CLASSIFICATION ALGORITHMS FOR TERRORISM PREDICTION IN MIDDLE EAST AND NORTH AFRICA

V. Fokker, T.J.C. Meulenbroek, K. Raijmann, Ruud. Warmels

1. Introduction

Terrorism is defined as the threatened or actual use of illegal force and violence by a non-state actor to attain a political, economic, religious, or social goal through fear, coercion, or intimidation (GTD Codebook, 2018). Since 1970, the Study of Terrorism and Responses to Terrorism, from now on referenced to as START, has been collecting data regarding terrorism. This data has all been collected in the Global Terrorism Database, from now on referenced to as GTD. For an incident to be defined as a terrorist attack, the following attributes need to be present: the incident must be intentional, the incident must entail some level of violence or immediate threat of violence and the perpetrators of the incident must be sub-national (GTD Codebook, 2018). In 2017 alone, 10.900 terrorist attacks were reported in the whole world, causing 26.400 people to die of which 10.819 in the Middle East and North Africa. (START GTD Overview, 2018).

In 2015, Soliman et al used data from the GTD for terrorism prediction in the Middle East and North Africa using hybrid machine learning algorithms and published a detailed comparison among the used hybrid methods and other standard, and ensemble models (Soliman et al, 2015). Machine Learning is the process of estimating unknown dependencies of structures in a system using a limited number of observations, which algorithms are used to retrieve hidden information and used in decision making (Bonnaccorsi, 1992). Machine learning techniques can be combined as proposed to improve the performance of individual machine learning algorithms (Soliman et al, 2015). These hybrid machine learning techniques combine and integrate different or heterogeneous machine learning techniques and decision making models and may considerably improve quality of reasoning and boost adaptivity of the entire solutions (Miškovic, 2014).

The study by Soliman et al (2015) is based on predicting terrorist groups responsible of attacks in Middle East and North Africa from year 2009 up to 2013 by comparing various standard, ensemble, hybrid, and hybrid ensemble machine learning methods namely; Naïve Bayes, K-Nearest Neighbours, Decision Tree, Support Vector Machine; Hybrid Hoeffding Tree, Functional Tree, Hybrid Naïve Bayes with Decision Table, Classification via Clustering; Random Forests; and Stacking classifiers (Soliman et al, 2015). Based on the techniques used in the study by Soliman et al (2015), this paper will conduct a similar research using data from the GTD gathered between 2013 and 2016. This research will use a selection of machine learning techniques namely; Naïve Bayes, K-Nearest Neighbours, Decision Tree, and Support Vector Machines to predict the terrorist groups behind terrorist attacks in the Middle East and North Africa and compare among the results of the used methods.

This research will demonstrate that various machine learning techniques are useful in predicting terrorism groups by the hand of several attributes. Unfortunately, due to the decisions that had to be made during the pre-processing, not all machine learning techniques used in this research could be proven to aid the war on terror.

2. Problem Definition and Algorithm

2.1 Task Definition

Besides similarities with the study of Soliman et al (2015) there are some important differences which makes this study relevant as a novel research. In the more recent data used in this study gathered between 2013 and 2016, some changes have been made with regard to the data used in the study of Soliman et al (2015). Since then, country and nationality variables were updated, target and weapon subtypes were adjusted and geo-codes were added for the entire dataset, weapon types unaffiliated individuals variable were added and removed from gname, and also number of wounded US was added as number of wounded was influenced by US statistics due to 9/11 (GTD Codebook, 2018). These changes in the dataset could have a significant impact of the results of this study.

Another problem with the study of Soliman et al (2015) is that the lack of describing the implementation of the distinct machine learning techniques used in the research, which makes it impossible to replicate the study. The aim of this research is to precisely describe the implementation and results of the distinct machine learning techniques used in the research so that the comparison between these machine learning techniques will be clearer for future research.

2.2 Algorithm Definition

2.2.1 Decision Tree

Decision Trees (DTs) are supervised learning method used for classification and regression. DTs are non-parametric; hence data is not required to fit into a normal distribution (Breiman, 2001). The main aim is to create a model that is able to predict the value of a target variable by learning simple decision rules, which are derived from the data features. DTs break a data set down into smaller and smaller subsets while at the same time the development of an associated decision tree is incrementally developed. DTs consist of multiple decision nodes, which have two or more branches. The leaf node is representing a classification or decision. A metric that is used to measure the impurity of the attributes is called entropy. The decrease of entropy after a dataset is split on the basis of an attribute is called information gain. The higher the information gain, the lower the entropy, hence the lower the randomness of the data (Donges, 2018).

The DT places the best attribute of the dataset at the root of the tree. Then the training set gets split into subsets, these subsets should be created in such a way that each subset contains data with the same value for an attribute. These steps are repeated until leaf nodes exist in all the branches of the tree (Breiman, Friedman, Olshen, and Stone, 1984). Some algorithms used for DTs are ID3, CART, and CHAID.

To illustrate the algorithm, a simple example is given below. Damien is going to plan a city trip in the Netherlands. In order to decide which cities to visit, he asks a friend, who knows him, to give him advice. Before that friend gives him advice, questions are asked to Damien about previous city trips. Based on those answers, his friend will

give him advice. This is a DTs algorithm approach. Rules to guide the decision are made by Damien's friend by using the answers of Damien.

Some advantages with DTs are simple to understand, little data preparation is necessary, both numerical and categorical data can be handles, and visualizations can be made. A major disadvantage is overfitting. Overfitting can be dealt with by pruning by maximum leaf nodes, setting a maximum depth, or setting a minimum number of samples required at the leaf node.

2.2.2 Random Forests

Random forests (RFs) are an ensemble learning method for tasks such as classification and regression. The algorithm for RFs is a supervised learning algorithm. RFs operate by constructing multiple decision trees at training time and output the class that appears most often (Breiman, 2001). While growing the trees, randomness is added to the trees. Instead of searching for the most important feature while splitting nodes, it searches for the best feature in a random subset of features. This provides a wide diversity that generally results in a better model. Hence, only a random subset of the features is taken into consideration for splitting a node by the algorithm.

An advantage of RFs is the ability to correct the DTs' occurrence of overfitting to the trainings set. RFs perform in general good on large datasets, since they create accurate predictions using multiple DTs (Donges, 2018).

According to Gu (2017) The pseudocode of the RFs consists of the following steps:

1. Randomly select 'f' features from total 'm' features, where f << m;
2. Amidst the 'f' features, the node 'd' is calculated using the best split point;
3. Split the node into child nodes using the best split;
4. Repeat step one to three until 'l' number of nodes have been reached;
5. Build forest by repeating the steps above for 'n' numbers of time, which creates 'n' number of trees.

The random forest classifiers being created in the above code, predictions can be made with the following pseudocode:

1. Take test features and utilize the rules of each randomly created decision tree to predict an outcome and store the predicted outcome;
2. Calculate the 'votes' for each predicted target;
3. The highest voted predicted target is the final prediction from the random forest algorithm.

To illustrate the pseudocode, an example is given. Consider the example given in the description of the decision tree. After Damien received advice from that one friend, he is going to ask more of his friends for advice. All these friends will ask him different questions from which the advice is derived. At the end, having received multiple recommendations, Damien makes his decision by choosing the places that were recommended the most.

2.2.3 Naive Bayes

Naive Bayes is a form of design to construct classifiers using (over)simplified assumptions according to Bayes Theorem, which is a theorem that describes the probability of an event, given prior knowledge related to that event. In naive bayes the features have strong (naive) independence assumptions between each other. The models that are build with naive bayes assume that each feature on its own is independent of other features, given the class variable. Possible correlations between the features are not included. Despite the simplified setup of the naive bayes design, the classifiers have shown to work quite well in complex situations.
Naive Bayes modeling is done by conditional probability modeling. An problem instance is classified as a vector, In which the amount of features (independent variables), are assigned to a certain probability. Using bayes theorem you then get that the posterior probability is the prior probability times the likelihood, divided by the evidence. For Naive Bayes, one have to take into account that each feature is conditionally independent of every other feature given the class.

With Naive Bayes one tests the probabilities of the different features, and calculate

posterior probabilities for the different outcomes, in our case terrorist groups. The group with the highest posterior probability will be classified as the responsible group. This is a form of supervised learning. For example: when you are assessing whether to go for a run, depends on a certain amount of features; what temperature it is, if it is raining or not and if it is windy. These features together define if you are going for a run or not, but they can also independently influence the likelihood of going. If it is raining the likelihood decreases, independent of the other features. From previous instances; you can calculate the likelihood of running in different scenarios, with different features included. Also the probability of the features given the choice , go running or not, can be calculated. From these probabilities new scenarios can be calculated, defining the final choice of the model. Naive Bayes models are a good fit for unbalanced data.

Multiple types of Naive Bayes models are available: Multinomial, Bernoulli and Gaussian Naive Bayes. Multinomial is used when your data describe discrete frequency counts, such as word count for example. Bernoulli is good for making predictions from binary features, such as man versus woman and Gaussian is good for making predictions about data that is normally distributed in the features.

2.2.4 K- Nearest Neighbours

K-Nearest Neighbors (KNN) is an supervised method of classifying a feature vector in feature space. This is done by calculating the Euclidean distance between the nearest features. The output is a classification based on its k nearest neighbors, in which k is an positive integer. Training of a KNN is done in a multidimensional feature space representing a class label. During this training phase, feature vectors are stored use in the classification phase. During the classification phase, an unlabeled vector is classified according to the most frequent classification labels in the range of the k nearest training samples to that point. The value of k can be tuned and depends on the data. Higher values of k reduce noise in the classification, but also make the boundaries between classes less distinct. Example; if there are 3 point of class A in feature space, and 3 points of class B in feature space, and the three points closest to the unlabeled vector X are A,A,B, and $k=3$ the classifier will classify X as A. Instead

if *k=5* and the next closes points beside A,A,B, are B,B, making A,A,B,B,B the classifier will classify X as B because B is more frequent.

2.2.5 Support Vector Machine (SVM)

Support vector machines (also, support vector networks) is a means of supervised learning algorithms used for classification and regression analysis. The theory behind the SVM is based on the geometric properties of all instances, where the instances are vectors. Therefore you are able to plot all the vectors in a place and calculate the best hyperplane between the vectors.
This is also the goal of SVM: to design a hyperplane that classifies all training vectors in a number of classes. The best hyperplane leaves the maximum margin from the classes. The maximum margin is the shortest distance between the hyperplane and the closest elements.

An important consequence of the geometric description is that the maximum margin hyperplane is completely determined by the group of points which lie nearest to the hyperplane. Therefore the SVM is notable more efficient and faster as other algorithms. Also, the SVM always points to one single point, the minimum. Support Vector Machine is considered to be good for large feature sets.

For example, the points can be plotted for two classes a and b in a plane. They are plotted on a Cartesian plane as can be found in the figure 1 below. Next the best hyperplane that has the biggest margin between the support vectors of the two groups based on the two features (X1 and X2) can be found. The hyperplane is the best way to classify the two groups.

When introducing more groups, the classification becomes more difficult. The different groups can have overlap when looking at certain features. Now there is no clear dividing margin with which the different groups can be separated. This is where the Kernel trick can be used. The Kernel trick is, in short, an algebraic means to transform the data in another dimension so that there will be a margin between the different groups so that the groups can be better separated.

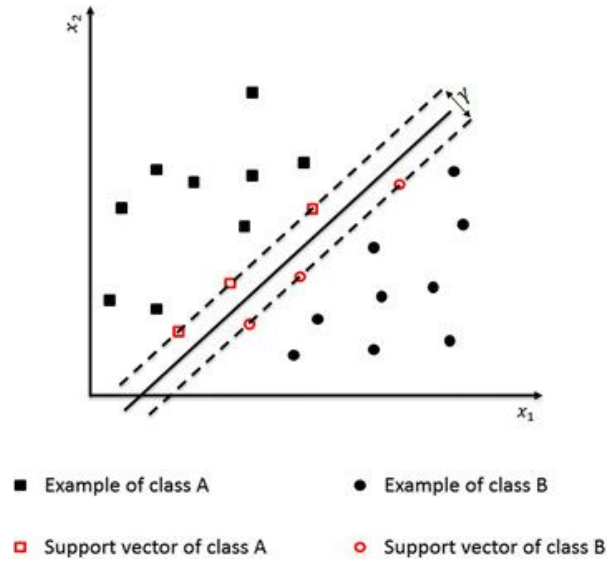

Fig. 1: Support vector machine

3. Experimental Evaluation

3.1 Methodology

The focus in this research was on the question: what standard supervised machine learning techniques aid in classifying terrorist groups responsible for terrorist attacks in the Middle East and North Africa based on open source intelligence?

This question will be answered by comparing five different algorithms: Decision Trees, Support Vector Machines, Naive Bayes, Random Forests and K-means. If the chosen supervised model classifies the terrorist groups with an accuracy above 80% and the method is reasonably applicable to our problem, then the conclusion can be made that this technique aids in classifying terrorist groups. All Algorithms used in this research will be discussed below.

3.1.0 Dealing with unbalanced data

To account for unbalanced data, a form of balancing needs to be applied. The Scikit-learn kit imbalanced-learn provides us with the possibility to apply multiple undersampling or oversampling methods for the different algorithms. For exploratory purposes we include 3 types of undersampling using a pipeline; Random

undersampling, which is one of the most simple undersampling methods, instance hardness threshold undersampling, which removes the samples with lower probabilities and condensed nearest neighbours undersampling, which looks at the 1 nearest neighbor to iteratively decide of a sample should be removed or not. The Pipelines for each of the models will be applied during model fitting.

3.1.1 Naive Bayes

For the Naive Bayes analysis, four available models provided by the Scikit learn are fitted on the data. The Gaussian classifier is most straight forward. The assumption must be that the features are normally distributed. To be certain that this classifier can be used, additional checks on normality of the features need to be done. For the check on normality, the stats functions from Scipy will be used.

The multinomial is also quite straightforward. The assumption for the data for this model is that the features describe discrete frequency counts. Five of the variables used are consistent with this assumption.

The Bernoulli model is a model that is fit for binary data. When the categorical variables in the dataset are binerized, this model would suit best when binarization value is set to True. The binarize value can be tuned using a simple loop that loops over the different binarize settings in a range from 0 to 2. If the accuracy measure is higher with a value that is above 1, binarization set to “True” should be used for this model.

An alternative way of using the Naive bayes classification is by looking at the complement Naive Bayes. The complement Naive bayes assumes that the object also occurs in the other classes next to what it belongs to, and classify the object to the group it least belongs to. With complement Naive Bayes one needs to worry less about skewed data, as it handles data skews better.
After model fitting the classification report is printed, to report the precision, recall and f1-measure. Also a confusion matrix is printed to give an idea of the correctly classified instances.

3.1.2 K-Nearest Neighbors (KNN)

The KNN models are made with the Scikit Learn environment. KNN models can be tuned in the *k* value. With an increase in *k* value, the noise in the classification can be reduced, but the boundaries between the classes can decrease. Therefore a tradeoff needs to be made to define the best k value. The k value is tuned by calculating the Mean Error for each k value and looking at the most favorable value. The Mean Error and k value tradeoff for this model is best at a k value of 3.

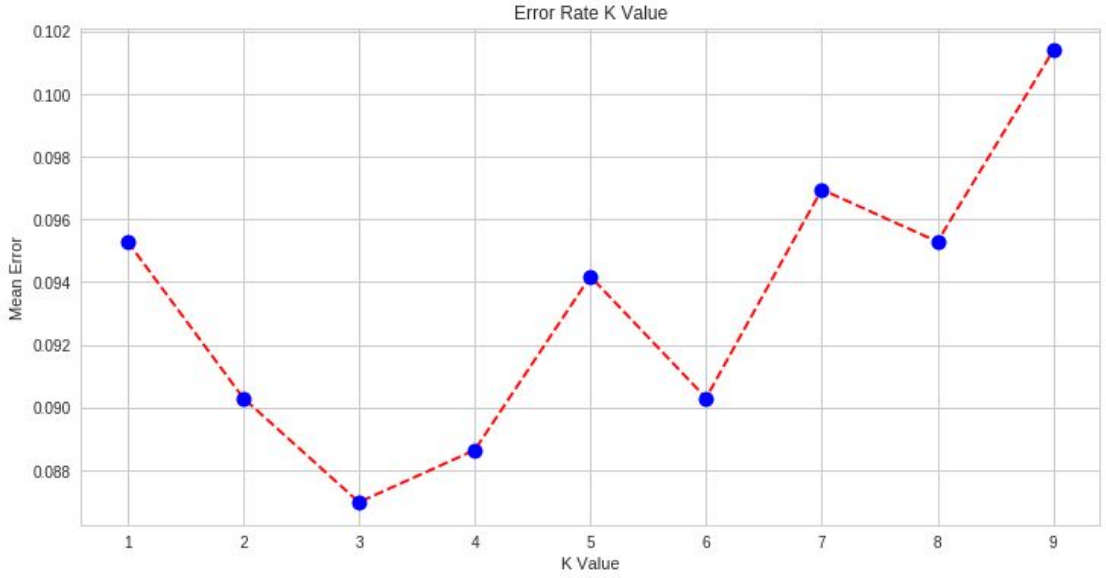

Fig 2. Mean Error plotted against the k value for the KNN model. Best tradeoff is at k =3.

3.1.3 Decision Trees (DTs)

The DTs are created using Scikit Learn. Scikit learn provides simple machine learning tools for data mining and data analysis. The DT tool, provided by Scikit Learn, uses an improved version of the CART algorithm. The Iterative Dichotomiser 3 makes use of a greedy strategy which selects the locally best attribute to split the dataset on each iteration. The CART algorithm creates binary trees using the feature and threshold which yield the largest information gain at each node. Hence, it aims to maximize the information gain and minimize the entropy. DTs have the tendency to overfit. Overfitting is an error of modelling which occurs when a function is too closely fit to a limited set of data points. Overfitting a model takes the form of making an overly complex model to explain behaviour of the data. Mechanisms to mend with

overfitting are pruning, setting a minimum number of samples, and implementing a maximum depth of the tree (Dietterich, 1995). In this research, these three mechanisms have been used to increase the accuracy of the DTs. To map the scores of the DTs, learning curves have been plotted and confusion matrices have been drawn. If, from the learning curves, the event of overfitting is discovered, mechanisms to deal with overfitting are applied to the DTs code.

3.1.4 Random Forests (RFs)

RFs are created by using tools from Scikit Learn. The RFs are included in this research in order to check if they perform better than DTs. An advantage, opposing to DTs, RFs perform generally better with big data sets, since RFs are composed of multiple trees. In order to limit the length of this research, no attempts are made to increase the performance of the RFs. The performance is measured by the use of a learning curve and a confusion matrix, both derived via Python.

3.1.5 Support Vector Machine (SVM)

Support Vector Machine is created using the SKLearn package in Python. This research only focussed on the Gaussian Distribution (RBF) with random state 42. The C value, which 'trades off correct classification of training examples against maximization of the decision function's margin.'(scikit-learn, 2018), has a value of [0.001, 0.01, 0.1, 1, 10], where the model uses 10 as the best value for C. This means that the model accepts a smaller margin while fitting the training set. Results are given in a confusion matrix and metrics report.

3.1.6 Problem Statement and hypothesis:

This research aims to answer the question: What standard supervised machine learning techniques aid in classifying terrorist groups responsible for terrorist attacks in the Middle East and North Africa based on open source intelligence?

This question is answered for every machine learning technique used in this research. To clearly answer this, the following question must be answered: Which features are most important classifying. These questions lead to the hypothesis:

If our chosen supervised model classifies the terrorist groups with an accuracy above 80% and the method is reasonably applicable to our problem, then the conclusion can be made that this technique aids in classifying terrorist groups.

3.2 Results

3.2.1 Pre-processing

Data from the Global Terrorism Database from the period 2009-2017 was used. The following variables were included in the dataset: 'iyear', 'imonth', 'gname', 'latitude', 'longitude', 'attacktype1', 'weaptype1', 'targtype1', 'nkill' and 'nwound'. All instances with missing values for 'gname', 'lattitude', 'longitude', 'nkill' and 'nwound' were dropped. This resulted in 10832 entries in the dataset.

In addition to the dataset the population for the locations, based on latitude and longitude were included in the dataset. This information is obtained with an API from Geonames (2018). This resulted in an two extra attributes: population and environment (rural/urban).

Fig. 3: Heatmap of the terrorist attacks

For the target variable, the terrorist groups, inactive terrorist groups were omitted. The assumption was made that a terrorist group which is inactive since 2013 can be considered a inactive terrorist group and therefore not included in our research.

The dataset consists of lots of small terrorist groups, while it is not possible to classify a terrorist group based on little data, the smaller terrorist groups are combined in one 'other' group. While the biggest four terrorist groups together are responsible for 68% of all terrorist attacks. These 4 groups and the 'other' group, the fifth group, are the target variables.

Table 1: Target variable: terrorist groups
Islamic State of Iraq and the Levant (ISIL)
Kurdistan Workers' Party (PKK)
Al-Qaida in the Arabian Peninsula (AQAP)
Houthi extremists (Ansar Allah)
Other

3.2.2 Decision Tree

From the learning curve, displayed in figure x, can be derived that the decision tree model is overfitting since the training score is almost equal to 1, hence the maximum. Mechanisms against overfitting are to be applied.

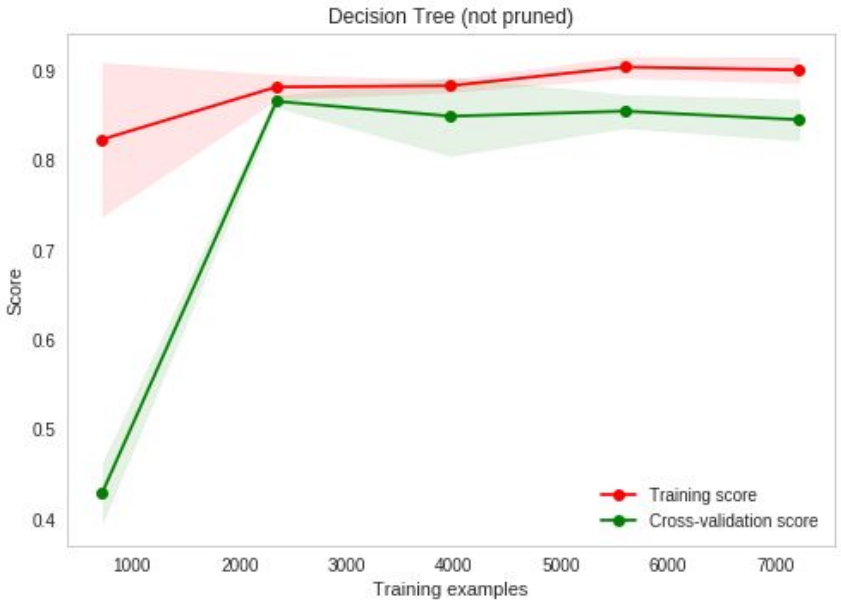

Fig. 4: Learning Curve Decision Tree (not pruned)

Table 2: Decision Tree models

Decision Tree		**Accuracy and performance measure**				
		corre. class.	**incorr. class.**	**precision**	**Recall**	**F-Measure**
Decision tree (no mechanisms applied)	Random undersampling	91,25%	8,75%	0.91	0.91	0.91
	Instance Hardness Threshold	80.31%	19.69%	0.84	0.91	0.86
	Condensed nearest Neighbour	40,22%	59,88%	0.41	0.40	0.37
Decision Tree (max leaf nodes = 10)	Random undersampling	89,70%	11,30%	0.90	0.90	0.90
	Instance Hardness Threshold	87.48%	12.52%	0.89	0.87	0.88
	Condensed nearest Neighbour	41.39%	58.61%	0.28	0.41	0.32
Decision Tree (max depth = 5)	Random undersampling	90.42%	9.58%	0.91	0.90	0.91
	Instance Hardness Threshold	87.53%	12.47%	0.89	0.88	0.88
	Condensed nearest Neighbour	35.79%	64.21%	0.35	0.36	0.33
Decision Tree (min sample leafs = 5)	Random undersampling	90.42%	9.58%	0.91	0.90	0.91
	Instance Hardness Threshold	87.53%	12.47%	0.89	0.88	0.88
	Condensed nearest Neighbour	38.06%	61.94%	0.35	0.36	0.33

From table 2 can be derived that decision tree and decision trees with mechanisms, random undersampling provides the best results. For example, the precision rate of all DTs models is either 0.9 or 0.91. The model with the highest accuracy score is the decision tree where no mechanisms are applied to. The Condensed nearest Neighbour provides in all cases the worst results, since all rates are below 50 percent.

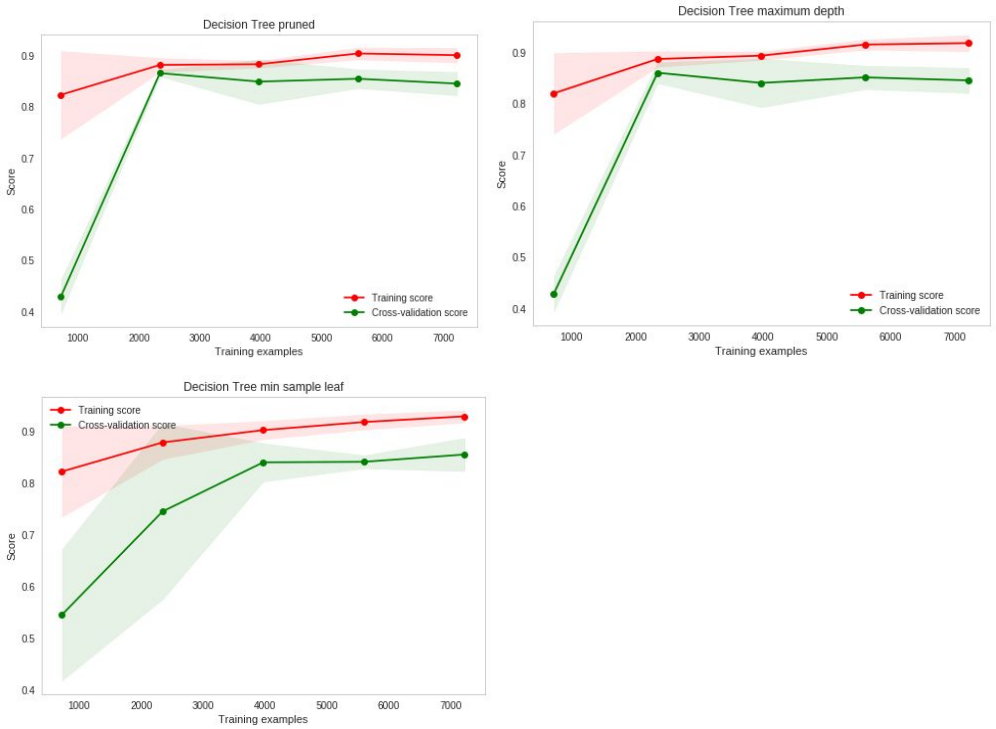

Fig. 5: Learning Curves of the Decision Trees with Mechanisms for overfitting

In figure x are the learning curves of all decision tree models displayed. From the plots can be derived that all tend to overfit. The model with the best plot is the decision tree where the mechanism of pruning is applied, since the lines intersect at a high score.

3.2.3 Random Forest

The random forest model has a precision of 0.95, which is derived from table 3. The correct classification class is 94,47%.

Table 3: Random Forest

Random Forest	**Accuracy and performance measure**				
	corre. class.	**incorr. class.**	**precision**	**Recall**	**F-Measure**
Random undersampling	**95.01%**	**4.99%**	**0.95**	**0.95**	**0.95**
Insurance Hardness Threshold	**95.01%**	**4.99%**	**0.95**	**0.95**	**0.95**
Condensed nearest Neighbour	**95.01%**	**4.99%**	**0.95**	**0.95**	**0.95**

The rates for every undersampling method are the same for the random forest. The precision rate is the highest of all models, with a rate of 0.95.

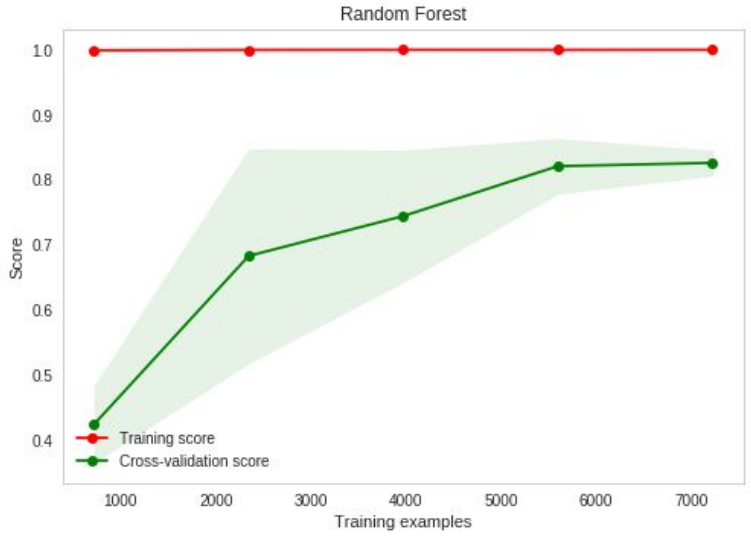

Fig. 6: Learning Curve Random Forest

The learning curve of the random forest in figure x shows that the training score is exactly equal to 1. The cross-validation score is increasing up to 5500 training examples, afterwards it seems to be almost constant.

3.2.4 Naive Bayes

The Gaussian Naive Bayes classifier classifies with the highest precision and recall with correct classification of 83,05% and a F-measure of 0.84. The other Naive Bayes models are all not great models with precision below 0.50. For BernoulliNB, the binarize value is set to false, as this gives higher performance on accuracy and precision.

Table 4: Naive Bayes

Naive Bayes		**Accuracy and performance measure**				
		corre. class.	**incorr. class.**	**precision**	**Recall**	**F-Measure**
gaussianNB	Random undersampling	83.05%	26.95%	0.87	0.83	0.84
	Instance Hardness Threshold	80.33%	19.67%	0.85	0.80	0.81
	Condensed nearest Neighbour	48.25%	51.75%	0.37	0.48	0.37
Multinominal NB	Random undersampling	37.12%	62.88%	0.70	0.37	0.41
	Instance Hardness Threshold	35.40%	65.60%	0.64	0.35	0.39

	Condensed nearest Neighbour	26.81%	73.19%	0.39	0.27	0.27
BernoulliNB	Random undersampling	46.43%	53.57%	0.56	0.46	0.48
	Instance Hardness Threshold	39.17%	60.83%	0.55	0.39	0.41
	Condensed nearest Neighbour	16.07%	83.93%	0.25	0.16	0.11

For the usage of the Gaussian Naive Bayes model, the assumption that the features are normally distributed need to be tested as well. None of the feature seem to be normally distributed, all P values are 0. Normality rejected in all cases. (figure 7)

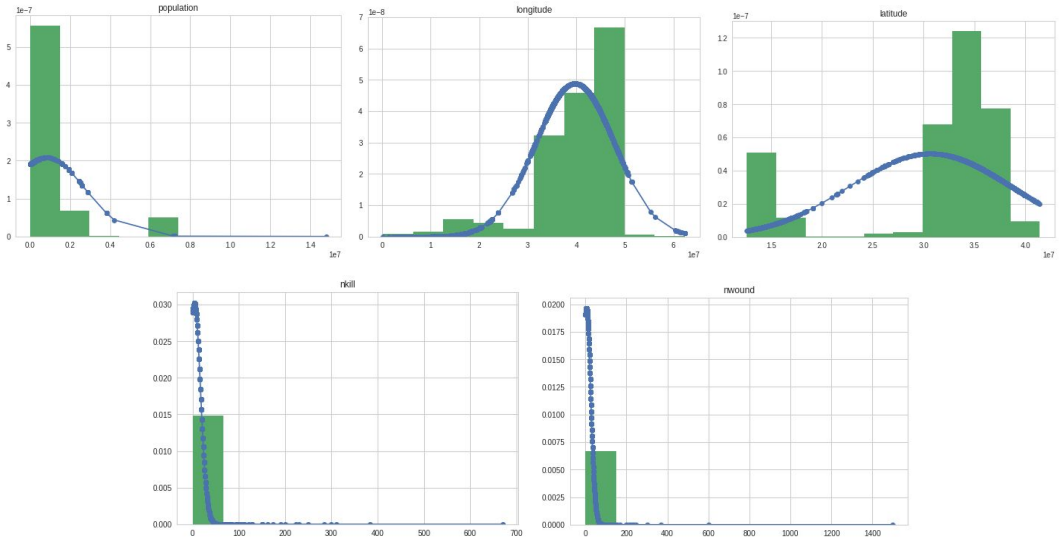

Figure 7: Distributions of the features “population”, “longitude”, “latitude” , “nkill” and “nwound”. These features are most important according to the feature importance.

3.2.5 K -Nearest Neighbors (KNN)

The KNN classifier performs better then the Naive Bayes classifier. The precision is 0.92 and the recall 0.91 making an F-measure of 0.91. (table 5).

Table 5: K-Nearest Neighbours

K- nearest Neighbors	**Accuracy and performance measure**				
	corre. class.	**incorr. class.**	**precision**	**Recall**	**F-Measure**
Random undersampling	91.30%	8.70%	0.92	0.91	0.91
Instance Hardness Threshold	88.59%	11.41%	0.90	0.89	0.89
Condensed nearest Neighbour	67.09%	33.91%	0.62	0.67	0.63

3.2.6 Support Vector Machine

The SVM classifier performed poorly. These are the result for a SVM with the Gaussian Kernel. The accuracy is 58%, recall 58% and the F-measure 0.58 for random undersampling. Especially Recall is low in this model.

Table 6: Support Vector Machine

Support Vector Machine	**Accuracy and performance measure**				
	corre. class.	**incorr. class.**	**precision**	**Recall**	**F-Measure**
Random undersampling	58.00%	42.00%	0.86	0.58	0.58
Instance Hardness Threshold	51.97%	48.03%	0.88	0.52	0.52
Condensed nearest Neighbour	7.31%	92.69%	0.07	0.07	0.07

3.3 Discussion

One weakness for the models is that there is a lot of categorical data that is not strong in the feature importance for the classification. The models heavily weigh on the features for location (latitude and longitude), which make the classification very location dependent. Terrorist groups targeting new attack space will easily be faulty classified when the attack is outside their current region.

The focus was now on a few variables, while there are more in the dataset. The scope of variables was narrowed down during this research, however in a more concise model all reasonable variables should be considered.

The population variable was included by using a API from geonames (Geonames, 2018). While using this API, the assumptions were made that the most relevant location with respect to the latitude and longitude was the right result. Also the locations that gave no result were given an population of 0. Which is probably not true, however this locations can be considered as small while there is no information of population available. Next to this, the population were received from wikipedia,

which itself uses other different websites to receive the populations for different countries. It is likely that the populations were measured in different years.

Furthermore, while only the largest terrorist groups are included in our model, the model is not useful in real life. A small 40% of terrorist attacks are still performed by smaller terrorist groups, which are harder to predict due to lack of data. The lack of data for the smaller groups is a bad thing for our research, however less attacks is a better for the world.

While the perfect model could possible predict the terrorist group behind an attack with a high level of accuracy, in real life the classification of terrorist attacks could not be predicted with certainty. However, a good model could come to aid for the anti terrorist organisations which could start their investigations more focussed with help of a model such as ours.

Another discussion point is the algorithms used. Many algorithms exist for executing the machine learning models. However, to limit the scope of this research, not all algorithms have been used. For example, the CART algorithm is used for the DTs. A disadvantage of this algorithm is that it could make the DTs unstable.

One of the weaknesses of the Support Vector Machine model is that only the Gaussian Kernel was tested, due to lack of time. While this kernel proved to give the best results for the SVM model in most cases, it is always best to test for the other kernels as well.

4. Related Work

The most relatable study is that of Soliman et al (2015). The aim in the research is to compare predictive accuracy and comprehension of several machine learning techniques as Naïve Bayes, K-nearest neighbours, Support Vector Machines, Decision Tree, and Random Forest. In this paper, these mentioned machine learning techniques are tested as well on a dataset of the Global Terrorism Database. Soliman

et al (2015) combine several techniques as a hybrid method and compares these. Which makes it different from this research.

Besides the hybrid machine learning techniques which make this research a novel one. The data set used in Soliman et al (2015) is quite different from the one used in this research. Even though the data from both studies were obtained through the same organization. It cannot be concluded why these data sets differ so much because the data preprocessing technique is not described extensively in the study of Soliman et al (2015). It can be concluded that numerous differences were present in the data preprocessing techniques between this research and that of Soliman et al (2015).

5. Future Work

For future research the data set should contain more data to work with. Because of the a lot of 'unknown' values a significant part of the data could not be used in the research. Also, a bigger variance of algorithms and parameters could be tested to investigated whether better results could be obtained.
The machine learning techniques could use more variables present in the data set in a future study. There may be another variable which could help predict terrorism attacks better. As final recommendation this study would encourage the use of closed source intelligence.

6. Conclusion

Using five different Machine Learning Models, this research reveals that there are a lot of predictive insights in osint data of Terrorist Attacks in the middle east and North Africa. Unfortunately, due to limiting the terrorist groups at the preprocessing phase of this research, the conclusion that the models come to aid in classifying all terrorist attacks in the Middle East and North Africa cannot be made. However, this research shows that when focussing on the larger terrorist groups, Machine Learning can come to aid in helping to classify these larger terrorist groups.

Multiple models have shown to classify our 5 terrorist groups with an accuracy above 80%, which are defined in the hypothesis as a good model. All the decision trees models proved to have an accuracy above 80%. Random forests have also shown to be a good model, however the overfitting has to be taken in consideration. The Naive Bayes model with the Gaussian algorithm have shown to be a good predictor. As last, the K-nearest Neighbors also has an accuracy above 80%.

The learning curves, derived from the DT, tend to overfit when using the pipeline method. Further time and research should be put into this observation.

This research shows that there is a lot of value in the combination of Machine Learning and the fight against terrorism. The possibility of classifying the larger terrorist groups behind terrorist attacks using Machine Learning is demonstrated, further research must be made to investigate to what extend Machine Learning Models could help anti terrorism in practise.

References

- Bonnaccorsi, “On the Relationship between Firm Size and Export Intensity,” Journal of International Business Studies, XXIII (4), pp. 605-635, 1992. (journal style)

- Breiman, L. (2001, April 11). Random Forests. *Machine Learning, 45,* 5-32. https://doi.org/10.1023/A:1010933404324

- Breiman L., Friedman J., Olshen R., and Stone C. Classification and Regression Trees. Wadsworth, Belmont, CA, 1984.

- Dietterich, T. (1995, September). Overfitting and Undercomputing in Machine Learning. *ACM Computing Surveys, 27,* 326-327. Doi: 10.1145/212094.212114

- Donges, N. (2018). The Random Forest Algorithm. *Towards Data Science.* Retrieved from https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd

- Geonames (2018) retrieved from https://www.geonames.org/export/ws-overview.html

- GTD Codebook. (2018). Global Terrorism Database Codebook: Inclusion criteria and variables.

- Gu, S. (2017, October 2017). How Random Forest Algorithm Works in Machine Learning. *Synced.* Retrieved from https://medium.com/@Synced/how-random-forest-algorithm-works-in-machine-learning-3c0fe15b6674

- Khorshid, M. M., Abou-El-Enien, T. H., & Soliman, G. M. (2015). Hybrid Classification Algorithms For Terrorism Prediction In Middle East And North Africa. International Journal of Emerging Trends & Technology in Computer Science, 4(3).

-scikit-learn (2018), RBF SVM parameters retreived from https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html on 28-11-2018.

- START GTD Overview. (2018). START National consortium for the study of terrorism and responses to terrorism background report.

- Miškovic, V. (2014). Machine learning of hybrid classification models for decision support. Sinteza 2014-Impact of the Internet on Business Activities in Serbia and Worldwide, 318-323.