Mini-project #3:

Using textual features for malware detection

# Brief summary

In this mini-project, you will use various methods to extract features and information from textual columns in the data. Based on these features you will train a classifier, aiming to detect malicious files.

# Initial data exploration

The dataset contains various attributes (columns) per file. Explore the textual ones. Look for the ones which can contain meaningful information regarding the maliciousness of the file. For example, the file name or the domain from which it was downloaded can be used.

*Include in your report a list of the columns you think can be used. In each, give examples for interesting values, and some intuition or explanation to why this column should be used.*

# Text analysis features

*For future use (time-split test) make sure you perform the feature extraction procedure separately for the first 5 days and the last 2 days.*

There are various methods to extract features from text. In this mini project, we will focus on the two most common ones. Each method is to be used on each column you selected in the first part **separately** (unless there is a *good* reason to do otherwise).

1. N-gram – read [here](https://en.wikipedia.org/wiki/N-gram). Use character level n-gram. Use various values of n (try in the range of one to five). We recommend you work with 3.

Extra: compare results to using other values.

Bag of Words (BoW) – read [here](https://en.wikipedia.org/wiki/Bag-of-words_model). Select your tokenizer smartly – your delimiters may be dots, dashes, spaces, etc. Consider limiting the length of a word.

Extra: consider using “stop-words” (read [here](https://en.wikipedia.org/wiki/Stop_words)) – do you want to have any? Which?

Several things you should consider when using these methods:

1. Removing digits – sometime the pattern is more important than the content. For instance, the pattern of an IP address is much easier to recognize if all digits are replaced by a special character.
2. Casing – is there any difference between a lower and upper case? Consider changing the input to have the same casing.
3. Aggregating the data – the same file may have different names, or it can be downloaded from several domains. Consider listing these values per file, and only then performing the feature extraction procedure.
4. TF/TF-IDF­ (term frequency – inverse document frequency) – read [here](https://en.wikipedia.org/wiki/Tf%E2%80%93idf). TF and TF-IDF are ways to assign weight to each feature which take into account the number of times it appeared in the entire data set (corpus). We recommend using TF-IDF.

Extra: compare results with/without TF/TF-IDF.

(TIP: most machine-learning suites have a built-in n-gram and BoW functionality, as well as a TF-IDF weighting mechanism.)

*Include in your report a description of the methods you used, the selections you made answering the issues described above etc.*

Checkpoint

# Textual features analysis

For each column you selected (file name, domain etc.) and each feature-extraction method (n-gram, BoW), you have many features. To better understand them, analyze them across the dataset. How many distinct values have you got? For each, you can count the number of files it appears in. What is the average and median value of these number? How is it distributed?

Note that our data contains only malicious labels. As for clean files, use files which are common and prevalent in the data. Analyze the distribution again, this time comparing clean vs malicious files. Plot the two distributions in a manner that will show the differences/similarities between them (TIP: you may want to exclude extreme values, and maybe use a log-scale).

*Include in your report graphs and statistics of the analysis you perform, and describe any insights you gained such as: which columns are better than the other? Which feature extraction method may yield better results? Which weighting function should be used?*

# Global features extraction

Other than the textual features, extract per file “global” features (size, prevalence, number of source domains etc.). Analyze per feature its distribution, comparing clean vs malicious files. Plot them in a manner that will show the differences/similarities of them (TIP: you may want to exclude extreme values).

*Include in your report the graphs and statistics of the analysis you perform, and describe any insights you gained, like what feature seems more helpful, why the distribution is as it is etc.*

Checkpoint

# Machine learning

Note that our dataset contains only malicious labels. As for clean files, use files which are common and prevalent in the data.

* 1. Use the features extracted to classify files to clean and malicious. Make sure you use two (or more) types of classifiers and select the most appropriate one.
  2. Use cross-validation to check your results and select the model.
  3. Make sure to use also a time-split to test your results. Be careful with what files you test yourself on. The files should only appear in the last two days (otherwise they are part of the training data.).

*Include in your report the graph and statistics of the results of the classification. Make sure you compare the models you use. Do the results from the cross-validation align with those of the time-split?*

Checkpoint

# Bonus – adjusting TF-IDF for our needs

TF-IDF counts the number of times a “word” appeared in the entire corpus. Consider the case where a “word” is seen in all of the malicious files (but in none of the clean). As the number of malicious files can be high, it will get a low weight, while, in practice, this feature is a strong indicator of the file being malicious. This issue is not a problem with TF-IDF, but rather with its usage in our domain.

Offer an alternative weighting formula which can deal with this issue.