# Project Report: USA Terrorism Attack Success Prediction

Using the CRISP-DM Model

## CRISP-DM – Model

The Cross industry Standard Process for Data Mining is a process model that serves as the base for the process and steps one should follow for data mining, analytics & data science projects. It includes 6 phases we followed excluding the Deployment phase.

## Business Understanding

In this phase it is crucial to approach the project with a deep understanding of what the needs are. In this case it would be what problem are we trying to solve with our dataset. We have decided to use terrorism data from the Global Terrorism Database (GTD) who have collected data about global terrorism starting from the 1970's. From this data we have decided to filter the results and focus on the events that occurred in the USA. It is our hope that we can use this data to provide law enforcement agencies like the FBI (Federal Bureau of Investigation) in the US with a model that will help them predict the success of a terror attack based on certain variables.

The FBI is the USA's domestic intelligence and security service. They are its primary federal law enforcement agency. Their main priority is protecting the citizens of its nation from things like cyber-crimes, terrorism, and counterintelligence as its primary areas of focus however, they also investigate in the areas of organized crime & violent crimes, upholding and protecting civil rights to child predators and serial killers to name a few more.

Our Objective: Create a tool/model that helps them assess the probability of an attacks success before it is carried out including other independent variables. We define the term 'success' the same way as the description given in the GTD to ensure consistency & reliability.

**Successful Attack**

"Success of a terrorist strike is defined according to the tangible effects of the attack. Success is not judged in terms of the larger goals of the perpetrators. For example, a bomb that exploded in a building would be counted as a success even if it failed in bringing the building down or inducing government repression.

The definition of a successful attack depends on the type of attack. Essentially, the key question is whether the attack type took place. If a case has multiple attack types, it is successful if any of the attack types are successful, apart from assassinations, which are only successful if the intended target is killed." (National Consortium for the Study of Terrorism and Responses to Terrorism | University of Maryland, 2021)

- **Resources Available**
  - Witek & Other Lecturers
  - ChatGPT
  - globalterrorismdb_0718dist.csv (Data Set)
  - prior expertise & knowledge

- **Risk**
    - Risk of our findings & models not being useful/insightful for their intended purpose

- **Contingencies**
    - Use our findings/model to see if there is anything that we learned in the process or have any interesting finding to present/put forward.

# Data Understanding

## Variables description

There are several variables related to the attack in the database that are considered as following up after the event. For this reason, we picked only certain attributes related to the attack that would not give an indication of the aftermath of the crime (independent variables). The description of each variable has been extracted from the GTD Codebook National Consortium for the Study of Terrorism and Responses to Terrorism | University of Maryland, 2021):

- **motive**: Motive mentioned on the incident report. This field may also include general information about the political, social, or economic climate at the time of the attack if considered relevant to the motivation underlying the incident.
- **provstate:** State where the event occurs
- **attacktype1_txt**: General method of attack. It consists of nine categories. Up to three attack types can be recorded for each incident.
- **targetype1_txt:** General type of target or victim of the attack
- **weaptype1_txt**: General type of weapon used. Thirteen different categories (including Other or Unknown)
- **weapsubtype1_txt:** More specific value for the weapon type identified.

Regarding the dependent variable related to the outcome of the attack we are using the “success” field. Where 1 indicates the attack was successful and 0 that it wasn’t.

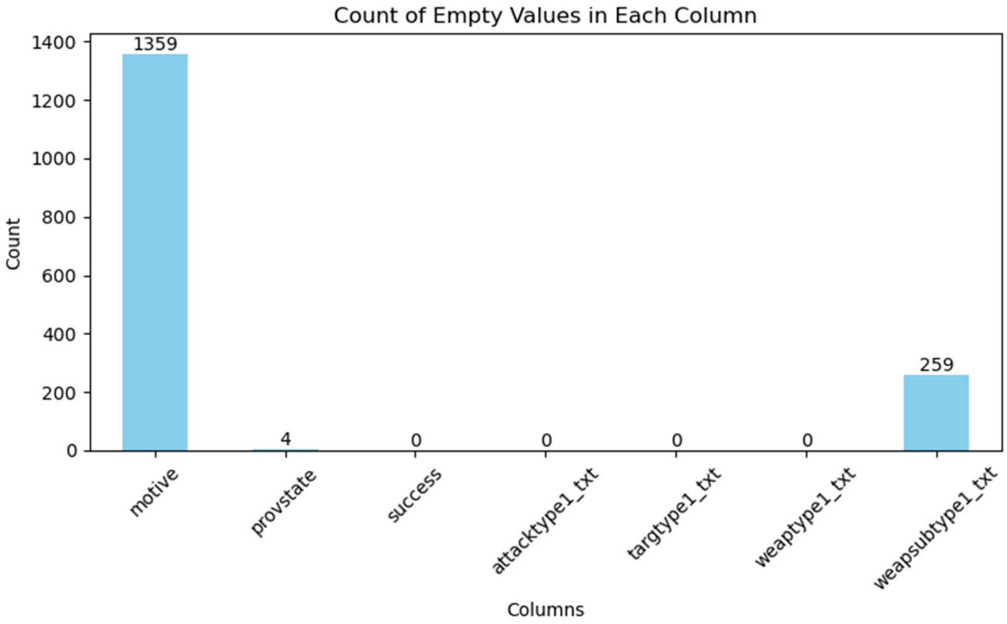
## Variable Understanding

# Data Preparation

The original global terrorism dataset size is has 135 columns and 181,691 entries. After filtering only the variables that we want to consider for the analysis and the model, and filtering out of the database only entries from the United States we end up with 7 columns and 2,836 entries.

## Dealing with missing values

Now, we calculate the empty values that may be on each column. We do this by using the function pd.DataFrame.isna().sum(). We show the results in the following bar chart:



From the resulting plot, motive has about 1,300 empty values, which is not a negligible amount and about half of the dataset, meaning that we can't delete these entries from the dataset. Therefore, we choose to fill in the missing values with "Unknown".

In contrast, we see that the variable "weaptype1_txt" is not empty while its more specific variation "weapsubtype1_txt" has about 300 empty values. Since the only difference between these two fields is that the first one provides a more general description than the second one, we will fill these empty values in "weapsybtype1_txt" with the values from "weaptype1_txt".

Finally, we also have to deal with the variable "provstate" which has 4 missing values, since this amount is very small when compared to the total dataset size, we choose to remove the entries that contain that variable as empty.

## Data Transformation

To make the variables compatible with their utilization in the models, it is necessary to convert all non-numeric variables. This section is dedicated to the conversion of textual and categorical data, followed by the merging of the outcomes from both procedures into a unified dataframe that is prepared for utilization in a machine learning model.

**1. TF-IDF Vectorization for "motive"**

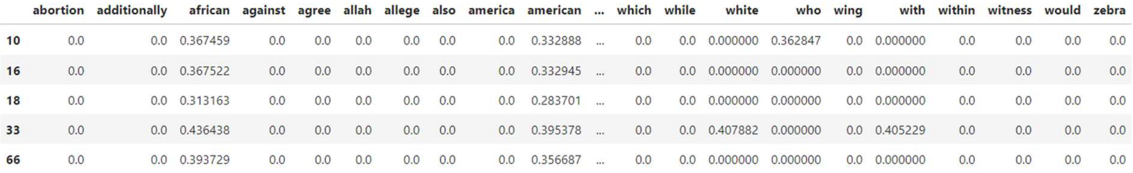
Among the dataset's variables, the "motive" feature stands out as it contains textual data rather than predefined categories. To effectively utilize this textual information, we employed TF-IDF vectorization.

TF-IDF, an acronym for "Term Frequency - Inverse Document Frequency," is a method used to assign numerical values to words in a collection of documents. It involves calculating a score for each word to indicate its significance within both a specific document and the entire corpus. This approach is a commonly employed technique in the fields of Information Retrieval and Text Mining (Scott, 2019).

Steps Taken:

- We utilized the scikit-learn library's TfidfVectorizer to perform TF-IDF vectorization.
- The max_features parameter was set to 200 to limit the number of features generated.
- TF-IDF assigns numerical values to each term within the "motive" column, reflecting their importance within the dataset.

The result is a transformed dataset that quantifies the relevance of words in the "motive" text data for subsequent analysis:

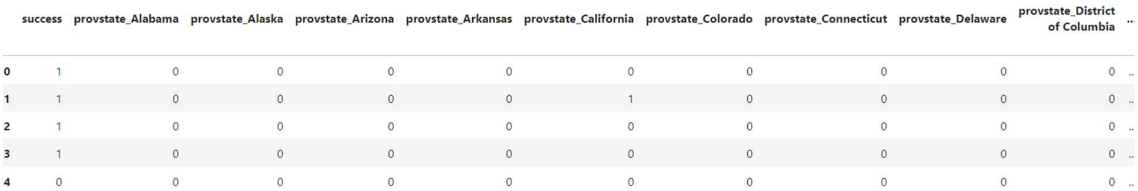


| | abortion | additionally | african | against | agree | allah | allege | also | america | american | ... | which | while | white | who | wing | with | within | witness | would | zebra |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **10** | 0.0 | 0.0 | 0.367459 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.332888 | ... | 0.0 | 0.0 | 0.000000 | 0.362847 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 |
| **16** | 0.0 | 0.0 | 0.367522 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.332945 | ... | 0.0 | 0.0 | 0.000000 | 0.000000 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 |
| **18** | 0.0 | 0.0 | 0.313163 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.283701 | ... | 0.0 | 0.0 | 0.000000 | 0.000000 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 |
| **33** | 0.0 | 0.0 | 0.436438 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.395378 | ... | 0.0 | 0.0 | 0.407882 | 0.000000 | 0.0 | 0.405229 | 0.0 | 0.0 | 0.0 | 0.0 |
| **66** | 0.0 | 0.0 | 0.393729 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.356687 | ... | 0.0 | 0.0 | 0.000000 | 0.000000 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 |

**2. Transformation for categorical variables (dummies)**

Initially, we identified the categorical variables in our dataset. These are features that have distinct categories, such as "provstate", "attacktype1_txt", "targtype1_txt", and "weapsubtype1_txt". The transformation is used using the get_dummies function provided by the Pandas library. This process transforms categorical attributes into binary columns (0 or 1), that represent the presence or absence of each category.

The outcome of this transformation is a new DataFrame named "df_encoded." It includes the one-hot encoded versions of the selected categorical variables, which are now represented by binary columns for each category:



| | success | provstate_Alabama | provstate_Alaska | provstate_Arizona | provstate_Arkansas | provstate_California | provstate_Colorado | provstate_Connecticut | provstate_Delaware | provstate_District of Columbia | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | .. |
| **1** | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | .. |
| **2** | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | .. |
| **3** | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | .. |
| **4** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | .. |

**Dimensions After Transformation:**

- The "wordsDF" DataFrame, which contains the TF-IDF-transformed textual data, has 200 features (columns).
- The "df_encoded" DataFrame, with the one-hot encoded categorical variables, includes 124 features (columns).

**Concatenating DataFrames:**

To consolidate all the transformed data, we merged the "wordsDF" and "df_encoded" DataFrames using the Pandas concat function. The resulting dataframe, named "fullUS," now encompasses both the textual data processed through TF-IDF and the one-hot encoded categorical attributes. After concatenating the dataframes, “fullUS” has 324 features in total.

# Modelling

## Separating in Train and Test

We start the modelling phase from our already created “fullUS” dataset. We separate this dataset into two different ones: “X_features” (which contains all the features, except for the target variable) and “y” (which only contains the “success” variable). We split it into 70-30, meaning that 70% of the data will go into the training and 30% will go into testing.

## Multinominal Naïve Bayes Model

The Naïve Bayes (NB) Model is a straightforward efficient model that works well for classification tasks. It makes decisions based on simple probabilities while assuming the varied factors are independent from each other. Naïve Bayes calculates the probability of an item being a particular class based on the probabilities of its features.

In our case we had to use the Multinominal variant of the NB model because it is specifically designed for text classification tasks. Multinominal NB counts how often words occur and creates a frequency table of the word occurrences. Multinominal distribution assumes that each word's occurrence is independent of the other words. It then uses smoothing techniques to handle unseen words which is why we thought it would be a good model to start out with.

## KNN (K Nearest Neighbor) Model

The KNN model is another simple and effective classification algorithm that can be used for both classification and regression problems. In our situation however, we are using it for classification. KNN operates on the idea that similar data points tend to be near each other, so it assigns the classification of a data point based on the majority class of its nearest neighbors. This model was one of the first we were introduced to data mining because it is easy to understand and implement.

The first thing we had to do was set the number of neighbors (2) and fit the KNN classifier to our training data to evaluate its performance on the test data. Once the classifier is fit and the predictions are made, the model calculates its accuracy. As we continued to explore our options, we kept experimenting with other models to see which of them would be the most effective.

## Random Forest Model

Random Forest is a powerful machine learning algorithm that is often used for both classification and regression. It works in the same way that one would decide by asking the opinions from multiple sources, and then making the final decision based on the majority, but with decision trees instead. We felt that the Random Forest algorithm would be a suitable choice for our problem as it is known for its versatility and ability to handle large and complex datasets.

This means that it would be suitable to analyze the various contributing factors to the success of a terror attack. With this model, however, it was crucial for us to take certain measures to prevent overfitting. If we did not take these steps, then it would be possible that our model would do well with data it has already seen but not generalize well to new unseen data.

Some of the parameters we had to adjust in order to prevent overfitting and achieve better results were the number of estimators, or decision trees (n_estimators), maximum depth of the trees (max_depth) and minimum samples per leaf (min_samples_leaf). After testing multiple options, we settled for the following configuration, which provided the best results from the test set:

- "n_estimators":100
- "max_depth": 55
- "min_samples_leaf": 1

### Gradient Boosting

Gradient Boosting is another popular and powerful machine learning algorithm that is often used for both classification and regression problems. It was another suitable choice because of its ability to deal with complex relationships within the data. Gradient Boosting builds a model step by step where each step focuses on making improvements. Each step the algorithm pays close attention to the data points where it did not predict correctly in the previous model. It focuses on learning from its mistakes to improve the overall prediction. It also makes sure to combine the predictions from the many weaker models to create a strong overall prediction. Each model's prediction is then weighted based on its performance and these weighted predictions are then combined to make the final prediction. Models that perform better are of course given more weight while those that perform bad are weighed less. In the end due to its intricate nature and its performance in comparison to the other previous models, this was the model that we in the end chose to use for the assignment.

Similar to Random Forest, we also made configurations to the parameters of the Gradient Boosting Model to achieve better results. The parameters we adjusted were the number of estimators (n_estimators), the learning_rate, which determines the "speed" the model learns from its previous errors (learning_rate) and maximum depth of the estimators (max_depth). The configuration that gave us the better results is the following:

- "n_estimators":100
- "learning_rate": 0.05
- "max_depth": 15

## Evaluation & Reporting

### Evaluation Metrics

In classification problems, various evaluation metrics help us understand how well a model performs. When dealing with binary classification, like our task of predicting successful or unsuccessful terrorist attacks, overall accuracy can be a helpful measure to assess how good the model is at predicting all the classes.

However, in our dataset, there's a challenge we need to consider—dataset imbalance. This means that a large portion (82%) of the data contains records of successful attacks, while the remaining (18%) belongs to unsuccessful attacks. While our primary focus is on predicting successful attacks, we also want our model to be precise and avoid wrongly classifying most attacks as successful.

Because of this, we can't solely rely on accuracy as our performance metric. Instead, we turn to the F1-Score, a metric that provides a balanced assessment for each class. The F1-Score combines two crucial aspects: "recall" and "precision".

Recall measures how accurately the model identifies instances of a specific class out of all the instances that truly belong to that class. It tells us how well the model captures the "success" class in our case.

Precision measures the percentage of correctly predicted instances of a certain class out of all the instances predicted as that class. It helps us understand the model's accuracy when it says an attack is "successful."

The F1-Score combines these two aspects into a single metric, allowing us to evaluate how well the model predicts each class. Ideally, we'd like to see a high and similar F1-Score for both classes. However, because of our imbalanced data, we expect that our models may struggle to accurately predict both classes and may favor the “successful attacks” class due to its higher representation in the dataset. Another criterion we can use for evaluation is to ensure that the overall accuracy of the model is greater than the percentage of successful attacks in the dataset, which is 82%. This would mean that the model is better at predicting the success of an attack than if it simply predicted everything as a success.

## K-Fold Cross Validation

As part of our model evaluation, we employ K-Fold Cross-Validation in addition to the results obtained from our test data. Cross-validation is a method that mimics how our model might behave when faced with new and unseen data. K-Fold Cross-Validation involves dividing the dataset into "K" subsets or folds. Each fold is utilized as a validation set while the remaining folds collectively constitute the training set. This process is repeated "K" times, and the outcomes from these iterations are then averaged to determine a final performance metric for our model, such as accuracy.

The adoption of this technique enhances the stability of our performance assessment. In our case, we have chosen to use "K=10," meaning we iterate through 10 different train-test cycles for each model. Consequently, we obtain the average accuracy of the results and the standard deviation, which informs us about the variation in these results. A small standard deviation indicates that our model's performance remains relatively consistent across different datasets, providing us with valuable insights into its reliability and generalization capacity.

## Naïve Bayes Evaluation Results

### Test Set Results

| Model | Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|---|
| **MN Naive Bayes** | 0 | 0.55 | 0.25 | 0.35 | 147 |
| **MN Naive Bayes** | 1 | 0.86 | 0.96 | 0.91 | 703 |
| **MN Naive Bayes** | Overall Accuracy | | | 0.84 | |

When we used this model originally, we did not produce the desired results and so we decided to continue to explore other models to determine if they would be a better fit for our problem. Only after we explored our further options and did some additional tweaking were we able to finally receive the above results. Our model has an .84 accuracy meaning that the model correctly predicted the right class of successful or unsuccessful for 84% of the samples we provided it. However, for us to judge these results properly it is also important for us to consider the different F1-Scores for each class. We consider that the F1-Score of 0.91 for predicting successful attacks (class 1) is very good, however this model is not very good at differentiating and predicting unsuccessful attacks (class 0) with a low F1-Score of 0.35.

### K-Fold Cross Validation Results

| Model | Metric | Value |
|---|---|---|
| **MN Naive Bayes** | Mean Accuracy | 0.811 |
| **MN Naive Bayes** | Standard Deviation | 0.023 |

After analyzing the results obtained from the test set, we go on to the evaluation of the results on the K-Fold Cross Validation (10 folds) which will give us an insight on if the model will perform as good when encountering new sets of data. The "Mean Accuracy" provides an average accuracy score across the 10 different iterations of cross-validation. In our case, it indicates that, on average, the Naive Bayes model correctly predicts the outcomes with an accuracy of approximately 81.1%. This lower than the previous obtained accuracy on the test set of 84%

The "Standard Deviation" measures the variation in accuracy scores across the 10 folds. In our case, the relatively small standard deviation of approximately 0.023 suggests that the model's accuracy remains consistent and reliable when confronted with different subsets of data. This consistency is a positive sign that the model performs fairly uniformly, regardless of the specific data it encounters.

## KNN Model Evaluation Results

### Test Set Results

| Model | Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|---|
| **KNN** | 0 | 0.33 | 0.59 | 0.42 | 147 |
| **KNN** | 1 | 0.90 | 0.76 | 0.82 | 703 |
| **KNN** | Overall Accuracy | | | 0.73 | |

After having now tried MN Naïve Bayes as well as the KNN model and receiving the above results we knew that there was room for improvement and so thought it would be worth giving another model an attempt. The results of the KNN model are worse than those of the MN NB model. This model has an accuracy of .73 meaning 73 % accuracy overall. When we compare the F1 Score of the KNN model we can see that the F1 score of class 0 is higher and the class 1 score is lower than those of the MN NB model. This means that the KNN Model did better at differentiating and predicting unsuccessful attacks (class 0) but worse in predicting successful attacks (class 1) we want a model that does well in both without sacrificing the other.

### K-Fold Cross Validation Results

| Model | Metric | Value |
|---|---|---|
| **KNN** | Mean Accuracy | 0.706 |
| **KNN** | Standard Deviation | 0.037 |

Going on to the evaluation of the model with 10-Fold Validation, we encounter that the "Mean Accuracy" indicates that on average the KNN model correctly predicts the outcomes with an accuracy of 70.6%. This performance is not far apart from the accuracy obtained in the test set of 73%; however, it is still exceptionally low in comparison to the results obtained with the Multinomial Naïve Bayes Model.

The "Standard Deviation" is approximately 0.037, meaning that the model is fairly consistent across different datasets.

## Random Forest Evaluation Results

### Test Set Results

| Model | Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|---|
| **Random Forest** | 0 | 0.62 | 0.31 | 0.41 | 147 |
| **Random Forest** | 1 | 0.87 | 0.96 | 0.91 | 703 |
| **Random Forest** | Overall Accuracy | | | 0.85 | |

As we continue to attempt to find the most effective model, we now come to see the results from the random forest model. We can immediately see that the overall accuracy is 85%, which is of course higher than the scores we received using the KNN model. If we take a closer look, we can see that this is because this model does better in classifying both successful (class 1) and unsuccessful (class 0). With high precision in both classes the recall of the unsuccessful class is reason for the low F1 score. This model overall is quite effective for only identifying successful attacks, but this is not good enough. This is, however, enough to prove a good indicator to us that we are getting closer to our most desired model.

## K-Fold Cross Validation Results

| Model | Metric | Value |
|---|---|---|
| **Random Forest** | Mean Accuracy | 0.821 |
| **Random Forest** | Standard Deviation | 0.021 |

Proceeding with the evaluation of the k-folding of the Random Forest model, we get that the "Mean Accuracy" indicates that on average the Random Forest model correctly predicts the outcomes with an accuracy of approximately 82.1%.  This accuracy is also not quite different from the test set accuracy of 85% and could be considered acceptable. Additionally, it's better than the average accuracy obtained from the Multinomial Naïve Bayes model, meaning that the model performs better with unseen data.

The "Standard Deviation" is approximately 0.021. This is even lower than the standard deviation obtained with the Multinomial Naïve Bayes Model, meaning that Random Forest must be more consistent when encountering different datasets.

## Gradient Boosting Evaluation Results

| Model | Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|---|
| **Gradient Boosting** | 0 | 0.64 | 0.33 | 0.43 | 147 |
| **Gradient Boosting** | 1 | 0.87 | 0.96 | 0.91 | 703 |
| **Gradient Boosting** | Overall Accuracy | | | 0.85 | |

With an overall accuracy of 85% this Gradient Boosting model has proven to be highly effective in making accurate predictions for both successful and unsuccessful attacks. Both the precision for the successful and unsuccessful attacks are high meaning that the model predicts an attack as unsuccessful accurately 64 % of the time and correctly successful 87%. Now looking at the recall of 33% for unsuccessful attacks means that the model identifies 33% of the actual unsuccessful attacks and 91% of the actual successful attacks. After going though those scores for more perspective we arrive at the F1 scores which provide a balanced measure of the models' accuracy in identifying successful and unsuccessful attacks. The F1 Score of 0.43 helps ensure that the model effectively minimizes both false positives and false negatives. The F1 score of 0.91 suggests a high balance between precision and recall which demonstrates our model's ability to accurately identify potential successful attacks.

This model's minimization of false positives would allow law enforcement to allocate resources efficiently and effectively help them protect the country from such attacks. Given how serious in nature our problem is we believe that the Gradient Boosting model's performance is the best as it shows us how well it can make accurate predictions.

## K-Fold Cross Validation Results

| Model | Metric | Value |
|---|---|---|
| **Gradient Boosting** | Mean Accuracy | 0.815 |
| **Gradient Boosting** | Standard Deviation | 0.025 |

Now, regarding the results of the Cross Validation, the "Mean Accuracy" indicates that on average the Gradient Boosting Model correctly predicts the outcomes with an accuracy of approximately 81.5%.  This average accuracy is not as good as the one obtained from Random Forest, but it is also not very far apart. Meaning that this model also performs well when encountering different datasets.

The "Standard Deviation" is approximately 0.025. This is also very close to the one obtained in the Random Forest, meaning their consistency across different datasets is very similar.

Considering both the K-Folding Cross-Validation results and the test set F1 Scores obtained, we consider that as well as providing an effective distinction between unsuccessful and successful attacks the model would also perform very consistently if law enforcement chose to apply it to new unseen data.

# Personal Reflections

## Antonella Portugal

At the beginning I found the course a little confusing as I believe we had very little context to start on. However, after progressing more in all the different classes I believe it gave us the necessary tools to complete this assignment. I have to admit that I do have some previous experience working on assignments like this, so I found myself wanting to add a bit of a challenge. I think this came in the form of using both categorical variables and textual variables to arrive at results, as I have not really worked with TF-IDF much before. I believe the use of both types of variables added to the complexity of the development of the model. I thought it was exciting that our models had very good results, even one that I had not used previously and thought wouldn’t give very good results which was MN Naïve Bayes. Another thing I’d like to mention is that previous to my enrolment in the minor I didn’t know about the CRISP-DM model, I knew how to apply different parts of it on their own, but I had not seen in it in an structured form until now, and it has made my workflow much more organized and even for myself to understand better what I am doing. I’m glad I had the opportunity to apply this framework and definitely will use it in my professional career, which is very much in Data Science roles.

## Haley Raynor

I found this assignment particularly challenging because this class was the first class that we had during the minor. I personally found myself a little behind during the lessons and would fully understand by the end of the week as we had completed the rest of the lessons. As I originally understood it, data mining was about taking data and ‘mining’ it for understanding. I had little to no knowledge of python, data models and the combination of using them to solve real world business problems. I can now confidently say that I have a more in-depth understanding of the concept and practice of data mining and what it entails. Data mining is about searching for valuable information, patterns, and insights from large sets of data. Then using these insights and information to make better decisions. In the beginning I found it easy to understand the dataset and the problem that we were attempting to solve. Using USA terrorist crime data to help make a model that would allow the FBI to predict the success of terror attacks based on variables like weapon and motive etc. I was lucky to have a partner like Antonella who was always very patient with me and always open and willing to answer any questions/ provide explanations. During this assignment's completion, I struggled most with the data preparation stage and the modelling stage. The modelling stage was difficult for me as I found it difficult in the beginning because I did not understand the models themselves. Once I understood them better it became clearer as to how and why they should/could be applicable to our problem. My favorite model which also happened to be one of our most effective was the Random Forest model; it became my favorite because I was able to easily understand and relate the model to our assignment. Random Forest can handle large datasets because it acts the same way as a large group of mixed and diverse people sharing their opinions on, in our case, predicting successful terror attacks. The area that I would say that I had the most trouble with was the data preparation phase, however, but that had less to do with the reasoning and more the lengths and steps that we had to take to make the data ready to use for modelling. I have found that the information that I have learned from data mining let alone the entire data driven minor can defiantly be used in my career going forward. I plan to become a change consultant back in Bermuda. It is also crucial that I also have other complementarity skills that I could use in consulting.

# References

National Consortium for the Study of Terrorism and Responses to Terrorism | University of Maryland. (2021, August 15). Global Terrorism Database Codebook. College Park, Maryland, United States of America.

Scott, W. (2019). *TF-IDF from scratch in python on a real-world dataset. | Towards Data Science*. https://towardsdatascience.com/tf-idf-for-document-ranking-from-scratch-in-python-on-real-world-dataset-796d339a4089