In reality, the majority of the events are considered to be “normal”, which leads to dataset on the left side of Figure 25. With such a dataset, an “algorithm” that always predicts the majority class for each input sample, could reach a very high accuracy without having learned anything:

$$Accuracy = \frac{Number\ of\ true\ positives + Number\ of\ true\ negatives}{Number\ of\ samples} = \frac{1000 + 0}{1000 + 10} \approx 99\%$$

In this case, no suspicious behaviour could ever be detected. Some possible ways to keep a dataset more balanced are:

- Collecting more data for the minority class
- Generating synthetic samples for the minority class
- Over-sampling of the minority class and / or under-sampling of the majority class

Selecting the right (amount) of data and features can also be very challenging and should always be taken with care. As mentioned in section 4.4.4, the wrong contents of a dataset can cause unfair bias which can lead to “automated discrimination and unfairness”. Some types of bias that should be considered while compiling a dataset are [115]:

- Historical Bias: already existing bias and socio-technical issues in the world
- Representation Bias: the way a population is defined and sampled from
- Measurement Bias: the way a feature is chosen, utilized and measured
- Sampling Bias: due to non-random sampling of subgroups
- Popularity Bias: More popular items / topics appear more often than less popular ones
- Social Bias: content coming from other people affects someone’s judgement

Regarding to data in tables such as Microsoft Excel, there are some additional aspects to keep in mind:

- All entries in a row should contain a value and should not be left empty or must not have a placeholder like “-“, etc.
- The datatypes of a column should be kept consistent and should not be mixed
- Digits must not contain a thousand separator (dot in Europe or comma in the US) to avoid misinterpreting the values
- Commas can be commas or dots and should not be used mixed

4.4.6 Implementation

The tool is developed in Java using the open-source[27] machine learning library Smile (Statistical Machine Intelligence and Learning Engine). Smile is a fast and comprehensive machine-learning engine with advanced data structures and algorithms, supporting different programming languages like Java, Scala or Kotlin. It supports various data input formats [116] like:

- Weka ARFF (attribute relation file format) is an ASCII text file format that is essentially a CSV file with a header that describes the meta data. ARFF was developed for use in the Weka machine learning software.

[27] Licensed under the Apache License, Version 2.0

- LibSVM is a very fast and popular library for support vector machines. LibSVM uses a sparse format where zero values do not need to be stored. Each line of a libsvm file is in the format: <label> <index1>:<value1> <index2>:<value2> ...
- Delimited Text and CSV (Comma-separated values): Any character may be used to separate the values, but the most common delimiters are the comma, tab, and colon.
- other formats that are not relevant for the intended use in PREVISION, mostly used by scientists: MicroArray, Coordinate Triple Tuple List, Harwell-Boeing Column-Compressed Sparse Matrix and Wireframe.

The entire application has been developed as a containerized microservice. This allows easy deployment and integration into the PREVISION platform, where everything that is needed to run the service (code, dependencies, runtimes, etc.) is already included within the container.

The decision tree algorithm implemented in Smile is based on the CART or Classification & Regression Trees methodology that was introduced in 1984 [117]. Smile supports three different split strategies based on different measures for calculating the score of a split criterion: The GINI index, the entropy measure and the classification error.

The general workflow of the implementation is depicted in Figure 26. The input for the tool is a Microsoft Excel file containing two sheets, where the first row of each sheet contains any name of the respective feature. Each sheet has a number of N features in the columns, where the order of the features must match. The first sheet in the Excel file is a table with J feature vectors (rows) of new data, that the trained predictor should classify. The second sheet is a table that contains a number of M training examples in the rows and with an additional last column, that holds the associated target class for each training example. For better data handling, the sheets are converted internally to the Weka ARFF format. Then, the supervised learning process is started, in which the decision tree is trained with the provided training dataset. Then, the trained decision tree is used as a predictor to predict the classes of the dataset to classify. Finally, the predicted classes are appended as a new column to the dataset to classify and converted back to Microsoft Excel format.

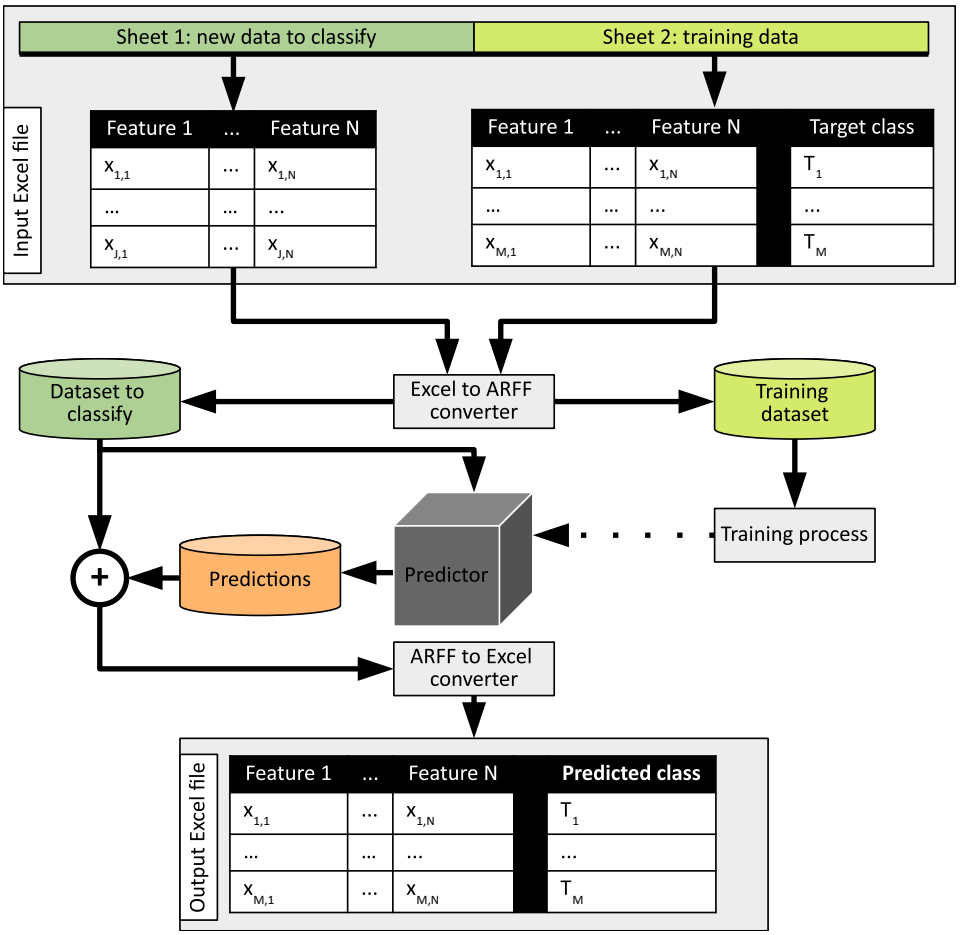

Figure 26. Implementation workflow of the PREVISION decision tree classifier.

I addition to the classified new data, the tool returns the graphic representation of the predictor in Graphviz dot format. Graphviz is an open-source graph visualization software representing structural information as diagrams of abstract graphs and networks like in [118]. In this case, Graphviz is used, to visualize the internal structure of the trained decision tree in order to better understand the decisions made by the tool. It can be embedded in the PREVISION web portal, ensuring that the requirements of explainability and accountability are satisfied, which refers to the requirement of "Explainability and algorithmic transparency" of PREVISION deliverable 8.1 [111]. An example tree, after the Financial Data Records dataset supplied by IGPR has classified, is shown in Figure 27 and Figure 28.

4.4.7 Evaluation

The Financial Data Records (FDR) dataset supplied by IGPR has been used for testing the Decision Tree Tool. The dataset contains more than 12000 transactions in Microsoft Excel format. The columns containing the initial balance, the amount and the final balance have been formatted as numbers (without the dot as separator for the thousand). All trailing spaces have been removed, because they are a problem when processing the data.

All transactions have been assigned to one of the following classes: “not suspicious”, “maybe suspicious”, and “suspicious”. For this purpose, an additional column named “Suspicious Transaction” has been added to the table that contains the training data. All transactions transferring money to a certain bank in Monaco with an amount of more than 600 Lei (the Romanian currency) have been labelled as “suspicious”, all debit-transactions with an amount of more than 300 Lei to this bank have been labelled as “maybe suspicious” and the rest of the transactions have been labelled with “not suspicious”.

The dataset has been split into two parts: The first 3000 transactions have been used as a test dataset, the rest has been used for the training of the Decision Tree. The test dataset contains 2805 entries marked as “not suspicious”, 126 entries marked as “maybe suspicious” and 70 “suspicious” entries.

Non-numeric values can be a problem for the algorithm: All non-numeric values are internally mapped to index numbers. But in the FDR, there are columns like the beneficiary bank that have no closed value set, meaning that there can occur new values in the datasets to be classified that have not occurred in the training dataset before. As a result, the csv-format has proven to be problematic, because the automatic generated mapping of non-numeric column values used for training is different from the mapping of the new data that is to be classified. The decision tree will not be applicable to the new datasets to be classified, because it doesn’t hold the correct index numbers.

The usage of ARFF-format however can ensure that this mapping is identical for both datasets, because it supports control of the mapping by explicitly defining the order (and respectively the index number) of the possible values of each non-numeric attribute set. A special CSV-to-ARFF converter must be used that creates the ARFF with the correct attribute value order by taking the training data’s attribute definitions and expanding them.

Figure 27 and Figure 28 show the Decision Trees that have been learned using different split rules. Both Decision Trees have then been tested by predicting the classes on the test dataset. Only 2 of 3000 predictions have been wrong. This leads to an accuracy of:

$$Accuracy = \frac{Number\ of\ true\ positives + Number\ of\ true\ negatives}{Number\ of\ samples} = \frac{2998}{3000} \approx 99{,}93$$

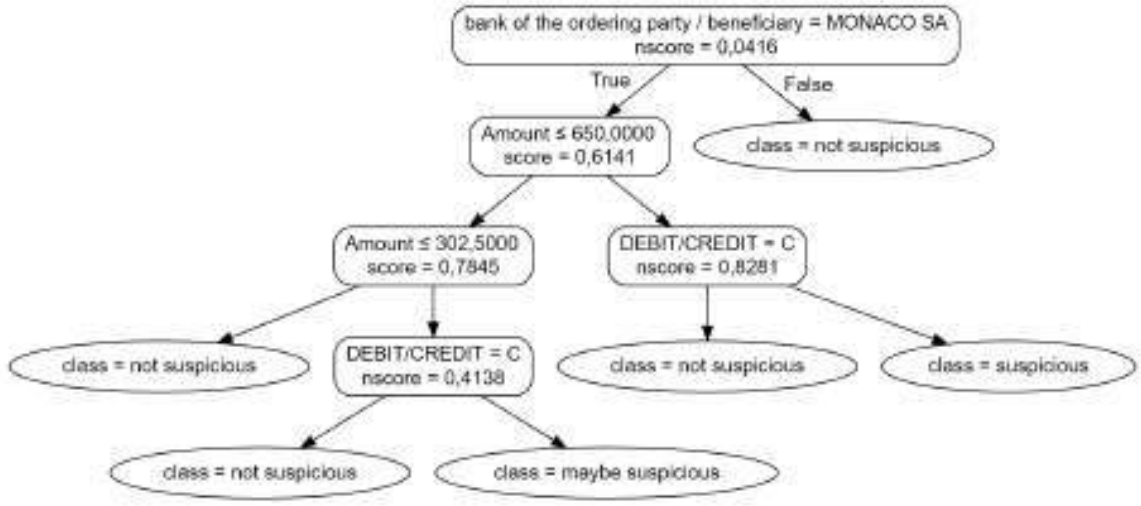

Figure 27. Decision Tree of the FDR dataset, when the split rule based on an entropy measure is applied.

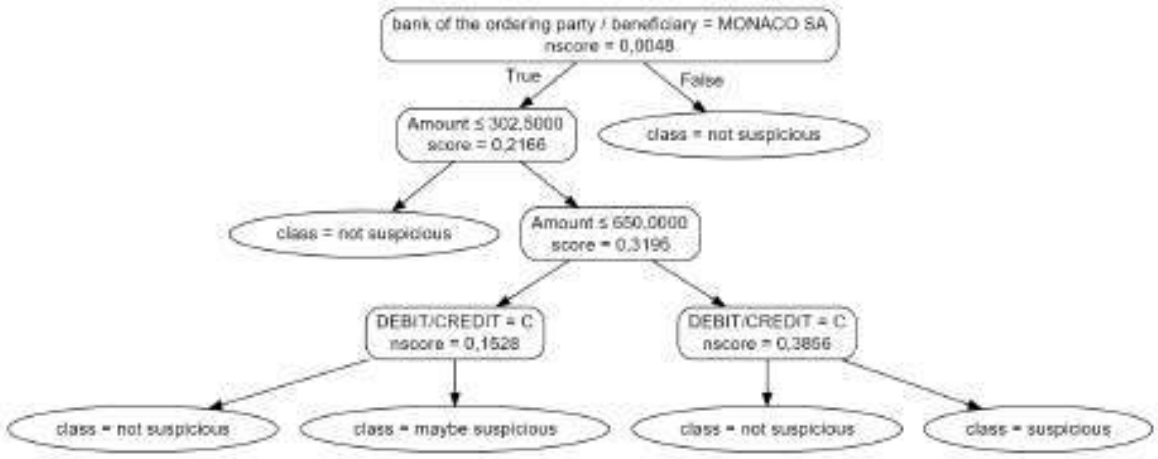

Figure 28. Decision Tree of the FDR dataset, when the split rule based on the GINI measure is applied.

4.4.8 Random Forest Classifiers

In order to make the classification more accurate and robust, a number of decision trees can be composed to an ensemble, called a random forest classifier. The principle is shown in Figure 29:

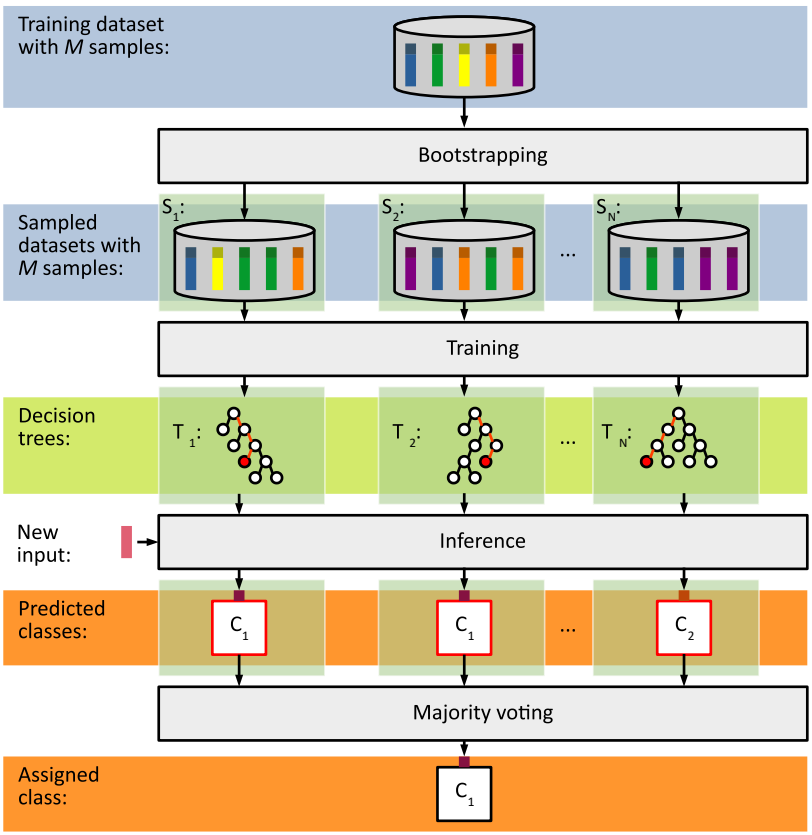

Figure 29. Principle of classification with random forest models.

As before, the training dataset consists of a set of M feature vectors, each associated with its desired target class (samples). The user specifies a number of N trees for the random forest model, which leads to a number of N sampled training datasets, one for each tree. Each sampled dataset consists of M randomly selected samples from the training dataset, where each sample may occur repeatedly (sampling with replacement). These sampled datasets are then used to build a decision tree which results in one tree per sampled dataset. During inference, the trained trees are used to predict the classes of new inputs, where each tree predicts a class individually. Finally, the class assigned to the new input is obtained by a majority voting [119].

Some advantages and disadvantages of random forests over decision trees are [120]:

- Ensembles reduce the variance of classifiers
- Combining the decision results of many trees helps to reduce the problem of overfitting
- As more trees are added, decision boundaries become more accurate and stable

Disadvantages are:

- More complex, as a consequence more time-consuming when constructing the forest and when predicting new inputs
- Additional hyperparameter: The number of trees N, which is usually obtained through cross-validation or grid-search

- More like a black box

Because of its complexity, a random forest model should only be used as an alternative to a decision tree, if the requirement of explainability and algorithmic transparency does not have to be fulfilled.

4.5 Information Processing

4.5.1 Overview

According to the PREVISION proposal, "In this task, correlation and machine learning techniques will be used to provide LEA officers with an automated capability to analyze vast amounts of heterogeneous data... Focus will be given classification algorithms (supervised learning), clustering techniques (unsupervised learning) and model fitting techniques to discover correlated evidences to support forensic investigation."

The whole process has several steps, organized in the layers as presented in the next figure.

- Data acquisition
- Preprocessing
- Features extraction
- Features representation and normalization
- Machine Learning processing, including classifications.

These layers are presented in the following subsections.

4.5.2 Data acquisition.

Data acquisition is subject of the tasks defined in WP2. It refers to

- Task 2.1 Crawling Tools
- Task 2.2 Interoperability with Traffic, Telecom and Financial Data Sources
- Task 2.3 Batch and Near-real-time Video and Image Analysis
- Task 2.4 Darknet, Web and Social Networks Data Analysis

4.5.3 Preprocessing

There are some ways of preprocessing data in order to make it more suitable for ML algorithms.

- Standardization and Normalization
- Detecting Outliers
- Treatment of missing data

4.5.3.1 Standardization and Normalization

In fact, the only family of algorithms that we could think of being scale-invariant are tree-based methods. Considering the general CART decision tree algorithm, without going into much depth regarding information gain and impurity measures, we can think of the decision as "is feature x_i >= some_val?" Intuitively, we can see that it really doesn't matter on which scale this feature is (centimeters, Fahrenheit, a standardized scale – it really doesn't matter). For this reason, standardization and normalization is not implemented in our CART systems.

4.5.3.2 Detecting Outliers

There are a couple of reasons outliers are important:

- Outliers distort the picture of the data we obtain using descriptive statistics and data visualization. When our goal is to understand the data, it is often worthwhile to disregard outliers.
- Outliers play havoc with many machine learning algorithms and statistical models. When our goal is to predict, our models are often improved by ignoring outliers.
- Outliers can be exactly what we want to learn about, especially for tasks like anomaly detection. Standardization of a dataset is a common requirement for many machine learning estimators. Typically, this is done by removing the mean and scaling to unit variance. However, outliers can often influence the sample mean / variance in a negative way. In such cases, the median and the interquartile range often give better results.

Our approach is to use from scikit-learn the **`RobustScaller`** component for processing outliers.

This Scaler removes the median and scales the data according to the quantile range (defaults to IQR: Interquartile Range). The IQR is the range between the 1st quartile (25th quantile) and the 3rd quartile (75th quantile).

Centering and scaling happen independently on each feature by computing the relevant statistics on the samples in the training set. Median and interquartile range are then stored to be used on later data using the `transform` method.

When using **RobustScaller** we will test the best approach on real data by playing with the following parameters: [121]

`with_centering` `boolean, True by default`

If True, center the data before scaling. This will cause `transform` to raise an exception when attempted on sparse matrices, because centering them entails building a dense matrix which in common use cases is likely to be too large to fit in memory.

`with_scaling` `boolean, True by default`

If True, scale the data to interquartile range.

`quantile_range` `tuple (q_min, q_max), 0.0 < q_min < q_max < 100.0`

Default: (25.0, 75.0) = (1st quantile, 3rd quantile) = IQR Quantile range used to calculate `scale_`.

New in version 0.18.

`Copy` `boolean, optional, default is True`

If False, try to avoid a copy and do inplace scaling instead. This is not guaranteed to always work inplace; e.g. if the data is not a NumPy array or scipy.sparse CSR matrix, a copy may still be returned.

4.5.3.3 Treatment of Observations with Missing Values

Missing data arise in almost all serious statistical analyses. In this chapter we discuss a variety of methods to handle missing data, including some relatively simple approaches that can often yield reasonable results.

To decide how to handle missing data, it is helpful to know why they are missing. We consider four general "missingness mechanisms," moving from the simplest to the most general.

1. Missingness completely at random.
2. Missingness at random.
3. Missingness that depends on unobserved predictors.
4. Missingness that depends on the missing value itself.

For each type we'll try the most appropriate solution including deletion of records with missing values (dataframe.dropna from pandas) and inputting missing values based on a KNN predicting system.

4.5.4 Features extraction

An ML system does not understand string values. They need numerical input to build models, sometimes they are also called numerical features. Feature extraction in ML is converting a set of text information into a set of numerical features. Any machine learning algorithm that you are going to train would need features in numerical vector forms as it does not understand the string. There are many ways text can be represented as numerical vectors. Some such ways are One hot encoding, TF-IDF, Word2Vec, and CountVectorizer.

Traditional clustering methodologies assume features are numeric valued, and represent data objects as points in a multidimensional metric space. These classical approaches adopt distance metrics, such as Euclidean and Mahalanobis measures, to define similarity between objects. On the other hand, conceptual clustering systems use conditional probability estimates as a means for defining the relation between groups or clusters. The measure partitions a data set in a manner that maximizes the probability of correctly predicting a feature value in a given cluster. These measures are tailored for nominal attributes. As application areas have grown from the scientific and engineering domains to the medical, business, and social domains, a majority of the useful data is described by a combination of numeric and nominal valued features.

In PREVISION, text preprocessing will be performed in modules, developed as part of the project, and which will implement the algorithms mentioned before. The usage of a specific algorithm will depend on the nature of the text processing. The preprocessing is fully based on the results of Task 3.1 "Semantic Information Processing"**,** which will advance the state of the art in semantic technologies, which provide a computable framework for systems to deal with knowledge, in a formalized manner. In the paradigm of semantic technologies, the metadata that represent data objects are expressed in a manner in which their deeper meaning and interrelations with other concepts are made explicit, by means of an ontology. This approach provides the underlying computing systems with the capability not only to extract the values associated with the data but also to relate pieces of data one to another, based on the details of their inner relationships

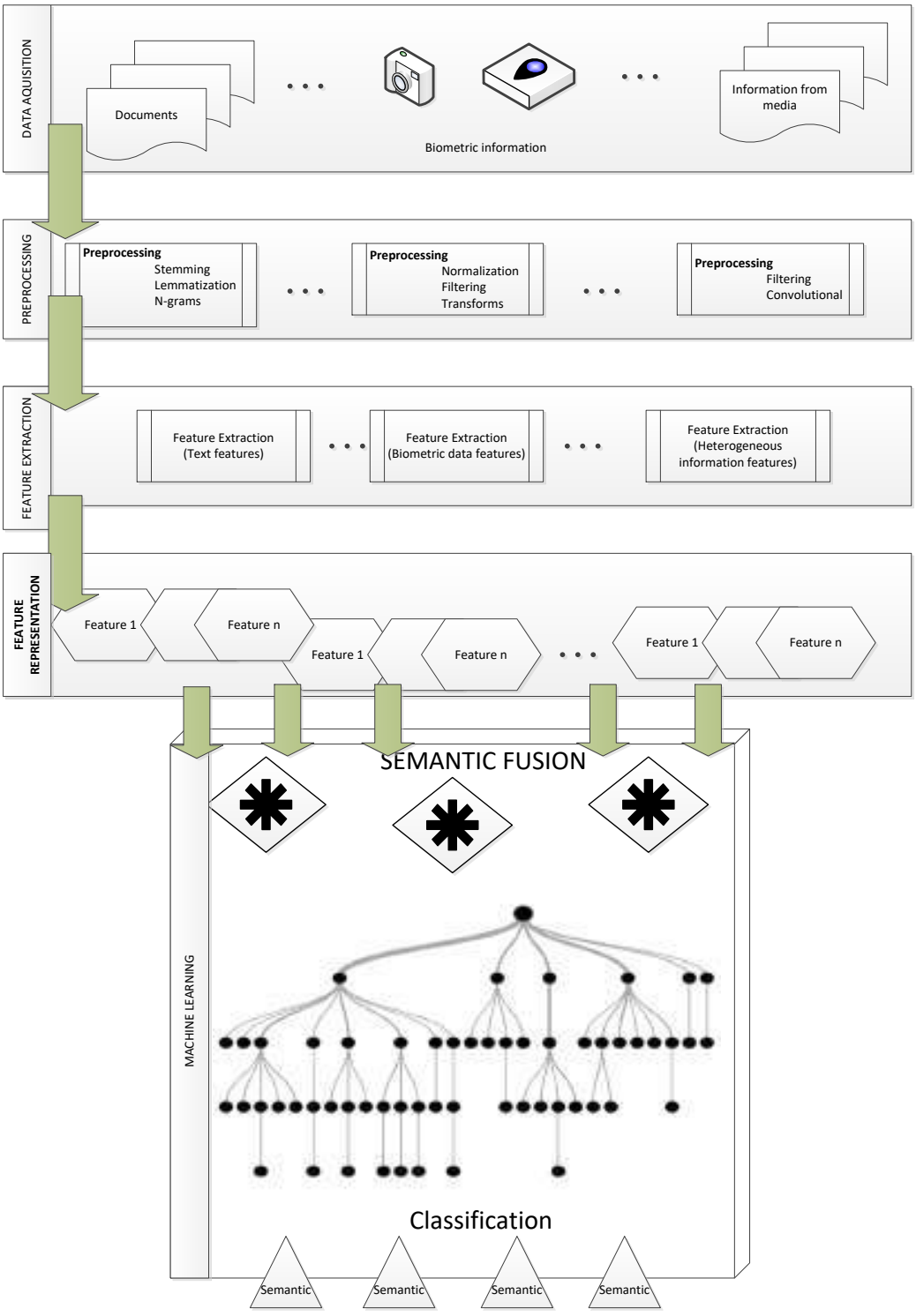

Figure 30. Information processing.

4.5.5 Data Classification [122]

Data classification is one of the very common use cases of PREVISION. Data classification can be used for use cases such as detection of a specific behavior, and sentiment analysis. This process is typically a classification problem wherein we are trying to identify a specific topic from a natural language source of a large volume of data. Within each of the data groups, we may have multiple topics discussed and hence it is important to classify the article or the textual information into logical groups. Text classification techniques help us to do that.

We will divide the data classification process in three parts:

- model training
- model verification,
- prediction

We will use the Clustering algorithms, Decision Trees and Random Forests algorithms for model training and prediction.

4.5.5.1 K-Means Clustering [123]

K-means Clustering is one of the most popular unsupervised algorithms for data clustering, which is used when we have unlabeled data without defined categories or groups. The number of clusters is represented by the k variable. This is an iterative algorithm that assigns the data points to a specific cluster based on the distance from the arbitrary centroid. During the first iteration, the centroids are randomly defined and the data points are assigned to the cluster based on the least vicinity from the centroid. Once the data points are allocated, within the subsequent iterations, the centroids are realigned to the mean of the data points and the data points are once again added to the clusters based on the least vicinity from the centroids. These steps are iterated to the point where the centroids do not change more than the set threshold.

In PREVISION k-means clustering will be used to identify groups of information, when the number of groups is previously known. This is usually used when some expected behavior is studied.

4.5.5.2 Decision Trees [123]

Classification and Regression Trees or CART for short is a term introduced by Leo Breiman [117] to refer to Decision Tree algorithms that can be used for classification or regression predictive modeling problems. Classically, this algorithm is referred to as “decision trees”, but on some platforms like R they are referred to by the more modern term CART. The CART algorithm provides a foundation for important algorithms like bagged decision trees, random forest and boosted decision trees.

In PREVISION, decision trees are used to identify when a user is expected to be categorized in a specific group of behavior.

CART Model Representation

The representation for the CART model is a binary tree. This is your binary tree from algorithms and data structures, nothing too fancy. Each root node represents a single input variable (x) and a split point on

that variable (assuming the variable is numeric). The leaf nodes of the tree contain an output variable (y) which is used to make a prediction.

Given a dataset with two inputs (x) of height in centimeters and weight in kilograms the output of sex as male or female, below is a crude example of a binary decision tree (completely fictitious for demonstration purposes only). With the binary tree representation of the CART model described above, making predictions is relatively straightforward. Given a new input, the tree is traversed by evaluating the specific input started at the root node of the tree.

A learned binary tree is actually a partitioning of the input space. You can think of each input variable as a dimension on a p-dimensional space. The decision tree split this up into rectangles (when p=2 input variables) or some kind of hyper-rectangles with more inputs.

New data is filtered through the tree and lands in one of the rectangles and the output value for that rectangle is the prediction made by the model. This gives you some feeling for the type of decisions that a CART model is capable of making, e.g. boxy decision boundaries.

Learn a CART Model from Data

Creating a CART model involves selecting input variables and split points on those variables until a suitable tree is constructed. The selection of which input variable to use and the specific split or cut-point is chosen using a greedy algorithm to minimize a cost function. Tree construction ends using a predefined stopping criterion, such as a minimum number of training instances assigned to each leaf node of the tree.

Greedy Splitting

Creating a binary decision tree is actually a process of dividing up the input space. A greedy approach is used to divide the space called recursive binary splitting.

This is a numerical procedure where all the values are lined up and different split points are tried and tested using a cost function. The split with the best cost (lowest cost because we minimize cost) is selected. All input variables and all possible split points are evaluated and chosen in a greedy manner (e.g. the very best split point is chosen each time).

For regression predictive modeling problems, the cost function that is minimized to choose split points is the sum squared error across all training samples that fall within the rectangle:

$$\sum (y - prediction)^2$$

Where y is the output for the training sample and prediction is the predicted output for the rectangle.

For classification the Gini index function is used which provides an indication of how “pure” the leaf nodes are (how mixed the training data assigned to each node is).

$$G = \sum (p_k * (1 - p_k))$$

Where G is the Gini index over all classes, p_k are the proportion of training instances with class k in the rectangle of interest. A node that has all classes of the same type (perfect class purity) will have G=0, where as a G that has a 50-50 split of classes for a binary classification problem (worst purity) will have a G=0.5.

For a binary classification problem, this can be re-written as:

$G = 2 * p_1 * p_2$
or
$G = 1 - (p_1^2 + p_2^2)$

The Gini index calculation for each node is weighted by the total number of instances in the parent node. The Gini score for a chosen split point in a binary classification problem is therefore calculated as follows:

$$G = \left(\left(1 - \left(g_{1,1}^2 + g_{1,2}^2\right)\right) * \left(\frac{ng_1}{n}\right)\right) + \left(\left(1 - \left(g_{2,1}^2 + g_{2,2}^2\right)\right) * \left(\frac{ng_2}{n}\right)\right)$$

Where

G is the Gini index for the split point,
$g_{1,1}$ is the proportion of instances in group 1 for class 1,
$g_{1,2}$ for class 2,
$g_{2,1}$ for group 2 and class 1,
$g_{2,2}$ group 2 class 2,
ng_1 and ng_2 are the total number of instances in group 1 and 2 and
n are the total number of instances we are trying to group from the parent node.

Stopping Criterion [48]

The recursive binary splitting procedure described above needs to know when to stop splitting as it works its way down the tree with the training data.

The most common stopping procedure is to use a minimum count on the number of training instances assigned to each leaf node. If the count is less than some minimum then the split is not accepted and the node is taken as a final leaf node.

The count of training members is tuned to the dataset, e.g. 5 or 10. It defines how specific to the training data the tree will be. Too specific (e.g. a count of 1) and the tree will overfit the training data and likely have poor performance on the test set.

Pruning the Tree

The stopping criterion is important as it strongly influences the performance of your tree. You can use pruning after learning your tree to further lift performance.

The complexity of a decision tree is defined as the number of splits in the tree. Simpler trees are preferred. They are easy to understand (you can print them out and show them to subject matter experts), and they are less likely to overfit your data.

The fastest and simplest pruning method is to work through each leaf node in the tree and evaluate the effect of removing it using a hold-out test set. Leaf nodes are removed only if it results in a drop in the overall cost function on the entire test set. You stop removing nodes when no further improvements can be made.

More sophisticated pruning methods can be used such as cost complexity pruning (also called weakest link pruning) where a learning parameter (alpha) is used to weigh whether nodes can be removed based on the size of the sub-tree

4.5.5.3 Random Forests [123] [122]

Random Forest is the class of algorithms that comes under the supervised learning algorithm category. It is based on forests of trees, which is similar to decision trees in certain contexts. Random Forest algorithms can be used for both classification and regression problems. A decision tree gives the set of rules that are used in building models, which can be executed against a test dataset for the prediction. In decision trees, we first calculate the root node. To calculate the root node, we use information gain. For example, if you want to predict whether your friend will accept a job offer or not. You need to feed the training dataset of the offers they have accepted to the decision tree. Based on this, the decision tree will come up with a set of rules that you will be using in the prediction. So let's say a rule can be if salary > 50K, then your friend will accept the offer. A decision tree algorithm can overfit as it is very flexible. To avoid this model overfitting in a decision tree, we can perform the pruning. The following is the pseudocode for the Random Forest algorithm:

1. Randomly select *k* features from total *m* features. Where $k << m$.
2. Among the *k* features, calculate the node, *d*, using the best split point.
3. Split the node into daughter nodes using the best split.
4. Repeat steps 1 to 3 until *l* number of nodes has been reached.
5. Build the forest by repeating steps 1 to 4 for *n* number times to create *n* number of trees.

Once we have trained the model using the previous steps, for prediction we need to pass the test features through all rules created by the different trees in the forest. If we want to understand by example, suppose you want to purchase a mobile phone and you have decided to ask your friends which phone is best for you. In this case, your friends might ask you some random question about the features you like and suggest a suitable phone. Here, each friend is the tree, and with the combination of all the friends, we form the forest.

Once you collect the suggestions from your friends (trees, in terms of the Random Forest algorithm), you will count which type of phone has the most votes, and you will might purchase that one. Similarly, in Random Forest, each tree will predict a different target variable that we will sum with respect to that key. The key with the highest count, predicted by the maximum number of trees, is the final

In PREVISION, decision trees are used to identify when a user is expected to be categorized in a specific group of behavior, when the complexity of data is much bigger that in the situation decision trees (CART) are used.

4.5.6 Development Approach [121]

Modules will be developed in Python, are platform independent, and are the main mechanism used to derive meaning from input data.

For machine learning processing we are using well known python libraries using Scikit-learn, version 0.22

From Scikit-learn the implementation of the following algorithms will be used:

- Feature transformations: standardization, normalization, hashing
- ML Pipeline construction
- Model evaluation and hyper-parameter tuning
- ML persistence: saving and loading models and Pipelines
- Decision trees, random forests, and gradient-boosted trees
- Recommendation: alternating least squares (ALS)
- Clustering: K-means,
- Frequent item sets, association rules, and sequential pattern mining

The software development architecture, proposed for data fusion and classification is presented in Figure 33.

4.5.6.1 Data Distribution Example

Before deciding on what strategy to be used for the clustering or classification, the distribution of values in the space of scores is presented.

In the example, a nonlinear distribution is seed, and the idea to use decision trees or random forests is applicable. Figure 31.

The training process will follow the steps:

```
#loading the data from csv files. For example:

train_data = pd.read_csv("C:/data/siveco/PREVISION_CLUSTER.csv",
low_memory=False)


#plot the data

features = ['ImageScore','TextScore']

targets = ['real','wrong']

X = train_data[features].values # matrix

y = train_data['RealExpected'].values
```

```
plt.plot(X[y==0, 1], X[y==0, 0], 'r*', markersize=3, label="spoof")

plt.plot(X[y==1, 1], X[y==1, 0], 'go', markersize=2, label="real")

plt.show
```

#create decision tree clasifier

```
decisiontree = DecisionTreeClasiffier (max_depth=2 , criterion = "mse",
random_state=0);
```

#**train the model**

```
model = decisiontree.fit(features, target);
```

The prediction process will follow the steps:

#loading the data from csv files. For example:

```
predicted_data = pd.read_csv("C:/data/siveco/PREVISION_predict.csv",
low_memory=False)
```

#predict the value

```
Res1=model.predict(predicted_data)

#get probabilities of prediction

Res2=model.predict_proba(predicted_data)
```

#visualize the decision tree model, by exporting a png file

```
dot_data = export_graphviz(decisiontree,
out_file=None,feature_names=features, \

    class_names = [ 'wrong','real'], filled=True,
leaves_parallel=True)

graph=pydotplus.graph_from_dot_data(dot_data)

Image(graph.create_png())

graph.write_png("C:/data/dectree_prevision.png")
```

The classifier can use different parameters, and the correct solution comes after a large set of tests on real data. Initially we are using the default values, then we play with criterion and max_depth, then we play with other parameters.

The list of possible parameters is below:

`criterion``{"gini", "entropy"}, default="gini"`

The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.

`splitter``{"best", "random"}, default="best"`

The strategy used to choose the split at each node. Supported strategies are "best" to choose the best split and "random" to choose the best random split.

`max_depth` `int, default=None`

The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

`min_samples_split` `int or float, default=2`

The minimum number of samples required to split an internal node:

If int, then consider `min_samples_split` as the minimum number.

If float, then `min_samples_split` is a fraction and `ceil(min_samples_split * n_samples)` are the minimum number of samples for each split.

`min_samples_leaf` `int or float, default=1`

The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least `min_samples_leaf` training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression.

If int, then consider `min_samples_leaf` as the minimum number.

If float, then `min_samples_leaf` is a fraction and `ceil(min_samples_leaf * n_samples)` are the minimum number of samples for each node.

`min_weight_fraction_leaf` `float, default=0.0`

The minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node. Samples have equal weight when sample_weight is not provided.

`max_features` `int, float or {"auto", "sqrt", "log2"}, default=None`

The number of features to consider when looking for the best split:

If int, then consider `max_features` features at each split.

If float, then max_features is a fraction and int(max_features * n_features) features are considered at each split.

If “auto”, then max_features=sqrt(n_features).

If “sqrt”, then max_features=sqrt(n_features).

If “log2”, then max_features=log2(n_features).

If None, then max_features=n_features.

Note: the search for a split does not stop until at least one valid partition of the node samples is found, even if it requires to effectively inspect more than max_features features.

random_state int or RandomState, default=None

If int, random_state is the seed used by the random number generator; If RandomState instance, random_state is the random number generator; If None, the random number generator is the RandomState instance used by np.random.

max_leaf_nodes int, default=None

Grow a tree with max_leaf_nodes in best-first fashion. Best nodes are defined as relative reduction in impurity. If None then unlimited number of leaf nodes.

min_impurity_decrease float, default=0.0

A node will be split if this split induces a decrease of the impurity greater than or equal to this value.

The weighted impurity decrease equation is the following:

```
N_t / N * (impurity - N_t_R / N_t * right_impurity
                    - N_t_L / N_t * left_impurity)
```

where N is the total number of samples, N_t is the number of samples at the current node, N_t_L is the number of samples in the left child, and N_t_R is the number of samples in the right child.

N, N_t, N_t_R and N_t_L all refer to the weighted sum, if sample_weight is passed.

class_weight dict, list of dict or “balanced”, default=None

Weights associated with classes in the form {class_label: weight}. If None, all classes are supposed to have weight one. For multi-output problems, a list of dicts can be provided in the same order as the columns of y.

Note that for multioutput (including multilabel) weights should be defined for each class of every column in its own dict. For example, for four-class multilabel classification weights should be [{0: 1, 1: 1}, {0: 1, 1: 5}, {0: 1, 1: 1}, {0: 1, 1: 1}] instead of [{1:1}, {2:5}, {3:1}, {4:1}].

The “balanced” mode uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data as `n_samples / (n_classes * np.bincount(y))`

For multi-output, the weights of each column of y will be multiplied.

Note that these weights will be multiplied with sample_weight (passed through the fit method) if sample_weight is specified.

`ccp_alpha` `non-negative float, default=0.0`

Complexity parameter used for Minimal Cost-Complexity Pruning. The subtree with the largest cost complexity that is smaller than `ccp_alpha` will be chosen. By default, no pruning is performed.

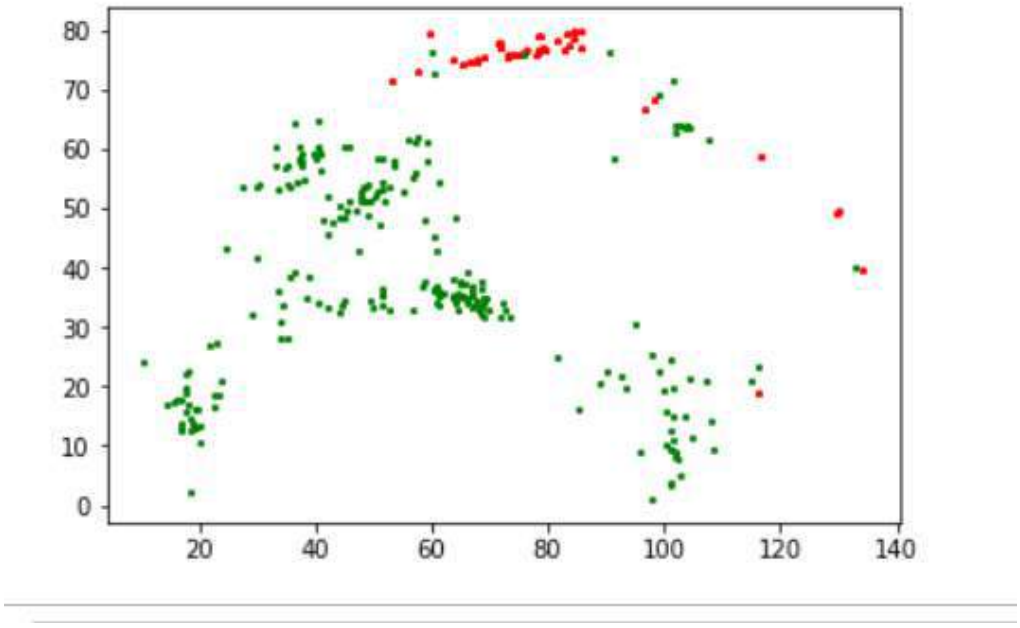

Figure 31. Data Distribution.

In the example we are presenting a decision tree for two scores.

The decision tree in the example separates the data in two categories, but many categories can also be created. Regression is also considered by using decision trees or random forests.

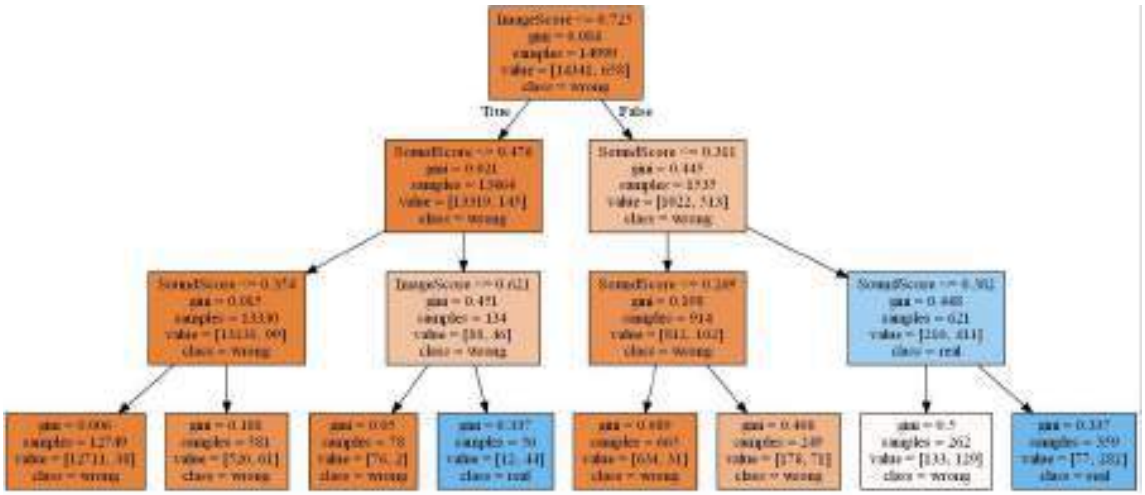

Figure 32. Decision Tree and Random Forests.

The tools used to train the system and create the decision trees or random forests were based on Python language, Python Jupyter and the Python libraries:

- Pandas (to read data stored in csv files)
- ScikitLearn (to train the system and get decision trees or random forests)
- Tensorflow (for complex parallel machine learning processes)

Data visualization uses Python Matplotlib library, with extension for saving data in png or jpg format

Exposure to other systems, based on REST services was created using Python Flask library.

The rest APIs are the communication point for complex clients, which are used to apply the decision based on the decision trees created previously during the training process.

Processed data are exposed in the format of csv files or REST lists of objects for presentation in Graphana visualization system.

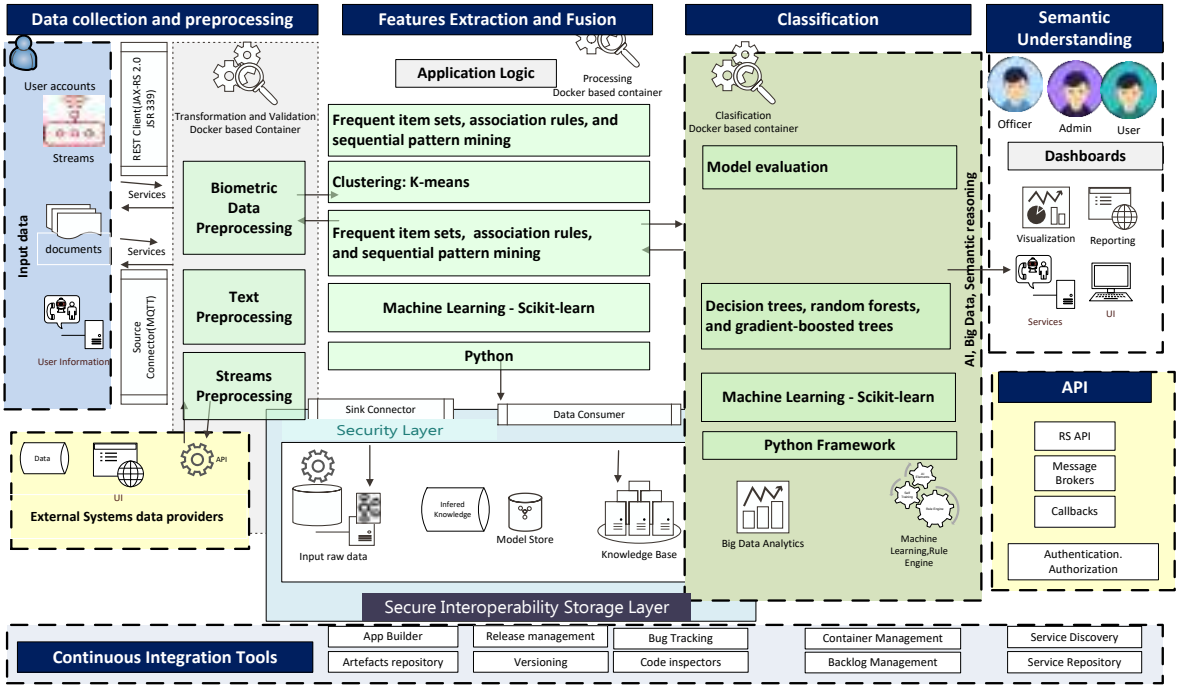

Figure 33. ML deployment architecture.

5 Predictive Analytics and Trend Analysis

This section is structured as follows. Section 5.1 deals with the fundamentals of predictive policing and puts an emphasis on the effect of propaganda messages in social networks on the radicalization of individual persons. Section 5.2 studies the detection of trends in the context of natural language processing. Section 5.3 generalizes the detectability of trends to more general datasets by proposing the use of regression trees and gradient boosted regression algorithms, while section 5.4 introduces a novel approach for checking the relevance of an automatically generated alarm signal in the AI augmented fight against crime and terrorism.

5.1 Predictive Analytics

5.1.1 Introduction

In Task 3.4 the latest machine learning techniques will be used. The data merged in Task 3.2 will serve as a data basis for prediction and trend analyses. In particular, machine learning algorithms for behavioral prediction will be used. In addition, there is a close connection to WP 4 because the developed applications are described there.

Predictive policing is the use of analytical and predictive methods (especially quantitative methods) to identify - potential targets for police intervention or prevention - and to solve crimes by statistical prediction. Several predictive police methods are currently used in law enforcement agencies and much has been written about their effectiveness. Another term that describes the use of analytical techniques to identify potential targets is prediction. Although there is a difference between prediction and forecasting, they are used interchangeably here.

Regardless of the forecasting methodologies used and the theories behind them, predictive policing is to be understood as a holistic police practice.

Due to the very low requirements from the use cases of LEAs so far regarding the possibilities offered by predictive policing, the development of targeted tools is extremely difficult. Therefore, generally valid and well-known methods and models are mentioned here first.

5.1.2 Related work

The majority of the work relates to predictive possibilities in connection with general crime, for example burglary, robbery and other offences of mass crime (see also Deliverable 1.2, para. 4 ff). Predicting acts of terrorism (specific time and place) solely on the basis of data relating to previous attacks is hardly possible, particularly because of the relatively small number of historical cases.

For this purpose, it is necessary, among other things, to analyse biographies of known offenders, to classify their origin, criminal career and change of behaviour and, if necessary, to draw conclusions.

A currently favoured approach is personal monitoring. Well-known so-called endangered persons are observed what they do, with whom they have contact, how they move and behave on the Internet/social media. From this, knowledge or statements about their future actions are to be derived. These possibilities are elaborated within PREVISION in Task 4.2 (Identification of Radicalization and Terrorist Propaganda).

Predictive policing refers to future perpetrators or victims of crime, i.e. it attempts to determine a crime risk on both the perpetrator and the victim side and to make this risk available to the police. In the case of personalised forecasting work, predictive profiling is also used ([124]: 149) or "person-based predictive targeting" ([125]: 34) or, more generally spoken, "individual-based predictive policing" (ibid.: 35; [124]: 149).

Other projects concerned with the topic:

‚RADAR-iTE' (rule-based analysis of potentially destructive perpetrators to assess the acute risk - Islamist terrorism'):

An application developed by the Federal Criminal Police Office in cooperation with the "Forensic Psychology" working group at the University of Constance, which evaluates potential perpetrators classified as dangerous or relevant persons with regard to their individual risk of violence and is intended to provide appropriate prioritisation decisions for surveillance measures. The aim of this procedure, which is based on three evaluation levels (moderate, conspicuous, high risk), is on the one hand to bring together the data available to the German security authorities on the persons concerned and to standardize the risk evaluation.[28]

‚RISKANT' (‚Risk analysis for Islamist motivated offenders'):

A research project is being carried out by the BKA in cooperation with the University of Konstanz (Working Group Forensic Psychology), among others, which is to develop a two-stage risk analysis system that "enables a case-oriented threat assessment and individual advice on measures for high-risk persons identified [by RADAR-iTE]".[29]

‚X-SONAR' (Extremist tendencies in social media networks: Identification, analysis and management of radicalisation processes):

The Federal Ministry of Education and Research (BMBF), for example, is currently funding a number of research projects that investigate the course and dynamics of radicalization processes with the explicit aim of developing an application that will predict the course of radicalization and guide suitable preventive measures. In this case, the focus of the security authorities is on persons who could possibly develop into threats, i.e. persons who could possibly carry out attacks.[30]

5.1.3 Proposed method

Social media have increasingly developed into a place where aggressively conducted social and political conflicts are carried out. Actors with an extremist background use social networks as a propaganda platform, including incitements to violence. These processes can be observed not only in social media, but

[28] https://www.bka.de/SharedDocs/Downloads/DE/AktuelleInformationen/Infografiken/Sonstige/infografikRADARiTE.jpg?__blob=publicationFile&v=4

[29] https://www.sifo.de/files/Projektumriss_RISKANT.pdf

[30] https://www.lka.polizei-nds.de/forschung/forschungsprojekt-x-sonar---extremistische-bestrebungen-in-social-media-netzwerken-identifikation-analyse-und-management-von-radikalisierungsprozessen-113539.html

also in the real world. However, the effects of propaganda events from the Internet on the real world have hardly been researched so far.

It will be examined which extremist ideas and symbolism are used on the Internet and in social media and how they contribute to radicalisation processes. Case studies (e.g. biographies and test files) will be used to investigate the effects that internet propaganda has on radicalisation and the use of violence in the real world.

As a result, a monitoring model for the analysis of extremist discourses, radicalisation processes and dynamics of violence should be developed. The findings will be used to develop tools and strategies for preventive measures and early detection of radicalisation processes.

5.1.4 Evaluation framework

In order to test the outlined approaches for their practical suitability, IfmPt will develop an evaluation scenario that can be applied to a data set created by IfmPt. IfmPt will make this data set available to the research community on request.

In the context of biographical research, the previous data set comprises 34 CVs of extremist/terrorist perpetrators from the right (18), left (7) and Islamist (9) spectrum:

Age, gender, nationality, migration background, school leaving certificate, education, military service, marital status, children, criminal record, psychological abnormalities, involvement in a radical network, police or intelligence findings on radicalisation, weapons possession card or possession of weapons, was the crime announced in the social media, execution of the crime, accomplices.

Text/image analysis: IfmPt provides the developers with keywords, phrases, symbols and images from the right, left and Islamist spectrum for evaluation.

5.1.5 Future work

Predictive methods of analysis enable the police to act proactively with limited resources in order to prevent crime or to make investigative efforts more effective. However, it must be communicated at all levels of security authorities that the use of predictive police methods is not the same as the famous look into a crystal ball. For a police strategy to be effective, it must produce tangible results. The aim of Prevision is to develop technical tools to optimize tactical crime analysis.

In order to recognize future radicalization tendencies at an early stage, it is necessary to break new ground methodically. It is not only the danger of attacks in the classical sense (e.g. bomb attacks, hijackings, suicide bombings) but also the threat to critical infrastructure (e.g. water and electricity supply, public transport). It is precisely for this purpose that the technical platforms and dependencies offer a risk environment and thus attractive targets for attack from the perspective of the perpetrators.

Computer programs and all algorithms alone will not be able to successfully counter this threat.

In the course of PREVISION, other projects will be checked for similar possibilities to use existing synergies.

5.2 Trend Analysis

5.2.1 Introduction

In addition to the requirements for predictive analytics described in section 5.1, Task 3.4 will also enable trend analyses.

A trend is a systematic increase or systematic decrease in a time series that also shows fluctuation or periodic fluctuations. A trend in a short time series can be fundamental or a temporary trend.

Trends exist wherever a reference value can change and a direction of development emerges from this change. Natural sciences, technology and economics in particular deal with trends in the form of time series. Trends are intended to provide the analyst with information about developments of a reference value within a certain period of time and/or location in order to gain insights into certain processes. The starting point is initially a trend free observation within a period/location. Then the mean value and variance of a time series remain constant over an entire observation period; this is called stationarity. Only non-stationary time series are subject to a trend development and therefore do not have a fixed mean value. A distinction is made between trend stationary time series with deterministic trend and differential stationary time series with purely random stochastic trends. [126]

In PREVISION, trends are primarily developed within the framework of text analyses. This is done in particular in the context of Task 4.2 (Identification of Radicalisation and Terrorist Propaganda). Available and self-created texts from various sources are used as a data basis.

In addition, the outcome of Task 3.2 could serve as a reference or complement to WP4 (Task 4.2), which aims to identify specific jargon authors with malicious intent and the means they use to spread their propaganda, using methods of network analysis and community identification.

5.2.2 Related work

The following statements serve to consider the question of the extent to which modern methods from the fields of machine language processing, corpus and computational linguistics, statistics and machine learning can be used to predict future trends on the basis of keywords and their contexts from available data. In the literature there are a number of examples of the performance of such analyses.

In the following, possible scientific methods and models are presented.

It applies to all of them that the existing data must first be preprocessed/prepared.

The collocation analysis (see below) requires a prior automatic analysis and annotation of the data with regard to morphological and syntactic properties (lemmatization, tagging, dependency parsing, if necessary morphological analysis according to compositional components and inflectional form). Lemmatization describes the abstraction of concrete inflected forms to an agreed basic form. These basic forms are necessary to identify related forms efficiently and automatically when counting trend indicators. The annotation of word parts enables an efficient filtering of the texts and helps to disambiguate, thus enables a faster and cleaner analysis. For a similar reason, the data must be automatically parsed for dependency in order to be able to include the hierarchical syntactical

relationships between the record components in the analysis later. The statistical procedures as well as the semantic analysis require their own data structures, which can be generated from the pre-processed (tagged, parsed) data as required. The processing should be done in environments that are optimized for statistical data analysis and machine language processing. These are mainly R (R Core Team, 2015) and Python with packages like numpy[31] [127].

5.2.2.1 Statistical Methods

5.2.2.1.1 Collocation Analysis

Collocations are combinations of words[32], that often occur together in a text. Frequently, with each other and in a text can and must be different concepts depending on the goal of the partial analysis. The coexistence of two words can manifest itself as pure presence in a window of ±k words, but also in concrete syntactic relationships (hierarchy vs. sequentiality of the words in a sentence). With regard to frequency, different scores can be included in the analysis in addition to the pure frequencies. This serves, among other things, to avoid overestimating combinations that occur randomly frequently due to the frequency of their components. A common measure for this is "pointwise mutual information" ((cf. 1), see also ([128], 178-183), which calculates the ratio of the probability that the components of a collocation occur together in relation to the probability of the independent occurrence of the components.

(1)
$$I_f(w_1, w_2) = \log \frac{f(w_1 w_2)}{f(w_1) \cdot f(w_2)}$$

Let us assume that an adjective and a noun are particularly common. Then a frequent occurrence of the collocation of both is at least less surprising than if adjective and noun are each rather rare. Pointwise mutual information is an established information theory measure, but in the present context it has obvious difficulties. These difficulties stem from the impossibility of estimating probabilities about a population from very few occurrences. One-off, random concurrences become unduly overweight.

This problem is countered with index-based procedures which, on the one hand, rate specific collocations whose relevance can be claimed with some certainty higher and, on the other hand, rate collocations whose information content is low or whose distribution cannot be separated from the unavoidable background noise lower.

The components of such an advanced score and its motivation are described in the following:

- Collocations whose absolute frequency f(w1w2) is high tend to be of greater interest and safer to prove than very selenium collocations. Since word frequencies tend to differ by orders of magnitude,

[31] In linguistics, disambiguation or term clarification refers to the dissolution of linguistic ambiguity. In this process, an unambiguous meaning is established for an expression in its context by syntactic or semantic assignment. (Wikipedia)

[32] In the field of corpus linguistics as well as machine language processing, under
In addition, due to its conceptual ambiguity, the term word for what is referred to in this text was avoided. Instead, we are talking about tokens that denote character sequences separated by punctuation and spaces. In the following text word is used synonymously with token, but also with lemma. For a detailed discussion see [233].

the logarithm log(f(w1w2)) or log(f(w1w2) + 1) is used to avoid collocations that appear only once, since log 1 = 0.

- The more frequent a word is w1, the more frequent its collocations tend to be. This relationship leads to a trivial proportionality of f(w1) and f(wi) for all possible collocation partners wi: f(w1wi) ∼ f(w1). The proportionality constant c1 = f(w1w2)/f(w1) quantifies the attraction that draws w1 into the collocation w1w2. Accordingly, for w2: c2 = f(w1w2)/f(w2). Collocation partners with high values for both c1 and c2 are much more frequent than expected due to the frequency of their components.
- Texts often have very specific topics. This can lead to strong repetitiveness. We do not want to overestimate collocations that are frequent in one or a few texts, but generally very rare. Therefore, the number of texts tA(w1w2) from category A in which the collocation in question occurs is positively reflected in the collocation index.
- In contrast, unspecific collocations can be recognized by the fact that they do not only occur in one category, but in many different ones. Therefore, the number of categories k(w1w2) in which w1w2 occurs inhibits the final score.

Combining all these elements results in the following proposal for a further developed collocation index:

$$I_A(w_1w_2) = \log\left(f(w_1w_2)\right) c_1c_2 \frac{t_A(w_1w_2)}{k(w_1w_2)}$$

5.2.2.1.2 Predictor analysis with Random Forests

Decision trees offer a possibility that is relatively abstract from the linguistic environment. The training of such decision trees can be used on data structures such as the present one to solve classification and regression tasks in order to decide which properties allow to sort the examined objects according to certain specifications. This procedure alone is very susceptible to so-called overfitting, an explanation of the data from itself, so that random fluctuations are given too much weight and are difficult to distinguish from actual predictors and structures. This makes predictions based on this data uncertain. One possible means to solve such problems is the use of Random Forests [129] (for a linguistic application see e.g. [130]). On the basis of resampling and bootstrapping procedures, it is not individual decision trees that are learned, but large quantities, in a sense entire forests. Thus, it is possible to compare the actual influence of many variables and to filter out the decisive ones.

5.2.2.2 Semantic analysis

While a statistical analysis in the narrower sense aims at abstracting linguistic structures to numerical values, the goal of a semantic analysis with the help of statistical and machine learning methods is, so to speak, an abstraction of concrete linguistic forms into the level of meaning. This abstraction makes it possible to classify texts into categories of meaning based on their tokens, even if specific terms are not used in the same way, and to compare the directions of meaning of different texts. Distributional semantics as well as semantic classification and morphological analysis are applied. An automatic morphological analysis of nouns and adjectives (e.g. [131]) is also useful for a similar question. With this method, words can be broken down into their components. For example, if two very complex words have a high overlap in their components, it can be assumed that they are at least related in meaning. Examples

1 and 2 describe construction-specific properties. By means of a morphological analysis, a correspondence of the last two components (morphemes) can be determined. By means of an overlap index such similarities would be measurable.

(1) Leicht|bau|weise
(2) Fertig|bau|weise

It would also be necessary to examine which word elements occur particularly frequently. However, such an analysis should be carried out semi-automatically to ensure a certain quality of the results. Especially for German, a morphological analysis is promising, since the underlying word formation process of many technical terms is mainly the composition typical for German. Under certain circumstances, tools from the field of Named Entity Recognition can also provide additional support. By analysing the distribution of the morphological individual components, the findings of the semantic analysis can also be meaningfully and methodically supplemented.

5.2.2.3 Keyword analysis

Keywords represent the course of a topic and can return key content (see [128]). The connection of keywords is not static. The meaning of keywords changes depending on the temporal distance of observation and the course of topic development. The goal of keyword analysis is to identify and reflect precisely these relationships. The correlation is used as a measure of the significance of a keyword for a topic. If correlations of keywords to a topic are recognizable, they reflect events occurring in them. In the context of trend research, particularly frequently occurring keywords can be an indication of a trend or a break in a trend.

5.2.3 Proposed method

In this paper we deal with the problem of trend analysis in news and texts from freely available sources (e.g. Internet, social media) by adapting existing methods and models to the needs of LEAs. Already developed solutions from corresponding projects (e.g. Tensor) will also be used.

5.2.4 Evaluation framework

In order to evaluate the proposed approach, we will develop an evaluation scenario that we will run on a data set we have created. We will make this data set available to the research community on request.

5.2.5 Future work

In future work, a promising module will be developed in consultation with the partners. This module will take into account the other existing modules and various PREVISION use cases in which the module could be used. Furthermore, we will develop a WP 5-compliant component. The evaluation will initially be based on use cases and data that simulate those of the LEAs.

5.3 Regression Trees and Boosting Algorithms

5.3.1 Overview

Considering the proposal of PREVISION, “This task aims to exploit the state-of-the-art machine learning techniques (both supervised and unsupervised algorithms) in order to perform predictive and trend

analysis to the fused data from Task3.2. Specifically, predictive analysis will rely on machine learning algorithms to predict behavior. Such algorithms include deep learning, random forest or learning to rank algorithms (RankLib or SVM_rank toolkits). Furthermore, non-supervised machine learning will be used to detect trends, groups, networks of terms/individuals/locations."

In this section we are proposing two steps for the creation of predictive analytics and trend analysis. First, we are proposing the usage of Regression Trees. Next considering the complexity of the problems, we are proposing a category of ensemble learning systems, specifically boosting algorithms. From the boosting algorithms we have chosen the Gradient Boosted Regression Trees (GBRT). [132]. It works by building a series of base estimators sequentially, with the bias of the combined estimator reduced gradually from one predecessor to another successor. This high flexibility makes the GBRTs highly customizable to any particular data-driven task. It introduces a lot of freedom into the model design thus making the choice of the most appropriate loss function a matter of trial and error. However, boosting algorithms are relatively simple to implement, which allows one to experiment with different model designs.

5.3.2 Regression Trees

Classification and Regression Trees or **CART** for short is a term introduced by Leo Breiman to refer to Decision Tree algorithms that can be used for classification or regression predictive modeling problems. Classically, this algorithm is referred to as "decision trees", but on some platforms like R they are referred to by the more modern term CART. The CART algorithm provides a foundation for important algorithms like bagged decision trees, random forest and boosted decision trees.

In PREVISION, regression trees are used to perform predictive and trend analysis. The full description of how decision trees will be implemented in PREVISION is presented in the previous chapter 4.5.5.2. We are mentioning here just the differences from using decision tree for the regression process instead of classification as described previously.

Regression trees works similarly to classification trees; however, instead of reducing Gini impurity or entropy, potential splits are by default measured on how much they reduce mean squared error (MSE):

$$\text{MSE}=\frac{1}{n}\sum_{1}^{n}(y_i - yp_i)^2$$

Where y_i is the true value of the target and yp_i is the predicted value.

As we are using scikit-learn, the regression can be conducted using `DecisionTreeRegressor`.

For reading data we plan to use pandas, and csv files.

There will be three major steps:

- Training
- Predicting
- Plotting results

The training process will follow the steps:

```
#loading the data from csv files. For example:

train_data       =       pd.read_csv("C:/data/siveco/PREVISION_ic.csv",
low_memory=False)


#plot the data

features = ['ImageScore','TextScore']

targets = ['real','fake']

X = train_data[features].values # matrix

y = train_data['RealExpected'].values

plt.plot(X[y==0, 1], X[y==0, 0], 'r*', markersize=3, label="spoof")

plt.plot(X[y==1, 1], X[y==1, 0], 'go', markersize=2, label="real")

plt.show

#create decision tree regressor

Decisiontree = DecisionTreeRegressor(max_depth=2 , criterion = "mse",
random_state=0);
#train the model

model = decisiontree.fit(features, target);
```

The prediction process will follow the steps:

```
#loading the data from csv files. For example:

predicted_data   =   pd.read_csv("C:/data/siveco/PREVISION_predict.csv",
low_memory=False)

#predict the value

Res1=model.predict(predicted_data)

# Plot the results

plt.figure()

plt.scatter(X, y, s=20, edgecolor="black",

            c="darkorange", label="data")
```

```
plt.plot(X_test, Res1, color="cornflowerblue",
         label="max_depth=2", linewidth=2)
plt.xlabel("data")
plt.ylabel("target")
plt.title("Decision Tree Regression")
plt.legend()
plt.show()
```

The training process and the prediction are working using different parameters. The decision on what the used values depends on real data. Initially we are using the default values, then we play with criterion and max_depth, then we play with other parameters.

The full list of parameters with their interpretation is presented below. [121]

criterion{"mse", "friedman_mse", "mae"}, default="mse"

The function to measure the quality of a split. Supported criteria are "mse" for the mean squared error, which is equal to variance reduction as feature selection criterion and minimizes the L2 loss using the mean of each terminal node, "friedman_mse", which uses mean squared error with Friedman's improvement score for potential splits, and "mae" for the mean absolute error, which minimizes the L1 loss using the median of each terminal node.

splitter{"best", "random"}, default="best"

The strategy used to choose the split at each node. Supported strategies are "best" to choose the best split and "random" to choose the best random split.

max_depth int, default=None

The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

min_samples_split int or float, default=2

The minimum number of samples required to split an internal node:

If int, then consider `min_samples_split` as the minimum number.

If float, then `min_samples_split` is a fraction and `ceil(min_samples_split * n_samples)` are the minimum number of samples for each split.

min_samples_leaf int or float, default=1

The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least `min_samples_leaf` training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression.

If int, then consider `min_samples_leaf` as the minimum number.

If float, then `min_samples_leaf` is a fraction and `ceil(min_samples_leaf * n_samples)` are the minimum number of samples for each node.

`min_weight_fraction_leaf float, default=0.0`

The minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node. Samples have equal weight when sample_weight is not provided.

`max_features` `int, float or {"auto", "sqrt", "log2"}, default=None`

The number of features to consider when looking for the best split:

If int, then consider `max_features` features at each split.

If float, then `max_features` is a fraction and `int(max_features * n_features)` features are considered at each split.

If "auto", then `max_features=n_features`.

If "sqrt", then `max_features=sqrt(n_features)`.

If "log2", then `max_features=log2(n_features)`.

If None, then `max_features=n_features`.

Note: the search for a split does not stop until at least one valid partition of the node samples is found, even if it requires to effectively inspect more than `max_features` features.

`random_state` `int or RandomState, default=None`

If int, random_state is the seed used by the random number generator; If RandomState instance, random_state is the random number generator; If None, the random number generator is the RandomState instance used by `np.random`.

`max_leaf_nodes` `int, default=None`

Grow a tree with `max_leaf_nodes` in best-first fashion. Best nodes are defined as relative reduction in impurity. If None then unlimited number of leaf nodes.

`min_impurity_decrease float, default=0.0`

A node will be split if this split induces a decrease of the impurity greater than or equal to this value.

The weighted impurity decrease equation is the following:

```
N_t / N * (impurity - N_t_R / N_t * right_impurity
                    - N_t_L / N_t * left_impurity)
```

where `N` is the total number of samples, `N_t` is the number of samples at the current node, `N_t_L` is the number of samples in the left child, and `N_t_R` is the number of samples in the right child.

`N`, `N_t`, `N_t_R` and `N_t_L` all refer to the weighted sum, if `sample_weight` is passed.

5.3.3 Gradient Boosted Regression

Gradient Boosted Regression is a kind of ensemble learning, with the base estimators belonging to the same class.

Gradient Boosting works by sequentially adding predictors to an ensemble, each one correcting its predecessor. The method tries to fit the new predictor to the residual errors made by the previous predictor.

GB builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage a regression tree is fit on the negative gradient of the given loss function.

In PREVISION we are using from scikit-lear the **`GradientBoostingRegresor`** component.

The **`GradientBoostingRegresor`** will be trained on the same data as **`CART,`** but in a loop, until no improvement will be found.

We will start by using default parameters. Then we will vary max_depth, then n_estimators, then other parameters.

The full list of parameters with their interpretation is presented below. [121]

`loss``{'ls', 'lad', 'huber', 'quantile'}, optional (default='ls')`

loss function to be optimized. 'ls' refers to least squares regression. 'lad' (least absolute deviation) is a highly robust loss function solely based on order information of the input variables. 'huber' is a combination of the two. 'quantile' allows quantile regression (use `alpha` to specify the quantile).

`learning_rate` `float, optional (default=0.1)`

learning rate shrinks the contribution of each tree by `learning_rate`. There is a trade-off between learning_rate and n_estimators.

`n_estimators` `int (default=100)`

The number of boosting stages to perform. Gradient boosting is fairly robust to over-fitting so a large number usually results in better performance.

Subsample float, optional (default=1.0)

The fraction of samples to be used for fitting the individual base learners. If smaller than 1.0 this results in Stochastic Gradient Boosting. subsample interacts with the parameter n_estimators. Choosing subsample < 1.0 leads to a reduction of variance and an increase in bias.

Criterion string, optional (default="friedman_mse")

The function to measure the quality of a split. Supported criteria are "friedman_mse" for the mean squared error with improvement score by Friedman, "mse" for mean squared error, and "mae" for the mean absolute error. The default value of "friedman_mse" is generally the best as it can provide a better approximation in some cases.

min_samples_split int, float, optional (default=2)

The minimum number of samples required to split an internal node:

If int, then consider min_samples_split as the minimum number.

If float, then min_samples_split is a fraction and ceil(min_samples_split * n_samples) are the minimum number of samples for each split.

min_samples_leaf int, float, optional (default=1)

The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least min_samples_leaf training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression.

If int, then consider min_samples_leaf as the minimum number.

If float, then min_samples_leaf is a fraction and ceil(min_samples_leaf * n_samples) are the minimum number of samples for each node.

min_weight_fraction_leaf float, optional (default=0.)

The minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node. Samples have equal weight when sample_weight is not provided.

max_depth integer, optional (default=3)

maximum depth of the individual regression estimators. The maximum depth limits the number of nodes in the tree. Tune this parameter for best performance; the best value depends on the interaction of the input variables.

min_impurity_decrease float, optional (default=0.)

A node will be split if this split induces a decrease of the impurity greater than or equal to this value.

The weighted impurity decrease equation is the following:

```
N_t / N * (impurity - N_t_R / N_t * right_impurity
                    - N_t_L / N_t * left_impurity)
```

where `N` is the total number of samples, `N_t` is the number of samples at the current node, `N_t_L` is the number of samples in the left child, and `N_t_R` is the number of samples in the right child.

`N`, `N_t`, `N_t_R` and `N_t_L` all refer to the weighted sum, if `sample_weight` is passed.

`min_impurity_split` `float, (default=1e-7)`

Threshold for early stopping in tree growth. A node will split if its impurity is above the threshold, otherwise it is a leaf.

Deprecated since version 0.19: `min_impurity_split` has been deprecated in favor of `min_impurity_decrease` in 0.19. The default value of `min_impurity_split` will change from 1e-7 to 0 in 0.23 and it will be removed in 0.25. Use `min_impurity_decrease` instead.

`Init` `estimator or 'zero', optional (default=None)`

An estimator object that is used to compute the initial predictions. `init` has to provide fit[33] and predict[34]. If 'zero', the initial raw predictions are set to zero. By default, a `DummyEstimator` is used, predicting either the average target value (for loss='ls'), or a quantile for the other losses.

`random_state` `int, RandomState instance or None, optional (default=None)`

If int, random_state is the seed used by the random number generator; If RandomState instance, random_state is the random number generator; If None, the random number generator is the RandomState instance used by `np.random`.

`max_features` `int, float, string or None, optional (default=None)`

The number of features to consider when looking for the best split:

If int, then consider `max_features` features at each split.

If float, then `max_features` is a fraction and `int(max_features * n_features)` features are considered at each split.

If "auto", then `max_features=n_features`.

If "sqrt", then `max_features=sqrt(n_features)`.

[33] https://scikit-learn.org/stable/glossary.html#term-fit
[34] https://scikit-learn.org/stable/glossary.html#term-predict

If “log2”, then `max_features=log2(n_features)`.

If None, then `max_features=n_features`.

Choosing `max_features < n_features` leads to a reduction of variance and an increase in bias.

Note: the search for a split does not stop until at least one valid partition of the node samples is found, even if it requires to effectively inspect more than `max_features` features.

`Alpha` `float (default=0.9)`

The alpha-quantile of the huber loss function and the quantile loss function. Only if `loss='huber'` or `loss='quantile'`.

`Verbose` `int, default: 0`

Enable verbose output. If 1 then it prints progress and performance once in a while (the more trees the lower the frequency). If greater than 1 then it prints progress and performance for every tree.

`max_leaf_nodes` `int or None, optional (default=None)`

Grow trees with `max_leaf_nodes` in best-first fashion. Best nodes are defined as relative reduction in impurity. If None then unlimited number of leaf nodes.

`warm_start` `bool, default: False`

When set to `True`, reuse the solution of the previous call to fit and add more estimators to the ensemble, otherwise, just erase the previous solution. See the Glossary[35].

`validation_fraction` `float, optional, default 0.1`

The proportion of training data to set aside as validation set for early stopping. Must be between 0 and 1. Only used if `n_iter_no_change` is set to an integer.

`n_iter_no_change` `int, default None`

`n_iter_no_change` is used to decide if early stopping will be used to terminate training when validation score is not improving. By default, it is set to None to disable early stopping. If set to a number, it will set aside `validation_fraction` size of the training data as validation and terminate training when validation score is not improving in all of the previous `n_iter_no_change` numbers of iterations.

`Tol` `float, optional, default 1e-4`

Tolerance for the early stopping. When the loss is not improving by at least tol for `n_iter_no_change` iterations (if set to a number), the training stops.

[35] https://scikit-learn.org/stable/glossary.html#term-warm-start

ccp_alpha non-negative float, optional (default=0.0)

Complexity parameter used for Minimal Cost-Complexity Pruning. The subtree with the largest cost complexity that is smaller than ccp_alpha will be chosen. By default, no pruning is performed.

5.4 Estimating Information Check-Worthiness

5.4.1 Introduction

PREVISION will provide means of detecting potential risks but it can lead to an overflow of alerts. It thus could be important to be able to grade the alarms in order to decide which are check-worthy. This CNRS-IRIT contribution tackles this issue.

A very similar problem arises in the case of information check-worthiness in fake news detection. Indeed, information technologies, including social media eases information spreading, makes information diffusion quicker, and reaches potentially more people than traditional media in many cases regardless of the information quality [133]. In that case, one of the challenges is to detect which claim should be prioritized for fact-checking. This challenge will thus be used by CNRS-IRIT to design and develop an approach to answer the PREVISION challenge of detecting which alarms should be prioritized for checking.

The originality of CNRS-IRIT approach resides in the way we represent the information. It relies on a variety of features, including the Information Nutritional Label for online documents [134]. This label was initially introduced to “help readers making more informed judgments about the items they read." The nutritional label provides scores for various criteria to describe the content of a written text. We combine the features from the information nutritional label with POS-tags or word-embedding representations and learn a machine learning model to predict the information check-worthiness.

5.4.2 Related Work

Related work on automated fact-checking mainly focuses on verifying the veracity of claims; a few studies address the challenge of identifying check-worthy statements. One of such first studies was ClaimBuster [135] that extracts check-able claims, classifies, and ranks the check-worthiness of these claims. The authors used the transcripts of all of the 30 US presidential debates until 2012 that were manually annotated as Non-Factual Sentence (NFS), Unimportant Factual Sentence (UFS), or Check-worthy Factual Sentence (CFS). The authors used an SVM with sentence-level features such as sentiment, length, TF-IDF, POS-tags, and Entity Types. They achieved an average precision (AP) of 0.223 for the top-100 sentences. One possible improvement of the ClaimBuster system was to consider the context of the claim to evaluate. Gencheva et al. [136] integrated several context-aware and sentence-level features to train both SVM and Feed-forward Neural Networks. This approach outperforms the ClaimBuster system in terms of MAP and precision. CheckThat! Lab at CLEF 2018 is the most recent challenge on this topic. Several teams participated, including Prise de Fer [137], Copenhagen [138], bigIR [139], UPV-INAOE-Autoritas [140], and RNCC [141]. The best performing system is Prise de Fer [137] that obtained a MAP score of 0.133, outperforming the second best system [138] by 18%. The authors represented the sentence using word-embedding combined with POS-tags, syntactic dependencies, and some new features including named entities, sentiment, and verbal forms. With gathering external training data, this sentence representation was used as input to train a multi-layer perceptron (MLP) which is composed of two hidden layers (with

100 units and 8 units, respectively) and the hyperbolic tangent (tanh) as an activation function. The other participants used different representations such as character n-grams [140] or topics [139]. The participants used different machine learning (ML) algorithms such as SVM [141], Random Forest [141], k-nearest neighbors [140] , or Gradient boosting [139]. In this paper, we investigate combining different types of features in the text representation, including some from the information nutritional label [134].

5.4.3 A variety of features to learn check-worthiness

5.4.3.1 Information nutritional label features

The Information Nutritional Label for Online Documents [134] aims at helping on-line information users in their information "consumption." The proposed label describes a textual information unit according to the following criteria: (1) **Factuality**: the number of facts it mentions, (2) Readability: the ease with which a reader can understand it, (3) Virality: the speed at which it is propagated, (4) **Emotion**: its emotional impact, (5) Opinion: the number of opinionated sentences it contains, (6) **Controversy**: the number of controversial issues it addresses, (7) Authority/Trust/Credibility: its credibility and the authority and trust of the source it belongs to, (8) **Technicality**: the number of technical issues it addresses and technical terms used, and (9) Topicality: its current interest which is time-dependent.

Although the information nutritional label aims at guiding the information reader only, we hypothesize that part of it could be used to decide whether a piece of information should be prioritized for checking or not. From the initial label, we have not used all the criteria since some are not straightforward applicable to the political transcribes that compose the data set we use (e.g. readability, topicality, or authority). We kept the ones that are described below.

5.4.3.1.1 Factuality

We develop two feature variants for the factuality as follows:

- **Factuality_Proba** that computes the probability of a sentence to be representative of a fact.
- **Factuality_Strict** is a binary feature which considers a sentence as either a fact (1) or an opinion (0). This feature is 1 if Factuality_Proba is ≥ 0.5 and is 0 otherwise.

These two features are extracted using Matatusko's classifier[36]. It is based on a multi-layer perceptron (MLP) using LBFGS gradient descent [142]. To train the classifier, we collected the data from Wikipedia articles for factual sentences (World_War_I, Industrial_Revolution, October_Revolution, Fermi_paradox, Steam_engine, Barack_Obama, Amazon (company), Netherlands, Triangular_trade, Song_dynasty, Nanking_Massacr, The_Holocaust) and from the Opinosis site[37] for opinion sentences. To represent a sentence as features, given each word of the sentence, we estimate the occurrence of POS tags, entity types, and dependency tags, adapted from the spacy annotation.

[36] https://github.com/matatusko/opinion-or-fact-sentence-classifier
[37] http://kavita-ganesan.com/opinosis

5.4.3.1.2 Emotion

We hypothesize that a sentence with a high emotional level might be used to deceive the “consumer" towards accepting false information and thus that it should be checked. We use the list of 2, 477 emotional words with evaluation from AFINN[38] [143], for example, abusive (-3), proud (2), etc. We develop three feature variants for the emotion criterion as follows:

- **Emotion_P** (resp. Emotion_N) is the sum of the positive (resp. negative) rating of the words in the sentence.
- **Emotion_U** considers both positive and negative emotions to get an overall level by summing the absolute value of the positive and negative rating of the words in the sentence.

5.4.3.1.3 Controversy

We hypothesize that a sentence addressing a controversial issue is more likely to be worth checking. To estimate the controversy level of a sentence, we count the number of controversial issues in the sentence based on the controversial entries in Wikipedia article “Wikipedia:List_of_controversial_issues." For each controversial issue, we also take into account the anchor text labels to find the synonyms and other designations of the issues. Thus, we build a dictionary of the controversial noun phrases which is used to estimate the controversy level of the sentence.

5.4.3.1.4 Technicality

We hypothesize that technical words will be associated less with check-worthiness. To estimate this criterion, we develop two feature variants by counting the number of domainspecific words found in a sentence in two different ways as follows:

- **Technicality_RE** uses a specific regular expression defined in [144] to find domain-specific words. First, we use NLTK [145] for POS tagging; then, we identify the terminological noun phrases (NPs) using the Python regular expression library. NPs represents domain-specific words. We calculate the number of these NPs that occur more than once.
- **Technical_List** uses technical words from the Academic Vocabulary Lists[39]. There are about 8,000 words that occur at least three times more frequently than expected in one of the nine COCA-Academic domains (e.g. Law, Medicine or Technology). We count the number of technical words in the sentence. All the features are normalized by dividing the feature value by the sentence length.

5.4.3.2 Features based on word embedding and spaCy

5.4.3.2.1 Word embedding

Word embedding refers to the representation of a word in semantic space as a vector of numerical values. Words that are semantically and syntactically similar tend to be close in this embedding space. “Word vectors" was trained on GoogleNews corpus using Word2Vec model [146]. We average the word vectors of every word in a sentence. When we could not find a word in the model, we represent it with a zero

[38] http://www2.imm.dtu.dk/pubdb/p.php?6010
[39] https://www.academicvocabulary.info

vector. Although zero vector affects the mean [147], this is indeed essential when we could not find any word of the sentence in the model.

5.4.3.2.2 SpaCy annotations

We use fine-grained POS tags, syntactic dependency tags, and the entities from a sentence as features using the information extraction library, spaCy[40]. To extract these features, we collect all the POS tags, dependency tags, and entity types mentioned in SpaCy. Then, for each word within a sentence, we measure the number of occurrences of all the collected tags. Simpler methods could also be used to extract key-phrases [148], but we keep this for future work.

5.4.4 Evaluation

5.4.4.1 Data collection

The collection used to evaluate our model is the one from CLEF18 CheckThat! 2018 share task (CT-CWC-18) [149]. It is composed of the transcriptions of political debates or speeches from the 2016 US Presidential campaign. The "CT-CWC-18" collection is divided into a training and test set. Each line in a transcription file consists of the line number (LN), the Speaker name, the transcription of the statement that the Speaker said, and for the training set a label indicating whether this statement is check-worthy (1) or not (0). The training set contains the Presidential debates (first and second) and the Vice-Presidential debate while the test set consists of the third Presidential and ninth Democratic debates along with five of Donald Trump's speeches.

Table 11 shows some statistics of the training and testing sets [149].

5.4.4.2 Evaluation measures

The evaluation measures used include the mean average precision (MAP) which was the official measure for the CLEF track [149] mean reciprocal rank (MRR), mean R-precision (MRP), and mean precision at 5 (MP@5). We can use the scripts provided by the CheckThat! Lab organizers[41].

Table 11. CT-CWC-18 collection: number of sentences (#Sent.) and the number of check-worthiness (#CW) on the training and test sets

	Type	Set	#Sent.	#CW
Debates	First Presidential	Train	1,403	37
	Second Presidential	Train	1,303	25
	Vice-Presidential	Train	1,358	28
	Third Presidential	Test	1,351	77
	Ninth Democratic	Test	1,464	17
Speeches	Donald Trump Acceptance	Test	375	21
	Trump at the World Economic Forum	Test	245	11
	Trump at a Tax Reform Event	Test	412	16
	Trump's Address to Congress	Test	390	15
	Trump's Miami Speech	Test	645	35
	Total		**8,946**	**282**

[40] https://spacy.io/

[41] http://alt.qcri.org/clef2018-factcheck

5.4.4.3 Various ML algorithms

Various ML algorithms can be used such as Random Forest (RF), Support vector machine with linear kernel (SVM_Linear), Support vector machine with non-linear kernel (SVM_RBF), Neural network (MLP_LBFGS), and Logistic regression (SGD_Logloss). These ML algorithms can be categorized as linear and non-linear models. Linear models are SGD_Logloss, SVM_Linear, and SVM_RBF. The non-linear models are Random forest (RF) and MLP_LBFGS.

The details of the implementation of the ML models are described as follows:

- SGD_Logloss: A stochastic gradient descent classifier training using ``logloss" function (AKA, Logistic regression) while using the default values of the other parameters from Scikit-learn (0.22.1),
- SVM_Linear: A support vector machine classifier with linear kernel considering the default parameters from Scikit-learn (0.22.1),
- SVM_RBF: A support vector machine classifier with Radial basis function (RBF) kernel while keeping the default parameters from Scikit-learn (0.22.1),
- Random Forest (RF): A random forest classifier with 100 trees considering the default parameter settings from Scikit-learn (0.22.1), and
- MLP_LBFGS: Multi-layer perceptron classifier where the configuration is one hidden layer with 100 units, the Relu activation function, and the ``log" loss function using LBFGS optimizer while keeping the default parameters from Scikit-learn (0.22.1).

Moreover, the distribution of the two classes (Check-worthy or not) in the training set (Table 11) is unbalanced which will probably be the case in the LEA data sets as well. To balance the distribution, we can apply the oversampling on the training set with the SMOTE algorithm [150].

6 Multivariate Behavioural Analysis and Anomaly Detection

This chapter aims to identify and report on the implementation of the appropriate machine learning techniques that will provide LEA officers with automated capabilities for detecting behavior changes, abnormality, outliers as well as weak signals and early changes that may be a signal of new trends in textual data, video data, financial data as well as in telecommunication and network data. The pre-processed and homogenized data in WP2 will be used together with machine learning techniques, in order to perform the multivariate behavioral analysis and detect abnormalities and new trends.

6.1 Behavioural Analysis and Anomaly Detection tool for text analysis and sentiment and radicalization detection

In this section, we present the complementary contributions from BPTI & CNRS-IRIT regarding text analysis. The model which is presented in Section 6.1.1 is illustrated for aggression identification but it is general enough so that it can be adapt to other tasks (e.g. radicalization detection, hate speech detection, ...) given appropriated annotated data and linguistic resources. In the same way the visual tolls presented in Section 6.1.2.

6.1.1 Aggression Identification in Posts - two machine learning approaches

6.1.1.1 Introduction

Within PREVISION identification of aggressive content can help in identifying outlawed aggressive content and provide some cues in cases of post physical aggression.

A **very similar problem** that has been tackle in different recent studies is the **detection of aggressive content in social media**. Indeed, social media have changed the way people communicate, [151], [152], [153], [154], One of these aspects is cyber-aggression and interpersonal aggression that can be catalysed by perceived anonymity [155]. Automatically monitoring user-generated content in order to help moderating social media is thus an important although difficult topic that researchers are trying to automate [156], [157]

In 2018, the Shared Task on Aggression Identification was organized as part of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC - 1) at COL- ING 2018, [158]. The objective of this task is to detect aggressive content and the level of aggressiveness. Thirty teams submitted their test runs.

CNRS-IRIT suggests two alternative models to answer the aggression identification task. The first model uses random forest and linear regression which can be considered as relatively mature approaches while the second model combines CNN and LSTM recent deep learning techniques. No strong conclusion could be made on the superiority of one or the other model since it depends on the collection.

6.1.1.2 Related work

Approaches based on features and supervised classifiers such as SVMs are often used in order to learn to detect whether a text contains aggressiveness [159] in recent years, deep learning has been also employed for this task, [160], [161].

Deep learning has also been used by TRAC challenge participants. TRAC challenge [162], is the first that focuses on detecting aggressive text. The task training set is composed of Facebook posts/comments; there is also two kinds of test sets: one from Facebook and another from Twitter. Among the thirty participants, Saroyehun [161], obtained the best results. The authors investigated the efficacy of deep neural network by experimenting different models: CNN, LSTM, BiLSTM, and combinations thereof. In their experiments they used translation technique to enlarge the training set and added an external dataset on hate speech[42]. The LSTM model which was trained on the augmented training set only, achieved the best weighted F1 score of 0.6425 on Facebook test set; it is the first ranked system on TRAC challenge; the same system does not perform as well on the Twitter data set. The other system of the same team which implements a combination of CNN and LSTM and which was trained on the augmented training set and the additional dataset, achieved a weighted F1 score of 0.5920 and the third rank on the twitter test set.

Raiyani *et. al.* [163], meanwhile, tested different models for text classification in TRAC, from classic machine learning model to deep learning models. At the end, they kept three models: FastText model, Dense neural networks, and Voting of the two. The Dense neural networks give better performance than the two others and achieved a weighted F1 score of 0.5813 on Facebook test set; it is the fourteenth rank on TRAC challenge. While it achieved the best weighted F1 score of 0.6009 and the first rank on the twitter test set, although it was trained on a Facebook dataset.

6.1.1.3 Machine learning based model

CNRS-IRIT conceived two supervised machine learning based models that we present in this section. CNRS-IRIT has also evaluated them on the TRAC shared task which is a very similar problem than the ones we can find in PREVISION as mentioned in the introduction of this section. The first method combines random forest and logistic regression while the second approach is deep learning based. We also developed a model based on CNN only for which results can be found in [164] it performs in between the two models reported hereafter.

6.1.1.3.1 RF LR: combination of two classifiers

This model combines random forest (RF) based on surface features and linguistic features with logistic regression (LR) based on document vectorization. We chose this combination because a combination of multiple machine learning models placed first in many prestigious machine learning competitions [165], such as Netflix Competition, Kaggle, etc. Moreover, when using non-combined models on the training dataset, the results were lower in the case of TRAC as well and this was confirmed on the test set.

RF Classifier. The random forest model uses different features extracted from the comments as presented in Table 12. Some are adapted from [166], [167], where the authors tried to detect depression from texts; another source of inspiration is [168] where the authors suggested an information nutritional label for describing text qualities.

[42] https://github.com/ZeerakW/hatespeech, accessed on January 10, 2020

Table 12. List of features used in FR to represent texts (Facebook comments and tweets)

Name	Hypothesis or tool/resource used
Part-of-speech frequency	Normalized frequencies of each tag: adjectives, verbs, nouns and adverbs (four features).
Negation	Normalized frequencies of negative words like*: no, not, didn't, can't, ...* The idea behind is to detect non direct aggressiveness.
Capitalized	The idea behind is that aggressive texts tend to put emphasis on the target they mention. It can indicate feelings or speaking volume.
Punctuation marks	! or ? or any combination of both can emphasize offensiveness of texts.
Emoticons	Another way to express sentiment or feeling.
Sentiment	Use of NRC-Sentiment-Emotion-Lexicons to trace the polarity in text.
Emotions	Frequency of emotions from specific categories: anger, fear, surprise, sadness and disgust. The idea behind is to check the categories related to aggressiveness.
Gunning Fog Index	Estimate of the years of education that a person needs to understand the text at first reading.
Flesch Reading Ease	Measure how difficult to understand a text is.
Linsear Write Formula	Developed for the U.S. Air Force to calculate the readability of their technical manuals.
New Dale-Chall Readability	Measure the difficulty of comprehension that persons encounter when reading a text. It is inspired from Flesch Reading Ease measure.
Swear words	The intuition behind is that the texts containing insults are often aggressive.
Lexical analysis with python library empath	Empath is a tool for analysing text across lexical categories. By default, it has 194 lexical categories and each category is considered as feature.

Some of these features are used for abusive language detection, hate speech, cyber- bullying and the others are used for sentiment or personality analysis that we judged useful for aggression detection.

A RF classifier was trained on train and validation sets by representing each text (Facebook comment or tweet) with a vector composed by the features we mentioned in Table 12.

The following parameters were used during the training: class weight="balanced", max features="sqrt", n estimators=60, min weight fraction leaf=0.0, criterion='entropy', random state=2.

At prediction time, a text from the test set is represented with features and then run the trained model. The output is the estimated probabilities for the three classes (overtly aggressive, covertly aggressive and non-aggressive).

LR Classifier. This model is based on document vectorization using *Doc2vec* [169]. *Doc2vec* is used to represent sentences, paragraphs, or whole documents as vectors and it can be trained on small corpora, which is case of the task datasets.

Before building the LR Classifier, we first trained two separate *Doc2vec* models: a Distributed Bag of Words and a Distributed Memory model [169]. For the training, we used the same configuration as in [170], for representing user's text. The two Doc2vec models were trained on the train and validation sets. We used the Python package *gensim*[43] [171]. We also concatenated the output vectors of these two models, as done in [170] resulting in a representation by a 200-dimension vector per text.

Then a logistic regression classifier was trained on the vectors for both the train and validation sets with the following parameters: class weight="balanced", random state=1, max iter=100, solver="liblinear".

At prediction time, the texts from the test set were vectorized by using the two *Doc2vec* models and the 200-dimension vectors were given as input of trained classifier. The output is also a set of class probabilities.

Combination of two classifiers. The class probabilities obtained from RF classifier and LR Classifier were averaged and finally the class with the highest probability was considered as the class the text belongs to. We also tested different ways to combine the output probabilities obtained from the two classifiers RF and LR, such as maximum, minimum, etc., but the average method gave the best results.

6.1.1.3.2 CNN LSTM: Combination of CNN and LSTM

This model combines two deep learning techniques: CNN and LSTM. The main idea is to pass input representation (sentence matrix in Figure 34) to the CNN and pass the local features learnt by the CNN (concatenated vectors in Figure 34 to the LSTM. Indeed, CNN and LSTM are complementary due to the fact that each of them captures information at different scales [161].

The architecture of our combined model is illustrated in Figure 34 It is as follows: first, we convert sentences/texts into *sentences matrix*[44] where each row is a vector representation[45] of each word in the

[43] https://radimrehurek.com/gensim/index.html

[44] The dimension of a sentence matrix is *l x d*, where *l* is the length of the longest text/sentence in the dataset and *d* is the dimension of word vector representation.

[45] The word vector representation is obtained with word2vec model [21] trained on the training and validation sets.

sentences/texts. Then, convolutions are applied on the sentences matrix where we used three filter region sizes: bigrams (height = 2), trigrams (height = 3) and quadrigrams (height = 4). Each region has 100 filters; thus, in total there are 300 filters. The result of convolutions is called feature maps; vectors with variable-length according to the region filter and each filter region has 100 feature maps. Afterwards, a 1-max pooling is performed over feature maps. More precisely, for each region the largest number from each feature map is kept and then concatenated to form a vector. As a result, we obtain one vector of size 100[46] per region filter. Then, these three vectors are concatenated to form a feature vector and a dropout is applied on this feature vector. The concatenated feature vector is passed to the LSTM layer. Then, we added one fully connected hidden layer to reduce the dimension of the concatenated vector, followed by a dropout. Finally, an output layer, which is also a fully connected layer with three possible output states, is added. On the output layer, the activation function used is the softmax function.

The architecture of our model is inspired from the CNN architecture Zhang *et. al.* [172] proposed and which is used for sentences classification. In that task, their CNN architecture outperforms baseline methods which use SVM as well as the one that used CNN in [173].

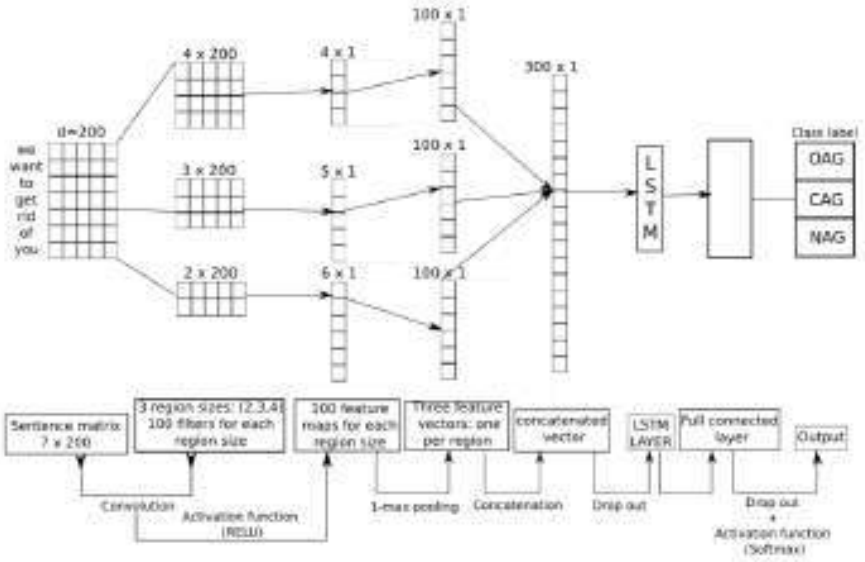

Figure 34. Illustration of a CNN+LSTM architecture of aggression detection inspired from [172].

[46] Because there are 100 feature maps.

6.1.1.4 Evaluation

6.1.1.4.1 Data Set

The evaluation presented below is based on the TRAC 2018 shared task [158] which we believe can provide a good insight of what could be the results on PREVISION confidential data.

The task dataset is a subset of Kumar et al' [162] and consists in English randomly sampled Facebook comments. In this study, we focused on the English part while a Hindi part also exists. The English part detailed in Table 13 is composed of (a) 11,999 Facebook comments for training and 3,001 comments for validation. It is annotated with 3 levels of aggression - Overtly Aggressive (OAG), Covertly Aggressive (CAG) and Non- Aggressive (NAG), (b) 916 English comments for test. Additionally, 1,257 English tweets were given as a second test set 2.

Table 13. Distribution of training, validation and testing data on TRAC 2018 data collection

Number of	Train	Validation	Test	
			Facebook	Twitter
texts (=posts+comments)	11,999	3,001	916	1,257
Overt aggression	2,708	711	144	361
Covert aggression	4,240	1,057	142	413
No aggression	5,051	1,233	630	483

6.1.1.4.2 Evaluation measure

The evaluation metric used in this paper is the weighted F1 which was also used in the TRAC shared task. The weighted F1 is equal to the average, weighted by the number of instances for each label, of the F1 (given by Equation 1) of each class label.

$$F1 = 2\frac{R * P}{R + P} \quad (1)$$

where $P = \frac{tp}{tp+fp}$ is the precision, $R = \frac{tp}{tp+fn}$ is the recall, tp denotes the true positives, fp the false positives, and fn the false negatives.

Equation 1

6.1.1.4.3 Results

Table 14 reports the results we obtained with the two models presented above. For comparison, we report also results obtained with the RF classifier only and with the LR classifier only. The baseline mentioned in the first row was given by the TRAC shared task organizers while the second row is the best result from participants in the TRAC workshop.

We can see that our two models outperform the baseline on both Facebook and Twitter subsets. Trac-RF-LR is better than Trac-CNN-LSTM on the Facebook collection while it is the opposite on the Twitter collection. This could be due to the train dataset which is only composed of texts crawled from Facebook. Indeed, we can observe the same behaviour for the other systems that participated to the challenge [158]. The only exception is for Saroyehun [161] system which performs better on the Twitter dataset.

Table 14. Result for the English (Facebook and Twitter) task. Bold values are the best performances for our approaches

System	Weighted F1	
	Facebook	Twitter
Random Baseline	0.354	0.348
Saroyehun [2]	0.642	0.592
Trac-RF_LR	**0.581**	0.409
Trac-CNN_LSTM	0.559	**0.511**
Trac-RF_only	0.573	0.397
Trac-LR_only	0.569	0.452

6.1.1.5 Future development

The program that has been written by CNRS-IRIT needs now to be made as a component compatible with the PREVISION architecture; it also needs some adaptation to make it easily generalizable to various detection tasks.

6.1.2 Visual Analytics for trends and dynamics of radicalized content

Collecting information, text analytics and Artificial Intelligence tools allows identification of trending topics in different media sources, while exploratory visual analytics tools provide means to identify prevalence of topics in different sources, and their dynamics. Such tools, among others, include co-occurrence network analysis, sentiment-based storyline analysis and structural topic modelling.

6.1.2.1 Word Co-occurrence Network

Network analysis describes a group of methods that characterize relationships between units to be analysed. A network is constituted of nodes (units to be analysed) and edges (connections between nodes). Network analysis quite often is used in the context of social network, where nodes are often individual people, while edges refers to friendships, affiliations or other types of [174](p.10). Though network analysis is most often used in terms of relationships between people, it can also be applied to represent relationships between words, e. g. [175]; [176]; [177]. Word co-occurrence network analysis is text length insensitive method thus it is suitable even when dataset is made of short texts or just texts of uneven length [178]. For word co-occurrence network analysis, we use package *textnets* for R. See an example of word co-occurrence network in Figure 35.

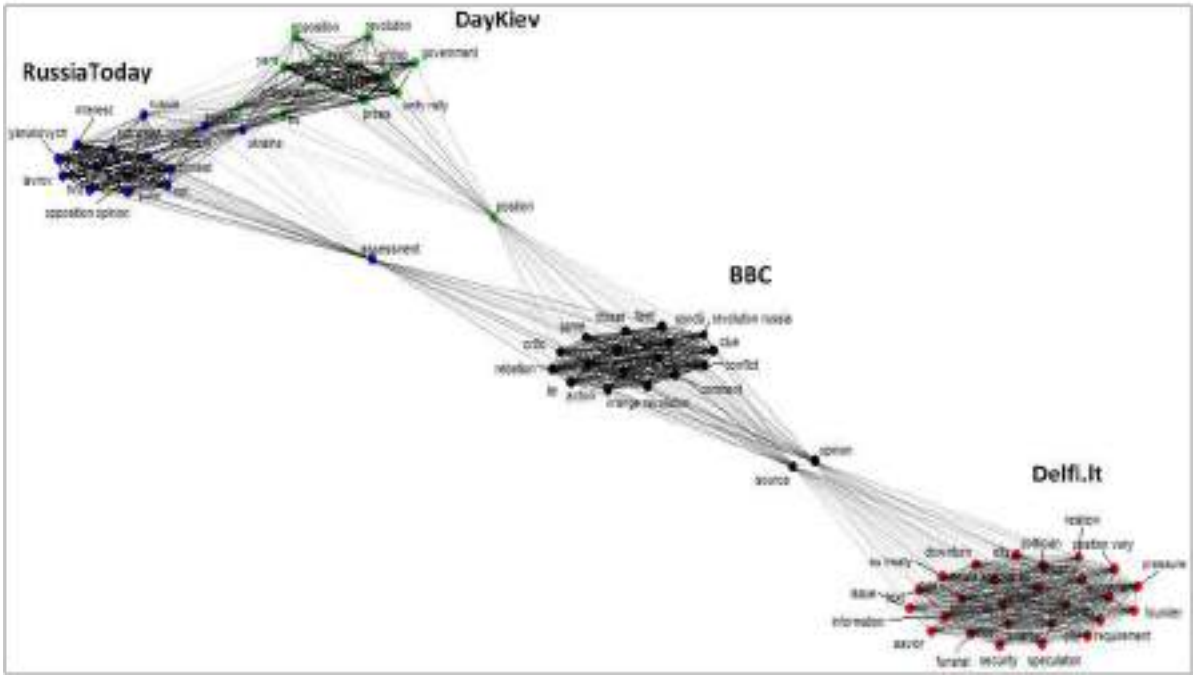

Figure 35. Word co-occurrence network: 1st stage of Ukrainian conflict (alpha=0.5).

6.1.2.2 Sentiment-Based Storyline Analysis

Sentiment analysis is important task of artificial intelligence, with applications such as automated analysis of reviews and social media, monitoring of political issues, etc. [179]. Sentiment analysis is based on the assumption that emotions are important in communication among humans. Some recent work in terms of analysis of literary texts has suggested that shifts in sentiment can be used for analysis of plot development [180](p. 73-110). As the raw sentiment time series is very noisy, a smoothing filter is usually applied for the raw data [181]. The result is a fairly smooth curve which represents generalized trend of the sentiment development of the text. This method is based on controlled vocabulary and data compression. For sentiment-based storyline analysis we use package *syuzhet* for R. See an example of sentiment-based storyline analysis in Figure 36.

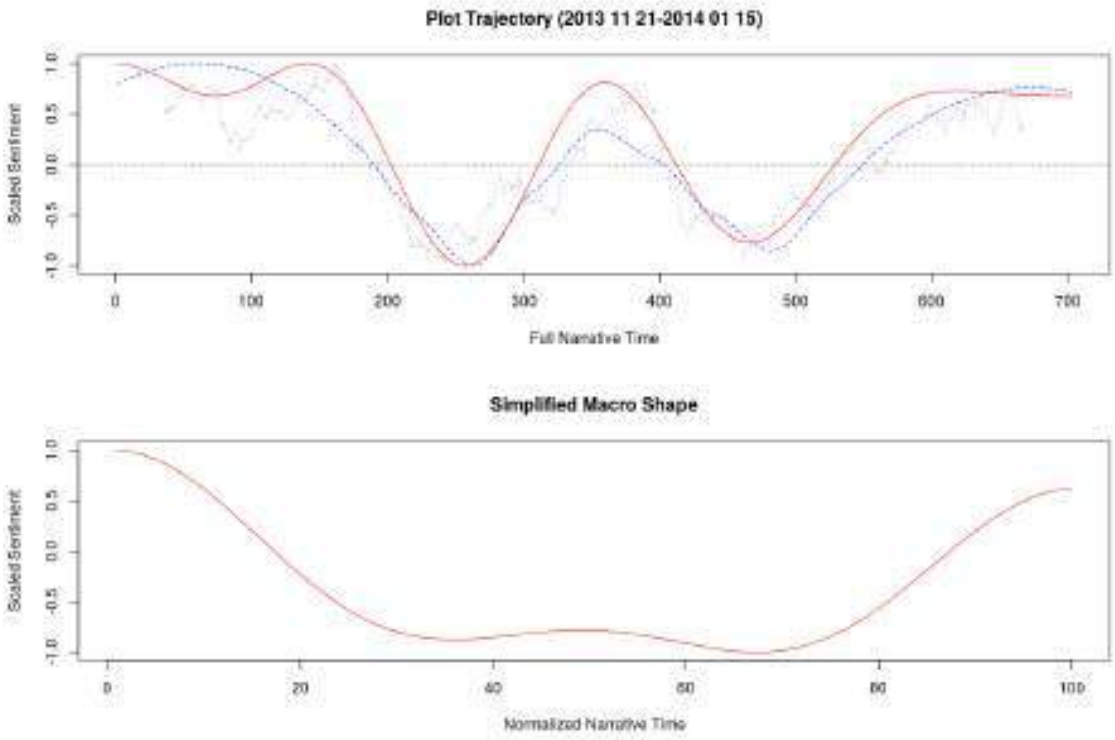

Figure 36. Sentiment-based Narrative Trajectory: 1st stage of Ukrainian conflict.

6.1.2.3 Structural Topic Modelling

The structural topic model allows researchers to flexibly estimate a topic model that includes document-level metadata [182]. Estimation is accomplished through a fast variation approximation. R library *stm*[47] provides many useful features, including rich ways to explore topics, estimate uncertainty, and visualize quantities of interest. Structural topic modelling operating principle:

1. The generative model begins at the top, with document-topic and topic-word distributions generating documents that have metadata associated with them:

    a. a topic is defined as a mixture over words where each word has a probability of belonging to a topic.

    b. a document is a mixture over topics, meaning that a single document can be composed of multiple topics. As such, the sum of the topic proportions across all topics for a document is one, and the sum of the word probabilities for a given topic is one.

2. Topical prevalence refers to how much of a document is associated with a topic (described on the left-hand side) and topical content refers to the words used within a topic (described on the right-

[47] https://cran.r-project.org/web/packages/stm/index.html

hand side). Hence metadata that explain topical prevalence are referred to as topical prevalence covariates, and variables that explain topical content are referred to as topical content covariates.

Topic modelling is part of a class of text analysis methods that analyse “bags” or groups of words together—instead of counting them individually–in order to capture how the meaning of words is dependent upon the broader context in which they are used in natural language. So foremost investigation was for finding the expected proportions in the data (see Figure 37.).

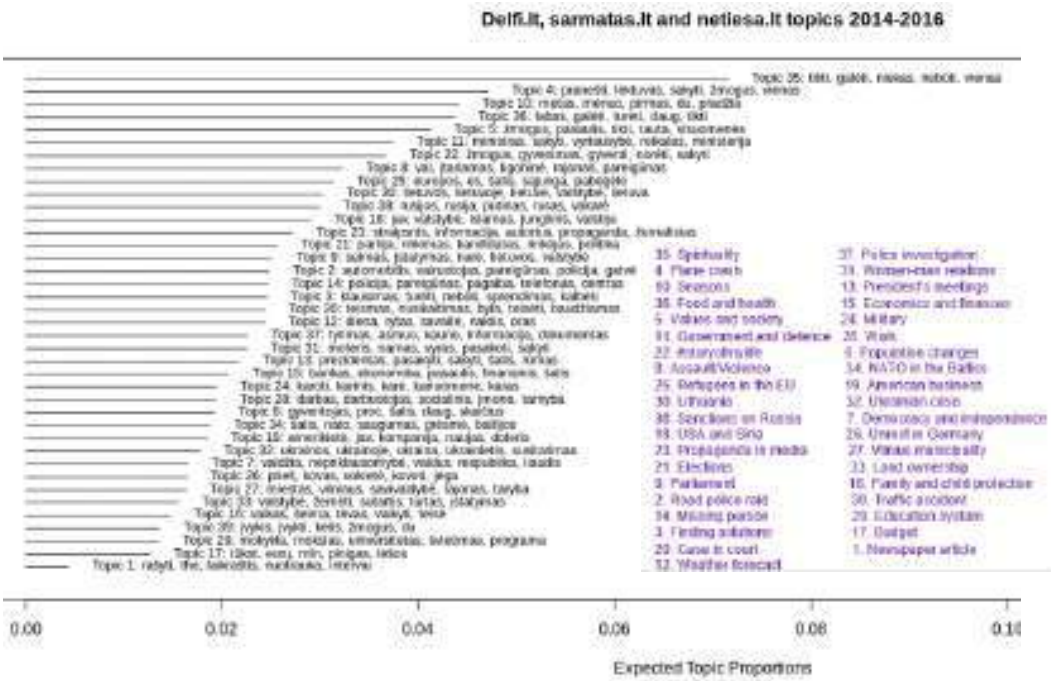

Figure 37. Topics by expected proportions in the Lithuanian data.

Topical prevalence refers to how much of a document is associated with a topic and topical content refers to the words used within a topic. Expected difference in topic probability be media type (with 95 % confidence intervals) is shown below (see Figure 38.).

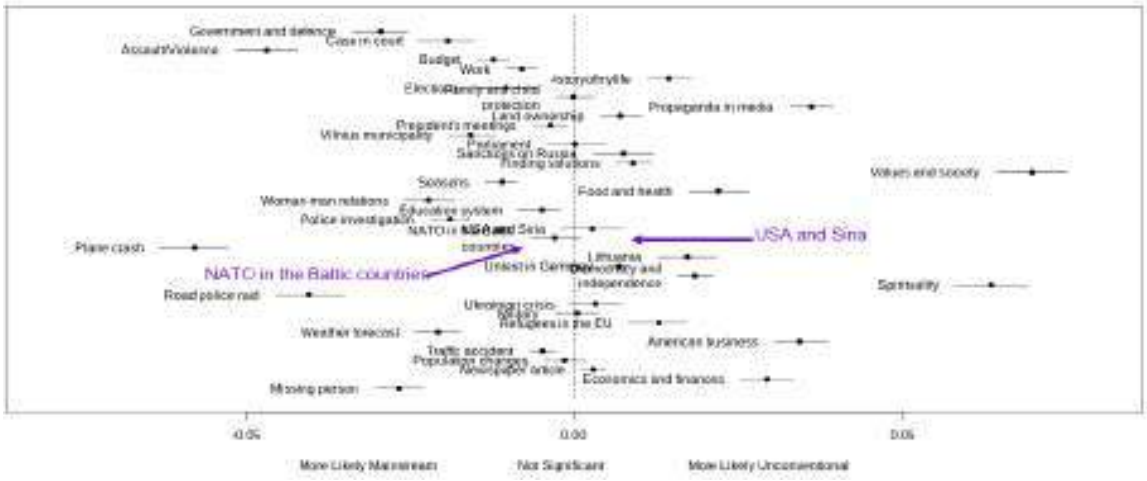

Figure 38. Effect of media Type (Mainstream vs Unconventional (in radical sense) on Topic Prevalence in Lithuanian dataset.

The last simulation initiated was topic correlation network creation (see Figure 39). The way these algorithms work is by assuming that each document is composed of a mixture of topics, and then trying to find out how strong a presence each topic has in a given document. This is done by grouping together the documents based on the words they contain and noticing correlations between them. A topic model captures this intuition in a mathematical framework, which allows examining a set of documents and discovering, based on the statistics of the words in each, what the topics might be and what each document's balance of topics is.

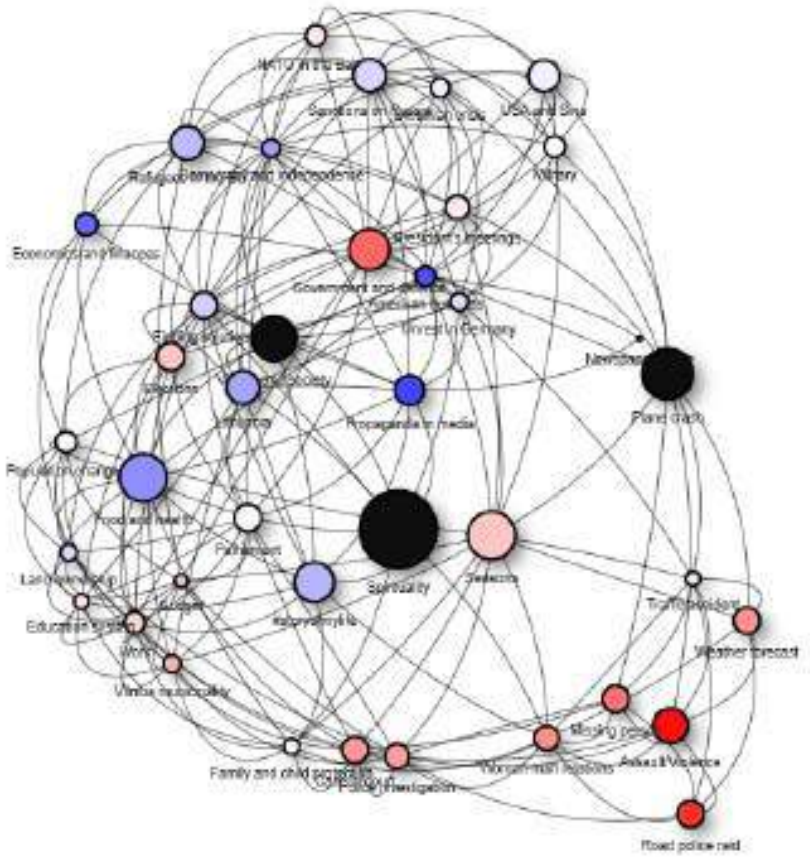

Figure 39. Topic (Correlation) Network of Lithuanian media (mainstream & unconventional). Explanation: Blue – more typical to unconventional media; Red – more typical to mainstream media; Black – topics that differ delfi.lt (mainstream) and alternative/unconventional (sarmatas (sarmatas.lt and netiesa.lt) news sources most.

6.1.2.4 Application of Visual Analytics for Radicalization Analysis

Based on the available data and resources, we are planning to apply the above-mentioned methods, such as word co-occurrence networks, sentiment-based storyline analysis and structural topic modelling for radical content identification, its dynamics, and relations. Such tools allow visually identifying change of dynamics, narrative, and hence activity of different groups, birth of new ones. Moreover, relations between different groups could be seen, both direct or just ideological (similar rhetoric), which could help predicting future evolvement of both, narrative and groups.

6.2 Behavioural Analysis and Anomaly Detection tool for videos

6.2.1 Introduction

The applications for detection abnormalities are varied, from fraud detection, medical images analysis, and sensor networks monitoring to video surveillance tracking and damage detection systems used in industries. As anomalies are declared all non-normal states or unknown states, everything abnormal,

deviant or outliers. In recent literature, a lot of methods have been proposed that are focused on abnormality detection using visual content. Moreover, the research community examines the advantages of deep learning-based techniques that can detect most efficiently the abnormalities when a large scale of visual data is used. The learning strategies of deep learning-based approaches for anomalies detection can be assigned to three categories - considering the nature of given data-: Supervised [183], Semi-supervised [184], and Unsupervised [184], see Figure 40 (left). Due to the fact that is very difficult to capture and annotate samples that describe abnormal activities into hours of videos, the generated datasets are strongly unbalanced and subsequently, the supervised approaches are not popular. Semi-supervised approaches are focused on detection of outliers of the input data. This carried out by the usage of datasets that describe a lot of normal activities (that is more easily to be captured) and as abnormalities are defined the outliers. So, the target of semi-supervised is to separate the outliers from a normal state. For example, a deep autoencoder that has been trained to reconstruct the input image can work as an outlier detector. To address this, an autoencoder will try to reconstruct the (unseen) input visual content that will lead to producing higher reconstruction error as has never been trained on this. Following similar to semi-supervised approaches, unsupervised approaches aim to detect outliers without using the small amount of labeled data that used on semi-supervised approaches. From another aspect, researchers have tried to classify abnormally detection methods taking into account the training objective. They have declared two categories as can be depicted in Figure 40 (right): The “Hybrid” category, in this category belong the methods that mainly consist of a deep neural network as feature extractor followed by a “One Class Support Vector Machine” (OC-SVM) classifier. In the “One-Class Neural Networks” (OC-NN) category belong the methods that are tried to learn and classify samples in one step [185].

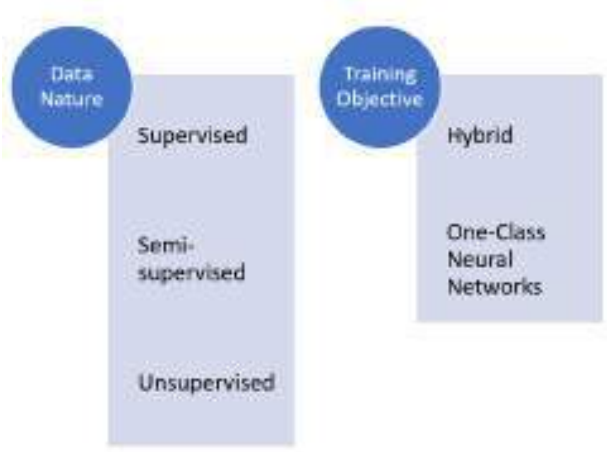

Figure 40. Abnormally detection: A methods categorization by data nature (left) and training objective (right).

Beyond the categorization of the abnormal detection approaches considering the data and the training objective, the methods in the recent literature are categorized into three categories taking into account the type of anomalies that aim to detect. “Point anomalies” comprising the most popular category in recent literature, and represents something that occurs randomly without having any connection with previous or future actions. The goal of the detection of a point anomaly will be for example to detect a

man which starts running into a store. “Contextual anomalies” are referred to the specific content of anomalies detection that includes both contextual and behavioral anomalies. Finally, “group anomalies” are focused on the detection of a group of abnormalities such as crowd-centered abnormalities. For instance, detection of panic events into crowd scenes that include a group of point anomalies occurred by a lot of persons [185].

6.2.2 Related work

A lot of recent works for the detection of abnormalities from surveillance video footage have been proposed so far. Most of them make use of unlabeled data -due to the lack of labeled data availability- and generally follow semi-supervised/unsupervised training approaches. The majority of these methods belong to the Hybrid category, where a deep neural network is used as the feature extractor and subsequently a binary classifier takes the decision if the sample is an outlier. Based on the feature extractor neural network that is used we will present indicative methods from each category. Hasan et al. [186] proposed a fully convolutional autoencoder that trained to reconstruct sequentially the input frames. During training, this autoencoder-decoder shifts the stacked frames (every forward pass) by 1 (frame) into the third dimension (channels) so as to learn the correlation of the temporal domain. Mahalanobis distance was used to calculate the distance between the input frames and the reconstructed one. Vincent et al. [187] proposed De-noising Autoencoders (DAE) and Stacked Denoising Autoencoders (SDA) in order to extract robust representations of input data to improve the performance of unsupervised approaches. SDAs are trained using two targets, the input image that the goal is to be reconstructed and the motion information in the form of optical features or trajectories. Finally, Vu et al. [188] propose an anomaly detection framework for surveillance videos based on Restricted Boltzmann Machine (RBM). The proposed framework works directly to the image pixels rather than hand-crafted features and learns the representations in an unsupervised manner. Subsequently, it reconstructs the input date in order to localize the possible abnormalities.

Approaches that aim to reconstruct and predict the anomalies based on Long Short-Term Memory (LSTM) neural networks have been also proposed so far. Srivastava et al. [189] propose a combination of two models, an autoencoder and a predictive LSTM model. Due to the fact that the autoencoders have a lack to memorize the information they use a parallel scheme of an autoencoder an LSTM model and as the predictor. Munawar et al. [190] proposed a convolutional feature representation that gives input to an LSTM model to predict the latent space representation and its prediction error is used to evaluate anomalies in a robotics application. Finally, it noted that the approaches that based on convolutional autoencoders and reconstructing the input frames are needed only one or two frames – in case that optical flow is calculated between two sequential frames-, on the other side predictive methods such as [189] and [190] are needed a number more than two past frames [186].

6.2.3 Datasets

In this section, the datasets that commonly used for training and evaluation of abnormally activity detection are presented. The datasets bellow consist of visual content (images and videos) and are available for research purposes online. Figure 41 and Figure 42 illustrate indicative samples of the datasets discussed below.

UCSD dataset [191]**:** is a dataset that consists of videos of pedestrians in open areas. As anomalies have been annotated all non-normal activities could occur into a pedestrian walkway. These anomalies are performed by cars, bikes, and cyclists. Furthermore, people that are moved to non-usual locations are considered anomalous. The dataset clips are approximately 200 frames. Two distinct subsets comprising the dataset the Peds1 and Peds2. The Peds1 contains 34 videos for training and 36 for testing that describe groups of people walking towards and away from the camera. The Peds2 contains 16 videos and 12 for train and test sets respectively. The videos describe person movements parallel to the camera. Both Peds1 and Peds2 the annotation provided by a binary flag (normal, abnormal).

CUHK Avenue dataset [192]**:** is a widely used dataset in the anomaly detection research area. The dataset consists of annotated anomalies of humans that are doing a strange action such as running, the movement to wrong directions on subways stations and humans that kept or interact with abnormal objects such as bicycles. The whole dataset consists of 37 videos, 16 belong to the train set and the rest to the test set. The total number of frames is 30652, 15328 and 15324 for training and testing, correspondingly. The annotation is provided in the form of bounding boxes that enclose the abnormal events.

UMN dataset [193]: is a dataset that describes abnormal activities such as panic, of crowd-based scenes both for indoor and outdoor environments.

Figure 41. Indicative dataset samples: UCSD (left), CUHK Avenue (center), UMN (right).

Train, Belleview and Subway exit dataset [194]**:** is a dataset that describes abnormalities of persons that movement on train rails, cars that entering thoroughfare from left or right and surveillance cameras that observing pedestrians at the subway exit movement in the wrong way.

Queen Mary University of London U-turn dataset [195]**:** is a dataset that contains videos of outdoor environments and specifically roads. The anomalies are annotated U-turns movements of cars.

LV dataset [196]**:** is the recent dataset in the literature regarding abnormal detection. LV consists of realistic events without actors performing predefined actions. Approximately 4 hours of videos and variance of frame rate (7-30 fps) comprising the dataset. Totally, 30 videos had been captured for training and test sets and generate approximately 70K of frames. Finally, the different types of abnormal events of this dataset counted to 34 with some of them having occurred in a short time. Indicative events are fighting, people clashing, arm robberies, thefts, car accidents, hit and runs, fires, panic, vandalism, kidnapping, homicide, cars in the wrong-way with many of them have been captured from indoor and outdoor environments.

Figure 42. Indicative dataset samples: Train, Belleview and Subway exit (left), U-turn (center), LV (right).

6.3 Behavioural Analysis and Anomaly detection tool for telecom and financial data

Outlier detection [197] refers to the problem of finding out the patterns in the massive datasets that does not show the accordance with the generalized expected behavior. Some outliers provide useful information and are a source of new knowledge. However, sometimes they are just noisy data points. Detection of outliers is a mature area of research and has been successfully applied to various areas like network intrusion detection, credit-card fraud detection, email-based network analysis etc. and can be used to identify system faults or frauds before potentially catastrophic consequences occur.

6.3.1 Financial data

Financial fraud [198] poses a challenging problem due to a variety of factors, including (a) the very small fraction of fraudulent transactions compared to the sheer volume of legitimate transactions processed by most financial institutions ("needle in the haystack" problem); (b) real- or near real-time transaction processing ("high velocity" of big data); (c) frequent lack of traceability of transactions from ultimate source to final destination; and (d) lack of information sharing among financial institutions.

Fraud detection is a highly complex function to be performed where there is no system which can guarantee a 100% satisfaction result rate. All the existing methods can likely predict fraud transactions and not assure you about the results. Let's consider the properties of good fraud detection method:

- It must be able to identify the frauds accurately.
- It must quickly detect fraud cases.
- In any case a genuine transaction should not be considered as fraud.

6.3.2 State-of-the-art

Machine learning refers to analytic techniques that "learn" patterns in datasets without being guided by a human analyst. It helps data scientists efficiently to determine which transactions are most likely to be fraudulent, while significantly reducing false positives. The techniques are extremely effective in fraud prevention and detection, as they allow the automated discovery of patterns across large volumes of streaming transactions. State-of-the-art [199], [200], [201] refers to many different clustering and classification machine learning algorithms that can be used in credit card fraud detection. Such algorithms include neural networks, distance-based methods (e.g. KNN, K-mean, etc.), density-based methods (e.g.

local outlier factor (LOF), connectivity-based outlier factor (COF), etc.), Parametric methods (e.g. logistic regression) and more. The Table 15 below summarizes the core differences between those methods:

Table 15. Comparison of Outlier Detection Methodologies

Methodology used	Computational Complexity	Efficiency	Feature Space	Practical Applicability
Parametric Method	Less Complex	More Efficient	Univariate	Data sets with prior Knowledge
Non-parametric Method	Less Complex	Efficient	Univariate/ Multivariate	profile of the data is maintained
Distance based method	Easy	Efficient	Multivariate	Based on closeness of individual points
Density based method	Very Complex	Very Efficient	Multivariate	Based on the closeness of points and nearest neighbor too
Neural Networks	Less Complex	Very Efficient	Multivariate	Applied on the normal training data

In outlier detection, unsupervised learning is preferred to detect the fraud because it can lead to new explanations and representation of the observation data. Unsupervised outlier detection techniques do not require training data, and thus are most widely applicable. The techniques in this category make the implicit assumption that the normal instances are far more frequent as compared to the outliers, if this is not the case then, these techniques are not that much successful, and they suffer from the high false alarm rate. Another advantage of using unsupervised data, it does not require prior labeling of data or knowledge about fraudulent methods or transactions. So, it need not be trained to discriminate between a legal and illegal transaction. It simply follows the normal behavior pattern as an unusual activity or fraudulent. The major advantage of using unsupervised method over supervised data is that it need not be trained to discriminate between a legal and illegal transaction.

6.3.2.1 Distance based proposed algorithm

Due to their computational simplicity and their efficiency for multivariate data analysis, distance-based outlier techniques [202] are one of the most widely accepted and frequently used techniques that are used in machine learning and data mining and it completely depends on the concept of local neighborhood (KNN) of the data points. This concept can also be termed as Nearest Neighbor Analysis and it can also be applied for different purposes such as classification, clustering and most importantly outlier detection. The most significant feature of the nearest neighbor-based outlier detection technique is that they have an explicit notion of proximity that is defined in the form of a distance and similarity measure for any of two individual data instances, or a set or a sequence of instances.

K-nearest neighbor algorithm [199] is used largely in detection systems. It is also proved that KNN works extremely well in credit card fraud detection systems. In this method the new instance query will be classified depending on the KNN category. The results of KNN depend on the below three factors:

- The distance metric used to decide the nearest neighbors.
- The distance rule that is used for the classification from K-nearest neighbors.

- The number of neighbors considered to classify the new sample.

The idea behind the KNN method is rather simple. Considering an annotated data collection, for an unknown data sample the class label is assigned based on the majority of its k-nearest neighbors. Even though it is a rather simple method, it has some indisputable advantages such as: simplicity, effectiveness, intuitively, non-parametric nature, and high performance for different classification tasks. The proper selection of parameter k responsible for the neighborhood size, and the distance function responsible for the quality of the topological representation can have a significant impact on the underlying results. To select the K that's right for the data, we run the KNN algorithm several times with different values of K and choose the K that reduces the number of errors we encounter while maintaining the algorithm's ability to accurately make predictions when it's given data it hasn't seen before.

KNN captures the idea of similarity (sometimes called distance, proximity, or closeness) calculating the distance between points on a graph. Some distances that you can use include Euclidean distance, Manhattan distance (also known as the city block distance or the taxicab distance), Minkowski distance (a generalization of the Manhattan and Euclidean distances), and Mahalanobis distance.

Minkowski-type distances assume that data is symmetric; that in all dimensions, distance is on the same scale. Mahalanobis distance, on the other hand, takes into account the standard deviation of each dimension.

- Euclidean: $D(X,Y) = \sqrt{\sum_1^n (x_i - y_i)^2}$
- Minkowski: $D(X,Y) = (\sum |x_i - y_i|^p)^{1/p}$
- Mahalanobis: $D(x,y) = \sqrt{\sum (\vec{x} - \vec{y})^T * S^{-1} * (\vec{x} - \vec{y})}$

6.3.2.2 Initial results

In order to validate the proposed approach and present some initial results, we used the credit and fraud detection dataset that is available in Kaggle[48]. The dataset contains 284.807 transactions made by credit cards in September 2013 by European cardholders. The dataset has been preprocessed using PCA and the original data are not provided. The only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction amount. Though this dataset is tagged in order to be used for classification purposes, we will remove the tags in order to apply unsupervised learning, which is the case expected in PREVISION. Furthermore, for the current analysis we will also ignore the PCA components as we do not have more information about the features that they represent.

In Figure 43, a plot of the Amount with respect to the Time is depicted. As it can be seen from the 2-dimensional plot, there are certain points (transactions) in the 2d space that present a different behavior compared to the majority of the transactions and it is expected to be identified as potential outliers.

[48] https://www.kaggle.com/mlg-ulb/creditcardfraud

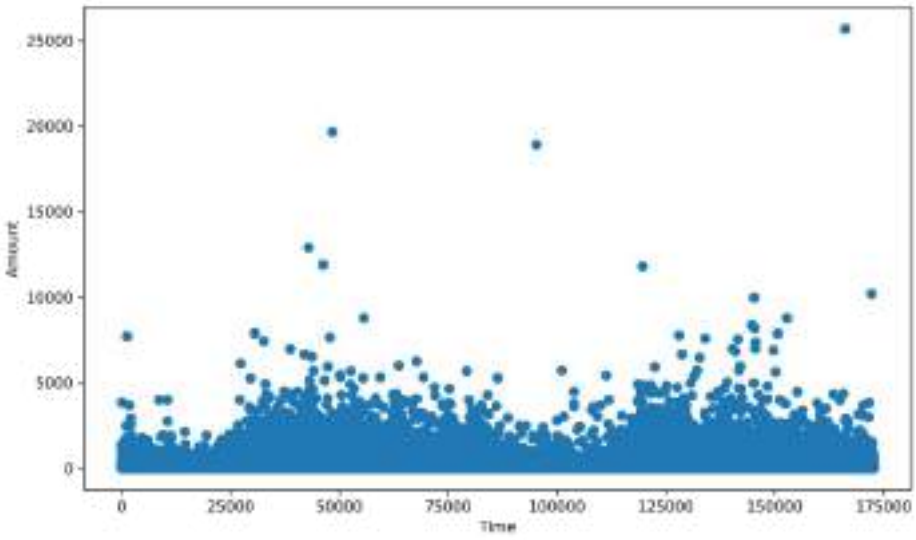

Figure 43. Amount of the transaction in the financial data records with respect to time.

Indeed, by applying the KNN algorithm[49] described above, we can observe in Figure 44 that these points are identified as potential outliers. However, further analysis is required, that will also be based on rules provided by LEAs in order to identify the specific outliers that are of particular interest to the end-users.

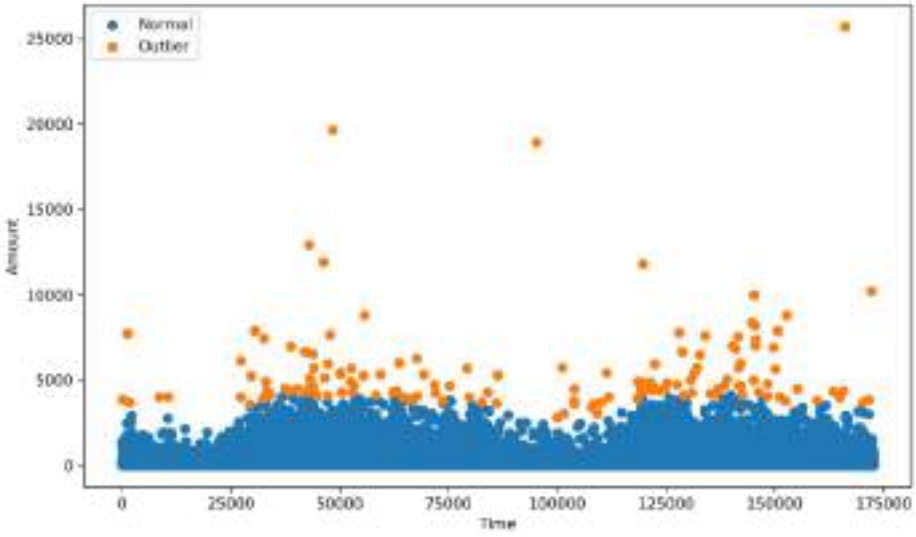

Figure 44. Outliers identified in the dataset and plotted in the graph of the amount of the transaction in the financial data records with respect to time.

[49] https://www.cs.umd.edu/hcil/VASTchallenge08/

6.3.2.3 Internal Interface Design

The specification of the internal interface for the financial data anomaly detection tool is described in Table 16.

Table 16. Interface description of the tool that identifies outliers in the financial data

Interface Attributes	Description
Interface Name	financialDataOutlierDetection
Module	Anomaly Detection
Author	ICCS
Type of interface	RESTful Webservice
Exposition	Private
Exposed Methods	financialAbnormalities • Description: Provides the outliers in the financial data records. The outliers refer to situation where abnormal behavior is detected using data mining techniques • In: Financial Data Records. • Out: List of outliers
Further remarks	The financial data outlier detection tool is based on data mining techniques and it can be parameterized for each PREVISION client and use case. Based on the needs of the LEAs different hyperparameters can be selected in order to provide more accurate results.

6.3.3 Telecommunication data

Telecommunication data [203] are of multidimensional nature. It is impossible to analyze them by using only statistics. The main features and technical parameters are: type of call, calling time, duration of call, call destination, location of caller, etc. The multidimensional analysis of such data can be used to forecast the data traffic, group user behavior, detect abnormalities, and so on. This kind of data creates a great demand for efficient data mining techniques in order to help identify telecommunication patterns, catch fraudulent activities, make better use of resources, and in consequence improve the quality of service. Specifically, for the purposes of PREVISION Call Data Records (CDRs) are provided by LEAs that contain information about the calls made or received by the subscriber of a service provider. Each call is represented by a caller ID, a receiver ID, the time of the call initiation, the duration of the call, the tower used and other details regarding the handset used (like IMEI). The properties typically used for analysis of CDR are those that capture individual user behavioral properties like number of calls made or received, number of contacts, towers used for making calls, call duration, etc. Users can be associated with

statistical properties like average number of calls made per day, min/max/average duration per call, etc., and the interaction between two users is described by the calls between them.

6.3.3.1 State of the art

There are many works in the literature that deals with the problem of anomaly detection in the telecommunication domain. These efforts can be grouped in two main categories. In the first category, the proposed methods are based on rules provided by domain experts, while in the second category data mining techniques are applied.

Specifically, for the rule-based systems, the authors in [204] treat the CDRs as a special kind of social network where nodes represent phone numbers and edges represent the call between two numbers, and propose an automated anomaly detection mechanism based on fuzzy logic that assigns anomaly score to nodes in the CDR network. The authors claim that their proposed method can achieve better results as it uses attributes of links of higher depth than the features of a single subscriber and his contacts. In [205], the authors present a call detail record-based anomaly detection method that analyzes the users' calling activities and detects the abnormal behavior of user movements in a real cellular network. Their method is based on the Hadoop ecosystem and it is rule based, i.e., it requires domain experts to set the rules and modify them accordingly.

On the other hand, data mining techniques and specifically cluster-based methods have been proposed in [206] and [207]. In particular, the authors in [206] propose clustering techniques in order to extract different calling patterns and behaviors from CDR that will help detecting outliers. In their work, the authors present a thorough description of the CDR analysis approaches and they conclude that their proposed method can work in real time and update the clusters periodically if required. In a similar way, the authors in [207] apply clustering methods and specifically k-means in order to detect anomalies in CDRs with the scope of fault detection and redesigning resource distribution.

6.3.3.2 Cluster-based approach

Based on the above analysis of the state of the art, a cluster-based approach is adopted for the CDR analysis of telecom data. Clustering-based approaches detect outliers by examining the relationship between objects and clusters, in an unsupervised way without requiring any labeled data. Once the clusters are obtained, clustering based methods need only to compare any object against the clusters in order to determine whether the object is an outlier. An outlier is an object that belongs to a small and remote cluster, or does not belong to any cluster. There are three general approaches [208] to clustering-based outlier detection:

- An outlier is an object that does not belong to any cluster
- An outlier is an object that has a large distance to the cluster to which it is closest
- A set of objects that form a small and sparse cluster are considered as outliers.

Though the first two approaches work well on individual objects, they fail when the outliers form a cluster of their own. This is because they compare the individual objects one at a time against clusters in the data set. In order to overcome this shortcoming, CBLOF algorithm [209] is adopted for the purposes of the

telecom data of PREVISION. CBLOF algorithm can identify small or sparse clusters and declare the objects in those clusters to be outliers. The algorithm works as follows:

1. Initially it finds the clusters in a data set, and sorts them according to decreasing size. The algorithm assumes that most of the data points are not outliers. It uses a parameter α (0<=α<=1) to distinguish large from small clusters. Any cluster that contains at least a percentage α (e.g. α=90%) of the dataset is considered a "large cluster". The remaining clusters are referred to as "small clusters".
2. To each data point, assign a cluster-based local outlier factor, called CBLOF. For a point belonging to a large cluster, its CBLOF is the product of the cluster's size and the similarity between the point and the cluster. For a point belonging to a small cluster, its CBLOF is calculated as the product of the size of the small cluster and the similarity between the point and the closest large cluster.

CBLOF defines the similarity between a point and a cluster in a statistical way that represents the probability that the point belongs to the cluster. The larger the value, the more similar the point and the cluster are. The CBLOF score can detect outlier points that are far from any clusters. In addition, small clusters that are far from any large cluster are considered to consist of outliers. The points with the lowest CBLOF scores are suspected outliers.

6.3.3.3 Initial results

In order to validate the proposed approach and present some initial results, we used the open telecommunication dataset that was provided by IEEE for the VAST 2008 Challenge. The dataset contains CDR information covering a 10-day period in June 2006 and 400 unique cell phones. The dataset includes the following attributes:

- From: Identifier for the calling cell
- To: Identifier for the receiving cell
- Datetime: a yyyymmdd hhmm format date and time
- Duration: duration of the call in seconds
- Cell Tower: Location of the call origination cell tower

The statistics of the CDR can be seen in Table 17. Based on these statistics, the call durations have a mean value of 1065 seconds and they also contain errors as the duration for certain calls has a negative value. However, as the intention of the current analysis is only to provide an overview of the proposed method, we will not perform data cleaning and we will expect from the algorithm to identify such points as outliers as well.

Table 17. Statistics of the CDR dataset

	From	To	Duration(seconds)	Cell Tower
count	9834.000000	9834.000000	9834.000000	9834.000000
mean	199.473764	130.446105	1065.250966	19.064877
std	115.618109	117.919222	303.768111	8.726197
min	0.000000	0.000000	-145.000000	1.000000
25%	99.000000	24.000000	861.000000	11.000000
50%	198.000000	92.000000	1064.500000	20.000000
75%	301.000000	217.750000	1267.000000	28.000000
max	399.000000	399.000000	2171.000000	30.000000

In Figure 45, a plot of the phone id (From) that initiated the call with respect to the duration of the call is presented. Furthermore, the duration of each call with respect to the date is depicted in Figure 46. As it can be seen from the 2-dimensional plot, there are certain points in the 2d space that it is expected to be identified as outliers.

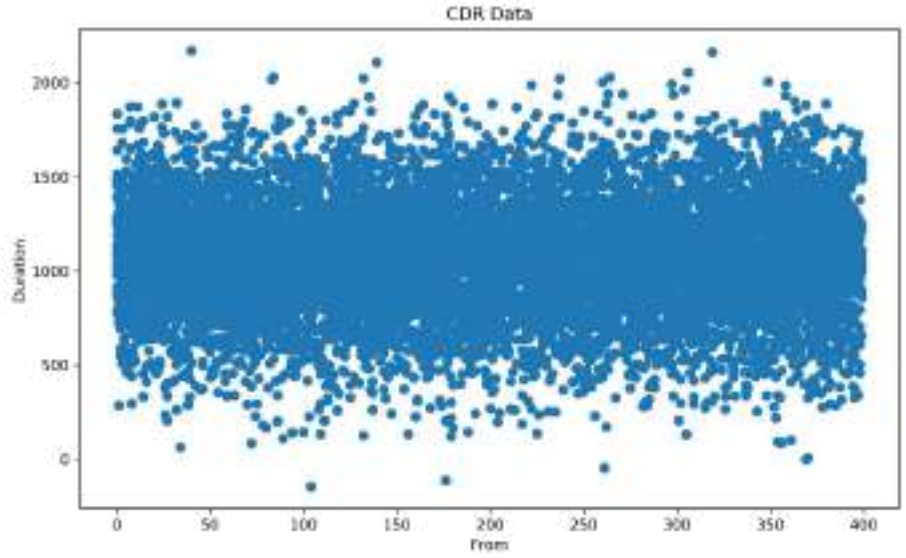

Figure 45. Duration of calls with respect to the cell phone that initiated the call.

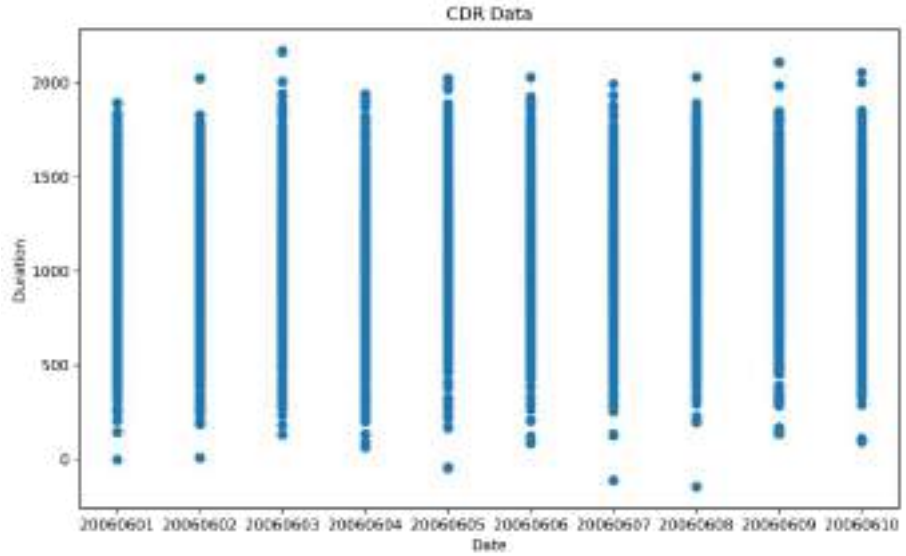

Figure 46. Duration of calls with respect to the date.

Indeed, by applying the CBLOF algorithm[50], we can observe in Figure 47 and Figure 48 that these points are identified as potential outliers. However, further analysis is required, that will also be based on rules provided by LEAs in order to identify the specific outliers that are of particular interest to the end-users.

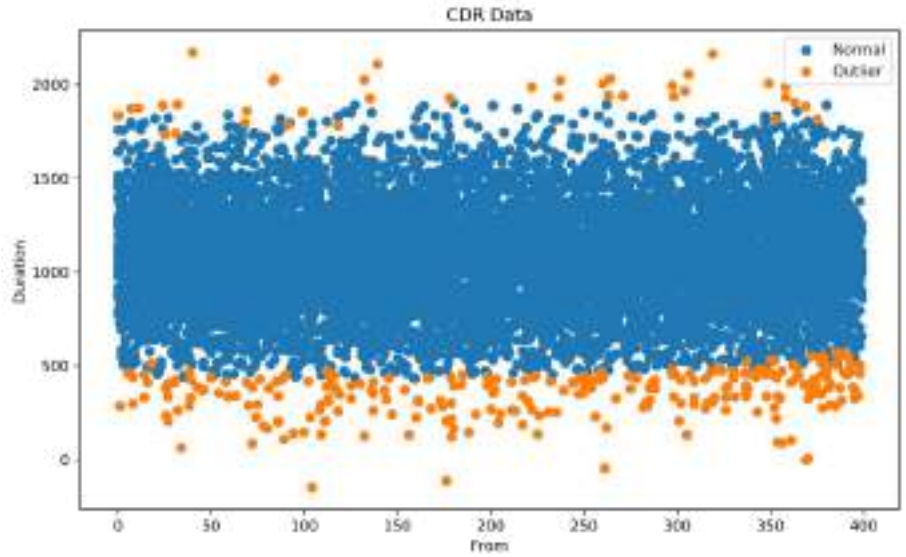

Figure 47. Outliers identified in the dataset and plotted in a graph of the duration of the calls with respect to the cell phone that initiated the call.

[50] https://www.cs.umd.edu/hcil/VASTchallenge08/

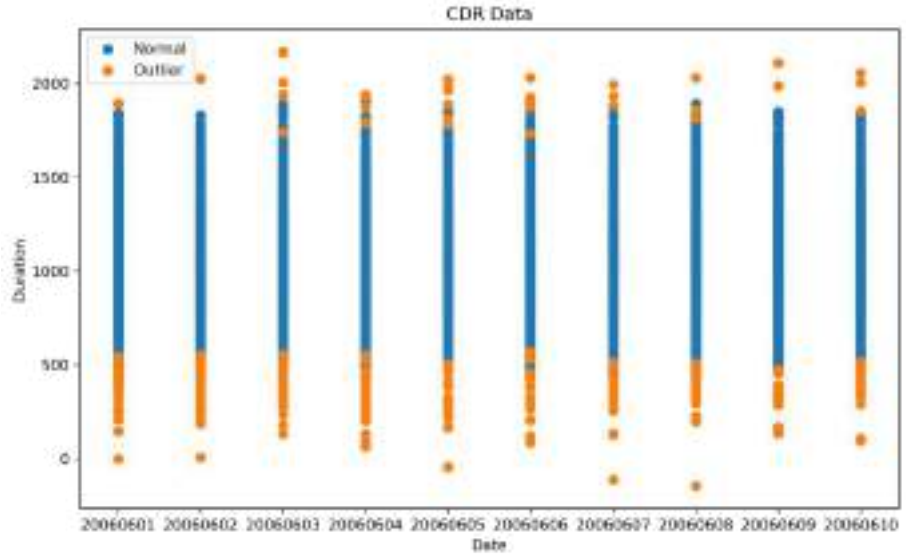

Figure 48. Outliers identified in the dataset and plotted in a graph of the duration of the calls with respect to the date.

6.3.3.4 Internal Interface Design

The specification of the internal interface for the financial data anomaly detection tool is described in Table 18.

Table 18. Interface description of the tool that identifies outliers in the telecommunication data

Interface Attributes	Description
Interface Name	telecomDataOutlierDetection
Module	Anomaly Detection
Author	ICCS
Type of interface	RESTful Webservice
Exposition	Private
Exposed Methods	telecomAbnormalities • Description: Provides the outliers in the call data records (telecommunication data). The outliers refer to situation where abnormal behavior is detected using data mining techniques • In: Call Data Records. • Out: List of outliers

Further remarks	The telecommunication data outlier detection tool is based on data mining techniques and it can be parameterized for each PREVISION client and use case. Based on the needs of the LEAs different hyperparameters can be selected in order to provide more accurate results.

6.4 Cyber-attack detection tool

One of the recent developments in the field of deep learning is attacks against deep neural networks (DNNs), the so-called adversarial attacks [210]. This type of attack aims at modifying the input data in a way that is almost indistinguishable or slightly noticeable by the human observer, whereas, this slight modification leads to drastically different prediction results produced by DNNs. Many adversarial proof-of-concept attacks have already been devised by the research community, varying from the traffic sign physical modification to mislead an autonomous driving [211] up to the malware detection avoidance by malicious software [212]. The potential damage of exploiting such types of attacks can be rather profound and pervasive since the DNNs are applied almost in every modern intelligent system related to data and information processing.

Currently, there is no universal defense mechanism available that guarantees protection against this kind of attack. Nevertheless, there are several techniques proved to be promising in mitigating the potential damage and consequences: adversarial training [213] and generative modeling [214]. Each of them is described in more detail below.

6.4.1 Adversarial Training

It has been discovered that different architectures of DNNs trained to tackle the same classification problem on similar datasets tend to have similar fairly piece-wise linear decision boundaries that separate categories in the input data domain [210]. This property is called transferability, and it allows us to devise an adversarial attack that universally targets all DNNs with a similar final objective in a black-box manner [215]. However, on the other side, it also means that it is possible to generate the known adversarial examples in advance automatically and add them to the training set before starting your training. Such an approach forces the DNN model to take into consideration the adversarial perturbations and increases the robustness of DNNs to the adversarial attacks. The advantage of this method is its simplicity. The most significant disadvantage is the requirement to generate adversarial examples in advance, which may defend only against those attacks that have been known at the moment of the DNN model training. Taking into consideration that adversarial examples can be generated with a vast diversity of different techniques, this approach can be useful only if the model retraining and redeployment procedures are relatively cheap. Thus, it is possible to keep the model up-to-date with the development of new attacks.

6.4.2 Generative Modeling

Modern DNN classifiers, such as Convolution Neural Networks (CNN), for example) approximate conditional probability density $p(y|x)$ where y is the class label and x is the input data. Such an approach is called discriminative since it directly maps input to the output category. The generative modeling, on the other side, attempts to maximize the likelihood of the input data $p(x)$. It is an unsupervised approach

that may exploit different optimization objectives, depending on the underlying method. In PREVISION, we focus on the deep Bayesian inference coupled with the intermediate latent semantically disentangled representation that attempts to approximate both intractable posterior density $p(z|x)$ (where z is the latent variable responsible for semantical representation) and generative model that consists of $p(z)$ and $p(x|z)$ that allows sampling and mapping the latent representation back to the input space. The $p(x)$ represents the marginal in this case. Such an approach provides several ways to the adversarial defense. First of all, the generative capabilities of the model may be used for input filtering before the model under protection gets the adversarial example, so this example is purified before further usage [216]. Secondly, the latent representational layer can be used for the detection of such adversarial examples through the variation of the semantical features of the input and the computation of the appropriate DNNs prediction results [217]. Thirdly, the generative models allow out-of-distribution examples detection [218]. We concentrate on the first two applications since they are most relevant to the adversarial defense.

6.4.3 Methodology

In PREVISION, we will use the generative modeling approach to enhance the robustness against the adversarial attacks. Two deep generative models are widely used in recent research: generative adversarial networks (GANs) [219] and variational auto-encoders (VAEs) [220].

GANs demonstrated diverse generative capabilities [221], including the input filtering approach against adversarial attacks [216]. The only disadvantage of GANs is the usage of two objective functions that should be optimized simultaneously in a game-theoretic manner. Such training may fail to converge, and that is the reason why GANs are notoriously difficult to train. Moreover, since GANs exploit implicit density [222], there is no possibility to make inference queries that limit GANs' applicability.

VAEs, on the other hand, allows to approximate intractable posterior densities within the framework of the *directed probabilistic graphical models (DPGMs)* [223] or *Bayesian networks*. They have only one objective to optimize, which provides them a significant advantage in comparison with GANs making the training process faster and more stable. Moreover, it makes the model to learn the semantically meaningful representations that subsequently may be used in discriminative modeling. The meaningfulness is achieved by the prior of the latent variable since we force the model to map the input to the latent space (encoder) in such a manner, so it is possible to map the latent space back to the input space (decoder). In addition, since both encoder and decoder may be approximated by universal function approximators such as neural networks, it provides extra layers of abstraction allowing mapping of the predefined random variable that follows a tractable and easy-to-handle prior distribution over the latent space into any other sufficiently complicated distribution. Finally, it provides the possibility to generate output samples that have never been observed during the training phase, and such generation will be based on the task-inherent representational features encapsulated in the latent space, which is useful for the latent features manipulation and devising the corresponding defense methods [217].

In PREVISION, we will analyze and apply different types of variational autoencoders to enhance the adversarial robustness of the DNNs. The first step is to leverage the latent manifolds learned by VAEs in order to:

- Identify the influence of numbers of dimensions of the latent space to the robustness and generative capabilities of VAEs together with the analysis of the latent manifold representation
- Recognize the most appropriate balancing factor between KL-divergence and reconstruction loss used in the objective function of VAEs
- Apply different neural network topologies to both encoder and decoder

After that, as we identified the relevant parameters and their influence on the result, we will investigate the conditioning of the VAE generation process. The standard objective of traditional VAE is the maximization of the evidence lower bound (ELBO) [224]. This approach provides a latent representation that can be sampled for generation purposes, but in a very general way, i.e., there is no control over the output generation based on the distinct representational features encapsulated in the prior distribution over the latent space. For the more controllable generation that may allow specifying the desired characteristics of the output, we will investigate and apply the conditioning of the VAEs that will enable us to introduce the conditioning variable based on which the particular output is generated [225]. This way, we will be able to devise a defense mechanism that will exploit the conditioning in the following ways:

- Enhance our VAE DPGM with additional frozen DNNs to guide the training process to consider the appropriate decision boundaries together with the proper class label
- Manipulate the latent intra-class representational features given the particular class and calculating the appropriate statistics separating normal versus adversarial examples

This approach, it will allow us to combine adversarial examples filtering [226] together with adversarial examples detection [217].

6.4.4 Evaluation

The evaluation of the output will be done utilizing the specifically designed framework for testing the robustness of a defense mechanism against adversarial attack: CleverHans [227]. This framework allows us to benchmark the defense against well-known attacks such as fast gradient sign method (FGSM) [228] and notorious Carlini-Wagner attack (CW) [229] on several benchmark datasets such as MNIST [230] and CIFAR-10 [231].

7 Summary and Conclusions

This deliverable constitutes the initial report on the results of work package 3 on machine learning and automated cognitive system capabilities. It is going to be released in a refined version as deliverable 3.2.

Multiple system components, technical methods and algorithms have been proposed. The central semantic knowledge database of platform PREVISION is constituted by the ontology documented in sections 2.1 and 4.2. The facts residing in the ontology are delivered by the heterogeneous data stream processing units developed in work package 2 as sketched in section 4.1, as well as by various system components introduced in this work package.

Natural candidates for these might be the jargon detection tool documented in section 2.3, the tools for data fusion and completion developed in section 3, the data preprocessing and classification methods introduced in sections 4.4 and 4.5, the predictive policing methods treated in section 5, as well as the behavioural analysis and anomaly detection tools presented in section 6.

On the other hand, information contained in the PREVISION ontology can serve as a source for further data analysis. First of all, it can be retrieved through SPARQL queries. Also it can be combined via logical calculus as first order logic or probabilistic logic like Markov logic networks, for example. This is the subject of section 2.2. An example of use in the case of one of the designated PREVISION use cases is given in section 4.3.

In the refined release of this report, more of the system components introduced here might make use of the information gathered in the PREVISION ontology. Also clear interfaces might be defined for enabling different system components to work together. Putting the information flux in the center of attention might be a natural approach for arranging all proposed modules in a use-oriented way. This could also be the guideline for the subdivision of the document into sections, which is structured according to the tasks of the work package in this initial release.

8 References

[1] “Ontology components,” [Online]. Available: https://en.wikipedia.org/wiki/Ontology_components.

[2] „Protégé,“ [Online]. Available: https://protege.stanford.edu/.

[3] „Protege Desktop Features,“ [Online]. Available: https://protegewiki.stanford.edu/wiki/Protege4Features.

[4] S. Falconer, „OntoGraf,“ [Online]. Available: https://protegewiki.stanford.edu/wiki/OntoGraf.

[5] M. Horridge, „OWLViz,“ [Online]. Available: https://protegewiki.stanford.edu/wiki/OWLViz.

[6] M. Sintek, „OntoViz,“ [Online]. Available: https://protegewiki.stanford.edu/wiki/OntoViz.

[7] „ProtégéVOWL: VOWL Plugin for Protégé,“ [Online]. Available: http://vowl.visualdataweb.org/protegevowl.html.

[8] J. W. Lloyd, „Foundations of logic programming (second, extended edition),“ Springer series in symbolic computation. Springer-Verlag, New York, 1987., 1987.

[9] M. Richardson und P. Domingos, „Markov logic networks,“ Machine learning 62.1-2: 107-136., https://homes.cs.washington.edu/~pedrod/mln.pdf, 2006.

[10] B. X. T.G. Dietterich, „"Integrating Multiple Learning Components through Markov Logic",“ in *23. AAAI Conference on Artificial Intelligence*, Chicago, 2008.

[11] F. Niu, C. Ré, A. Doan und J. Shavlik, „Tuffy: Scaling up statistical inference in markov logic networks using an rdbms,“ in *Proceedings of the VLDB Endowment 4.6, pages 373-384.*.

[12] D. Venugopal, S. Sarkhel und V. Gogate, „Just Count the Satisfied Groundings: Scalable Local-Search and Sampling Based Inference in MLNs,“ in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence 2015*, 2015.

[13] A. Kuwertz, D. Mühlenberg, J. Sander und W. Müller, „Applying Knowledge-Based Reasoning for Information Fusion in Intelligence, Surveillance, and Reconnaissance,“ in *Multisensor Fusion and Integration in the Wake of Big Data, Deep Learning and Cyber Physical System. MFI 2017*, Springer, Cham, 2018.

[14] P. Oliveira, „Probabilistic Reasoning in the Semantic Web using Markov Logic”, MSc Thesis,“ Department of Informatics Engineering, University of Coimbra, Portugal, Coimbra, Portugal, 2009.

[15] Computer Science Department Stanford, „Tuffy: A Scalable Markov Logic Inference Engine,“ 19 09 2019. [Online]. Available: http://i.stanford.edu/hazy/tuffy/doc/.

[16] A. Doan, F. Niu, C. Ré, J. Shavlik und C. Zhang, „User Manual of Tuffy 0.3," 1 May 2011. [Online]. Available: http://i.stanford.edu/hazy/tuffy/doc/tuffy-manual.pdf.

[17] M. A. Butnaru and R. T. Ionescu., "Moroco: The moldavian and romanian dialectal corpus," in *arXiv preprint arXiv:1901.06543*, 2019.

[18] T. Kocmi and O. Bojar, "Lanidenn: Multilingual language identification on character window," in *arXiv preprint arXiv:1701.03338*, 2017.

[19] A. Ali, N. Dehak, P. Cardinal, S. Khurana, S. Yella, J. Glass, P. Bell and S. Renals, "Automatic dialect detection in arabic broadcast speech," in *arXiv preprint arXiv:1509.06928*, 2015.

[20] K. Darwish, H. Sajjad and H. Mubarak, "Verifiably effective Arabic dialect identification," in *In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (pp. 1465-1468)*, 2014.

[21] M. Zampieri, B. Gebre and S. Diwersy, "N-gram Language Models and POS Distribution for the Identification of Spanish Varieties," in *In Proceedings of TALN 2013 (Volume 2: Short Papers) (pp. 580-587)*, 2013.

[22] M. Lui and T. Baldwin, "langid. py: An off-the-shelf language identification tool," in *In Proceedings of the ACL 2012 system demonstrations (pp. 25-30)*, 2012.

[23] N. Chittaragi, A. Prakash and S. Koolagudi, "Dialect identification using spectral and prosodic features on single and ensemble classifiers," in *Arabian Journal for Science and Engineering, 43(8), pp.4289-4302*, 2018.

[24] T. Mikolov, I. Sutskever, K. Chen, G. Corrado and J. Dean, "Distributed representations of words and phrases and their compositionality," in *In Advances in neural information processing systems (pp. 3111-3119)*, 2013.

[25] J. Devlinv, M. Chang, K. Lee and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *arXiv preprint arXiv:1810.04805*, 2018.

[26] E. Rahm and P. A. Bernstein, " A survey of approaches to automatic schema matching," in *the VLDB Journal, 10(4), 334-350.*, 2001.

[27] H. Ben Hamadou, "Querying heterogeneous data in NoSQL document stores," in *Doctoral dissertation, Université de Toulouse, Université Toulouse III-Paul Sabatier*, 2019.

[28] P. Shvaiko and J. Euzenat, "A survey of schema-based matching approaches," in *In Journal on data semantics IV (pp. 146-171). Springer, Berlin, Heidelberg*, 2005.

[29] K. Lakshminarayan, S. A. Harp and T. Samad, "Imputation of missing data in industrial databases," in *Applied intelligence, 11(3), 259-275*, 1999.

[30] W. Nutt, S. Razniewski and G. Vegliach, "Incomplete databases: missing records and missing values," in *In International Conference on Database Systems for Advanced Applications (pp. 298-310). Springer, Berlin, Heidelberg*, 2012.

[31] A. J. Henry, N. D. Hevelone, S. Lipsitz and L. L. Nguyen, "Comparative methods for handling missing data in large databases," in *Journal of vascular surgery, 58(5), 1353-1359*, 2013.

[32] B. Khaleghi, A. Khamis, F. O. Karray and S. N. Razavi, "Multisensor data fusion: A review of the state-of-the-art," in *Information fusion, 14(1), 28-44*, 2013.

[33] W. Xu, M. Li and X. Wang, "Information fusion based on information entropy in fuzzy multi-source incomplete information system," *International Journal of Fuzzy Systems,* vol. 19(4), pp. 1200-1216, 2017.

[34] M. Li and X. Zhang, "Information fusion in a multi-source incomplete information system based on information entropy," in *Entropy, 19(11), 570*, 2017.

[35] B. Ball, B. Karrer and M. E. Newman, "Efficient and principled method for detecting communities in networks," in *Physical Review E, 84(3), 036103*, 2011.

[36] B. Karrer and M. Newman, "Stochastic blockmodels and community structure in networks," in *Physical review E 83(1), 016107*, 2011.

[37] X. Wang, P. W. J. Cui, J. Pei, W. Zhu and S. Yang, "Community preserving network embedding," in *Thirty-first AAAI conference on artificial intelligence*, 2017.

[38] T. Yang, R. Jin and Y. Z. S. Chi, "Combining link and content for community detection: a discriminative approach," in *In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 927–936*, 2009.

[39] X. Wang, D. Jin, X. Cao, L. Yang and W. Zhang, "Semantic community identification in large attribute networks," in *In: Thirtieth AAAI Conference on Artificial Intelligence* , 2016.

[40] J. Yang, J. McAuley and J. Leskovec, "Community detection in networks with node attributes," in *2013 IEEE 13th International Conference on Data Mining. pp. 1151–1156. IEEE*, 2013.

[41] M. Qin, D. Jin, K. Lei, B. Gabrys and K. Musial-Gabrys, "Adaptive community detection incorporating topology and content in social networks," in *Knowledge-Based Systems 161, 342–356* , 2018.

[42] D. Surian, D. Nguyen, G. Kennedy, M. Johnson, E. Coiera and A. Dunn, "Characterizing twitter discussions about hpv vaccines using topic modeling and community detection," in *Journal of medical Internet research 18(8), e232* , 2016.

[43] Y. Ruan, D. Fuhry and S. Parthasarathy, “Efficient community detection in large networks using content and links,” in *In: Proceedings of the 22nd international conference onWorldWideWeb. pp. 1089–1098*, 2013.

[44] T. Zhao, H. Huang and X. Fu, “Identifying topical opinion leaders in social community question answering,” in *In: International Conference on Database Systems for Advanced Applications. pp. 372–387. Springer*, 2018.

[45] M. Rosvall and C. Bergstrom, “Maps of random walks on complex networks reveal community structure,” in *PNAS 105(4), 1118–1123. https://doi.org/10.1073/pnas.0706851105*, 2007.

[46] U. Raghavan, R. Albert and S. Kumara, “Near linear time algorithm to detect community structures in large-scale networks,” in *Physical Review E 76(036106). https://doi.org/10.1103/PhysRevE.76.036106*, 2007.

[47] M. Newman, “Finding community structure in networks using the eigenvectors of matrices,” in *Phys. Rev. E 74(036104). https://doi.org/10.1103/PhysRevE.74.036104*, 2006).

[48] V. Blondel, J. Guillaume and R. Lambiotte, “Fast unfolding of communities in large networks,” in *Journal of Statistical Mechanics: Theory and Experiment 2008. https://doi.org/10.1088/1742-5468/2008/10/P10008*, 2008.

[49] J. Reichardt and S. Bornholdt, “Statistical mechanics of community detection,” in *Phys. Rev. E 74(016110). https://doi.org/10.1103/PhysRevE.74.016110*, 2006.

[50] P. Pons and M. Latapy, “Computing communities in large networks using random walks,” in *Lecture Notes in Computer Science, Springer 3733. https://doi.org/10.1007/1156959631*, 2005.

[51] J. H. Lau and T. Baldwin, “ An empirical evaluation of doc2vec with practical insights into document embedding generation,” in *arXiv preprint arXiv:1607.05368*, 2016.

[52] R. T. Ionescu, M. Popescu and A. Cahill, “Can characters reveal your native language? A language-independent approach to native language identification,” in *In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (pp. 1363-1373)*, 2014.

[53] M. Washha, A. Qaroush and F. Sedes, “Leveraging time for spammers detection on twitter,” in *In: Proceedings of the 8th International Conference on Management of Digital EcoSystems. pp.109–116. ACM*, 2016.

[54] J. Mothe, K. Mkhitaryan and M. Haroutunian, “Community detection: Comparison of state of the art algorithms,” in *In: 2017 Computer Science and Information Technologies (CSIT). pp.125–129. IEEE* , 2017.

[55] L. Danon, A. Diaz-Guilera, J. Duch and A. Arenas, “Comparing community structure identification,” in *Journal of Statistical Mechanics: Theory and Experiment 2005(09), P09008* , 2005.

[56] M. Haroutunian, K. Mkhitaryan and J. Mothe, "f-Divergence Measures for Evaluation in Community Detection (regular paper)," in *In: Workshop on Collaborative Technologies and Data Science in Smart City Applications (CODASSCA 2018). pp. 137–145. AUA NEWSROOM (American University of Armenia, affiliated with the University of California)), http://newsroom.aua.am/* , 2018.

[57] M. Haroutunian, K. Mkhitaryan and J. Mothe, "A New Information-Theoretical Distance Measure for Evaluating Community Detection Algorithms," in *Journal of Universal Computer Science 25(8), 887–903, http://www.jucs.org/jucs_25_8/a_new_information_theoretical/jucs_25_08_0887_0903_haroutunian.pdf*, 2019.

[58] M. Newman and M. Girvan, "Finding and evaluating community structure in networks," in *Physical review E 69(2), 026113*, 2004.

[59] T. Caliński and J. Harabasz, "A dendrite method for cluster analysis," in *Communications in Statistics-theory and Methods, 3(1), 1-27*, 1974.

[60] M. Halkidi, Y. Batistakis and M. Vazirgiannis, "On clustering validation techniques," in *Journal of intelligent information systems, 17(2-3), 107-145*, 2001.

[61] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," in *Journal of computational and applied mathematics, 20, 53-65*, 1987.

[62] Data Fusion Panel Joint Directors of Laboratories Technical Panel for C3, "Data Fusion Lexicon," 1991. [Online]. Available: https://apps.dtic.mil/dtic/tr/fulltext/u2/a529661.pdf.

[63] A. Artikis, A. Margara, M. Ugarte, S. Vansummeren and M. Weidlich, "Complex event recognition languages: Tutorial," in *In Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems (pp. 7-10)*, 2017.

[64] G. Cugola and A. Margara, "Processing flows of information: From data stream to complex event processing," in *ACM Computing Surveys (CSUR), 44(3), 1-62*, 2012.

[65] B. P. L. Lau, S. H. Marakkalage, Y. Zhou, N. U. Hassan, C. Yuen, M. Zhang and U. X. Tan, "A survey of data fusion in smart city applications," in *Information Fusion, 52, 357-374*, 2019.

[66] W. Luan, D. Sharp and S. Lancashire, "Smart grid communication network capacity planning for power utilities," in *In IEEE PES T&D 2010 (pp. 1-4). IEEE*, 2010.

[67] S. Consoli, D. Reforgiato Recupero, M. Mongiovi, V. Presutti, G. Cataldi and W. Patatu, "An urban fault reporting and management platform for smart cities," in *In Proceedings of the 24th International Conference on World Wide Web (pp. 535-540)*, 2015.

[68] F. Ricci, "Travel recommender systems," in *IEEE Intelligent Systems, vol. 17, no. 6, pp. 55–57*, 2002.

[69] N. Dawar and N. Kehtarnavaz, “A convolutional neural network-based sensor fusion system for monitoring transition movements in healthcare applications,” in *2018 IEEE 14th International Conference on Control and Automation (ICCA). IEEE, pp. 482–485*, 2018.

[70] L. Jayasinghe, N. Wijerathne, C. Yuen and M. Zhang, “Feature learning and analysis for cleanliness classification in restrooms,” in *IEEE Access, vol. 7, pp. 14 871–14 882*, 2019.

[71] A. Ghorpade, F. C. Pereira, F. Zhao, C. Zegras and B.-A. M., “An integrated stop-mode detection algorithm for real world smartphone-based travel survey,” in *Transportation Research Board 94th Annual Meeting, vol. 15, p. 6021*, 2015.

[72] J. Han, J. Pei and M. Kamber, “Data mining: concepts and techniques,” in *Elsevier*, 2011.

[73] C. Tsirogiannis, „Mapping the Supply: usual suspects and identified antiquities in "reputable" auction houses in 2013,“ 2013.

[74] G. R. Lock, „Beyond the map: Archaeology and Spatial Data,“ *NATO Science Series, Series A life Science,* Bd. 231, p. 232, 2000.

[75] „ICOM,“ [Online]. Available: https://icom.museum/en/resources/red-lists/.

[76] D. S. Fearon, Jr., C. L. Borgman, S. Traweek und L. Wynholds, „Curators to the Stars. Proceedings of the 73rd ASIS&T Annual,“ *Meeting on Navigating Streams in an Information Ecosystem,* Bd. 47, p. 162, 2010.

[77] J. Smith, „Motivations and Expectations for Central Database Creation at an Egyptian Archaeological Site,“ 2019.

[78] „CIDOC CRM,“ [Online]. Available: http://www.cidoc-crm.org/.

[79] E. LE GOFF und C. TUFFERY, „Comment l’INRAP exploite le CIDOC pour les archives de fouille et la documentation archéologique ?,“ *Ecole thématique DONIPAT, Aussois, 14-18 octobre 2019,* 2019.

[80] „CVAONLINE,“ [Online]. Available: http://www.cvaonline.org/cva/browse.htm.

[81] „Münzkabinett,“ [Online]. Available: http://www.cvaonline.org/cva/browse.htm.

[82] „IADB UK,“ [Online]. Available: https://www.iadb.co.uk/iadb2017.php.

[83] „ADS Archaeology Data Service,“ [Online]. Available: https://archaeologydataservice.ac.uk/.

[84] „ARACHNE Germany,“ [Online]. Available: https://arachne.dainst.org/.

[85] „PACTOLS DOLIA,“ [Online]. Available: https://masa.hypotheses.org/pactols.

[86] „Artefacts CNRS," [Online]. Available: http://artefacts.mom.fr/fr/forum.php.

[87] C. Chartrelle, A. Kerep und V. Michel, „Channels and actors of the illicit cultural trafficking: perspectives and solutions.," in *Social platform endangered cultural heritage and on illicit trafficking of cultural goods" Deliverable 2.4, Forum 1 Report, Lyon 25-26 February*, LYON, 2020.

[88] C. Hine, Databases as Scientific Instruments and Their Role in the Ordering of Scientific Work., S. S. o. Science, Hrsg., 2006.

[89] „News Artnet," [Online]. Available: https://news-artnet-com.cdn.ampproject.org/c/s/news.artnet.com/art-world/germany-illicit-antiquities-1826016/amp-page.

[90] R. Lacey, Bidding for Sothebys, Little Brown & Co; 1st edition, 1998.

[91] „EUROPOL," [Online]. Available: https://www.europol.europa.eu/newsroom/news/over-41-000-artefacts-seized-in-global-operation-targeting-illicit-trafficking-of-cultural-goods.

[92] N. Brodie, „"The Internet Market in Antiquities"," *Countering Illicit Traffic in Cultural Goods; The Global Challenge of Protecting the World's Heritage, edited by F. DESMARAIS, 11-20. Paris: ICOM.,* pp. 11-20, 2015.

[93] M. Wantuch-Thole, „Cultural property in cross-border litigation, Turning rights into claims".

[94] I. I. UNESCO, „Basic Actions Concerning Cultural Objects Being Offered for Sale Over the Internet.," UNESCO, Paris, 2006.

[95] M. Templ und P. A. Filzmoser, „Exploring incomplete data using visualization techniques," *Adv. Data Anal. Classif,* Bd. 6, Nr. 1, pp. 29-47, 2012.

[96] „Missing data? Quickly impute values using SAS Viya - SAS Users".

[97] L. Zhang, Y. Xie, L. Xidao und X. Zhang, „Multi-source heterogeneous data fusion," *International Conference on Artificial Intelligence and Big Data (ICAIBD),* pp. 47-51, 2018.

[98] F. E. White, „Data fusion lexicon, joint directors of laboratories, technical panel for C3, data fusion sub-panel," 1987.

[99] D. Pratihar, „Intelligent Autonomous Systems: Foundations and Applications," Bd. 275, 2010.

[100] H. Yeon, M. Pi, H. Son und Y. Jang, „Visual predictive modeling of incomplete time series panel data," in *International Symposium on Visual Information Communication and Interaction*, 2019.

[101] A. S., „Visual Analytics of Missing Data in Epidemiological Cohort Studies," in *VCBM*, 2017.

[102] M. F. R. M. S. a. J. S. P. Valero-Mora, „A Plot for the Visualization of Missing Value Patterns in Multivariate Data,“ Bd. 24, Nr. 1, p. 9, 2019.

[103] L. A. F. Park, J. C. Bezdek, C. Leckie, R. Kotagiri, J. Bailey und M. Palaniswami, „Visual assessment of clustering tendency for incomplete data,“ *IEEE Trans. Knowl. Data Eng.,* Bd. 28, Nr. 12, p. 3409–3422, 2016.

[104] D. Hall und J. Llinas, „An introduction to multisensor data fusion,“ *Proc. IEEE,* Bd. 85, Nr. 1, pp. 6-23, 1997.

[105] S. Schreiber-Ehle und W. Koch, „The JDL model of data fusion applied to cyber-defence—a review paper,“ in *Workshop on Sensor Data Fusion: Trends, Solutions, Applications (SDF)*, 2012.

[106] R. Zuech, T. Khoshgoftaar und R. Wald, „Intrusion detection and big heterogeneous data: a survey,“ *J. Big Data,* Bd. 2, Nr. 1, p. 3, 2015.

[107] V. Dragos, „Developing a core ontology to improve military intelligence analysis. International Journal of Knowledge-Based and Intelligent Engineering Systems,“ *IOS Press 17 (1),* pp. 29-36, 2013.

[108] „Intelligence source and information reliability,“ [Online]. Available: https://en.wikipedia.org/wiki/Intelligence_source_and_information_reliability.

[109] E. Casey, S. Barnum, R. Griffith, J. Snyder, H. van Beek, E. van Eijk, R. van Baar und A. Nelson, „The Evolution of Expressing and Exchanging Cyber-investigation Information in a Standardized Form,“ in *Chapter for EU EVIDENCE Project publication*.

[110] P. Mitzias, E. Kontopoulos, J. Staite, T. Day, G. Kalpakis, T. Tsikrika, H. Gibson, S. Vrochidis, B. Akhgar und I. Kompatsiaris, „Deploying Semantic Web Technologies for Information Fusion of Terrorism-related Content and Threat Detection on the Web,“ *WI '19 Companion,* pp. 193-199, 2019.

[111] TRIL, ICCS, „Deliverable D8.1 - Ethical and legal guidelines for the use of PREVISION tools,“ 2019.

[112] G. Sudhamathy und C. J. Venkateswaran, R Programming: An Approach to Data Analytics, MJP Publisher, 2019.

[113] L. Rokach und O. Maimon, „Top-down induction of decision trees classifiers - a survey,“ 24 October 2005.

[114] N. Tariverdiyev, „Machine Learning Algorithms : Decision Trees,“ 18 09 2019. [Online]. Available: https://mc.ai/machine-learning-algorithms-decision-trees/.

[115] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman und A. Galstyan, „A Survey on Bias and Fairness in Machine Learning,“ 17 September 2019.

[116] H. Li, „Smile - Statistical Machine Intelligence and Learning Engine,“ 19 09 2019. [Online]. Available: https://haifengl.github.io/smile/data.html.

[117] L. Breiman, J. H. Friedman, R. A. Olshen und C. J. Stone, Classification and regression trees, Monterey: Brooks/Cole Publishing, 1984.

[118] „Graphviz - Graph Visualization Software," 19 09 2019. [Online]. Available: http://www.graphviz.org/.

[119] H. Daumé, A Course in Machine Learning, 2017.

[120] H. Deng, „Why random forests outperform decision trees," 28 October 2018. [Online]. Available: https://towardsdatascience.com/why-random-forests-outperform-decision-trees-1b0f175a0b5.

[121] „scikit-learn," 2020. [Online]. Available: https://scikit-learn.org.

[122] A. Deshpande, Artificial Intelligence for Big Data, Packt, 2018.

[123] C. Albon, Nachine Learning with Python Cookbook, O'Reilly, 2018.

[124] L. Sommerer, „Geospatial Predictive Policing – Research Outlook & A Call For Legal Debate," *Neue Kriminalpolitik,* Bd. 29, Nr. 2, pp. 147-164, 2017.

[125] A. G. Ferguson, „The Rise of Big Data Policing," *New York University Press,* 2017.

[126] „Trend (Statistik)," [Online]. Available: https://de.wikipedia.org/wiki/Trend_(Statistik)#Allgemeines. [Zugriff am April 2020].

[127] „NumPy," [Online]. Available: https://numpy.org/index.html. [Zugriff am 16 April 2020].

[128] C. D. Manning und H. Schütze, Foundations of Statistical Natural Language Processing, Cambridge, Massachusetts, London, England: The MIT Press, 1999.

[129] T. K. Ho, „Random decision forests," in *Proceedings of the 3rd International Conference on Document Analysis and Recognition*, 1995.

[130] „Sali A. Tagliamonte und R. Harald Baayen. Models, forests, and trees of york english: Was/were variation as a case study for statistical practice," *Language Variation and Change,* Bd. 24, Nr. 2, p. 135–178, 2012.

[131] H. Schmid, A. Fitschen und U. Heid, „SMOR: A german computational morphology covering derivation, composition and inflection," *LREC,* 2004.

[132] H. H. Liu, Machine Learning - A quantitative approach, PerfMath, 2018.

[133] T. B. N. Hoang and J. Mothe, "Predicting information diffusion on Twitter–Analysis of predictive features," *Journal of computational science,* vol. 28, pp. 257-264, 2018.

[134] N. Fuhr, A. Giachanou, G. Grefenstette, I. Gurevych, A. Hanselowski, K. Jarvelin, J. R., L. Y., M. J., N. W. and I. Peters, "An information nutritional label for online documents," in *In ACM SIGIR Forum (Vol. 51, No. 3, pp. 46-66)*, New York, USA, 2018.

[135] N. Hassan, B. Adair, J. T. Hamilton, C. Li, M. Tremayne, J. Yang and C. Yu, "The quest to automate fact-checking," in *In Proceedings of the 2015 Computation+ Journalism Symposium*, 2015.

[136] P. Gencheva, P. M. L. Nakov, A. Barrón-Cedeño and I. Koychev, "A context-aware approach for detecting worth-checking claims in political debates," in *In Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017 (pp. 267-276).*, 2017.

[137] C. Zuo, A. Karakas and R. Banerjee, "A Hybrid Recognition System for Check-worthy Claims Using Heuristics and Supervised Learning.," in *In Working Notes of CLEF 2018 - Conference and Labs of the Evaluation Forum*, Avignon, France., 2018.

[138] C. Hansen, C. Hansen, J. G. Simonsen and C. Lioma, "The Copenhagen Team Participation in the Check-Worthiness Task of the Competition of Automatic Identification and Verification of Claims in Political Debates of the CLEF-2018 CheckThat! Lab," in *In Working Notes of CLEF 2018 - Conference and Labs of the Evaluation Forum*, Avignon, France, 2018.

[139] K. Yasser, M. Kutlu and T. Elsayed, "bigIR at CLEF 2018: Detection and Verification of Check-Worthy Political Claims," in *In Working Notes of CLEF 2018 - Conference and Labs of the Evaluation Forum*, Avignon, France, 2018.

[140] B. Ghanem, M. Montes-y-Gómez, F. M. R. Pardo and P. Rosso, "UPV-INAOE-Check That: Preliminary Approach for Checking Worthiness of Claims," in *In Working Notes of CLEF 2018 - Conference and Labs of the Evaluation Forum*, Avignon, France., 2018.

[141] R. Agez, C. Bosc, C. Lespagnol, J. Mothe and N. Petitcol, "IRIT at CheckThat! 2018," in *In Working Notes of CLEF 2018 - Conference and Labs of the Evaluation Forum*, Avignon, France, 2018.

[142] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, P. P. M. Blondel, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research ,* vol. 12, p. 2825–2830, 2011.

[143] F. Å. Nielsen, „AFINN,“ [Online]. Available: www2.imm.dtu.dk/pubdb/p.php?6010.

[144] J. S. Justeson and S. M. Katz, "Technical terminology: some linguistic properties and an algorithm for identification in text," *Natural language engineering,* vol. 1(1), pp. 9-27, 1995.

[145] S. Bird, E. Klein and E. Loper, "Natural language processing with Python," O'Reilly Media, Inc., 2009.

[146] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado und J. Dean, „Distributed representations of words and phrases and their compositionality.," in *In Advances in neural information processing systems (pp. 3111-3119).*, 2013.

[147] M. Z. Ullah, M. Shajalal, A. N. Chy and M. Aono, "Query subtopic mining exploiting word embedding for search result diversification," in *In Asia Information Retrieval Symposium (pp. 308-314). Springer, Cham.*, 2016.

[148] J. Mothe, F. Ramiandrisoa and M. Rasolomanana, "Automatic keyphrase extraction using graph-based methods," in *In Proceedings of the 33rd Annual ACM Symposium on Applied Computing (pp. 728-730)*, 2018.

[149] P. Nakov, A. Barrón-Cedeno, T. Elsayed, R. Suwaileh, L. Màrquez, W. Zaghouani, P. Atanasova, S. Kyuchukov and G. Da San Martino, "Overview of the CLEF-2018 CheckThat! Lab on automatic identification and verification of political claims," in *In International Conference of the Cross-Language Evaluation Forum for European Languages (pp. 372-387). Springer, Cham.*, 2018.

[150] G. Lemaître, F. Nogueira and C. K. Aridas, "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning," *The Journal of Machine Learning Research,,* vol. 18(1), pp. 559-563, 2017.

[151] J. Caron und J. Light, „"Social Media has Opened a World of 'Open communication:'" experiences of Adults with Cerebral Palsy who use Augmentative and Alternative Communication and Social Media.," *Augmentative and Alternative Communication,* Nr. 32, pp. 25-40, 2016.

[152] J. Lipschultz, „Social media communication: Concepts, practices, data, law and ethics," 2017.

[153] A. Marganski und L. Melander, „Intimate partner violence victimization in the cyber and real world: Examining the extent of cyber aggression experiences and its association with in- person dating violence.," *Journal of interpersonal violence,* Bd. 7, Nr. 33, p. 1071–1095, 2018.

[154] J. De´cieux, A. Heinen und H. Willems, „Social media and its role in friendship-driven inter- actions among young people: A mixed methods study.," *YOUNG,* Bd. 1, Nr. 27, p. 18–31, 2019.

[155] F. Mishna, C. Regehr, A. Lacombe-Duncan, J. Daciuk, G. Fearing und M. Van Wert, „Social media, cyber-aggression and student mental health on a university campus," *Journal of mental health,* Bd. 3, Nr. 27, p. 222–229, 2018.

[156] J. Chen, H. Xu und A. Whinston, „Moderated online communities and quality of user- generated content," *Journal of Management Information Systems,* Bd. 28, Nr. 2, p. 237–268, 2011.

[157] S. Myers West, „Censored, suspended, shadowbanned: User interpretations of content mod- eration on social media platforms," *New Media & Society,* Bd. 20, Nr. 11, p. 4366–4383, 2018.

[158] R. Kumar, A. Ojha, S. Malmasi und M. Zampieri, „Benchmarking Aggression Identifica- tion in Social Media," in *Proceedings of the First Workshop on Trolling, Aggression and Cyberbulling (TRAC)*, Santa Fe (USA), 2018.

[159] A. Schmidt und M. Wiegand, „A Survey on Hate Speech Detection Using Natural Language Processing.," in *Proceedings of the Fifth International Workshop on Natural Language Pro- cessing for Social Media. SocialNLP@EACL*, Valencia, Spain, 2017.

[160] G. Priyadharshini, „A pragmatic supervised learning methodology of hate speech detection in social media," 2019.

[161] S. Aroyehun und A. Gelbukh, „Aggression detection in social media: Using deep neural net- works, data augmentation, and pseudo labeling," in *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying*, 2018.

[162] R. Kumar, A. Reganti, A. Bhatia und T. Maheshwari, „Aggression-annotated corpus of hindi- english code-mixed data," *ArXiv,* p. arXiv preprint arXiv:1803.09402, 2018.

[163] K. Raiyani, T. Gonçalves, P. Quaresma und V. Nogueira, „Fully connected neural network with advance preprocessor to identify aggression over facebook and twitter," in *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying, TRAC@COLING*, Santa Fe, New Mexico, USA, 2018.

[164] F. Ramiandrisoa und J. Mothe, „Irit at trac 2018," in *Proceedings of the First Workshop on Trolling, Aggression and Cyberbulling, TRAC@COLING*, Santa Fe, New Mexico, USA, 2018.

[165] M. Osama und S. El-Beltagy, „A transfer learning approach for emotion intensity prediction in microblog text.," in *Proceedings of the International Conference on Advanced Intelligent Systems and Informatics 2019*, Cairo, Egypt, 26-28 October 2019.

[166] I. Abdou Malam, M. Arziki, M. Nezar Bellazrak, F. Benamara, A. El Kaidi, B. Es-Saghir, Z. He, M. Housni, V. Moriceau, J. Mothe und F. Ramiandrisoa, „IRIT at e-Risk (regular paper).," in *International Conference of the CLEF Association, CLEF 2017 Labs Working Notes.*, 2017.

[167] F. Ramiandrisoa, J. Mothe, F. Benamara und V. Moriceau, „IRIT at e-Risk 2018 (regular paper)," in *Conference and Labs of the Evaluation Forum, Living Labs*, Avignon, France, 2018.

[168] N. Fuhr, A. Giachanou, G. Grefenstette, I. Gurevych, A. Hanselowski, K. Jarvelin, R. Jones, Y. Liu, J. Mothe, W. Nejdl und e. al., „An information nutritional label for online documents," in *ACM SIGIR Forum*, 2018.

[169] Q. Le und T. Mikolov, „Distributed representations of sentences and documents.," in *Proceed- ings of the 31th International Conference on Machine Learning*, Beijing, China, 21-26 june 2014.

[170] M. Trotzek, S. Koitka und C. Friedrich, „Linguistic metadata augmented classifiers at the CLEF 2017 task for early detection of depression.," in *Working Notes of CLEF 2017 - Con- ference and Labs of the Evaluation Forum,*, Dublin, Ireland, September 11-14, 2017.

[171] R. Rehurek und P. Sojka, „Software framework for topic modelling with large corpora.," in *Pro- ceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. Citeseer*, 2010.

[172] Y. Zhang und B. Wallace, „A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification," *CoRR abs/1510.03820,* 2015.

[173] Y. Kim, „Convolutional neural networks for sentence classification," *CoRR abs/1408.5882,* 2014.

[174] J. Scott, „Social network analysis," *Sociology,* Bd. 22, Nr. 1, pp. 109-127, 1988.

[175] D. Paranyushkin, „Identifying the pathways for meaning circulation using text network analysis," *Nodus Labs,* Bd. 26, 2011.

[176] P. Ruiz, C. Plancq und T. Poibeau, „More than word cooccurrence: Exploring support and opposition in international climate negotiations with semantic parsing," in *LREC: The 10th Language Resources and Evaluation Conference, ELRA*, Portorož, Slovenia, 2016, May 2016.

[177] C. Bail, „Combining natural language processing and network analysis to examine how advocacy organizations stimulate conversation on social media," *National Academy of Sciences,* Bd. 133, pp. 11823-11828, 2016.

[178] A. Rule, J. Cointet und P. Bearman, „Lexical shifts, substantive changes, and continuity in State of the Union discourse 1790-2014," *National Academy of Sciences,* Bd. 112, pp. 10837-10844, 2015.

[179] E. Cambria, „Affective computing and sentiment analysis," *IEEE Intelligent Systems,* Bd. 31, Nr. 2, pp. 102-107, 2016.

[180] J. Archer und M. Jockers, „The bestseller code: Anatomy of the blockbuster novel," *St. Martin's Press,* 2016.

[181] J. Gao, M. Jockers, J. Laudun und T. Tangherlini, „A multiscale theory for the dynamical evolution of sentiment in novels," in *International Conference on Behavioral, Economic and Socio-cultural Computing*, 2016.

[182] M. Roberts, B. Stewart und D. Tingley, „Stm: An R Package for Structural Topic Models," *Journal of Statistical Software,* 2019.

[183] N. Görnitz, M. Kloft, K. Rieck und U. Brefeld, „Toward supervised anomaly detection.," *Journal of Artificial Intelligence Research,,* pp. 46, 235-262., 2013.

[184] B. R. Kiran, D. M. Thomas und R. Parakkal, „An overview of deep learning based methods for unsupervised and semi-supervised anomaly detection in videos.," *Journal of Imaging,,* pp. 4(2), 36., 2018.

[185] R. Chalapathy und S. Chawla, „Deep learning for anomaly detection: A survey.," in *arXiv preprint arXiv:1901.03407.*, 2019.

[186] M. Hasan, J. Choi, J. Neumann, A. K. Roy-Chowdhury und L. S. Davis, „Learning temporal regularity in video sequences.," in *In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 733-742).*, 2016.

[187] P. Vincent, H. Larochelle, Y. Bengio und P. A. Manzagol, „Extracting and composing robust features with denoising autoencoders.," in *In Proceedings of the 25th international conference on Machine learning (pp. 1096-1103).*, 2018.

[188] H. Vu, T. D. Nguyen, A. Travers, S. Venkatesh und D. Phung, „Energy-based localized anomaly detection in video surveillance.," in *In Pacific-Asia Conference on Knowledge Discovery and Data Mining (pp. 641-653). Springer, Cham.*, 2017.

[189] N. Srivastava, E. Mansimov und R. Salakhudinov, „Unsupervised learning of video representations using lstms.," in *In International conference on machine learning (pp. 843-852).*, 2015.

[190] A. Munawar, P. Vinayavekhin und G. De Magistris, „Spatio-temporal anomaly detection for industrial robots through prediction in unsupervised feature space.," in *IEEE Winter Conference on Applications of Computer Vision (WACV) (pp. 1017-1*, 2017.

[191] V. Mahadevan, W. Li, V. Bhalodia und N. Vasconcelos, „Anomaly detection in crowded scenes.," in *In 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (pp. 1975-1981). IEEE.*, 2010.

[192] C. Lu, J. Shi und J. Jia, „Abnormal event detection at 150 fps in matlab.," in *In Proceedings of the IEEE international conference on computer vision (pp. 2720-2727).*.

[193] „UMN. Unusual Crowd Activity Dataset.," [Online]. Available: http://mha.cs.umn.edu/Movies/Crowd-Activity-All.avi.

[194] A. Zaharescu und R. Wildes, „Anomalous behaviour detection using spatiotemporal oriented energies, subset inclusion histogram comparison and event-driven processing.," in *In European Conference on Computer Vision (pp. 563-576). Springer, Berl*, 2010.

[195] Y. Benezeth, P. M. Jodoin, V. Saligrama und C. Rosenberger, „Abnormal events detection based on spatio-temporal co-occurences.," in *In 2009 IEEE Conference on Computer Vision and Pattern Recognition (pp. 2458-2465). IEEE.*, 2009.

[196] R. Leyva, V. Sanchez und C. T. Li, „The LV dataset: A realistic surveillance video dataset for abnormal event detection.," in *In 2017 5th International Workshop on Biometrics and Forensics (IWBF) (pp. 1-6). IEEE.*, 2017.

[197] D. Agnieszka und L. Magdalena, „Detection of outliers in the financial time series using ARIMA models," Applications of Electromagnetics in Modern Techniques and Medicine (PTZE), 2018.

[198] R. Sean, M. Travis, G. Amitava und W. Alfred, „Abnormal Traffic Pattern Detection in Real-Time Financial Transactions," EasyChair Preprint no. 827, 2019.

[199] N. Malini und M. Pushpa, „Analysis on credit card fraud identification techniques based on KNN and outlier detection," Third International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB), 2017.

[200] S. Iman, S. Nawal und B. Faouzia, „Detection of credit card fraud: State of art," International Journal of Computer Science and Network Security, VOL.18 No.11, 2018.

[201] X. Zhou, S. Cheng, M. Zhu, C. Guo, S. Zhou, P. Xu und W. Zhang, „A state of the art survey of data mining-based fraud detection and credit scoring," MATEC Web of Conferences, 2018.

[202] M. Kamal, S. Harsh und K. Gursharanjeet, „Comparative Analysis of Outlier Detection Techniques," International Journal of Computer Applications, 2014.

[203] K. Choroś, „Real Anomaly Detection in Telecommunication Multidimensional Data Using Data Mining Techniques," Computational Collective Intelligence. Technologies and Applications. Springer., Berlin, Heidelberg, 2010.

[204] N. L. Dey, „Anomaly Detection from Call Data Records," In proc. of International Conference on Pattern Recognition and Machine Intelligence, New Delhi, India, 2009.

[205] I. A. Karatepe und E. Zeydan, „Anomaly Detection In Cellular Network Data Using Big Data Analytics.," 20th European Wireless Conference, Barcelona, Spain, 2014.

[206] J. A. Iglesias, A. Ledezma, A. Sanchis und P. Angelov, „Real-Time Recognition of Calling Pattern and Behaviour of Mobile Phone Users through Anomaly Detection and Dynamically-Evolving Clustering.," Applied Sciences vol. 7 no.798, 2017.

[207] K. Sultan, H. Ali und Z. Zhang, „Call Detail Records Driven Anomaly Detection and Traffic Prediction in Mobile Cellular Networks.," IEEE Access, vol.6, pp.2169-3536, 2018.

[208] J. Han, M. Kamber und J. Pei, „Data Mining Concepts and Techniques," 3rd edition, Elsevier, 2012.

[209] Z. He, Z. Xu und S. Deng, „Discovering cluster-based local outliers," Pattern Recognition Letters, Elsevier, vol. 24, pp.1641-1650., 2003.

[210] I. J. Goodfellow, J. Shlens und C. Szegedy, *Explaining and Harnessing Adversarial Examples,* 2014.

[211] I. Evtimov, K. Eykholt, E. Fernandes, T. Kohno, B. Li, A. Prakash, A. Rahmati und D. Song, „Robust Physical-World Attacks on Machine Learning Models," *CoRR,* Bd. abs/1707.08945, 2017.

[212] O. Suciu, S. E. Coull und J. Johns, „Exploring Adversarial Examples in Malware Detection," *CoRR,* Bd. abs/1810.08280, 2018.

[213] A. Madry, A. Makelov, L. Schmidt, D. Tsipras und A. Vladu, *Towards Deep Learning Models Resistant to Adversarial Attacks,* 2017.

[214] Y. Li, J. Bradshaw und Y. Sharma, *Are Generative Classifiers More Robust to Adversarial Attacks?,* 2018.

[215] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik und A. Swami, *Practical Black-Box Attacks against Machine Learning,* 2016.

[216] P. Samangouei, M. Kabkab und R. Chellappa, „Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models," *CoRR,* Bd. abs/1805.06605, 2018.

[217] S. Wang, T. Chen, S. Nepal, C. Rudolph, M. Grobler und S. Chen, *Defending Adversarial Attacks via Semantic Feature Manipulation,* 2020.

[218] A. Pol, V. Berger, C. Germain, G. Cerminara und M. Pierini, „Anomaly Detection with Conditional Variational Autoencoders," 2019.

[219] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville und Y. Bengio, *Generative Adversarial Networks,* 2014.

[220] D. P. Kingma und M. Welling, *Auto-Encoding Variational Bayes,* 2013.

[221] F. B. Aissa, M. Mejdoub und M. Zaied, „A survey on generative adversarial networks and their variants methods," in *Twelfth International Conference on Machine Vision (ICMV 2019)*, 2020.

[222] I. Goodfellow, *NIPS 2016 Tutorial: Generative Adversarial Networks,* 2016.

[223] D. Koller und N. Friedman, Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning, The MIT Press, 2009.

[224] D. P. Kingma und M. Welling, „An Introduction to Variational Autoencoders," *Foundations and Trends® in Machine Learning,* Bd. 12, p. 307–392, 2019.

[225] D. P. Kingma, D. J. Rezende, S. Mohamed und M. Welling, *Semi-Supervised Learning with Deep Generative Models,* 2014.

[226] U. Hwang, J. Park, H. Jang, S. Yoon und N. I. Cho, *PuVAE: A Variational Autoencoder to Purify Adversarial Examples,* 2019.

[227] N. Papernot, F. Faghri, N. Carlini, I. Goodfellow, R. Feinman, A. Kurakin, C. Xie, Y. Sharma, T. Brown, A. Roy, A. Matyasko, V. Behzadan, K. Hambardzumyan, Z. Zhang, Y.-L. Juang, Z. Li, R. Sheatsley, A. Garg, J. Uesato, W. Gierke, Y. Dong, D. Berthelot, P. Hendricks, J. Rauber und R. Long, „Technical Report on the CleverHans v2.1.0 Adversarial Examples Library," *arXiv preprint arXiv:1610.00768,* 2018.

[228] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke und A. Rabinovich, „Going Deeper with Convolutions," *CoRR,* Bd. abs/1409.4842, 2014.

[229] N. Carlini und D. Wagner, *Towards Evaluating the Robustness of Neural Networks,* 2016.

[230] *The MNIST Database.*

[231] A. Krizhevsky, V. Nair und G. Hinton, „CIFAR-10 (Canadian Institute for Advanced Research)".

[232] EUROB, ITTI, VML, SIV, TRT, IOSB, ICCS, PAWA, CBRNE, QMUL, KUL, UPV, „D2.3 Refined System Architecture and Representational Model," 2019.

[233] KUL, CBRNE, „Ethical and Legal Guidelines for the use and development of MAGNETO Tools," 2019.

[234] A. Saxena, „Implementing Decision Trees Using Smile," 19 09 2019. [Online]. Available: https://dzone.com/articles/implementing-decision-tree-using-smile-scala.

[235] S. S. Khan und A. Ahmad, „Cluster center initialization algorithm for K-modes clustering," *Expert Systems with Applications,* Bd. 40, pp. 7444–7456,, 2013.

[236] H. Zinsmeister und L. Lemnitzer, Korpuslinguistik: Eine Einführung, Narr Francke Attempto Verlag, 2015.