Malware Threat Detection Using Machine Learning

For CyBOK funded project:
Development of an active learning lesson plan and laboratory materials for AI for Security

Dr Hossein Abroshan

Senior Lecturer in Cyber Security

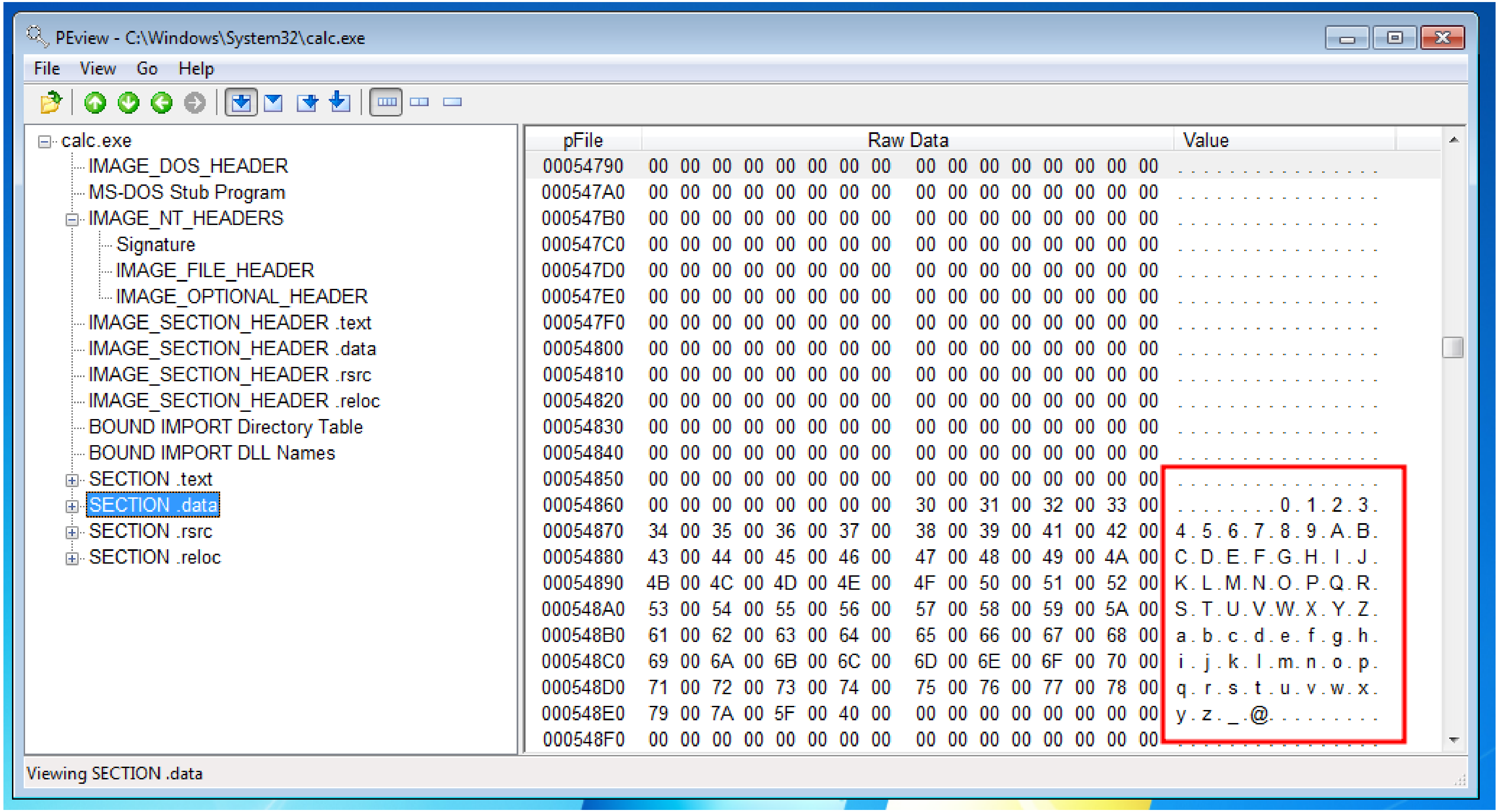
Anglia Ruskin University, Cambridge, UK

Static and Dynamic Malware Analysis (recap)

	Static Malware Analysis	Dynamic Malware Analysis
01	Static analysis is a process of analyzing a malware binary code without actually running the code.	Dynamic analysis requires programs to be executed in a closely monitored virtual environment.
02	It uses a signature-based approach for malware analysis.	It uses a behavior-based approach for malware detection and analysis.
03	It involves file fingerprinting, virus scanning, reverse-engineering the binary, file obfuscation, analyzing memory artifacts, packer detection, & debugging.	Dynamic analysis involves API calls, instruction traces, registry changes, network, and system calls, memory writes, and more.
04	It is ineffective against sophisticated malware programs and codes.	It is effective against all types of malware because it analyzes the sample by executing it.

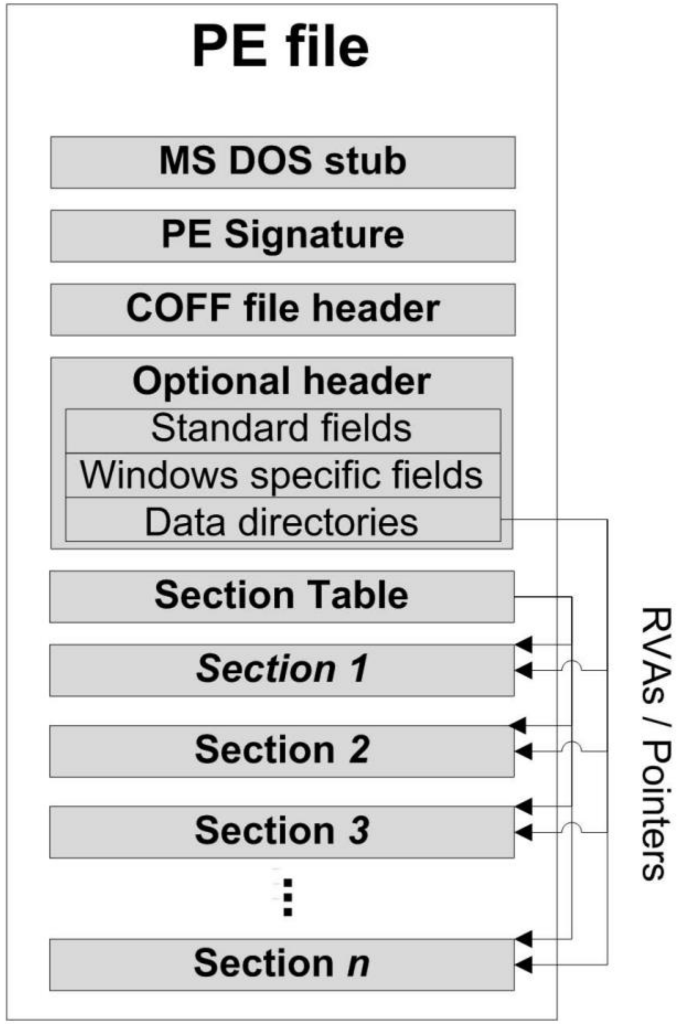
Source: differencebetween.net

PE Header (review)

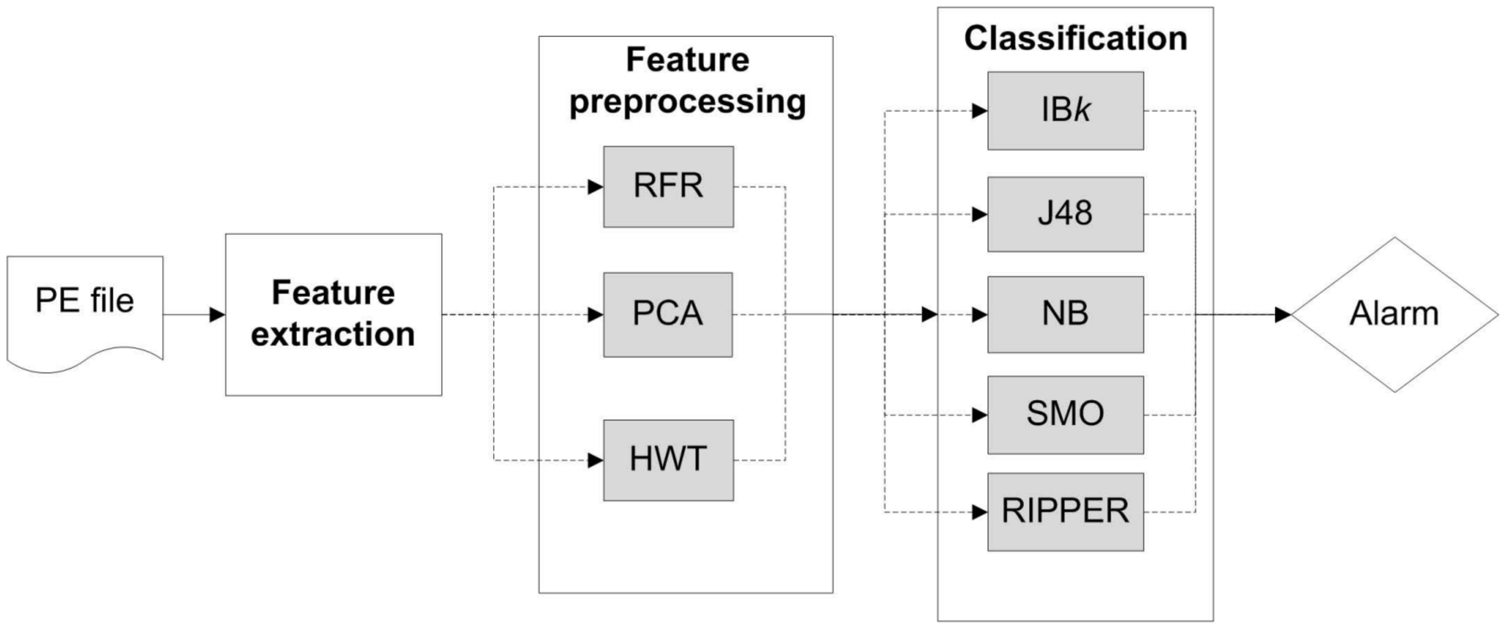


Source: RE -PE

PE Feature Extraction (PE Miner)



PE File Format



The architecture of our PE-Miner framework

Source: [PE-Miner: Mining Structural Information to Detect Malicious Executables in Realtime]

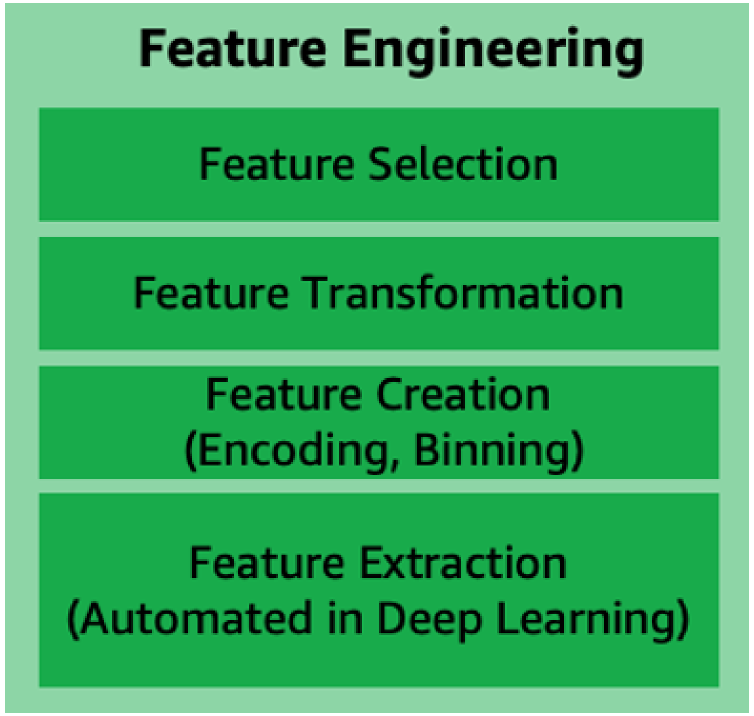
Feature Engineering

Feature creation refers to the creation of new features from existing data to help with better predictions. (e.g., splitting, calculated features)

Feature transformation and imputation include steps for replacing missing features or features that are not valid. (example techniques: forming Cartesian products of features, creating domain-specific features)

Feature extraction involves reducing the amount of data to be processed using dimensionality reduction techniques (example techniques: Principal Components Analysis (PCA), linear discriminant analysis (LDA)). This reduces the amount of memory and computing power required while still accurately maintaining original data characteristics.

Feature selection is the process of selecting a relevant subset of extracted features that contributes to minimising the error rate of a trained model. The feature importance score and correlation matrix can be factored into selecting the most relevant features for model training.



Source: [Amazon.com]

PE Feature Extraction (Example)

Feature Description	Type	Quantity
DLLs referred	binary	73
COFF file header	integer	7
Optional header – standard fields	integer	9
Optional header – Windows specific fields	integer	22
Optional header – data directories	integer	30
.text section – header fields	integer	9
.data section – header fields	integer	9
.rsrc section – header fields	integer	9
Resource directory table & resources	integer	21
Total		189

List of the features extracted from PE files

Source: PE-Miner: Mining Structural Information to Detect Malicious Executables in Realtime

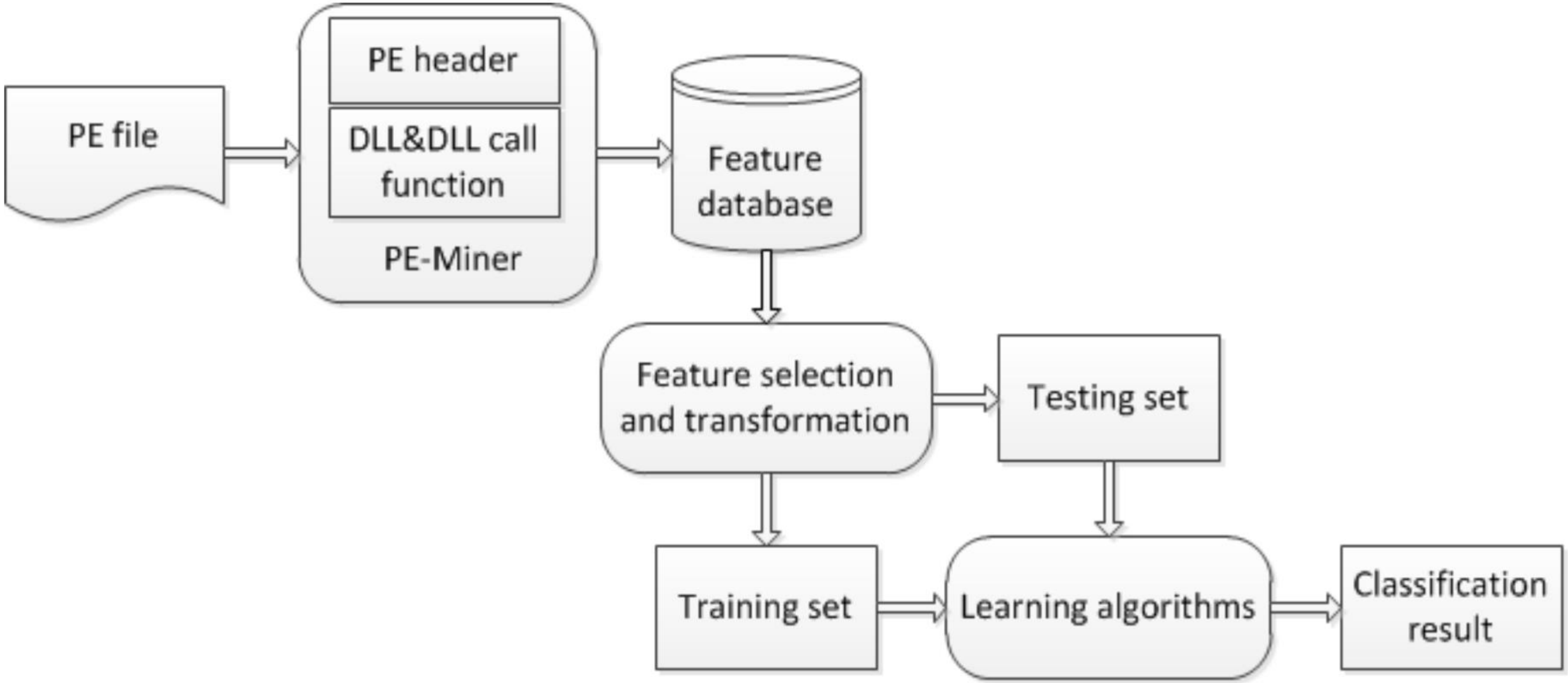
PE Features (Examples)

DLLs referred:
The list of DLLs referred to in an executable effectively provides an overview of its functionality. For example, if an executable calls WINSOCK.DLL or WSOCK.DLL, then it is expected to perform network-related activities. However, there can be exceptions to this assumption as well.

Optional header Windows-specific fields:
The Windows-specific fields of the optional header include information about the operating system version, the image version, etc.

Source: PE-Miner: Mining Structural Information to Detect Malicious Executables in Realtime

Malware Detection Using PE Header



Source: [A Static Malware Detection System Using Data Mining Methods]

Malware Detection Using PE Header (example study)

Dataset	VX Heavens									Malfease
Malware	Backdoor + Sniffer	Constructor + Virtool	DoS + Nuker	Flooder	Exploit + Hacktool	Worm	Trojan	Virus	Average	-
Scenario 1: Detection of packed benign and malicious PE files										
IBK	0.999	1.000	1.000	0.999	0.999	0.998	0.999	0.999	0.999	0.812
J48	0.996	1.000	1.000	0.999	0.999	0.998	0.993	0.999	**0.998**	**0.991**
NB	0.971	0.988	0.963	0.955	0.996	0.980	0.978	0.987	0.977	0.934
RIPPER	0.997	0.996	0.999	0.990	0.993	0.985	0.858	0.998	0.977	0.988
SMO	0.985	0.998	1.000	0.996	0.994	0.994	0.985	0.998	0.994	0.706

An analysis of the robustness of extracted features of PE-Miner (RFR) in one of the scenarios (you can find more scenarios in the research paper)

Source: PE-Miner: Mining Structural Information to Detect Malicious Executables in Realtime

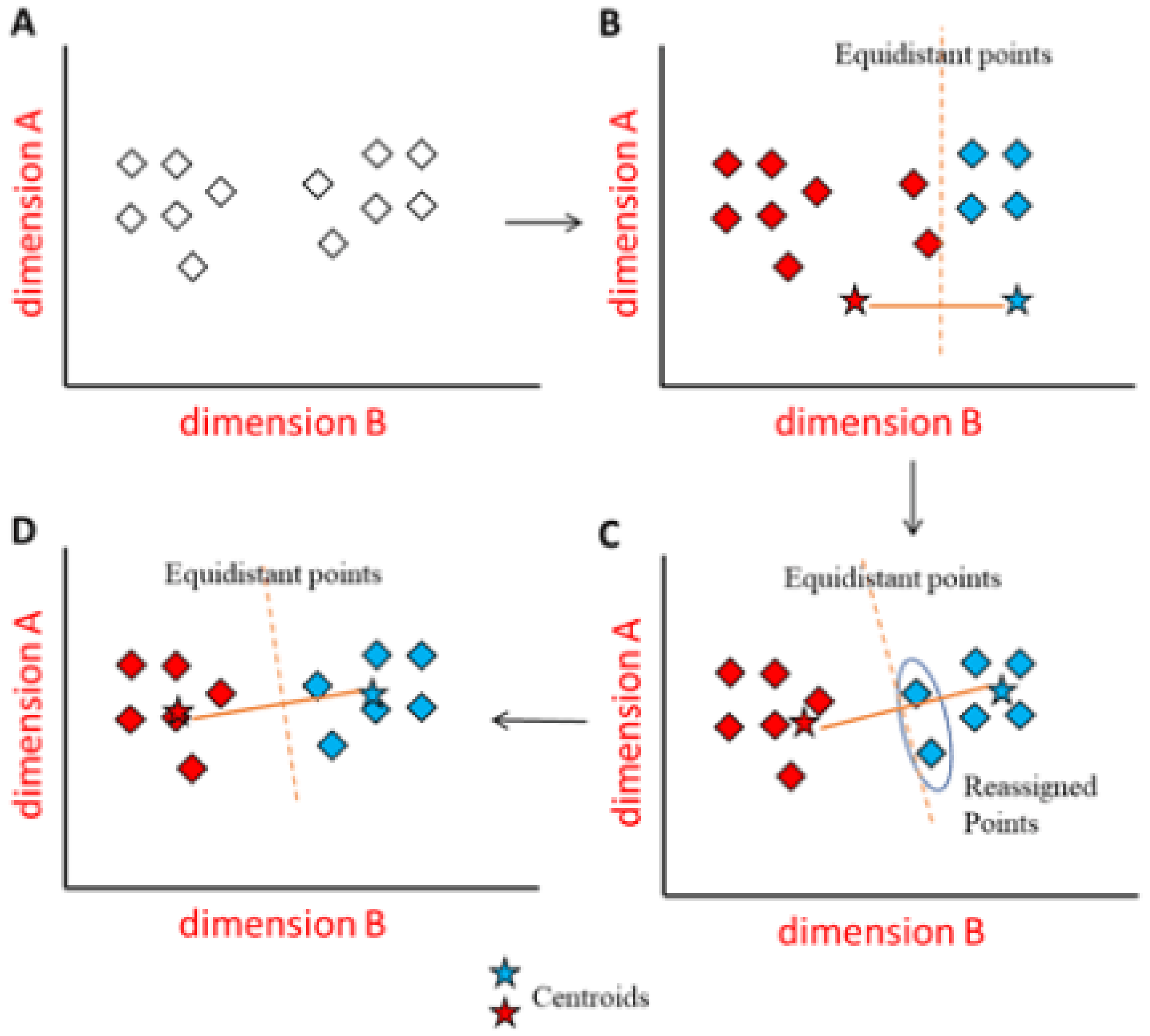
PE Header based Malware Detection

Index	Key Features	Malware (5598)	Normal (1237)	Difference
1	Size Of Initialized Data == 0	1626 (29%)	0 (0%)	29%
2	Unknown Section Name	2709 (48.4%)	16 (1.3%)	47.1%
3	DLL Characteristics == 0	5335 (95.3%)	401 (32.4%)	62.9%
4	Major Image Version == 0	5305 (94.8%)	486 (39.3%)	55.5%
5	Checksum == 0	5084 (90.8%)	474 (38.3%)	52.5%

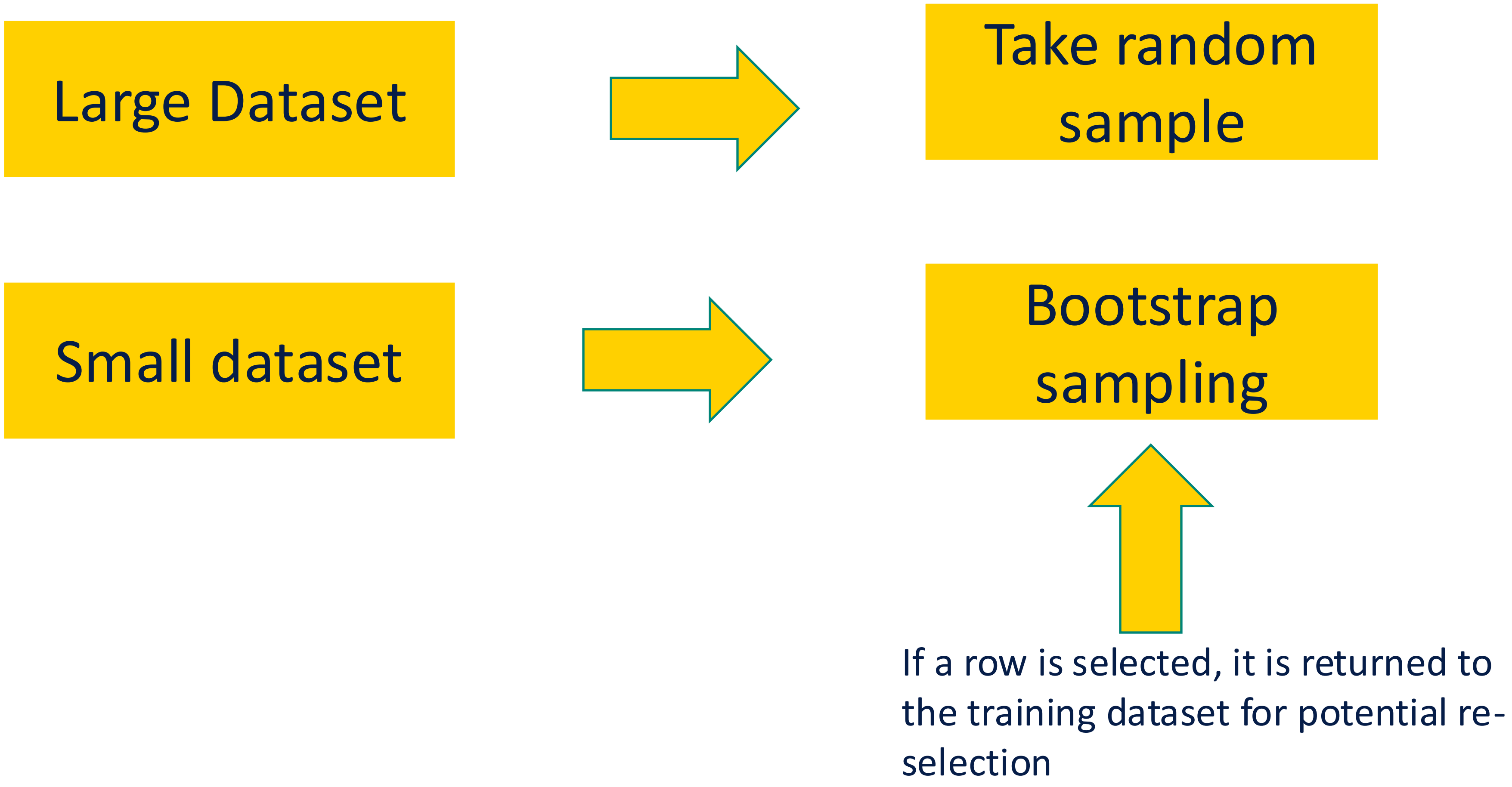
```
Read the file
if SizeOfInitializedData == 0 then
    return malware
else if UnknowSectionName then
    return malware
else if (DLLCharacteristics == 0
        and MajorImageVersion == 0
        and CheckSum == 0) then
    return malware
else
    return benign
end if
```

Source: PE-Header-Based Malware Study and Detection

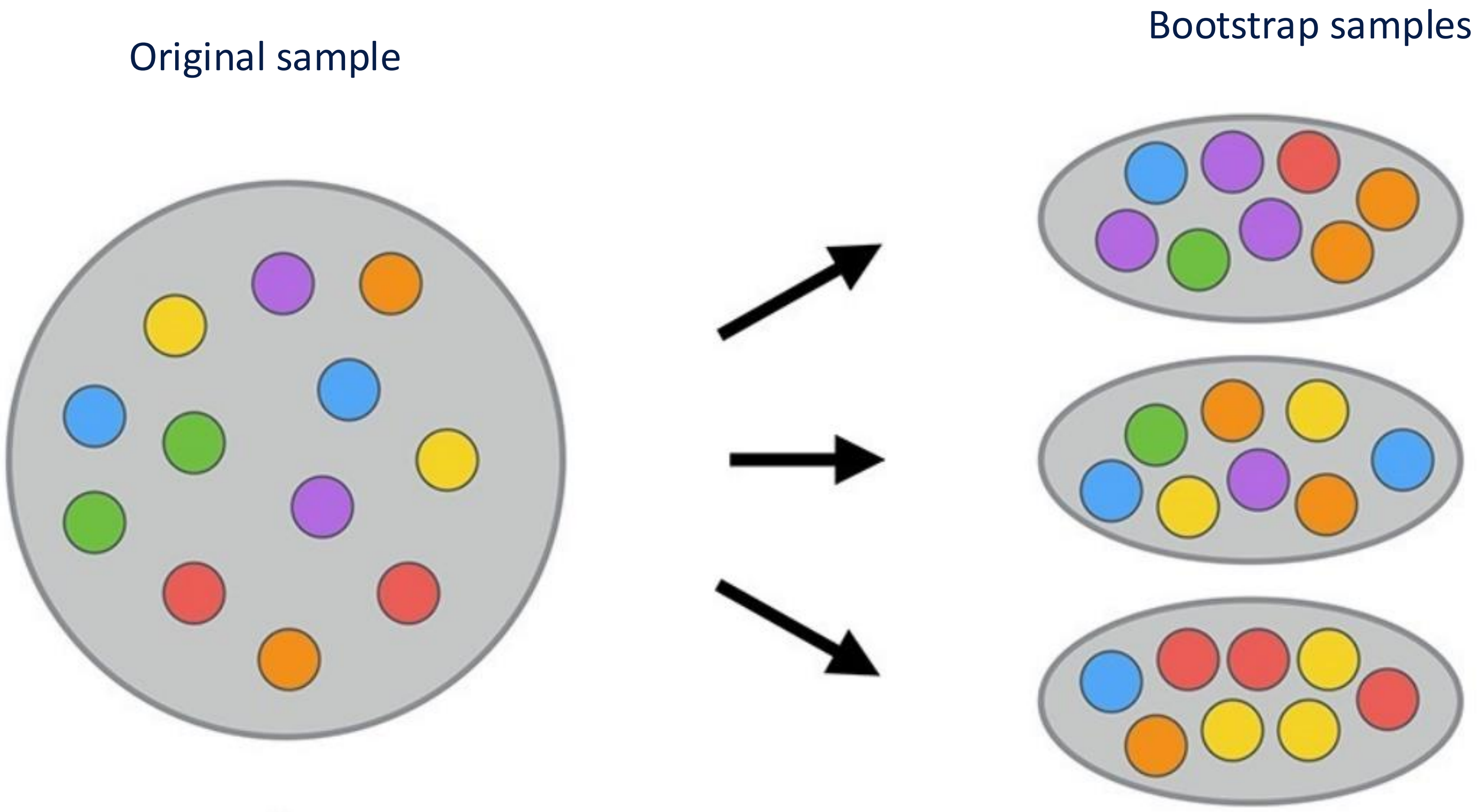
K-Means



Source: [Oxford Protein Informatics Group]

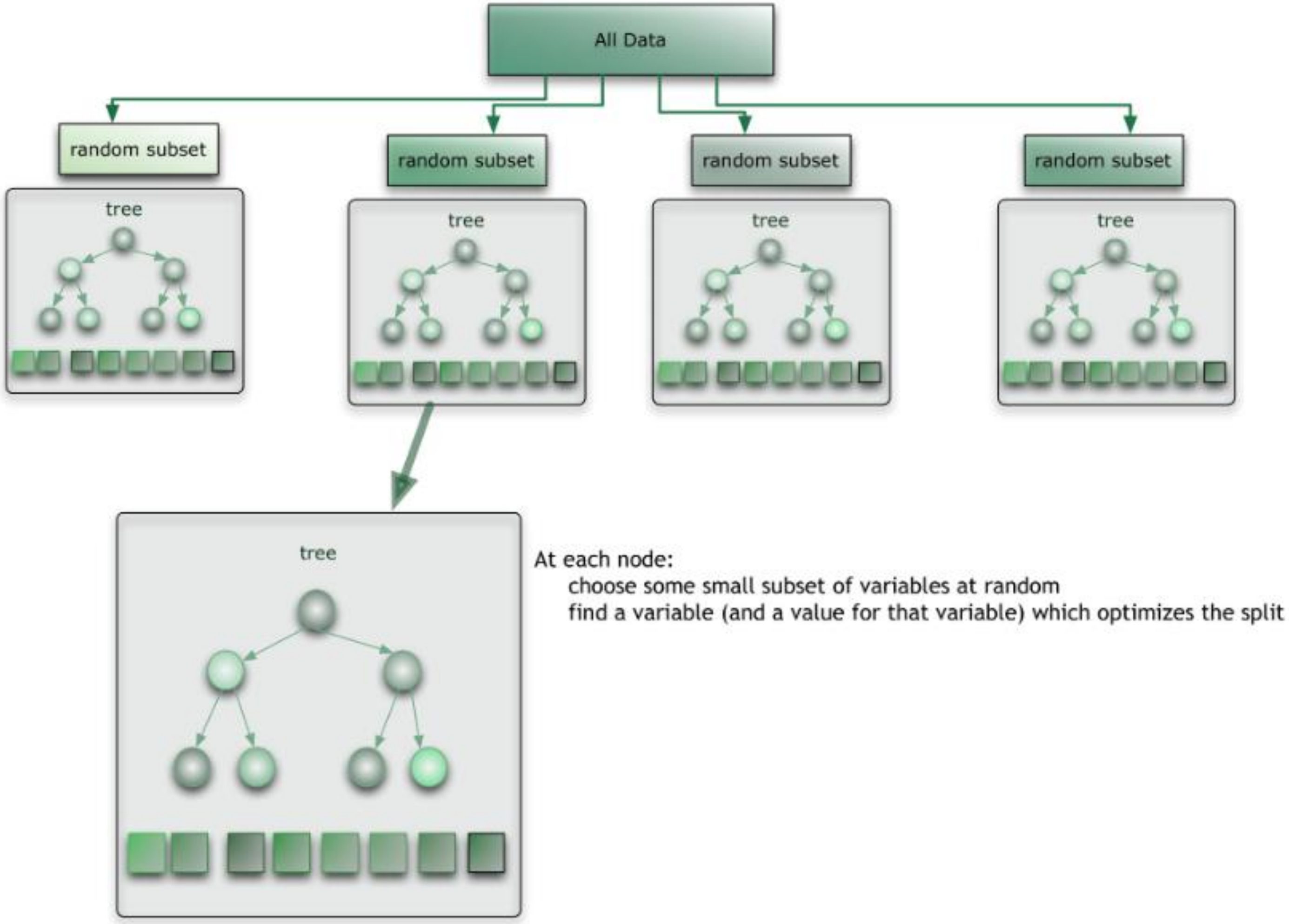

Large Dataset
Take random sample
Small dataset
Bootstrap sampling
If a row is selected, it is returned to the training dataset for potential re-selection

Bootstrap Sampling



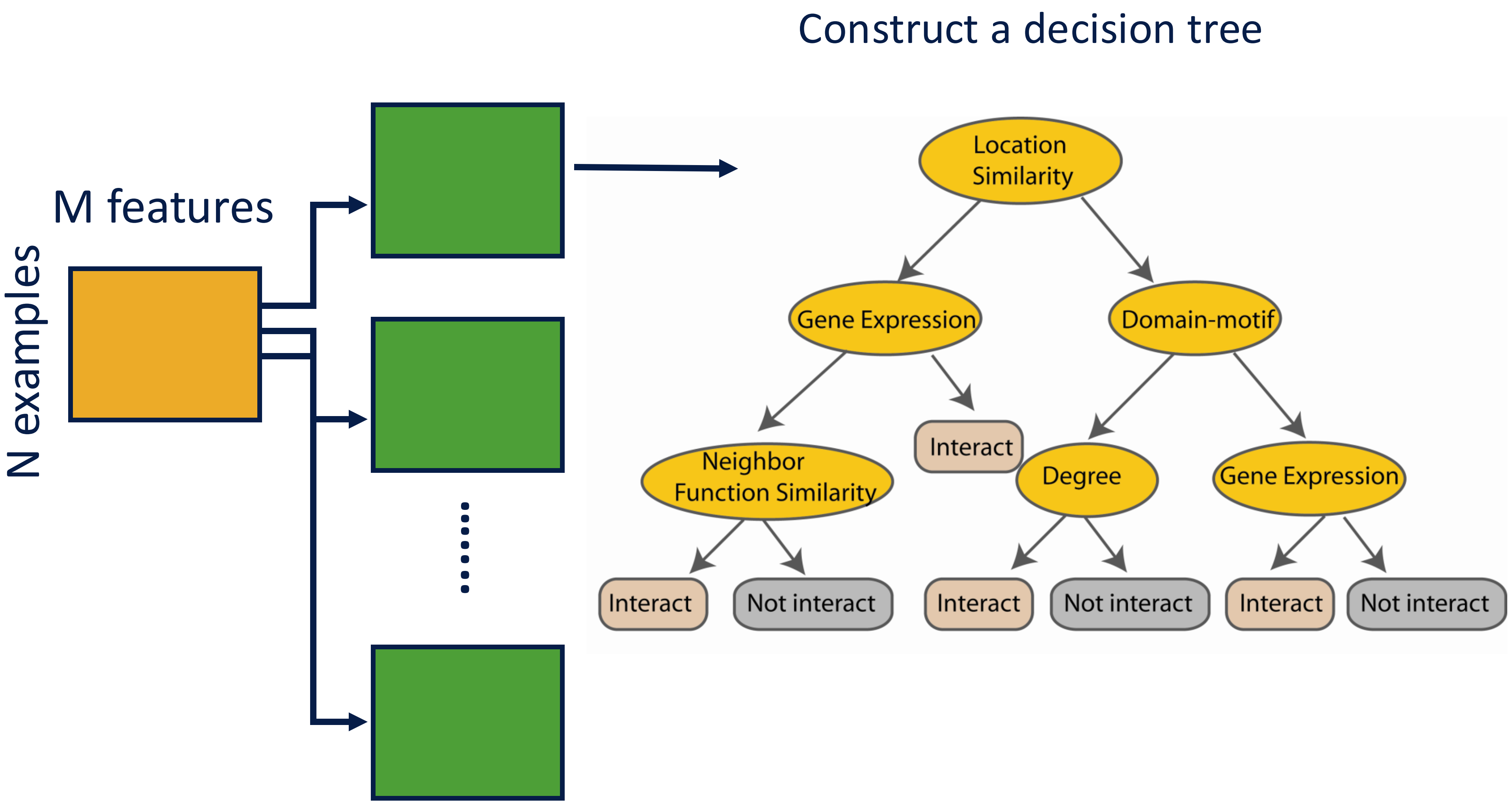
https://medium.datadriveninvestor.com/ensemble-techniques-bagging-bootstrap-aggregating-c7a7e26bdc13

Random Forest



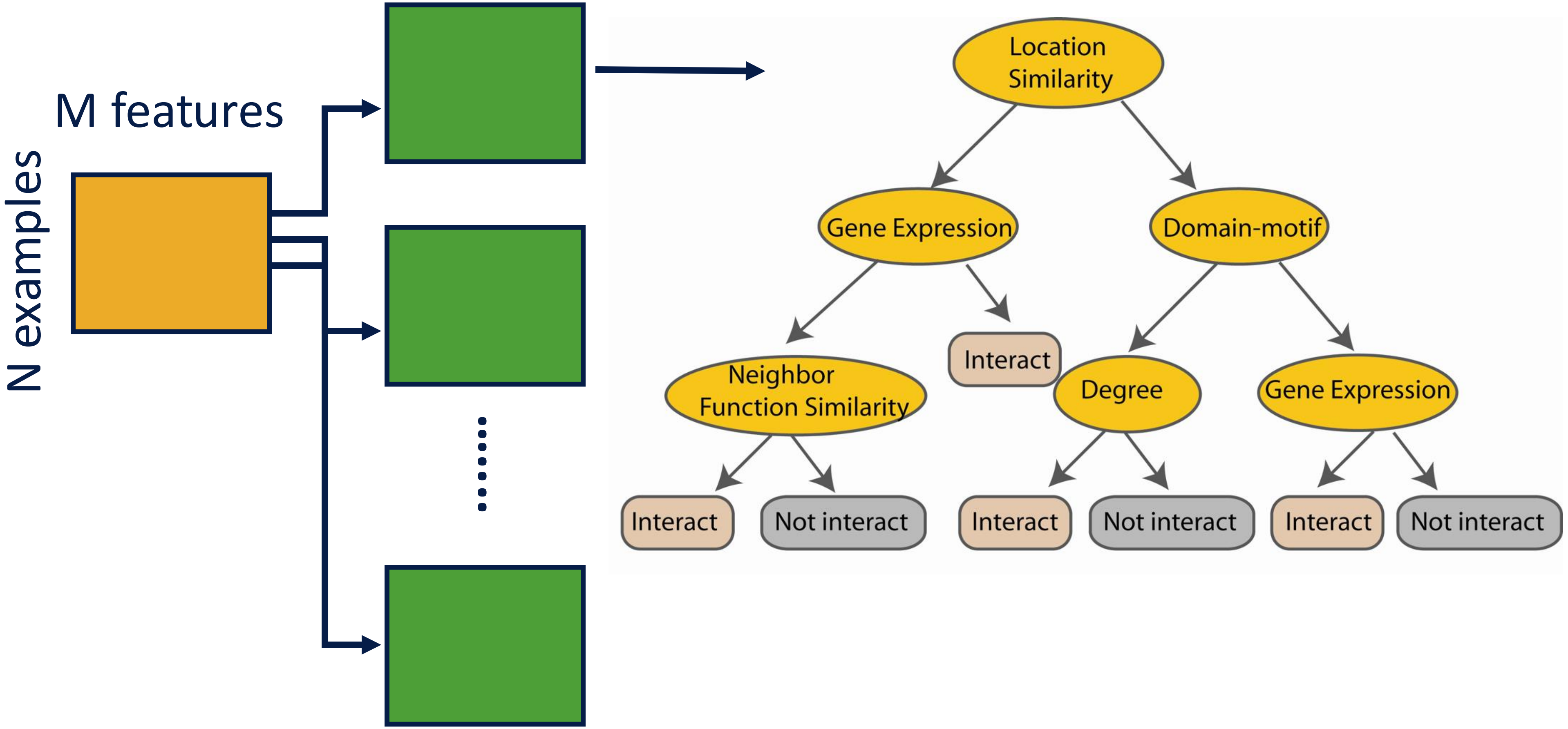
https://blog.citizennet.com/blog/2012/11/10/random-forests-ensembles-and-performance-metrics

Random Forest Classifier



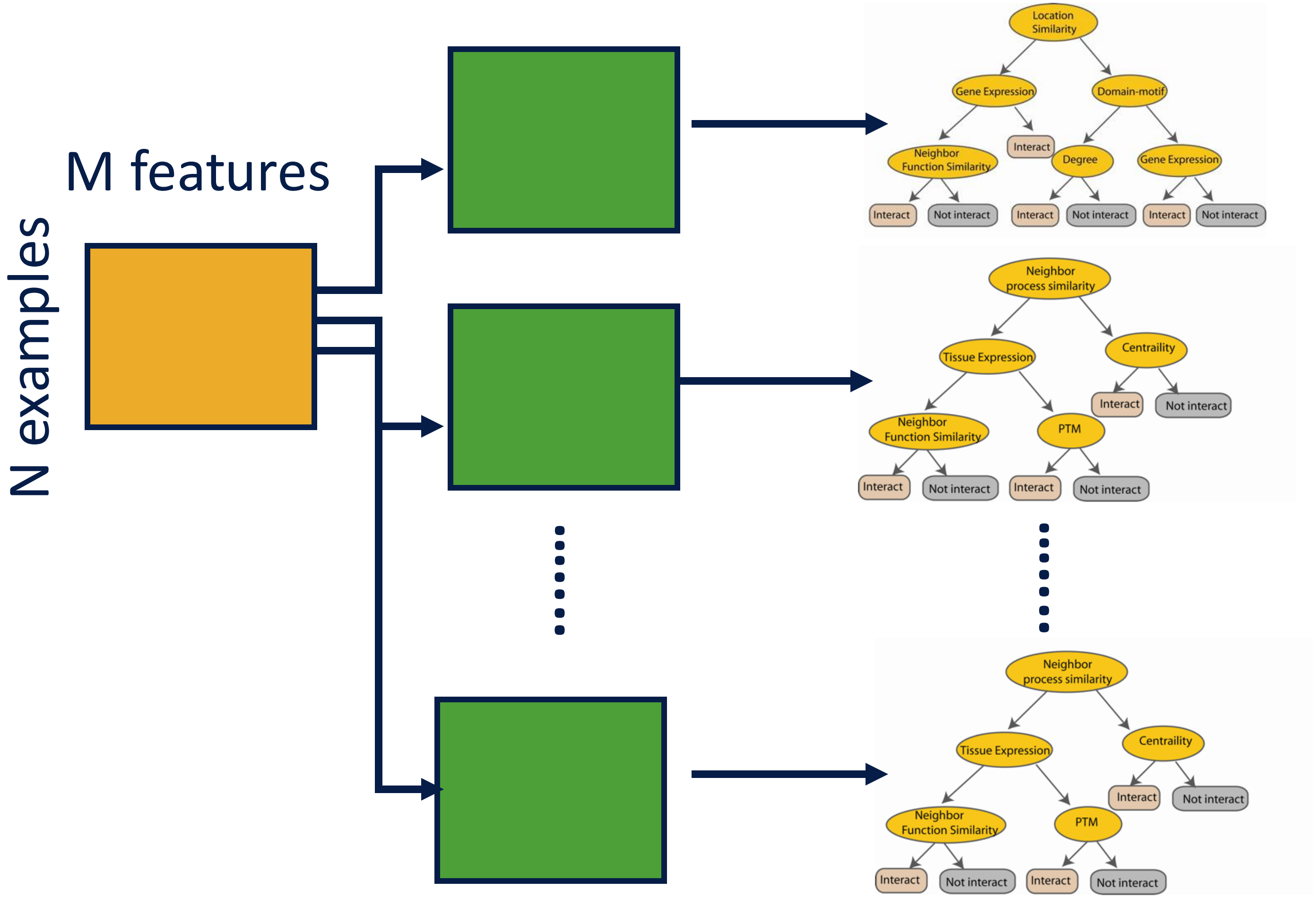
Random Forest Classifier

At each node in choosing the split feature choose only among *m*<*M* features

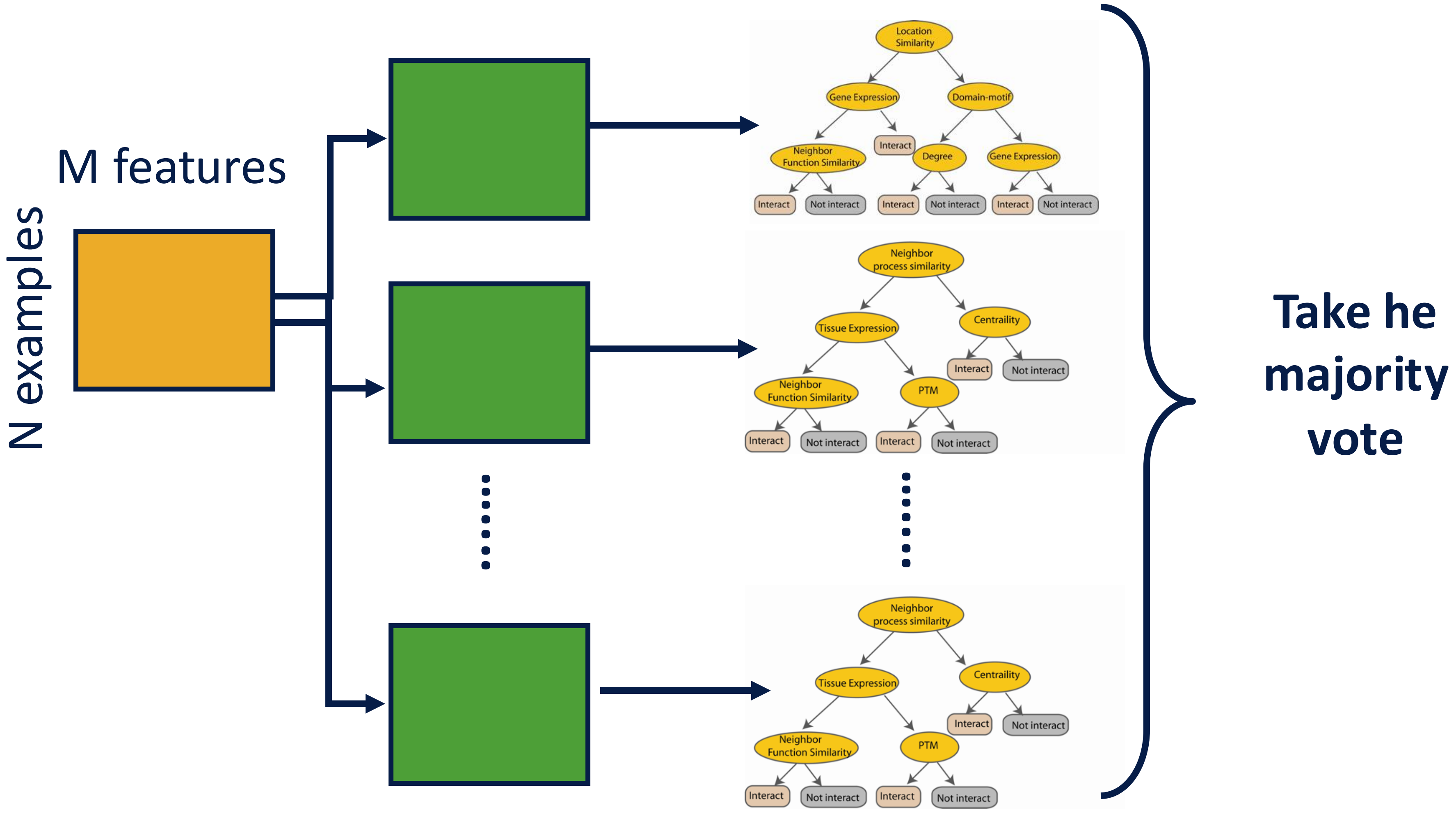


Random Forest Classifier

Create decision tree from each bootstrap sample



Random Forest Classifier



Random Forest - Sklearn

sklearn.ensemble.RandomForestClassifier

class `sklearn.ensemble.RandomForestClassifier`(*n_estimators=100, *, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='sqrt', max_leaf_nodes=None, min_impurity_decrease=0.0, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None*) [source]

A random forest classifier.

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the `max_samples` parameter if `bootstrap=True` (default), otherwise the whole dataset is used to build each tree.

Read more in the User Guide.

Parameters::

n_estimators : *int, default=100*

The number of trees in the forest.

Changed in version 0.22: The default value of `n_estimators` changed from 10 to 100 in 0.22.

criterion : *{"gini", "entropy", "log_loss"}, default="gini"*

The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "log_loss" and "entropy" both for the Shannon information gain, see Mathematical formulation. Note: This parameter is tree-specific.

max_depth : *int, default=None*

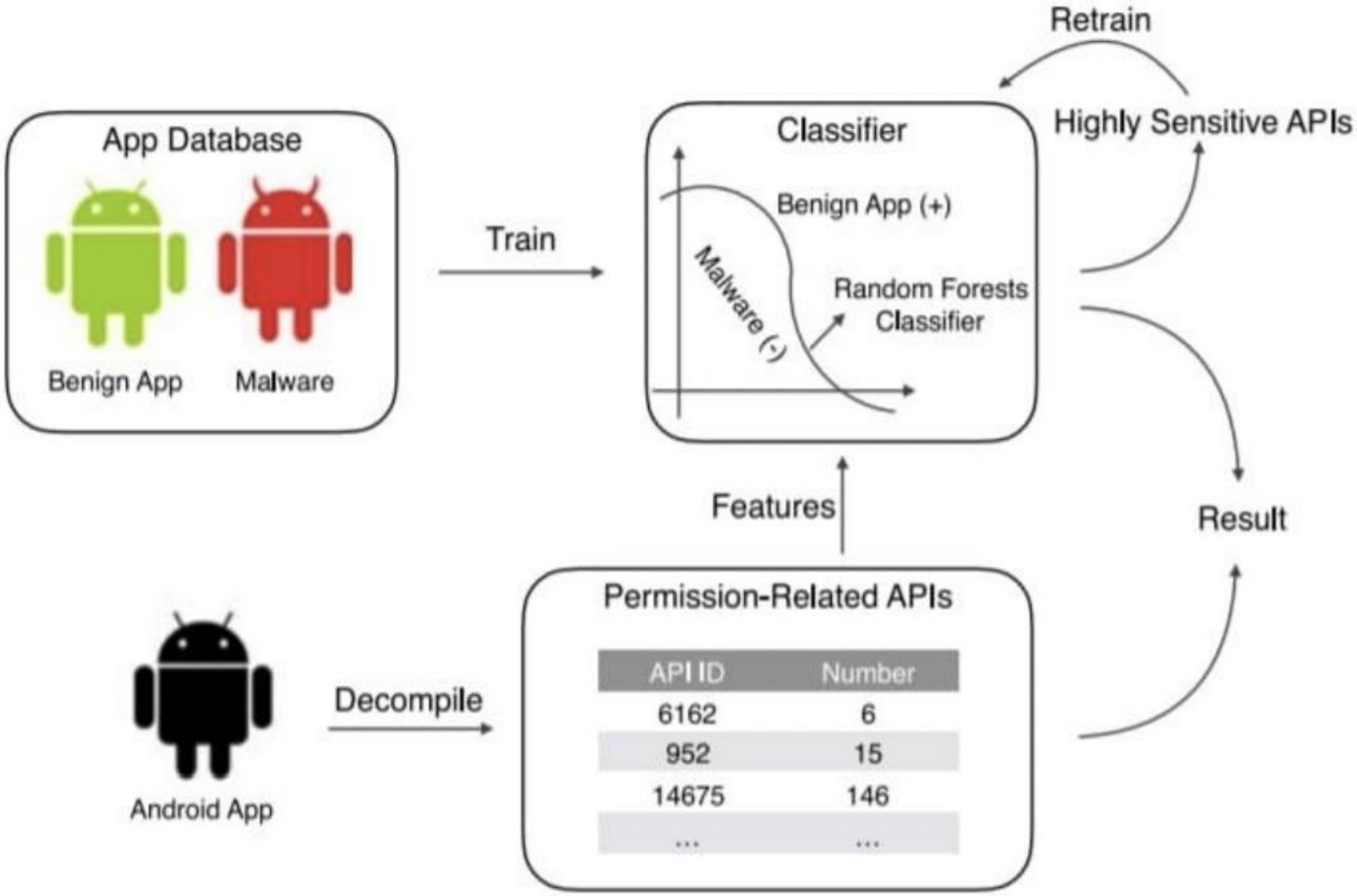
The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

min_samples_split : *int or float, default=2*

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

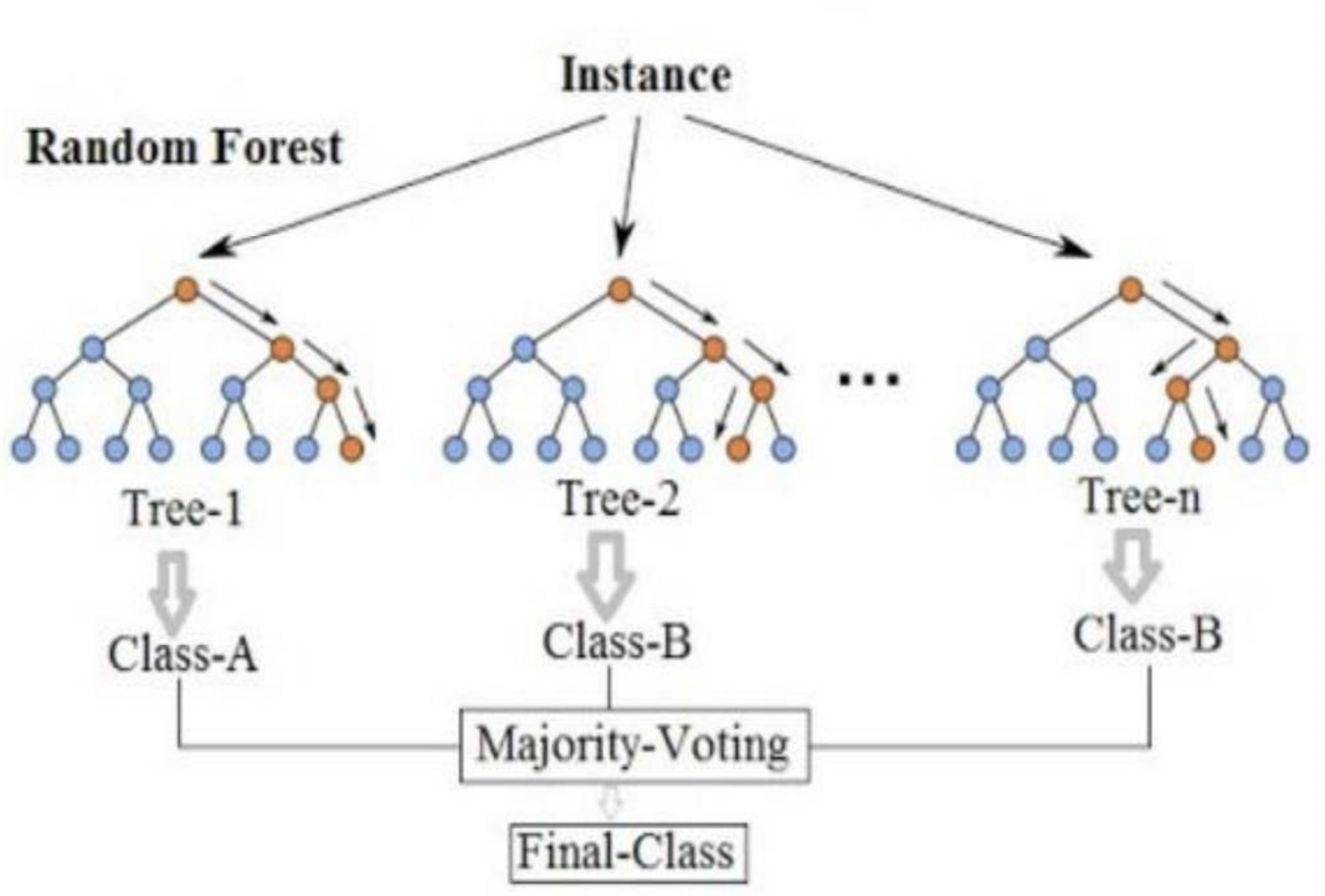
- If int, then consider `min_samples_split` as the minimum number.
- If float, then `min_samples_split` is a fraction and `ceil(min_samples_split * n_samples)` are the minimum

Example: Dynamic Malware Detection using Random Forest Algorithm (for Android)



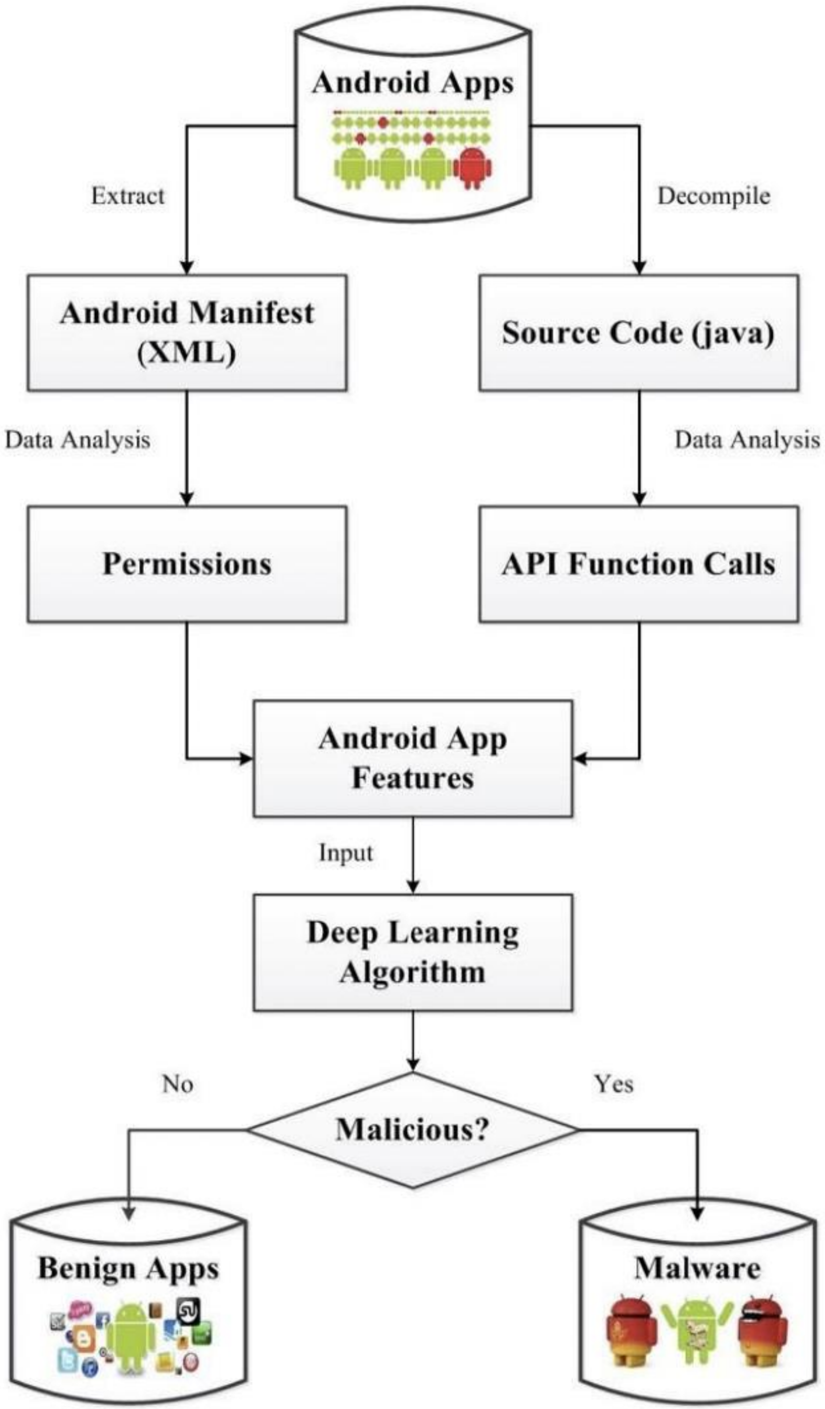
Source: [Maldroid: Dynamic Malware Detection using Random Forest Algorithm]

Example: Dynamic Malware Detection using Random Forest Algorithm (for Android)



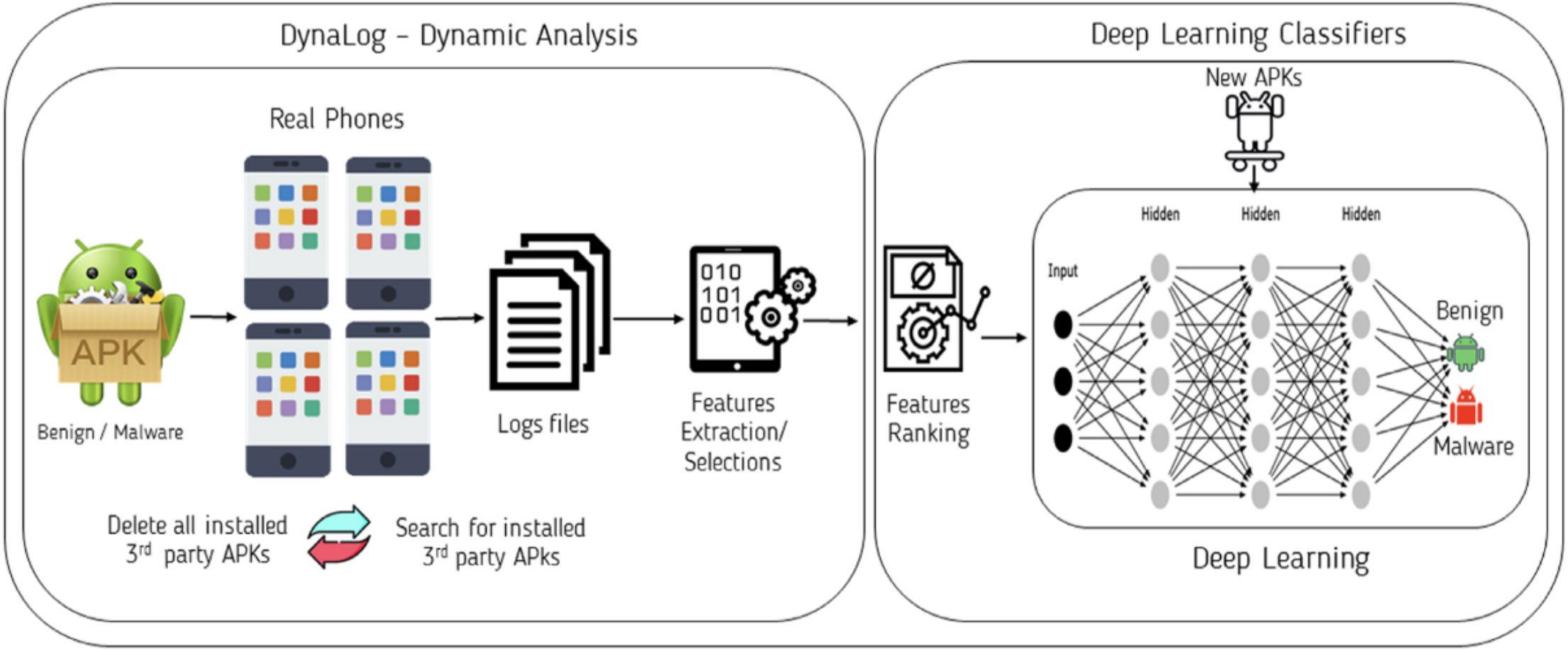
Source: [Maldroid: Dynamic Malware Detection using Random Forest Algorithm]

Example: Dynamic Malware Detection



Source: [Maldroid: Dynamic Malware Detection using Random Forest Algorithm]()

Malware Detection using Deep Learning (example)



Source: DL-Droid: Deep learning based android malware detection using real devices