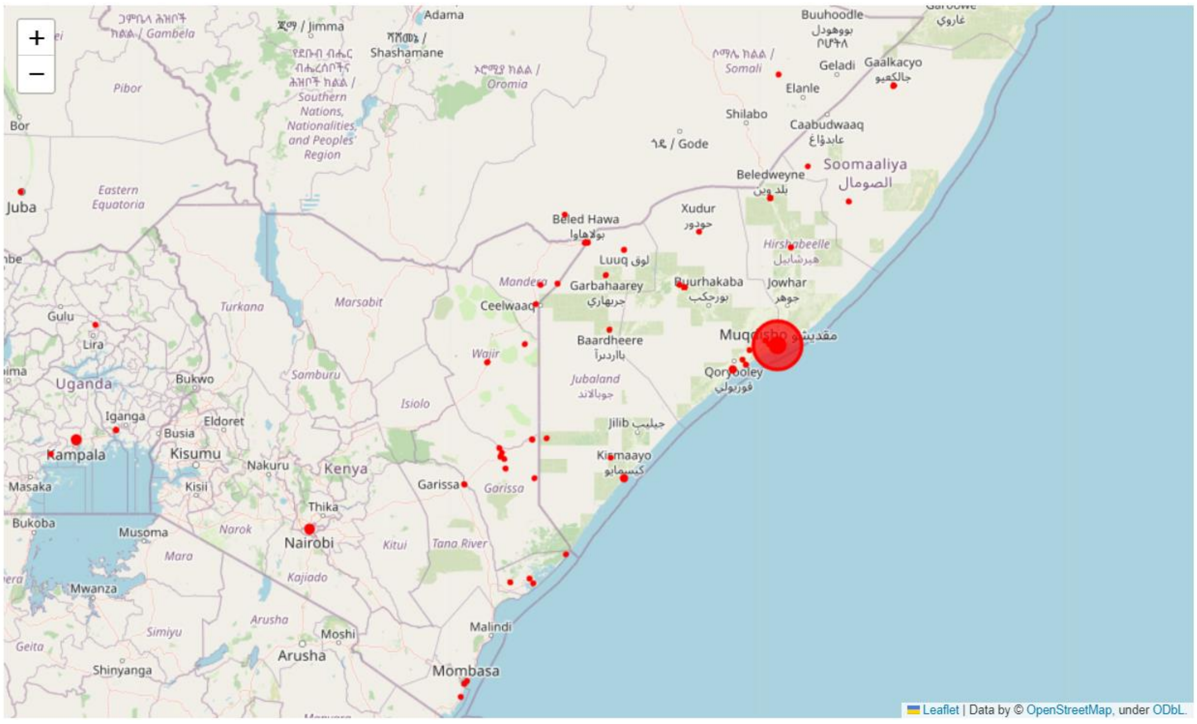


Figure 30: Severity of Somalia Region

The city of Muqdisho which is the capital city of Somalia was one of the deadliest in terms of those affected by explosive attacks. This is one of the many example cases that can be derived from the map which helps NATO in their effort to contribute to world security.

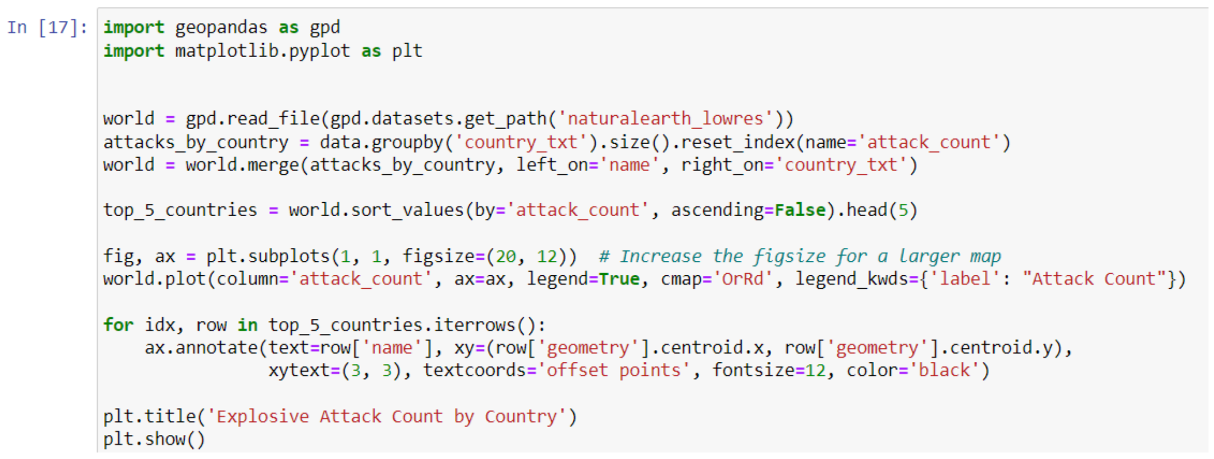
Such type of information can be used by NATO and other organisations for strategic planning and response. One of the methods is that they can allocate additional resources or specialised units to the highly affected areas. This target allocation ensures that NATO's efforts are focussed on regions with the greater need for security enhancement.

The map also helps in trend analysis. It helps NATO to identify trends in the attack patterns like if certain regions like in the case of Somalia are experiencing high death rate among the attacks, NATO can investigate the underlying causes and adapt its strategies accordingly, thereby helping in strategic planning for the future.

6.2.3 Choropleth map using Geopandas:

Using Geospandas, we have generated a chloropleth map showing the explosive attack count per i.e., we can now visualize the most affected countries by such attacks. Each country is coloured depending on the attack count. Higher the attack count, warmer is the colour. Also, we display the top 5 affected countries due to explosive attacks in the world.

The python code was executed in the Jupyter Notebook:



```
In [17]: import geopandas as gpd
         import matplotlib.pyplot as plt


         world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))
         attacks_by_country = data.groupby('country_txt').size().reset_index(name='attack_count')
         world = world.merge(attacks_by_country, left_on='name', right_on='country_txt')

         top_5_countries = world.sort_values(by='attack_count', ascending=False).head(5)

         fig, ax = plt.subplots(1, 1, figsize=(20, 12))  # Increase the figsize for a larger map
         world.plot(column='attack_count', ax=ax, legend=True, cmap='OrRd', legend_kwds={'label': "Attack Count"})

         for idx, row in top_5_countries.iterrows():
             ax.annotate(text=row['name'], xy=(row['geometry'].centroid.x, row['geometry'].centroid.y),
                         xytext=(3, 3), textcoords='offset points', fontsize=12, color='black')

         plt.title('Explosive Attack Count by Country')
         plt.show()
```

After importing both the geopandas and matplotlib library, we load the world map using the following piece of code:

gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))

We calculate the attacks per country using the groupby() which is shown below:

attacks_by_country = data.groupby('country_txt').size().reset_index(name='attack_count')

After which we merge both the attack count data with geospatial data. We sort the data to get the top 5 affected counties and then generate the choropleth map using the following code:

fig, ax = plt.subplots(1, 1, figsize=(20, 12)) world.plot(column='attack_count', ax=ax, legend=True, cmap='OrRd', legend_kwds={'label': "Attack Count"})

The size of the map can be altered by changing the values of figsize=(20, 12). We then label the top 5 countries and display the map.

Output:

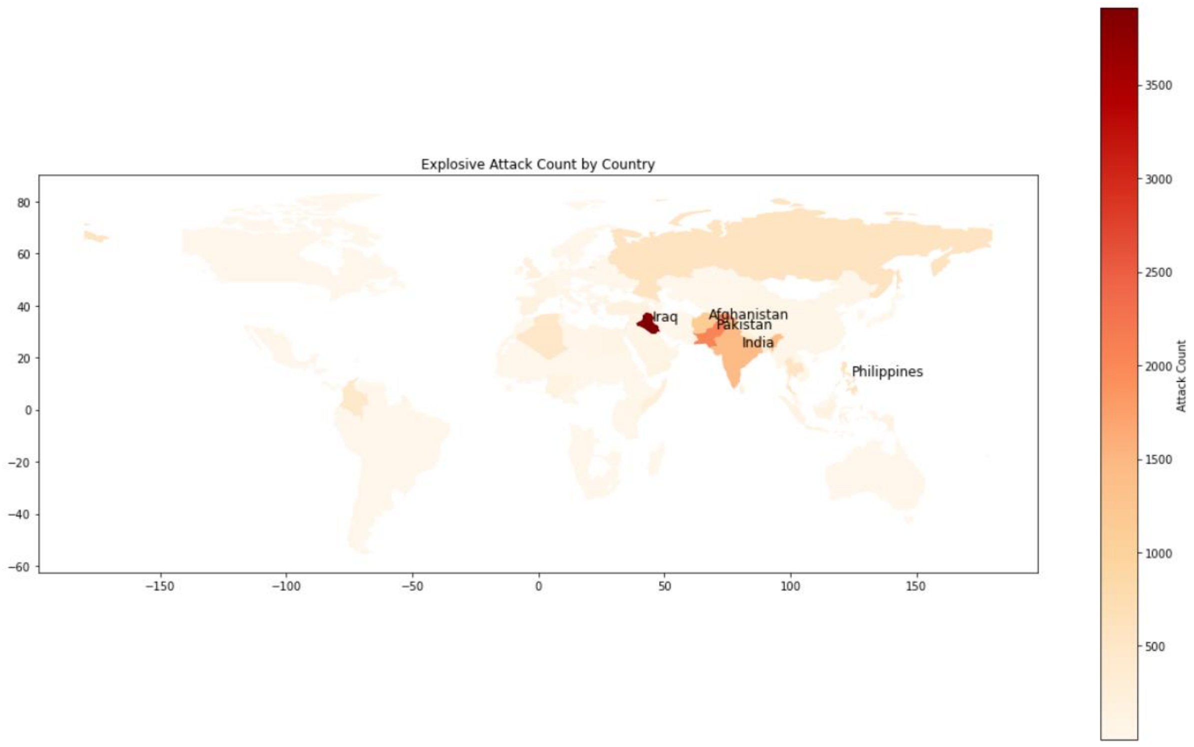


Figure 31: Choropleth map of attack locations per country

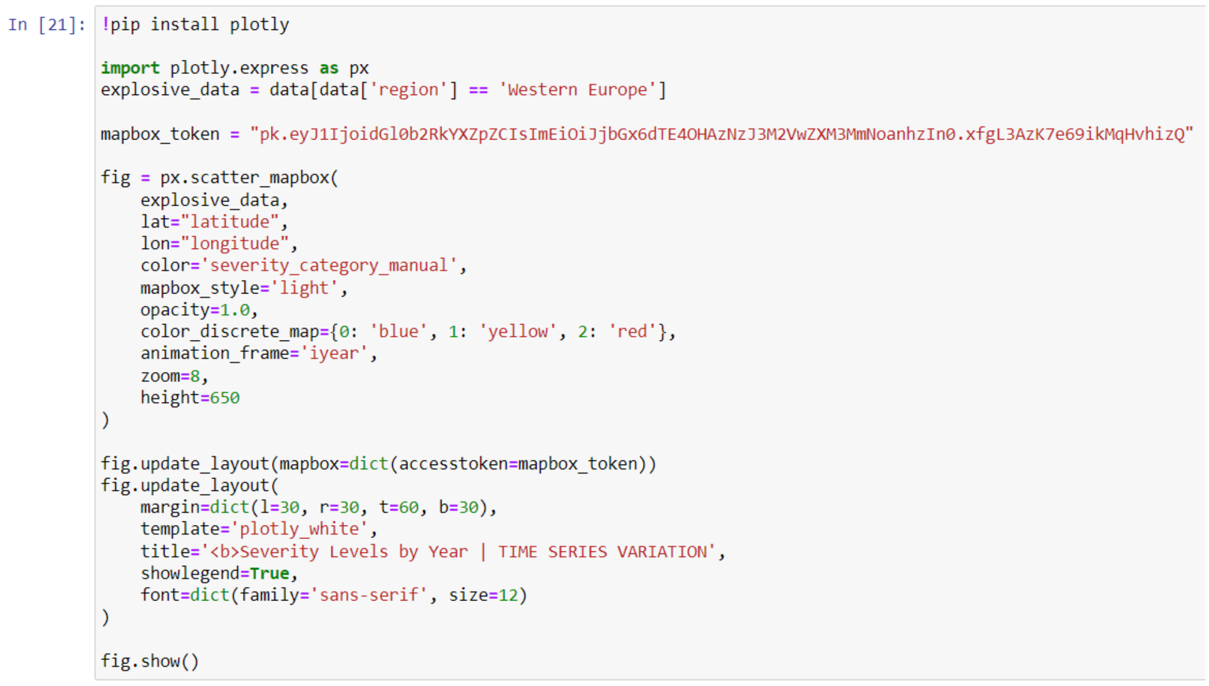
We can now see the distribution of attack counts per country over the years. Darker the colour is for a country (darker red it is) signifies more is the number of explosive attacks for that country. Of all the countries we can see that Iraq, Afghanistan, Pakistan, India and the Philippines were the most affected.

This map helps NATO to quickly identify countries with the highest counts of explosive attacks. The darker shade on the map like in the case of Iraq indicate countries with more security challenges. The map also assists in resource allocation decisions as they can prioritize such countries for the deployment of security assets and as a result aid in strategic planning.

6.2.4 Time Series Variation Map using plotly:

Using plotly we can generate a time series variation map or a scatter map which visualizes the geospatial data of explosive attacks specifically focussing on a particular region and displaying the severity levels over time.

The python code was executed in the Jupyter Notebook and is shown as follows:



```
In [21]: !pip install plotly

         import plotly.express as px
         explosive_data = data[data['region'] == 'Western Europe']

         mapbox_token = "pk.eyJ1IjoidGl0b2RkYXZpZCIsImEiOiJjbGx6dTE4OHAzNzJ3M2VwZXM3MmNoanhzIn0.xfgL3AzK7e69ikMqHvhizQ"

         fig = px.scatter_mapbox(
             explosive_data,
             lat="latitude",
             lon="longitude",
             color='severity_category_manual',
             mapbox_style='light',
             opacity=1.0,
             color_discrete_map={0: 'blue', 1: 'yellow', 2: 'red'},
             animation_frame='iyear',
             zoom=8,
             height=650
         )

         fig.update_layout(mapbox=dict(accesstoken=mapbox_token))
         fig.update_layout(
             margin=dict(l=30, r=30, t=60, b=30),
             template='plotly_white',
             title='<b>Severity Levels by Year | TIME SERIES VARIATION',
             showlegend=True,
             font=dict(family='sans-serif', size=12)
         )

         fig.show()
```

After importing the necessary libraries i.e. plotly we select a specific region for our analysis for e.g. Western Europe. This parameter can be changed to any desired location. The line of code which demonstrates this action is shown below:

explosive_data = data[data['region'] == 'Western Europe']

In order to view and use Mapbox which is a mapping and data platform which provides map visualisation services we need a unique access token from their website which can be obtained after creating a free account.

Next, we create the mapbox scatter plot, I which we only take the spatial data of the specific region. In previous chapters we have seen that we have transformed the severity values into 3 classes. For each class we select a colour to be displayed on the animated map which show their severity. The assigning of colours can be better understood with the help of a table:

Value	Colour Assigned	Description
0	Blue	Less severe
1	Yellow	Moderate
2	Red	Highly severe

Finally, we select an already existing template for the map and specify the map titles. After which we use the show () to display the map.

Output:

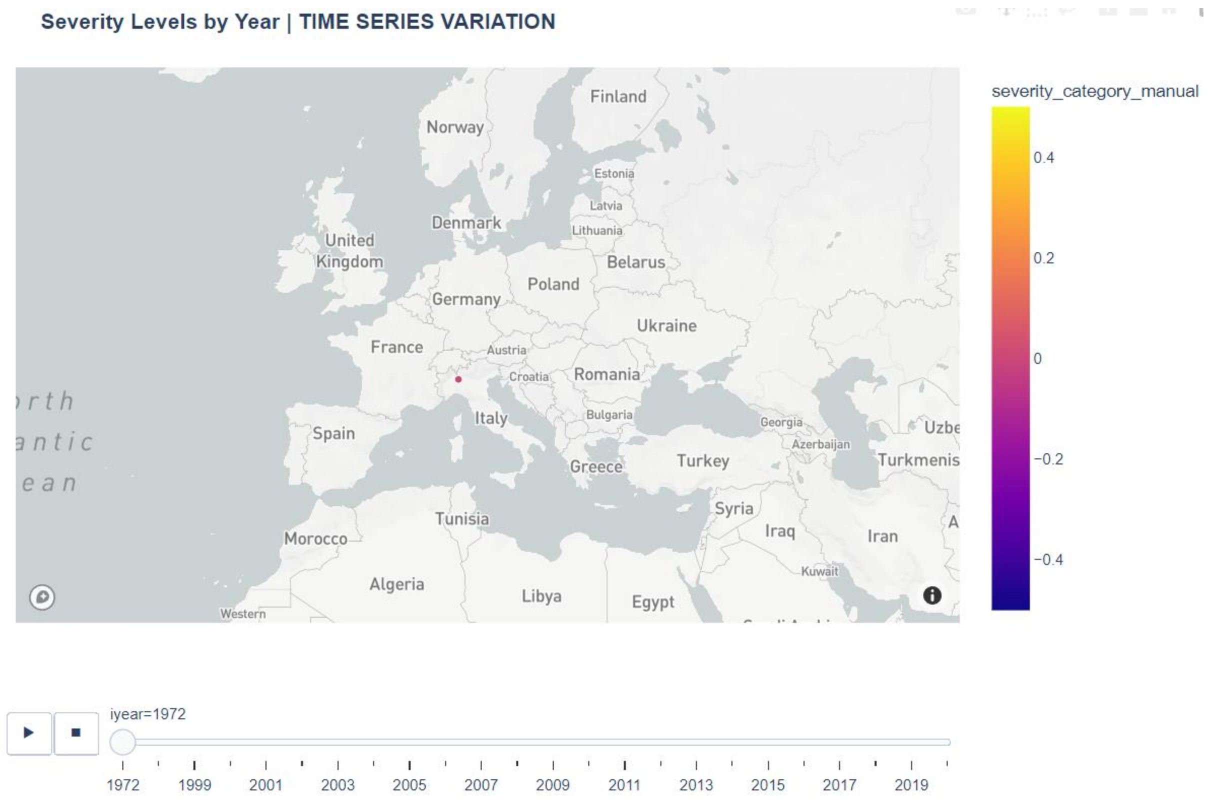


Figure 32: Time series variation map

The layout of the output map is straightforward as on the top we have the title with the map occupying the centre stage. The bottom of the map we can see a slider with all the years alongside a stop and play button. And on the right-hand side the colour code variation for the severity levels.

It must be noted that this is an interactive animated map which will help in analysing the distribution of explosive incidents and their severity levels from year to year.

Figure 32 shows us the area of attack in the year 1970 in eastern Europe which is in Italy with a red severity meaning highly severe attack (High number of causalities). Now when we click the play button, the map changes its distribution according to the year through which it passes.

In the year 1999 the distribution of attacks is seen as follows:

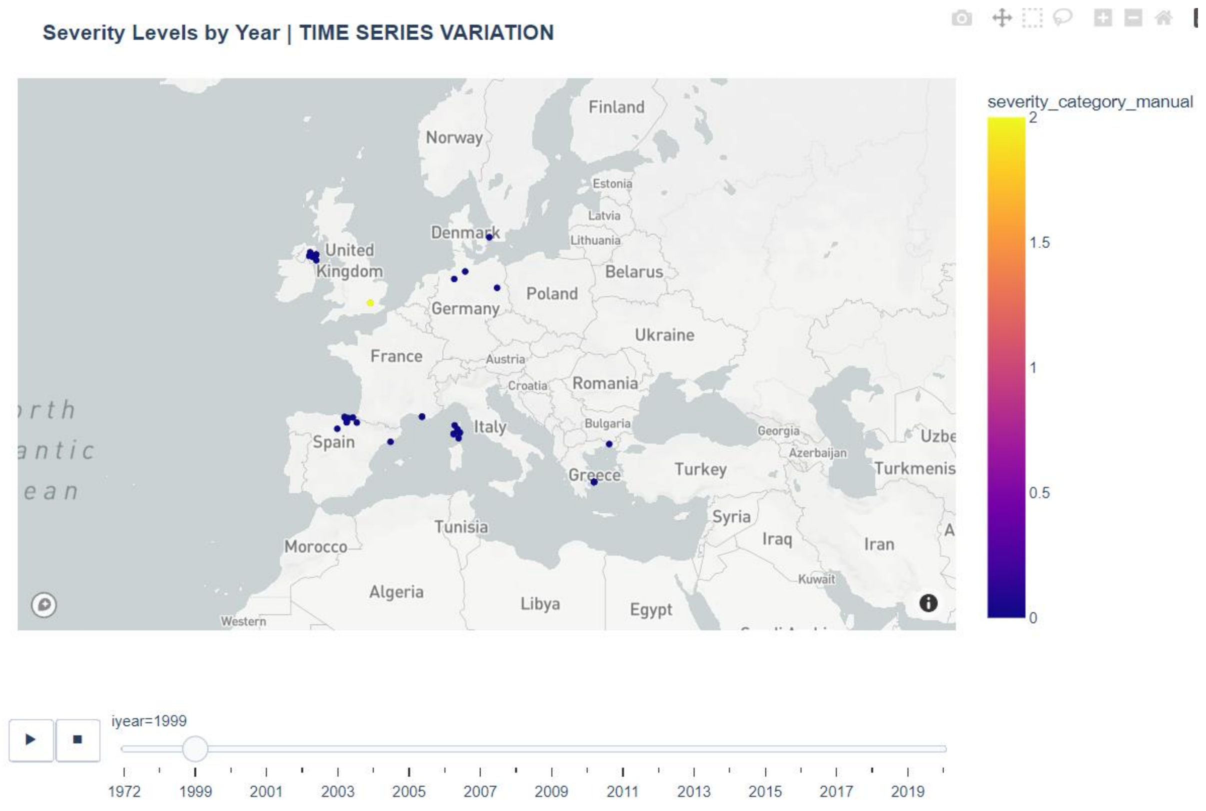


Figure 33: Map for year 1999

As we can see in the above figure, when compared to previous years the number of explosive attacks have significantly increased over time with attacks spreading across the Europe region.

Similarly in the year 2006:

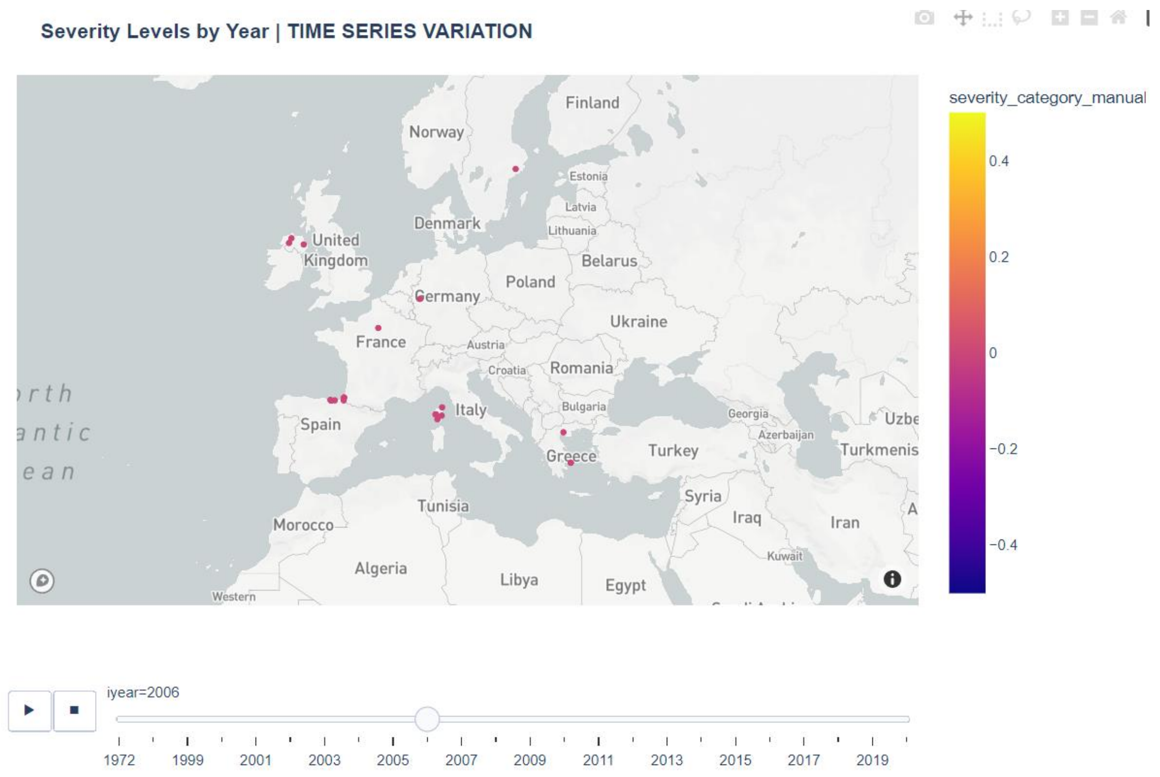


Figure 34: Map for year 2006

Similarly, when it comes to year 2006, the distribution of attacks although seems consistent and somewhat identical as in 1999, however the intensity or rather the severity has increased (meaning more deaths and wounded people).

And finally in the year 2016:

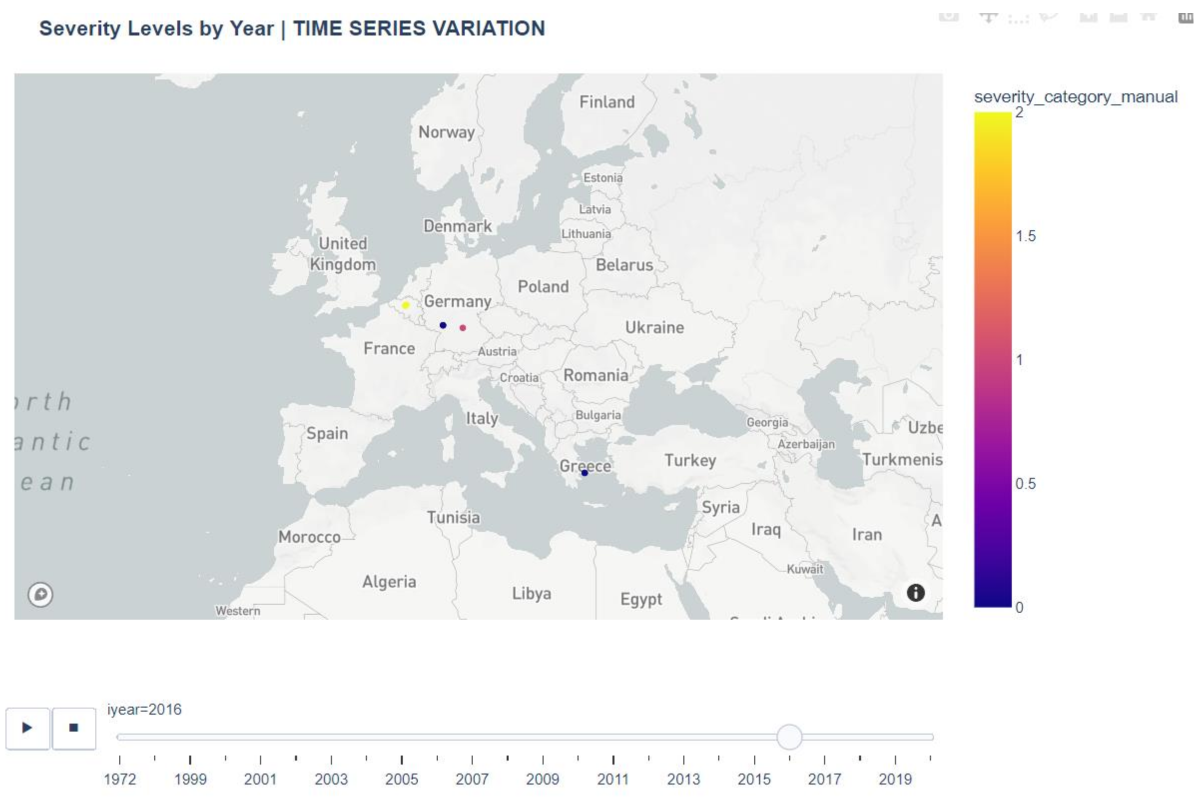


Figure 35: Map for year 2016

In the above figure we can see 3-4 regions with different levels of severity . One of the benefits of such analysis is that being an interactive and animated map, one can visualize the area of attacks with their severity level more easily.

Also, we can get further details about the data point by placing the cursor over that location which is shown in the figure below:

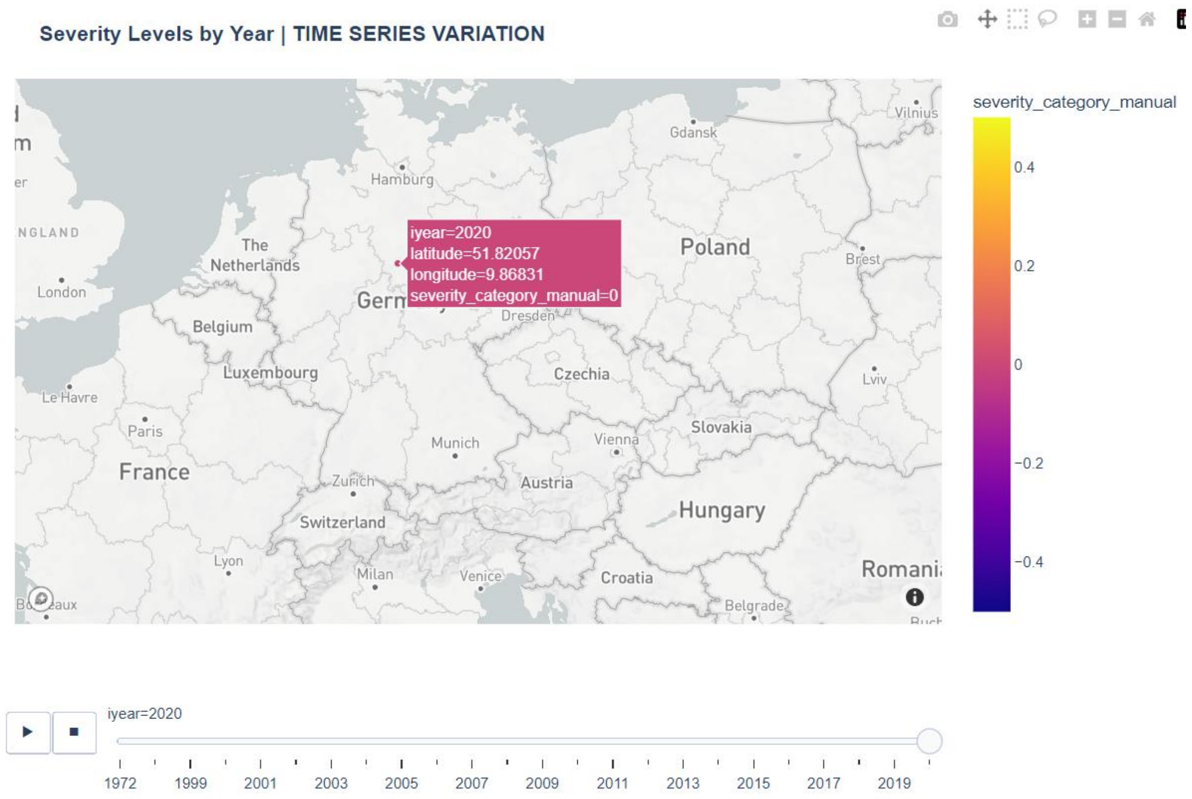


Figure 36: Point details of map

As we can see in figure 36, hovering over each point or location pops up a small window with the year, latitude, longitude, and the severity category of that spatial location. This map or visualisation helps the users or in this case NATO, interact by zooming in and out to explore the different areas of the specific region. They can also toggle between different years to analyse and study how the attacks evolved.

The animated map by year allows NATO to analyse temporal trends in explosive incidents. Answers to questions such are the incidents increasing or decreasing over time? Are certain areas becoming more potent or volatile? Can be found out using such analysis. Also, in the event of a security crisis, NATO can use this map to rapidly asses the current situation and hence identify areas that require immediate intervention. It always helps in crisis response planning and execution.

6.3 Summary

In this chapter we saw the use of python libraries and other tools for creating maps and other visualisations for geospatial analysis using spatial data and how it could be useful for NATO in their effort to make better decisions and plans for incidents which could arise in the future.

Chapter 7

7.0 Implementation

7.1 Introduction

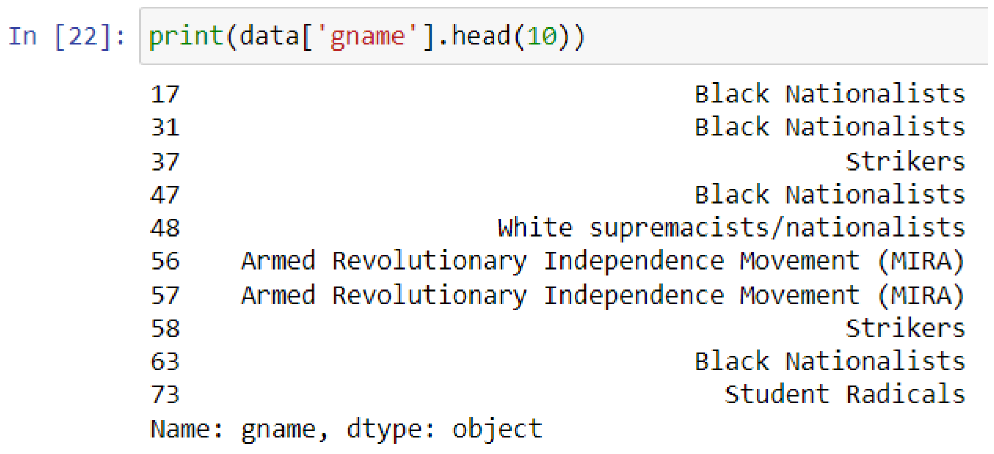
In this chapter, we will implement our novel machine learning algorithm model along with the benchmark models and other pre requisites needed for the successful implementation of the project in order to perfectly classify the severity of the attacks into the three relevant categories. The python code was run and executed in Jupyter Notebook.

7.2 Implementation of pre-requisites

7.2.1 Pre-processing before model implementation:

Before we proceed with PCA, as we have seen in the previous chapters, we have X and y which hold both the features and the target variable for performing machine leaning analysis. But we have one more pre- processing step to perform which is label encoding in which we convert the categorical data columns into numerical data which is required machine learning algorithms to process data effectively.

Original Data before conversion:



```
In [22]: print(data['gname'].head(10))

17                                     Black Nationalists
31                                     Black Nationalists
37                                               Strikers
47                                     Black Nationalists
48                          White supremacists/nationalists
56    Armed Revolutionary Independence Movement (MIRA)
57    Armed Revolutionary Independence Movement (MIRA)
58                                               Strikers
63                                     Black Nationalists
73                                       Student Radicals
Name: gname, dtype: object
```

Firstly we import the LabelEncoder from the sklearn.preprocessing module and create an instance of the same as shown below:

from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()

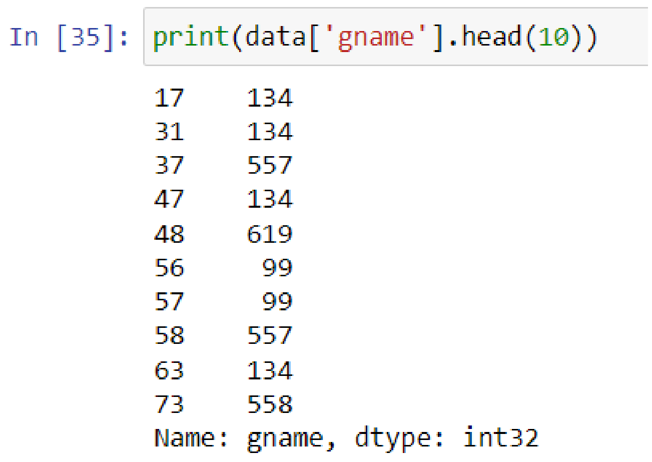
We then specify the list of columns which we need to convert and store it in a variable:

categorical_columns = ['attacktype', 'targtype', 'gname', 'natlty', 'target', 'motive', 'weapsubtype1_txt', 'weapdetail', 'country_txt', 'region', 'provstate', 'city']

Now using a for loop, we change and convert each value to their numerical configuration without creating new columns as shown below:

```
for column in categorical_columns:
    data[column] = label_encoder.fit_transform(data[column])
```

The newly transformed ‘gname’ column can be seen as follows:



```
In [35]: print(data['gname'].head(10))

17    134
31    134
37    557
47    134
48    619
56     99
57     99
58    557
63    134
73    558
Name: gname, dtype: int32
```

Next to we perform another common pre-processing step which is feature scaling by using the standard scaler from scikit-learn. For any machine learning algorithm, feature scaling is important because it ensures all feature variables have the same scale or magnitude.

```
scaler = StandardScaler()

X_scaled = scaler.fit_transform(X)
```

We create an object of the StandardScaler() having the name scaler after which we apply the fit.transform() to the set of features and finally store the scaled data to a new variable name ‘X_scaled’.From now on we will be using ‘X_scaled’ instead of ‘X’ which holds the scaled data.

7.2.2 PCA:

Using PCA or Principal Component Analysis which is a dimensionality reduction technique, we capture the most important information from the original data while reducing the number of features.

We first create an instance of the PCA class with n_components =2 which specifies that we want to reduce the dimensionality of the data by 2 components as shown below:

```
pca = PCA(n_components=2)
```

Next we apply the PCA to the scaled features that we obtained in the previous phase using the pca.fit_tranform().The modiied data is stored in a new variable (X_pca):

```
X_pca = pca.fit_transform(X_scaled)
```

We can visualize the result of PCA using a scatter plot as follows:

```
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y, cmap='viridis')

plt.xlabel('Principal Component 1')
```

plt.ylabel('Principal Component 2')

plt.show()

Output:

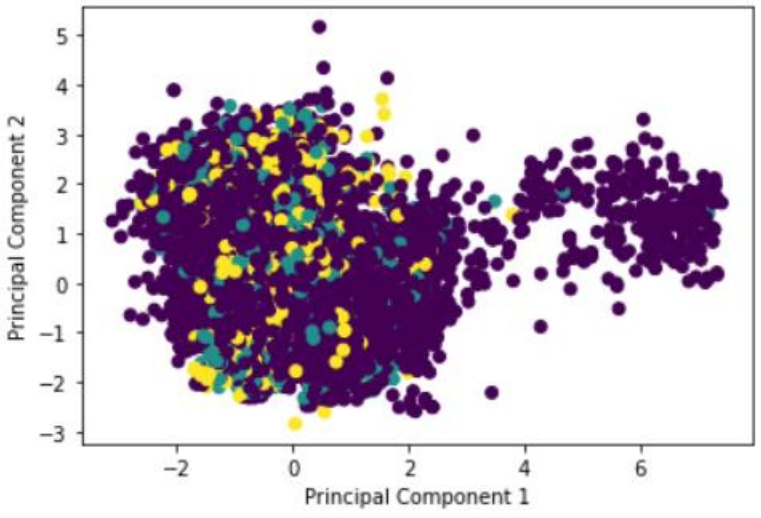


Figure 37: Result of PCA

This scatter plot allows to visualize how the data is distributed in the new two-dimensional feature space.

7.2.3 Random Forest Classifier using GridSearch

We now implement our novel random forest classification model to classify the severity of attacks into its three class labels.

Firstly, an instance of the random forest classifier is created namely rfclassifier. We set the parameter of random state= 2 which is used to control the randomness of the algorithm and in this case is set to deliver consistent results. This is achieved using the following code:

rf_classifier = RandomForestClassifier(random_state=0)

We now perform hyperparameter tuning for our classifier using the Grid search cross-validation.

The param_grid is the dictionary which is used to define the hyperparameter grid. We specify the following set of factors for the corresponding hyperparameter:

- 'n_estimators': which is the maximum number of trees,
- 'max_depth': maximum depth of tree
- 'min_samples_split': min number of samples to split internal node,
- 'min_samples_leaf': minimum number of samples required to be at leaf node

It has to be noted that these values were selected after repeated experimentation with different set of values to find the best combination for our model which gives the best performance. The final set of values we used in the project:

- 'n_estimators': [100, 200, 300],

- 'max_depth' : [10, 20, 30],
- 'min_samples_split': [2, 5, 10],
- 'min_samples_leaf': [1, 2, 4]

Now we perform hyperparameter tuning using the grid search method which is executed as follows:

```
grid_search = GridSearchCV(estimator=rf_classifier, param_grid=param_grid,
                           cv=5, n_jobs=-1, verbose=2)
```

The GridSearchCV has several parameters. The estimator defines the machine learning model to be used which in our case is the random forest classifier, hence we set it as:

estimator=rf_classifier

We then specify the hyperparameter grid to search which is already specified in the param_grid dictionary. The number of cross validation folds is specified, which we have selected as 5 (cv=5) which means that our input data will be split into 5 subsets after which the tuning process will be executed 5 times with a different validation fold each time.

Setting the parameter n_jobs=-1 enables the system to use all available cpu cores for parallel processing which will help in speeding up the grid search. The final parameter verbose is used to control the verbosity of the search. We set it as verbose=2 to get a detailed output.

Finally, we perform the grid search on the training data: X_train and y_train as shown below:

```
grid_search.fit(X_train, y_train)
```

This evaluates the random forest classifier with different combinations of hyperparameter values.

Next, we will select the best random forest model for our classification based on the grid search results. The python code for executing this phase is given as follows:

```
best_rf = grid_search.best_estimator_
```

The best_estimator_ is an attribute of the GridSearchCV ,which now stores the machine learning model or rather the estimator that achieved the best performance which in case is the random forest classifier. The variable best_rf now holds the best model for classification which can be used for making new predictions and further analysis as it is the one which performed the best on our training data which is the attack data.

Now that we have best classifier, we can now make predicitons on the test dataset as follows:

```
y_pred = best_rf.predict(X_test)
```

The predict(X_test) will return the set of new predictions baes on the applied test data. The variable y_pred finally holds the result data.

Finally, we use the following piece of code to view the confusion matrix and the classification report which hold the values of precision, recall, f- score and accuracy of our new model

```
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred)
```

7.2.4 Benchmark approach with SVM and other classifiers:

We now compare our model to other benchmark classification models namely the SVM, gradient boosting, logistic regression, Naïve bayes, KNN, Decision Tree, AdaBoost and Multi-Layer Perceptron. Firstly, we import all the necessary classifiers as follows:

```
from sklearn.svm import SVC

from sklearn.ensemble import GradientBoostingClassifier

from sklearn.linear_model import LogisticRegression

from sklearn.naive_bayes import GaussianNB

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import AdaBoostClassifier

from sklearn.neural_network import MLPClassifier

from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

Except for the random forest for which we already have an instance variable, we create new ones for each of the classifiers in use as follows:

```
svm_classifier = SVC()

gb_classifier = GradientBoostingClassifier()

logistic_classifier = LogisticRegression(max_iter=1000)

naive_bayes_classifier = GaussianNB()

knn_classifier = KNeighborsClassifier()

decision_tree_classifier = DecisionTreeClassifier()

ada_boost_classifier = AdaBoostClassifier()

mlp_classifier = MLPClassifier(max_iter=1000)
```

In order to facilitate the process, we list the classifiers and the names into respective lists:

```
all_classifiers = [

   best_rf, svm_classifier, gb_classifier, logistic_classifier,

   naive_bayes_classifier, knn_classifier, decision_tree_classifier,

   ada_boost_classifier, mlp_classifier

]
```

```
classifier_names = [
    "Random Forest", "Support Vector Machine", "Gradient Boosting",
    "Logistic Regression", "Naive Bayes", "K-Nearest Neighbors",
    "Decision Tree", "AdaBoost", "Multi-Layer Perceptron"
]
```

We also create an empty list which will hold all the accuracy values for the classifiers(accuracy_scores = []), confusion matrix (confusion_matrix=[]) and classification reports (classification_reports = []).

Using a for loop, we iterate through each classifier and evaluate the performance of each classifier:

```
for clf, clf_name in zip(all_classifiers, classifier_names):
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
```

Now we generate the confusion matrix, classification report which will show us the precision, recall and f score and the accuracy of each classifier. The python code to implement this segment is shown below:

```
accuracy = accuracy_score(y_test, y_pred)
accuracy_scores.append(accuracy)
report = classification_report(y_test, y_pred)
classification_reports.append(report)
matrix = confusion_matrix(y_test, y_pred)
confusion_matrices.append(matrix)
print(f"Classification Report for {clf_name}:\n{report}\n")
print(f"Confusion Matrix for {clf_name}:\n{matrix}\n")
print(f"{clf_name} Accuracy: {accuracy:.2f}")
report = classification_report(y_test, y_pred)
classification_reports.append(report)
print(f"{clf_name} Accuracy: {accuracy:.2f}")
print(f"Classification Report for {clf_name}:\n{report}\n")
```

For better visualisation of the achieved accuracy score and for better comparison and analysis, we use a horizontal bar chart to deliver the results. The python code is shown below:

```
plt.figure(figsize=(12, 8))
plt.barh(classifier_names, accuracy_scores, color=['blue', 'green', 'purple', 'orange', 'red', 'brown', 'pink', 'gray', 'cyan'])

plt.xlabel('Accuracy')

plt.title ('Classifier Comparison')

plt.xlim([0, 1.0])

plt.gca().invert_yaxis()

plt.show()
```

It must be noted that we have reversed the order of classifiers for better visualisation. The final step is to use the show () to view the plot.

7.3 Summary

In this chapter we saw the implementation aspect of our project where we began by performing some pre-processing steps and pre-requisites. Our novel approach using random forest classifier was discussed and explained in detail after which the benchmark models for classification were also analysed.

Chapter 8

8.0 Investigation and Evaluation

8.1 Introduction

In this chapter we will assess the performance of our machine learning model for classification. In order to draw meaningful results, we use several tools and evaluation metrics such as accuracy, precision, recall and F-score. The novel approach is compared to the benchmark models for evaluation.

8.2 Results and Discussion

8.2.1 Evaluation Metrics

In evaluating each model, following metrics will be considered:

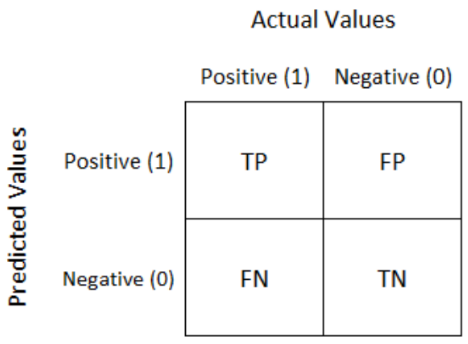
- Accuracy: It is the measure of how well a classifier can predict the data. It is the ratio of the number of correct predictions and the total number of predictions.

$$\textit{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Where,

1. TP(True Positive) : the prediction is positive and its true.
2. TN(True Negative) : the prediction is negative and its true.
3. FP(False Positive) : the prediction is positive and its false.
4. FN(False Negative): the prediction is negative and its false.

- Confusion Matrix: It is a table which is used to describe the performance of a classification model on a set of test data for which the true values are known.



Predicted Values		Actual Values: Positive (1)	Actual Values: Negative (0)
	Positive (1)	TP	FP
	Negative (0)	FN	TN

- Precision: It explains how many of the correctly predicted cases turned out to be positive.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

- Recall: It explains how many of the actual positive cases we were able to predict correctly with our model.

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

- F1-score: It gives a combined idea of both the above metrics.it is the harmonic mean of both precision and recall.

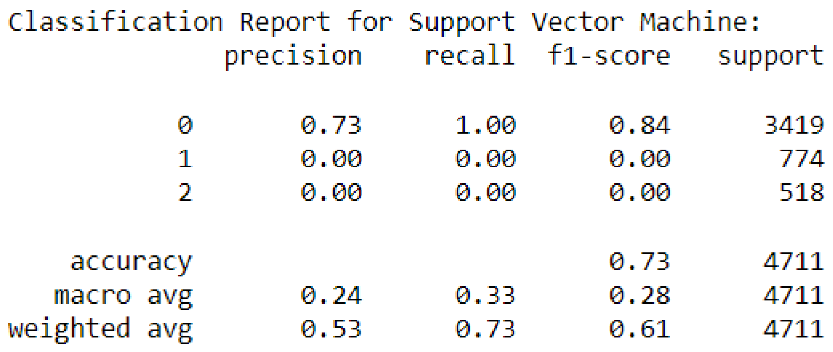
$$F1 = 2. \frac{Precision \times Recall}{Precision + Recall}$$

8.2.2 Benchmark models

The paper used as the benchmark application for most of the models is (Kanevski,M. et al., 2008). In this paper the author suggests the use of different classifiers with KNN being the prominent.

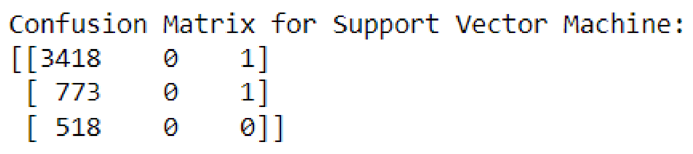
8.2.2.1 Support Vector Machine:

Classification Report having precision, recall and F1-score:

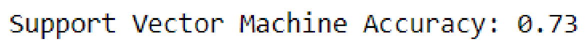

```
Classification Report for Support Vector Machine:
              precision    recall  f1-score   support

           0       0.73      1.00      0.84      3419
           1       0.00      0.00      0.00       774
           2       0.00      0.00      0.00       518

    accuracy                           0.73      4711
   macro avg       0.24      0.33      0.28      4711
weighted avg       0.53      0.73      0.61      4711
```

The three classes are 0,1,2 in which for two classes (1 and 2) the precision, recall and f score are 0.00 showing that the model was not able to correctly classify instance belonging to that class.

Confusion Matrix and accuracy:



```
Confusion Matrix for Support Vector Machine:
[[3418    0    1]
 [ 773    0    1]
 [ 518    0    0]]
```

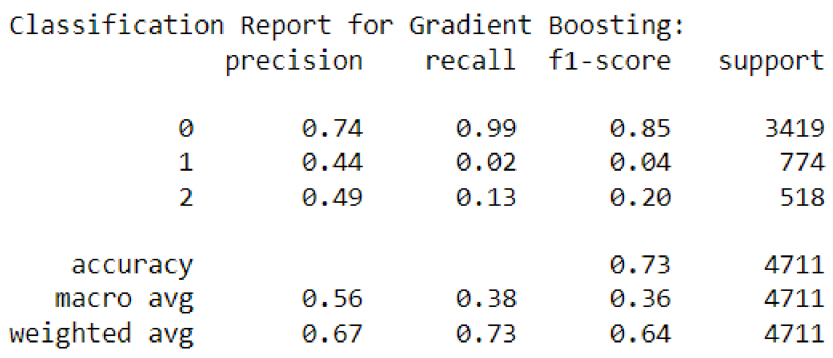


```
Support Vector Machine Accuracy: 0.73
```

We obtain an overall accuracy of 0.73 which in the end will be compared to other models and our novel model.

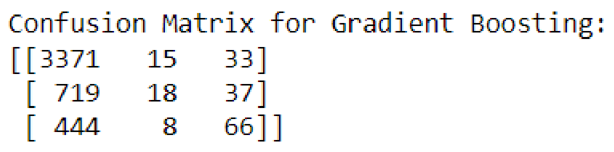
8.2.2.2 Gradient Boosting:

Classification Report having precision, recall and F1-score:

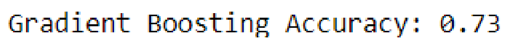


```
Classification Report for Gradient Boosting:
              precision    recall  f1-score   support

           0       0.74      0.99      0.85      3419
           1       0.44      0.02      0.04       774
           2       0.49      0.13      0.20       518

    accuracy                           0.73      4711
   macro avg       0.56      0.38      0.36      4711
weighted avg       0.67      0.73      0.64      4711
```

Confusion Matrix and accuracy:



```
Confusion Matrix for Gradient Boosting:
[[3371   15   33]
 [ 719   18   37]
 [ 444    8   66]]
```



```
Gradient Boosting Accuracy: 0.73
```

The overall accuracy obtained by gradient boosting is 0.73 which as we can see is the same as obtained by SVM.

8.2.2.3 Logistic Regression:

Classification Report having precision, recall and F1-score:

```
Gradient Boosting Accuracy: 0.73
Classification Report for Logistic Regression:
              precision    recall  f1-score   support

           0       0.73      1.00      0.84      3419
           1       0.00      0.00      0.00       774
           2       0.00      0.00      0.00       518

    accuracy                           0.73      4711
   macro avg       0.24      0.33      0.28      4711
weighted avg       0.53      0.73      0.61      4711
```

Out of the three classes, two classes (1 and 2) the precision, recall and f score are 0.00 showing that the model was not able to correctly classify instance belonging to that class.

Confusion Matrix and accuracy:

```
Confusion Matrix for Logistic Regression:
[[3419    0    0]
 [ 774    0    0]
 [ 518    0    0]]

Logistic Regression Accuracy: 0.73
```

The zeroes in the confusion matrix also suggest why the model has not performed well. The accuracy is 0.73 which will be compared to other models.

8.2.2.4 Naive Bayes:

Classification Report having precision, recall and F1-score:

```
Classification Report for Naive Bayes:
              precision    recall  f1-score   support

           0       0.73      0.99      0.84      3419
           1       0.43      0.03      0.05       774
           2       0.41      0.03      0.06       518

    accuracy                           0.73      4711
   macro avg       0.52      0.35      0.32      4711
weighted avg       0.65      0.73      0.63      4711
```

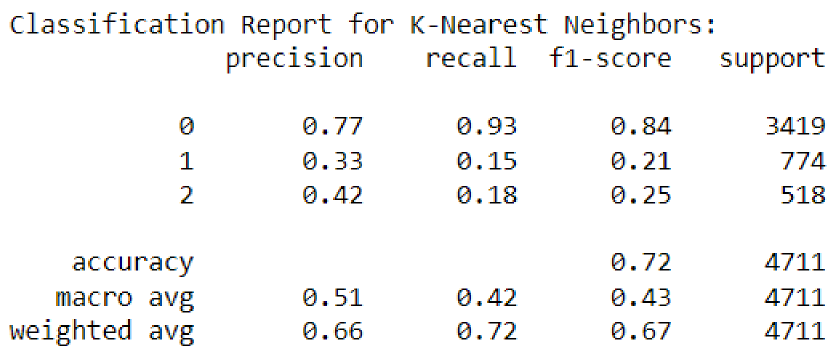
Confusion Matrix and accuracy:

```
Confusion Matrix for Naive Bayes:
[[3382   18   19]
 [ 746   21    7]
 [ 490   10   18]]

Naive Bayes Accuracy: 0.73
```

We obtain an accuracy of 0.73 when we use Naïve Bayes Classifier.

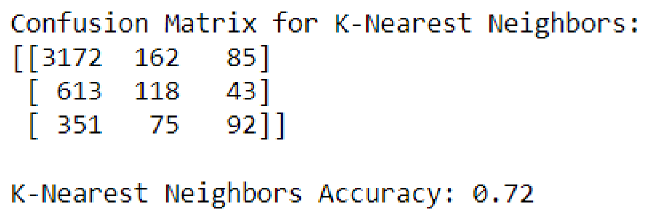
8.2.2.5 K-nearest neighbours:

Classification Report having precision, recall and F1-score:



```
Classification Report for K-Nearest Neighbors:
              precision    recall  f1-score   support

           0       0.77      0.93      0.84      3419
           1       0.33      0.15      0.21       774
           2       0.42      0.18      0.25       518

    accuracy                           0.72      4711
   macro avg       0.51      0.42      0.43      4711
weighted avg       0.66      0.72      0.67      4711
```

Confusion Matrix and accuracy:

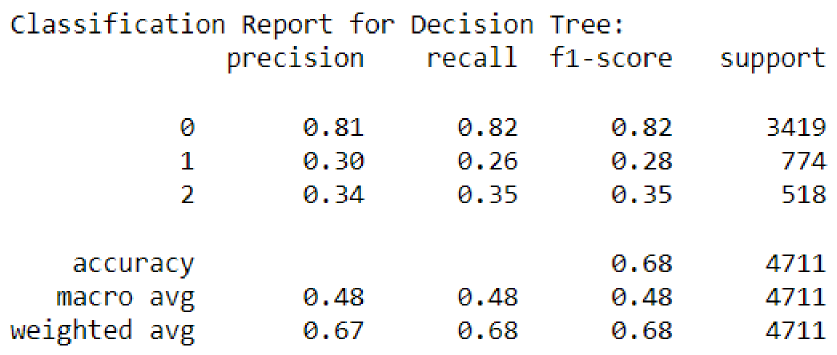


```
Confusion Matrix for K-Nearest Neighbors:
[[3172  162   85]
 [ 613  118   43]
 [ 351   75   92]]

K-Nearest Neighbors Accuracy: 0.72
```

Out of the 4 algorithms implemented till now, KNN is the least performing in terms of accuracy i.e., 0.72.

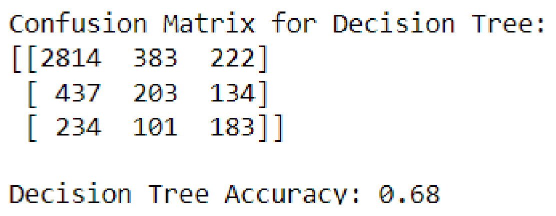
8.2.2.6 Decision Tree:

Classification Report having precision, recall and F1-score:



```
Classification Report for Decision Tree:
              precision    recall  f1-score   support

           0       0.81      0.82      0.82      3419
           1       0.30      0.26      0.28       774
           2       0.34      0.35      0.35       518

    accuracy                           0.68      4711
   macro avg       0.48      0.48      0.48      4711
weighted avg       0.67      0.68      0.68      4711
```

Confusion Matrix and accuracy:



```
Confusion Matrix for Decision Tree:
[[2814  383  222]
 [ 437  203  134]
 [ 234  101  183]]

Decision Tree Accuracy: 0.68
```

The decision tree algorithm for classification of severity with an overall accuracy of 0.68 is the least performing model when compared to other models.

8.2.2.7 AdaBoost:

Classification Report having precision, recall and F1-score:

```
Classification Report for AdaBoost:
              precision    recall  f1-score   support

           0       0.74      0.98      0.85      3419
           1       0.35      0.02      0.04       774
           2       0.46      0.12      0.19       518

    accuracy                           0.73      4711
   macro avg       0.52      0.37      0.36      4711
weighted avg       0.65      0.73      0.64      4711
```

Confusion Matrix and accuracy:

```
Confusion Matrix for AdaBoost:
[[3366   21   32]
 [ 720   15   39]
 [ 450    7   61]]

AdaBoost Accuracy: 0.73
```

We obtain an accuracy of 0.73 when we use AdaBoost Classifier.

8.2.2.8 MLP or Multi-Layer Perceptron:

Classification Report having precision, recall and F1-score:

```
Classification Report for Multi-Layer Perceptron:
              precision    recall  f1-score   support

           0       0.75      0.97      0.85      3419
           1       0.32      0.04      0.08       774
           2       0.37      0.13      0.19       518

    accuracy                           0.72      4711
   macro avg       0.48      0.38      0.37      4711
weighted avg       0.64      0.72      0.65      4711
```

Confusion Matrix and accuracy:

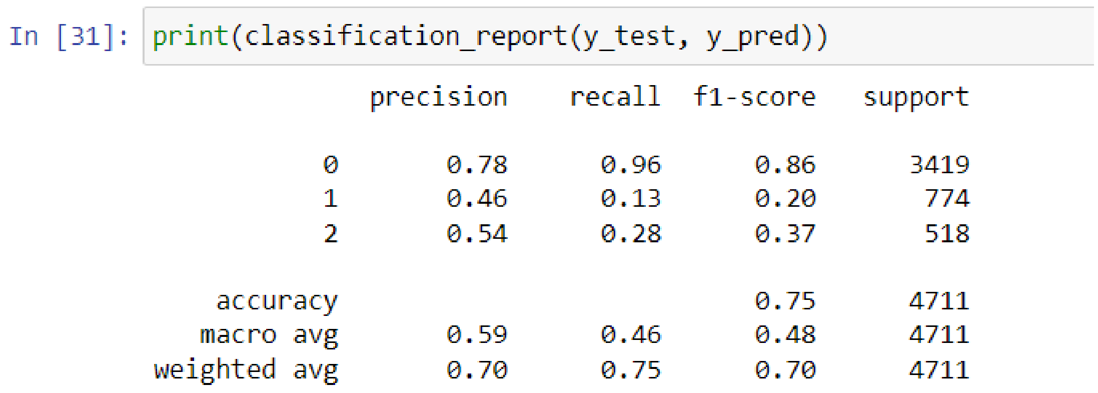
```
Confusion Matrix for Multi-Layer Perceptron:
[[3317   37   65]
 [ 693   33   48]
 [ 421   32   65]]

Multi-Layer Perceptron Accuracy: 0.72
```

We obtain an accuracy of 0.72 when we use Multi-Layer Perceptron Classifier for classification.

8.2.3 Novel Method

8.2.3.1 Random Forest Classifier using GridSearch:

Classification Report having precision, recall, F1-score and accuracy:



```
In [31]: print(classification_report(y_test, y_pred))

              precision    recall  f1-score   support

           0       0.78      0.96      0.86      3419
           1       0.46      0.13      0.20       774
           2       0.54      0.28      0.37       518

    accuracy                           0.75      4711
   macro avg       0.59      0.46      0.48      4711
weighted avg       0.70      0.75      0.70      4711
```

Precision:

- For class 0, of all the instance predicted as class 0 , 78% were actually Class 0.
- For class 1, 46% of instances predicted as class 1 were actually class 1.
- For class 2, 54% of instances predicted as class 2 were actually class 2.

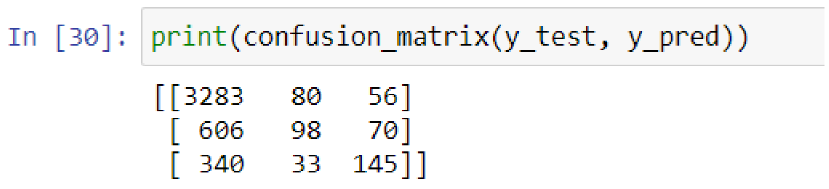
Recall:

- For class 0, the model captured 96% of actual class 0 instances.
- For class 1, the model captured only 13% of actual class 1 instances.
- For class 2, the model captured 28% of actual class 2 instances.

F1-score:

- For class 0, the score is 0.86, which is the harmonic mean of both precision and recall.
- For class 1, we have a low score of 0.20
- For class 2, the score is 0.37

Confusion Matrix:



```
In [30]: print(confusion_matrix(y_test, y_pred))

[[3283   80   56]
 [ 606   98   70]
 [ 340   33  145]]
```

Using the above confusion matrix, we can analyse the classes which need improvement like in the case of Class 0 and Class 2 which were better classified than Class 1.

Some of interpretations from the above results can be seen in the fact that the model performs the best in classifying instances for Class 0 and Class 2 than Class 1. The high precision (78%)

and recall (96%) of Class 0 are proof to this analysis. Class 0 has the greatest number of data instances in the input dataset and hence serves its purpose.

The overall accuracy which is 75% and the best among all the models which were used to compare, performs reasonably well to classify the data instances across the classes.

8.2.4 Test results summary

The Random Forest classifier emerged as a novel and highly accurate approach when compared to other classifiers, including the K-Nearest Neighbours (KNN) classifier. The Random Forest model achieved a remarkable accuracy rate in classifying geospatial terrorism data in which we classified the severity category of locations where the explosions took place which are 0 – less severe,1-moderately severe, 2-highly severe, outperforming support vector machines, gradient boosting, logistic regression, Naïve bayes, KNN, Decision Tree, AdaBoost and Multi-Layer Perceptron. This novel approach demonstrated its efficiency in handling complex, real-world datasets, showcasing robustness against overfitting, and providing valuable insights into feature importance. Its accuracy and reliability in geospatial analysis make it a standout choice for addressing the project's objectives, particularly when compared to the KNN classifier and other methods employed.

8.2.4.1 Summary under Accuracy

In the above two sections we have seen the achieved accuracy of the various models for classification. A more improved and better approach is to visualize the data on a graph such as a horizontal bar chart and visualize the data:

```
Random Forest Accuracy: 0.75
Support Vector Machine Accuracy: 0.73
Gradient Boosting Accuracy: 0.73
Logistic Regression Accuracy: 0.73
Naive Bayes Accuracy: 0.73
K-Nearest Neighbors Accuracy: 0.72
Decision Tree Accuracy: 0.68
AdaBoost Accuracy: 0.73
Multi-Layer Perceptron Accuracy: 0.72
```

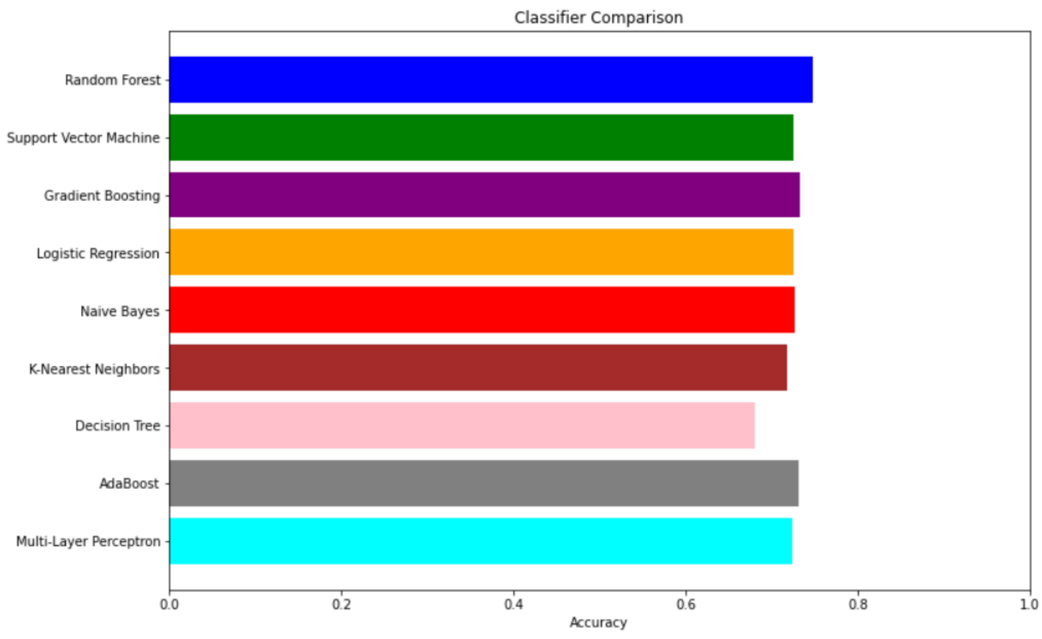


Figure 38: Classifier Performance

In the above plot each block highlighted by a particular colour represents a classifier. The x-axis holds the accuracy values i.e., from 0.0 to 1.0 whereas the y-axis holds the different classifiers.

S.NO	Classifier Model	Accuracy
1.	Random Forest (Novel Model)	75 %
2	SVM	73 %
3	Gradient Boosting	73 %
4	Logistic Regression	73 %
5	Naïve Bayes	73 %
6	KNN	72 %
7	Decision Tree	68 %
8	AdaBoost	73 %
9	MLP	72 %

From figure 38 and the above table, we can observe that all the algorithms perform very close to each other in terms of accuracy where our novel approach using random forest classifier **with an accuracy of 75% supersedes in detecting the potential intensity of explosions in different geographic locations than** Support Vector Machines **whose accuracy is 73%**, gradient boosting **whose accuracy is 73%** , logistic regression **whose accuracy is 73%**, Naïve bayes **whose accuracy is 73%,** KNN **whose accuracy is 72**%, Decision Tree **whose accuracy is 68%,** AdaBoost **whose accuracy is 73%** and Multi-Layer Perceptron **whose accuracy is 72%.**

8.3 Summary

Our approach for classifying the severity of explosive attacks using Random Forest, performed the best in terms of accuracy when compared to benchmark classifier models namely support vector machines, gradient boosting, logistic regression, Naïve bayes, KNN, Decision Tree, AdaBoost and Multi-Layer Perceptron. Each of the classifier's accuracy and performance was compared with the random forest model. We investigated the values of precision, recall, f1-score which were used to analyse the performance of the algorithm. Several features were chosen for the model such as the latitude, longitude etc which were used to train the model and the result is a machine learning model with an overall accuracy of 75% which can now be used for classification of the severity of various locations in the world.

Chapter 9

9.0 Conclusion

My project ‘Geospatial analysis of Explosive Data for Strategic Planning and decision making’, can be divided into 2 parts in which the first part we deploy some geospatial analysis on the explosive data and in the second part we use machine learning algorithms for classification of the severity of attack locations which are explained as follows:

1. In previous chapters we used location-based data or precisely spatial data from our input dataset and perform geospatial analysis using some of the maps offered by the python libraires. Interactive maps using Folium, choropleth maps, heatmaps and time series variation maps were some of the maps that were used to aid the analysis process. Geospatial analysis is very useful and informative for several organisations including our client which is NATO. Using the heatmaps, areas with higher density of incidents can be analysed and hence NATO can identify such regions which require heightened security measures. Such maps also help NATO in trend analysis, i.e., identify the trends in attacks which will help them investigate the underlying causes and therefore adapt its strategies for the future accordingly. We also visualized a map which shows the attack count per countries which will help NATO to prioritize their security assets which towards the end will help in strategic planning. Finally, we employed a time series variation map which also helps in trend analysis of explosive attacks.
2. In the second part of our project, we successfully classified the severity of attack locations into three categories
   - Low severity (0)
   - Medium severity (1)
   - High severity (2)

Hence, we provided NATO an effective tool to prioritize the response efforts which will always help them in strategic planning. Through our project results, the novel random forest classifier using GridSearch outplayed the benchmark models in terms of accuracy and hence was the best model for classification.

Together, both geospatial analysis and classification are very vital. Classification as we know provides NATO with a tool for prioritization meaning the resources such as personal, equipment and funding can be allocated more efficiently to the regions with a higher verity level. Whereas the maps will help in visualizing such areas, making it easier for NATO, to identify where their efforts are needed the most.

In conclusion, geospatial analysis of explosive data and classification of severity of attack locations go hand in hand in accomplishing the project aim for strategic planning. they help in addressing security challenges and helps NATO, make more informed decisions.

As mentioned, in their website about their fundamental role:

"To guarantee the freedom and security of its member countries by political and military means (NATO, n.d.)." Our project deliverables certainly help NATO in its mission towards promoting security and peace.

9.1 Further Research

In this section we will see the various areas that can use further research and exploration that help further in improving and enhancing the explosive data analysis and further aid NATO in their effort to promote security. Some of the areas for further research are as follows:

1. Using of natural language processing or NLP techniques to analyse textual data can be an area of further research. Textual data can help us in providing better and deeper insights in understanding the different types of data which can help in better assessment.
2. Using much more advanced machine learning algorithms or combinations of algorithms might yield different and better accuracy and analysis results such as deep neural networks
3. In our project for geospatial analysis, we used python libraries for implementing maps and other features, however there are other systems or platforms available such as the GIS software or Geographic Information System like the ArcGIS, QGIS provide environments for such analysis. They offer wide range of tools for data processing and other visualisations which are widely used in various industries. It is always beneficial to employ such different technologies for the same analysis and see if can deliver more promising results than the other.
4. Instead of using just one base dataset, we can also use multiple relevant datasets related to our parent data and gain a more comprehensive understanding of explosive data.

9.2 What to do when you find a suspicious item

Whether it be NATO or any other security organisation, all agencies work together in order to reach the common goal, which is safeguard its citizens from foreign and unknown threats. Attacks from explosive devices are on the rise since the turn of the century which makes it essential for the general public to be aware of 'what to do' when you find a suspicious item. The following data is from the America's Cyber Défense Agency (https://www.cisa.gov/news-events/news/what-do-bomb-threat)

According to (Cybersecurity and Infrastructure Security Agency [CISA], n.d.)

If you find a suspicious item...

Together we can help keep our communities safe—if you see something that is suspicious, out of place, or doesn't look right, say something.

A suspicious item is any item (e.g., bag, package, vehicle, etc.) that is reasonably believed to contain explosives, an Improvised Explosive Device (IED), or other hazardous material that requires a bomb technician and/or specialized equipment to further evaluate it. Examples that could indicate a bomb include unexplainable wires or electronics, other visible bomb-like components, and unusual sounds, vapors, mists, or odors. Generally speaking, anything that is Hidden, obviously suspicious, and not Typical (HOT) should be deemed suspicious. In addition, potential indicators for a bomb are threats, placement, and proximity of the item to people and valuable assets.

You may encounter a suspicious item unexpectedly or while conducting a search as part of your facilities or employer’s Bomb Threat Response Plan. If it appears to be a suspicious item, follow these procedures:

1.Remain calm.

2.Do NOT touch, tamper with, or move the package, bag, or item.

3.Notify authorities immediately:

> Notify your facility supervisor, such as a manager, operator, or administrator, or follow your facility's standard operating procedure. (See below for assistance with developing a plan for your facility or location.)
>
> •Call 9-1-1 or your local law enforcement if no facility supervisor is available.
>
> •Explain why it appears suspicious.

4.Follow instructions. Facility supervisors and/or law enforcement will assess the situation and provide guidance regarding shelter-in-place or evacuation.

5.If no guidance is provided and you feel you are in immediate danger, calmly evacuate the area. Distance and protective cover are the best ways to reduce injury from a bomb.

6.Be aware. There could be other threats or suspicious items.

Every situation is unique and should be handled in the context of the facility or environment in which it occurs. Facility supervisors and law enforcement will be in the best position to determine if a real risk is posed and how to respond.

References

A. S. Alsaedi, A. S. Almobarak and S. T. Alharbi, "Mining the Global Terrorism Dataset using Machine Learning Algorithms," 2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA), Abu Dhabi, United Arab Emirates, 2019, pp. 1-7, doi: 10.1109/AICCSA47632.2019.9035296.

P. Agarwal, M. Sharma and S. Chandra, "Comparison of Machine Learning Approaches in the Prediction of Terrorist Attacks," 2019 Twelfth International Conference on Contemporary Computing (IC3), Noida, India, 2019, pp. 1-7, doi: 10.1109/IC3.2019.8844904.

L. Markowsky and G. Markowsky, "Lattice-Based Technique to Visualize and Compare Regional Terrorism Using the Global Terrorism Database," 2021 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), Cracow, Poland, 2021, pp. 257-266, doi: 10.1109/IDAACS53288.2021.9660877.

T. Xia and Y. Gu, "Building Terrorist Knowledge Graph from Global Terrorism Database and Wikipedia," 2019 IEEE International Conference on Intelligence and Security Informatics (ISI), Shenzhen, China, 2019, pp. 194-196, doi: 10.1109/ISI.2019.8823450.

W. Han, Z. Yang, L. Di, B. Zhang and C. Peng, "Enhancing Agricultural Geospatial Data Dissemination and Applications Using Geospatial Web Services," in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 7, no. 11, pp. 4539-4547, Nov. 2014, doi: 10.1109/JSTARS.2014.2315593.

S. Nanda, M. Ranjan Mishra, A. Narayan Brahma, S. Chandra Swain, S. Shekhar Patra and R. Kumar Barik, "Performance Analysis of Geospatial Serverless Computing for Geospatial Big Data Analysis," 2022 3rd International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India, 2022, pp. 716-720, doi: 10.1109/ICESC54411.2022.9885470.

A. Yoshizumi et al., "A Review of Geospatial Content in IEEE Visualization Publications," 2020 IEEE Visualization Conference (VIS), Salt Lake City, UT, USA, 2020, pp. 51-55, doi: 10.1109/VIS47514.2020.00017.

I. Simonis, "Geospatial Big Data Processing in Hybrid Cloud Environments," IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium, Valencia, Spain, 2018, pp. 419-421, doi: 10.1109/IGARSS.2018.8519218.

H. Deng, P. Yue, D. Yu, Z. Cao, R. Liu and K. Wang, "A Geospatial Data and Model Hub for Online Geospatial Analysis," 2023 11th International Conference on Agro-Geoinformatics (Agro-Geoinformatics), Wuhan, China, 2023, pp. 1-6, doi: 10.1109/Agro-Geoinformatics59224.2023.10233547.

M. J. Hossain, S. M. Abdullah, M. Barkatullah, M. S. U. Miah, T. B. Sarwar and M. F. Monir, "Predicting the Success of Suicide Terrorist Attacks using different Machine Learning Algorithms," 2022 25th International Conference on Computer and Information Technology

(ICCIT), Cox's Bazar, Bangladesh, 2022, pp. 378-383, doi: 10.1109/ICCIT57492.2022.10055100.

K. Singh, A. S. Chaudhary and P. Kaur, "A Machine Learning Approach for Enhancing Defence Against Global Terrorism," 2019 Twelfth International Conference on Contemporary Computing (IC3), Noida, India, 2019, pp. 1-5, doi: 10.1109/IC3.2019.8844947.

A. Yoshizumi et al., "A Review of Geospatial Content in IEEE Visualization Publications," 2020 IEEE Visualization Conference (VIS), Salt Lake City, UT, USA, 2020, pp. 51-55, doi: 10.1109/VIS47514.2020.00017.

N. Thakur, S. S. Saini and A. K. Pathak, "Data Mining Model Framework for GTD (Global Terrorism Database)," 2022 International Conference on Cyber Resilience (ICCR), Dubai, United Arab Emirates, 2022, pp. 1-5, doi: 10.1109/ICCR56254.2022.9995939.

H. Deng, P. Yue, D. Yu, Z. Cao, R. Liu and K. Wang, "A Geospatial Data and Model Hub for Online Geospatial Analysis," 2023 11th International Conference on Agro-Geoinformatics (Agro-Geoinformatics), Wuhan, China, 2023, pp. 1-6, doi: 10.1109/Agro-Geoinformatics59224.2023.10233547.

Y. -Y. Lee and Y. L. Chang, "Uncovering Los Angeles Tourists' Patterns Using Geospatial Analysis and Supervised Machine Learning with Random Forest Predictors," 2019 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 2019, pp. 1275-1280, doi: 10.1109/CSCI49370.2019.00239.

Kanevski, M.; Pozdnukhov, A.; and Timonin, V., "Machine Learning Algorithms for GeoSpatial Data. Applications and Software Tools" (2008). International Congress on Environmental Modelling and Software. 53. https://scholarsarchive.byu.edu/iemssconference/2008/all/53

Cybersecurity and Infrastructure Security Agency (CISA). (n.d.). What to Do During a Bomb Threat. https://www.cisa.gov/news-events/news/what-do-bomb-threat

NATO. (n.d.). NATO Checklist. Fundamental Role: To guarantee the freedom and security of its member countries by political and military means. NATO Official Website. https://www.nato.int/nato-welcome/files/checklist_en.pdf

START. (n.d.). Overview of the GTD. Retrieved from https://www.start.umd.edu/gtd/about/

APPENDIX

Appendix A: Project Specification Form

UNIVERSITY OF PORTSMOUTH

School of Computing Postgraduate Programme

MSc in Data Analytics

Project Specification

TITO THOMAS DAVID

Project Specification

1. Basic details

Student name:	TITO THOMAS DAVID
Draft project title:	Geospatial Analysis of Explosive Data for Strategic Planning
Course and year:	MSc Data Analytics 2022-2023
Client organisation:	NATO ACT
Client contact name:	Dr Alexander Gegov
Project supervisor:	Dr Alexander Gegov

2. Outline of the project environment

The client is NATO ACT. In this project we will try to perform geospatial analysis using machine learning techniques and algorithms on explosive data. This analysis can be used by the NATO to gain insights into strategic planning and resource allocation.

3. The problem to be solved

The purpose or the aim of the project is to enable or help the NATO to gain insights into attacks caused by explosive materials, identifying the high-risk areas so that the resources can be allocated accordingly and steps can be taken so as to be prepared in advance. This project will also help in analysing the attack patterns and trends. Further predictive models can help several organisations to forecast the severity of such attacks in the future. Such measures will help NATO to enhance their preparedness for such attacks.

Even if there are several such methodologies in place, the newly proposed project will develop a system which will be much more efficient and provide better accuracy in terms of results.

4. Breakdown of tasks

- Combine different datasets from different sources and integrate them together.
- Enhance data quality or pre-processing.
- Use data/text mining and other advanced analytics methodology to provide insight to relevant data.
- Develop an ML system using different AI algorithms.
- Compare the different models and provide a valuable insight.

5. Project deliverables

To build and implement an AI machine learning system with project report, and python programming code.

7. Legal, ethical, professional, social issues

None of the issues would be contravened. The data is sourced from the following website and is accessible freely:

https://www.start.umd.edu/gtd/

I'll be adding more data from different websites to integrate all of them to form a final dataset for the project.

8. Facilities and resources

No particular facilities to be used.

9. Project plan

The project will be executed in the following steps:

- Collect all the required information and background knowledge about the existing methodologies and systems which could be improved upon – 1 week(approx.)
- Analyse any possible challenges one can face and develop a proper plan in order to overcome such problems – 1 week(approx.)
- Decide on the methodology to be used – 1 week(approx.)
- Implementation phase which includes the dataset integration to the final step of evaluating the results and comparing it with other algorithms – 1 week(approx.)
- Drawing conclusions from the results and further explanation – 1 week(approx.)
- Trying to modify the current model by making small changes so as to improve upon the existing system – 1 week(approx.)

10. Supervision meetings

A weekly meeting agreed with supervisor which is face to face but also could be online through Google meet or other platforms.

11. Project mode

If there are two possibilities for your project mode, after negotiation, please record your planned duration and submission date. It is also helpful to record your initial registration mode (i.e. are you a full time or a part time student). Remember, the exact dates will be announced through Moodle – these represent a generic guideline.

Please delete as appropriate	
Full Time	
Full Time	
22/09/23	

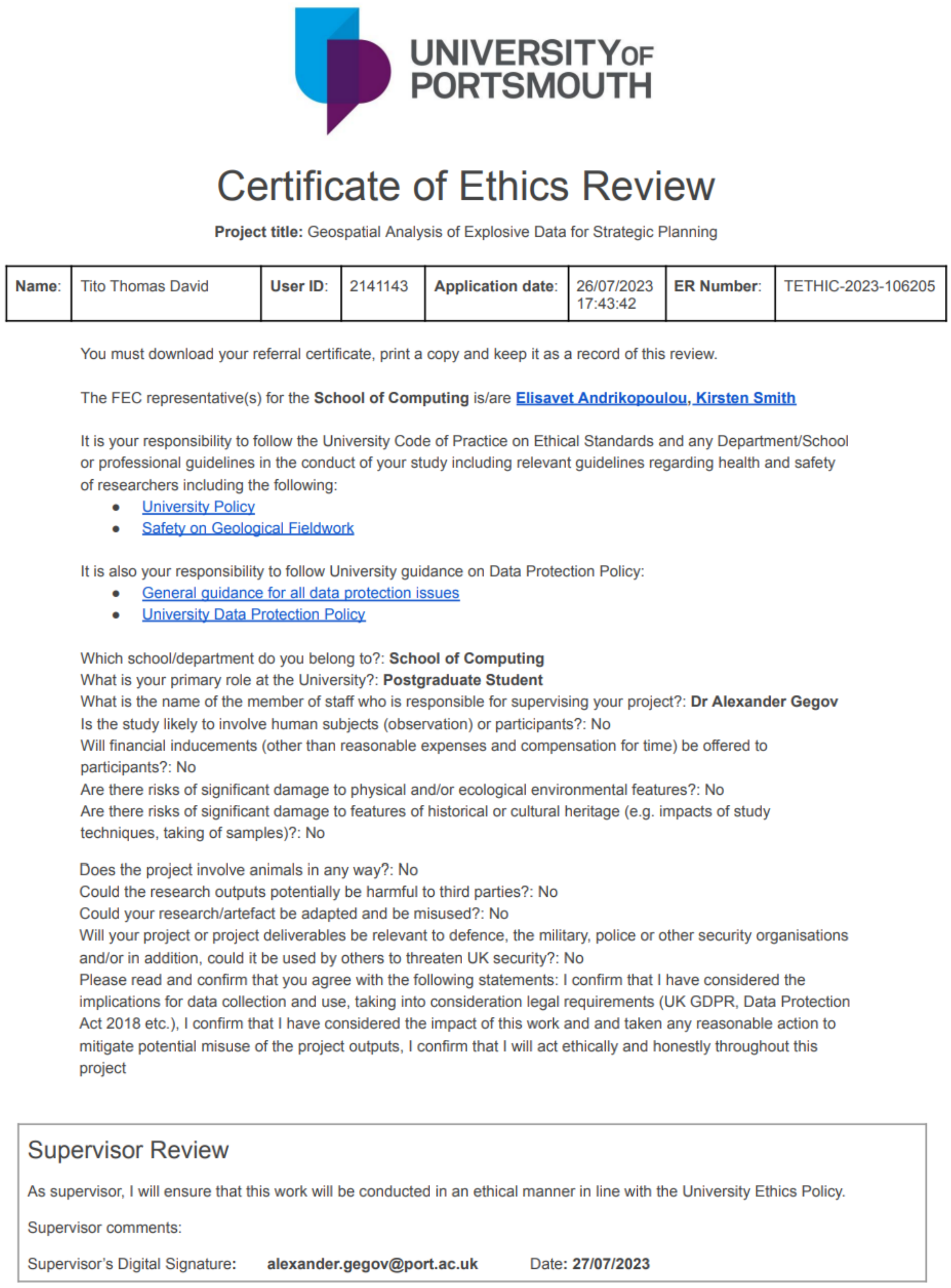
Registration mode

Project mode Planned submission deadline

12. Signatures

	Signature:	Date:
Student	Tito Thomas David	02/06/23
Client	NATO ACT	02/06/23
Project supervisor	Dr Alexander Gegov	02/06/23

Appendix B: Ethics Form



UNIVERSITY OF PORTSMOUTH

Certificate of Ethics Review

Project title: Geospatial Analysis of Explosive Data for Strategic Planning

Name:	Tito Thomas David	**User ID**:	2141143	**Application date**:	26/07/2023 17:43:42	**ER Number**:	TETHIC-2023-106205

You must download your referral certificate, print a copy and keep it as a record of this review.

The FEC representative(s) for the **School of Computing** is/are **Elisavet Andrikopoulou, Kirsten Smith**

It is your responsibility to follow the University Code of Practice on Ethical Standards and any Department/School or professional guidelines in the conduct of your study including relevant guidelines regarding health and safety of researchers including the following:

- University Policy
- Safety on Geological Fieldwork

It is also your responsibility to follow University guidance on Data Protection Policy:

- General guidance for all data protection issues
- University Data Protection Policy

Which school/department do you belong to?: **School of Computing**
What is your primary role at the University?: **Postgraduate Student**
What is the name of the member of staff who is responsible for supervising your project?: **Dr Alexander Gegov**
Is the study likely to involve human subjects (observation) or participants?: No
Will financial inducements (other than reasonable expenses and compensation for time) be offered to participants?: No
Are there risks of significant damage to physical and/or ecological environmental features?: No
Are there risks of significant damage to features of historical or cultural heritage (e.g. impacts of study techniques, taking of samples)?: No

Does the project involve animals in any way?: No
Could the research outputs potentially be harmful to third parties?: No
Could your research/artefact be adapted and be misused?: No
Will your project or project deliverables be relevant to defence, the military, police or other security organisations and/or in addition, could it be used by others to threaten UK security?: No
Please read and confirm that you agree with the following statements: I confirm that I have considered the implications for data collection and use, taking into consideration legal requirements (UK GDPR, Data Protection Act 2018 etc.), I confirm that I have considered the impact of this work and and taken any reasonable action to mitigate potential misuse of the project outputs, I confirm that I will act ethically and honestly throughout this project

Supervisor Review

As supervisor, I will ensure that this work will be conducted in an ethical manner in line with the University Ethics Policy.

Supervisor comments:

Supervisor's Digital Signature: **alexander.gegov@port.ac.uk** Date: **27/07/2023**

Appendix C: Python Code

```
# In[1]:
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.decomposition import PCA
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix

# In[2]:
data = pd.read_csv('mydata_gtd.csv')

# In[3]:
data.info()

# In[4]:
attack_types=data['attacktype'].unique()
for attack_type in attack_types:
   print(attack_type)

# In[5]:
data = data[data['attacktype'] == 'Bombing/Explosion']
```

```
# In[6]:
missing_values = data.isna()
missing_count = missing_values.sum()
print(missing_count)

# In[7]:
data = data.dropna()

# In[8]:
missing_values = data.isna()
missing_count = missing_values.sum()
print(missing_count)

# In[9]:
data.info()

# In[10]:
columns_to_drop = ['multiple', 'success', 'suicide',
'property','nkillus','nkillter','nwoundus','nwoundte']
data = data.drop(columns=columns_to_drop)

# In[11]:
data.info()

# In[12]:
weight_fatalities = 1
weight_injuries = 0.5
data['severity_score'] = (weight_fatalities * data['nkill']) + (weight_injuries * data['nwound'])
plt.figure(figsize=(10, 6))
plt.hist(data['severity_score'], bins=20, color='blue', alpha=0.7)
plt.xlabel('Severity Score')
plt.ylabel('Frequency')
plt.title('Distribution of Severity Scores')
plt.show()

# In[13]:
minimum_severity = data['severity_score'].min()
maximum_severity = data['severity_score'].max()
print("Minimum Severity Score:", minimum_severity)
print("Maximum Severity Score:", maximum_severity)

# In[14]:
def map_severity_manual(severity):
    if severity <= 3:
        return 0
    elif severity <= 10:
        return 1
    else:
        return 2
data['severity_category_manual'] = data['severity_score'].apply(map_severity_manual)

# In[15]:
severity_counts = data['severity_category_manual'].value_counts()
plt.figure(figsize=(8, 6))
severity_counts.plot(kind='bar', color='skyblue')
plt.xlabel('Severity Category')
plt.ylabel('Count')
plt.title('Count of Severity Categories')
plt.xticks(rotation=0)
plt.show()
```

```
# In[16]:

print(data[['severity_score', 'severity_category_manual']])


# In[17]:

plt.figure(figsize=(10, 6))

data['iyear'].value_counts().sort_index().plot(kind='bar')

plt.title('Number of Incidents by Year')

plt.xlabel('Year')

plt.ylabel('Number of Incidents')

plt.show()


# In[18]:

deaths_by_year = data.groupby('iyear')['nkill'].sum()

plt.figure(figsize=(10, 6))

deaths_by_year.plot(kind='bar')

plt.title('Number of People Killed by Year')

plt.xlabel('Year')

plt.ylabel('Number of People Killed')

plt.show()


# In[19]:

top_10_affected_targets = data['target'].value_counts().head(5)

plt.figure(figsize=(8, 8))

plt.pie(top_10_affected_targets, labels=top_10_affected_targets.index, autopct='%1.1f%%',
startangle=140)

plt.title('Top 10 Target Types Being Affected by Explosive Attacks', y=1.08)  # Adjust the y
parameter for title position

plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.

plt.show()
```

```
# In[20]:
N = 5  # Specify the number of top groups to visualize
top_N_groups = data['gname'].value_counts().head(5)
plt.figure(figsize=(10, 6))
top_N_groups.plot(kind='barh', color='skyblue')
plt.title(f'Top {N} Groups Responsible for Explosive Incidents')
plt.xlabel('Number of Incidents')
plt.ylabel('Group Name')

for index, value in enumerate(top_N_groups):
    plt.text(value, index, str(value))
plt.show()

# In[21]:

import plotly.express as px
N = 10  # Specify the number of top regions to visualize
top_N_regions = data['region'].value_counts().head(N)
region_df = pd.DataFrame({'region': top_N_regions.index, 'attack_count':
top_N_regions.values})
fig = px.treemap(region_df, path=['region'], values='attack_count', title=f'Top {N} Regions
with the Highest Number of Attacks')
fig.update_layout(margin=dict(t=50, l=0, r=0, b=0))
fig.show()

# In[22]:
import folium
from folium.plugins import HeatMap
m = folium.Map(location=[50.8791, 4.4262], zoom_start=2)
heat_data = [[row['latitude'], row['longitude']] for index, row in data.iterrows()]
HeatMap(heat_data).add_to(m)
```

```
m


# In[23]:
import folium
import numpy as np
m = folium.Map(zoom_start=2)
max_radius = 20
data['scaled_severity'] = np.interp(data['severity_score'], (data['severity_score'].min(),
data['severity_score'].max()), (1, max_radius))
for index, row in data.iterrows():
    folium.CircleMarker(
        location=[row['latitude'], row['longitude']],
        radius=row['scaled_severity'],
        color='red',
        fill=True,
        fill_color='red',
        fill_opacity=0.7
    ).add_to(m)
M


# In[24]:
import geopandas as gpd
import matplotlib.pyplot as plt
world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))
attacks_by_country = data.groupby('country_txt').size().reset_index(name='attack_count')
world = world.merge(attacks_by_country, left_on='name', right_on='country_txt')


top_5_countries = world.sort_values(by='attack_count', ascending=False).head(5)
fig, ax = plt.subplots(1, 1, figsize=(20, 12))  # Increase the figsize for a larger map
world.plot(column='attack_count', ax=ax, legend=True, cmap='OrRd', legend_kwds={'label':
"Attack Count"})
```

```
for idx, row in top_5_countries.iterrows():
    ax.annotate(text=row['name'], xy=(row['geometry'].centroid.x, row['geometry'].centroid.y),
                xytext=(3, 3), textcoords='offset points', fontsize=12, color='black')
plt.title('Explosive Attack Count by Country')
plt.show()


# In[25]:
explosive_data = data[data['region'] == 'Western Europe']
mapbox_token =
"pk.eyJ1IjoidGl0b2RkYXZpZCIsImEiOiJjbGx6dTE4OHAzNzJ3M2VwZXM3MmNoanhzIn
0.xfgL3AzK7e69ikMqHvhizQ"
fig = px.scatter_mapbox(
    explosive_data,
    lat="latitude",
    lon="longitude",
    color='severity_category_manual',
    mapbox_style='light',
    opacity=1.0,
    color_discrete_map={0: 'blue', 1: 'yellow', 2: 'red'},
    animation_frame='iyear',
    zoom=8,
    height=650
)
fig.update_layout(mapbox=dict(accesstoken=mapbox_token))
fig.update_layout(
    margin=dict(l=30, r=30, t=60, b=30),
    template='plotly_white',
    title='<b>Severity Levels by Year | TIME SERIES VARIATION',
    showlegend=True,
    font=dict(family='sans-serif', size=12)
)
```

```
fig.show()


# In[26]:

print(data['gname'].head(10))


# In[27]:

from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()

categorical_columns = ['attacktype', 'targtype', 'gname', 'natlty', 'target', 'motive',
'weapsubtype1_txt', 'weapdetail', 'country_txt', 'region', 'provstate', 'city']

for column in categorical_columns:

    data[column] = label_encoder.fit_transform(data[column])


# In[28]:

print(data['gname'].head(10))


# In[29]:

features = ['iyear', 'latitude', 'longitude','targtype', 'gname', 'natlty', 'target']

X = data[features]


# In[30]:

y = data['severity_category_manual']


# In[31]:

scaler = StandardScaler()

X_scaled = scaler.fit_transform(X)

# In[32]:

pca = PCA(n_components=2)

X_pca = pca.fit_transform(X_scaled)

# Visualize the reduced data

plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y, cmap='viridis')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.show()


# In[34]:
X_train, X_test, y_train, y_test = train_test_split(X_pca, y, test_size=0.3, random_state=0)


# In[35]:
rf_classifier = RandomForestClassifier(random_state=0)


# In[36]:
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}


# In[37]:
grid_search = GridSearchCV(estimator=rf_classifier, param_grid=param_grid,
                           cv=5, n_jobs=-1, verbose=2)


# In[38]:
grid_search.fit(X_train, y_train)


# In[39]:
best_rf = grid_search.best_estimator_


# In[40]:
y_pred = best_rf.predict(X_test)
```

```
# In[41]:
print(confusion_matrix(y_test, y_pred))

# In[42]:
print(classification_report(y_test, y_pred))

# In[51]:
from sklearn.svm import SVC
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.neural_network import MLPClassifier

svm_classifier = SVC()
gb_classifier = GradientBoostingClassifier()
logistic_classifier = LogisticRegression(max_iter=1000)
naive_bayes_classifier = GaussianNB()
knn_classifier = KNeighborsClassifier()
decision_tree_classifier = DecisionTreeClassifier()
ada_boost_classifier = AdaBoostClassifier()
mlp_classifier = MLPClassifier(max_iter=1000)

all_classifiers = [
    best_rf, svm_classifier, gb_classifier, logistic_classifier,
    naive_bayes_classifier, knn_classifier, decision_tree_classifier,
    ada_boost_classifier, mlp_classifier
]


classifier_names = [
    "Random Forest", "Support Vector Machine", "Gradient Boosting",
    "Logistic Regression", "Naive Bayes", "K-Nearest Neighbors",
    "Decision Tree", "AdaBoost", "Multi-Layer Perceptron"
]
accuracy_scores = []
classification_reports = []
confusion_matrices = []
for clf, clf_name in zip(all_classifiers, classifier_names):
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)

    accuracy = accuracy_score(y_test, y_pred)
    accuracy_scores.append(accuracy)

    report = classification_report(y_test, y_pred)
    classification_reports.append(report)

    matrix = confusion_matrix(y_test, y_pred)
    confusion_matrices.append(matrix)

    print(f"Classification Report for {clf_name}:\n{report}\n")
    print(f"Confusion Matrix for {clf_name}:\n{matrix}\n")
    print(f"{clf_name} Accuracy: {accuracy:.2f}")

# In[52]:
accuracy_scores = []
```

```
for clf, clf_name in zip(all_classifiers, classifier_names):
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    accuracy_scores.append(accuracy)
    print(f"{clf_name} Accuracy: {accuracy:.2f}")

plt.figure(figsize=(12, 8))
plt.barh(classifier_names, accuracy_scores, color=['blue', 'green', 'purple', 'orange', 'red',
'brown', 'pink', 'gray', 'cyan'])
plt.xlabel('Accuracy')
plt.title('Classifier Comparison')
plt.xlim([0, 1.0])
plt.gca().invert_yaxis()  # Reverse the order of classifiers for better visualization
plt.show()
```