



Anomaly Detection Challenge

CHALLENGE 4:

Malware Anomaly Detection w PEINFO

Team: Abracadata

Ishmeet Kaur(03677735)

Mustika Rizki Fitriyanti(03667399)

A decorative network diagram in the top-left corner, featuring a cluster of interconnected nodes. Some nodes are represented by concentric circles, while others are simple dots. The connections are thin grey lines.

1.

Introduction about the Data

1. Introduction about the Data

Aim: classify the binaries as malicious or benign in the test set using the features derived from the training set.

Problem Type: Binary Classification

Dataset

- Training Set: 9764
- Test Set: 46784



2.

Data Preprocessing and Data Analysis



2.1

Data Preprocessing

2.1 Data Preprocessing

```
JSON
├── label : "benign"
├── results
│   ├── sha256 : "001bd69f3e309ed655400f8145125a79ca0eb93479ac4f7f53ee45f9617c0a06"
│   └── peinfo
│       ├── imphash : "4a128e034b29626b284abefbf44f9089"
│       ├── exports
│       ├── pe_sections
│       ├── imports
│       └── rich_header
│           ├── sha256 : "4947798090ff53af2546e2606c4fad310bfa4480fbb3c6ae15540ed84710ba3e"
│           ├── values_raw
│           │   └── checksum : 173844575
│           ├── values_parsed
│           ├── pehash : "510c88d075f79fb337c3d3d77d7963b2bf1f0e78"
│           ├── version_info
│           ├── debug
│           │   └── 0
│           │       ├── MajorVersion : 0
│           │       ├── subtype : "pe_debug"
│           │       ├── DebugPath : "LXAA4_iesc.pdb"
│           │       ├── PointerToRawData : "0x7df8"
│           │       ├── SizeOfData : 39
│           │       ├── Type : 2
│           │       ├── TimeDateString : "2013-04-25 20:23:21"
│           │       ├── DebugAge : 1
│           │       ├── MinorVersion : 0
│           │       ├── result : "LXAA4_iesc.pdb"
│           │       ├── DebugSig : "RSDS"
│           │       ├── DebugGUID : "131cbebdcab7174990aa86f598d7db3d"
│           │       └── TimeDateStamp : 1366921401
│           ├── version_var
│           └── timestamp
```



2.2

Handling Categorical Data

2.2 Methods to handle categorical data

- **One Hot Encoding:**

Different sections of the json binaries were treated as categorical variables rich_header, the categorical variables were made as different features to train the dataset.

- **Limited One hot Encoding :**

As the entire one hot encoding of the features took an extensive memory and runtime, the feature set containing value counts greater than 5 were encoded .Features upon which OHE was applied:

- ★ Rich Header
- ★ Thread local Storage
- ★ PE Section
- ★ Debug
- ★ Version_var

- **Tf-Idf (N grams):**

N gram analysis(using one gram) increased the accuracy by 0.4 as compared to one hot encoding of the features .Tf- Idf Vectorizer was applied to the following features:

- Imports(Function and dll)
- Exports
- Error
- version -info(key,value)

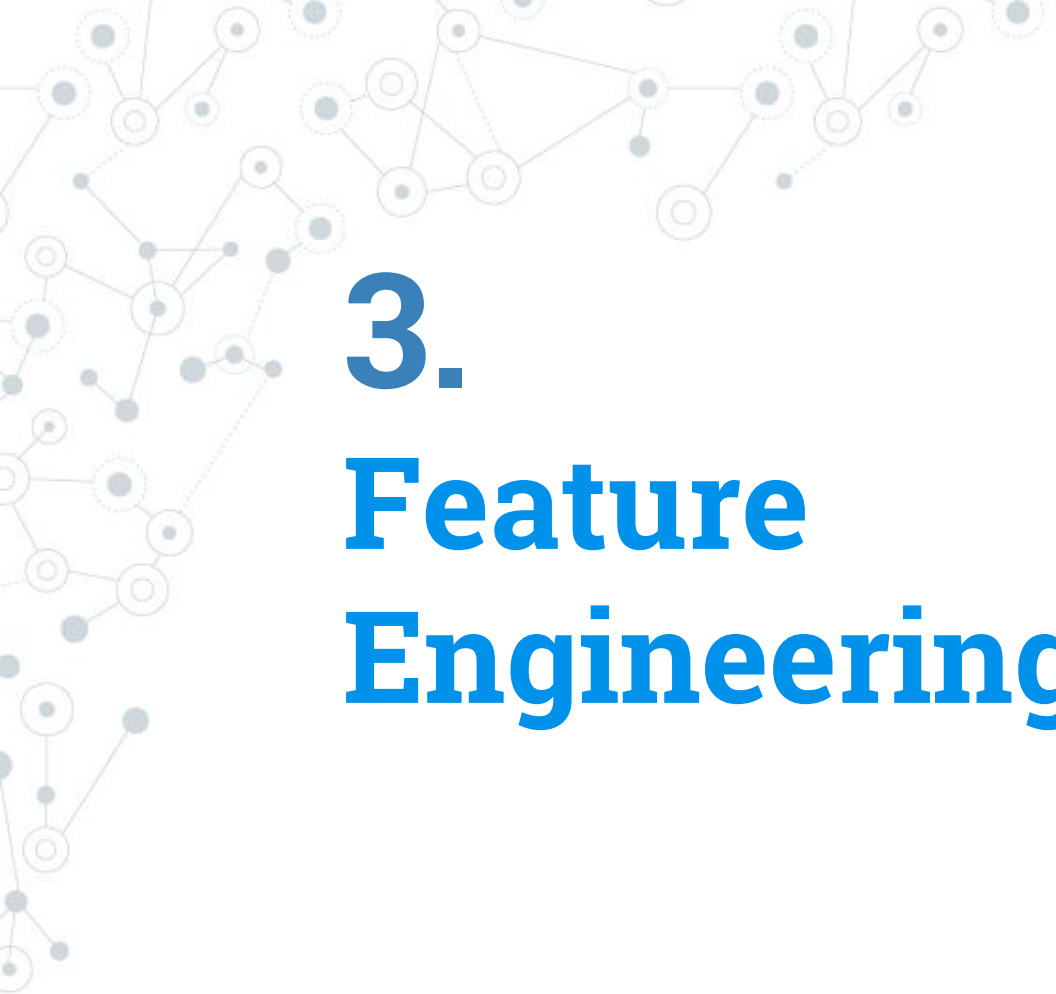
- About One Hot Encoding

One hot encoding transforms categorical features to a format that works better with classification and regression algorithms.

Sample	Category	Numerical
1	Human	1
2	Human	1
3	Penguin	2
4	Octopus	3
5	Alien	4
6	Octopus	3
7	Alien	4



Sample	Human	Penguin	Octopus	Alien
1	1	0	0	0
2	1	0	0	0
3	0	1	0	0
4	0	0	1	0
5	0	0	0	1
6	0	0	1	0
7	0	0	0	1

A decorative network diagram in the top-left corner, featuring a complex web of interconnected nodes and lines. The nodes are represented by circles of varying sizes, some with concentric rings, and the lines are thin and grey. The overall structure is organic and branching, resembling a molecular or biological network.

3.

Feature Engineering

3.1 Feature Engineering

- **Dimensionality Reduction by PCA :**
Out of 56000 features, selecting 100 features by iterating 5 times gave an accuracy of 0.9688 using PCA.
- **Dimensionality Reduction Univariate Selection:**
Selection of 56000 features , selecting 500 features using a simple method of univariate selection (using chi square test) resulted in an accuracy of 0.967

As the feature selection methods did not increase the accuracy considerably, all features were used to train the model in a sparse matrix format.



4.

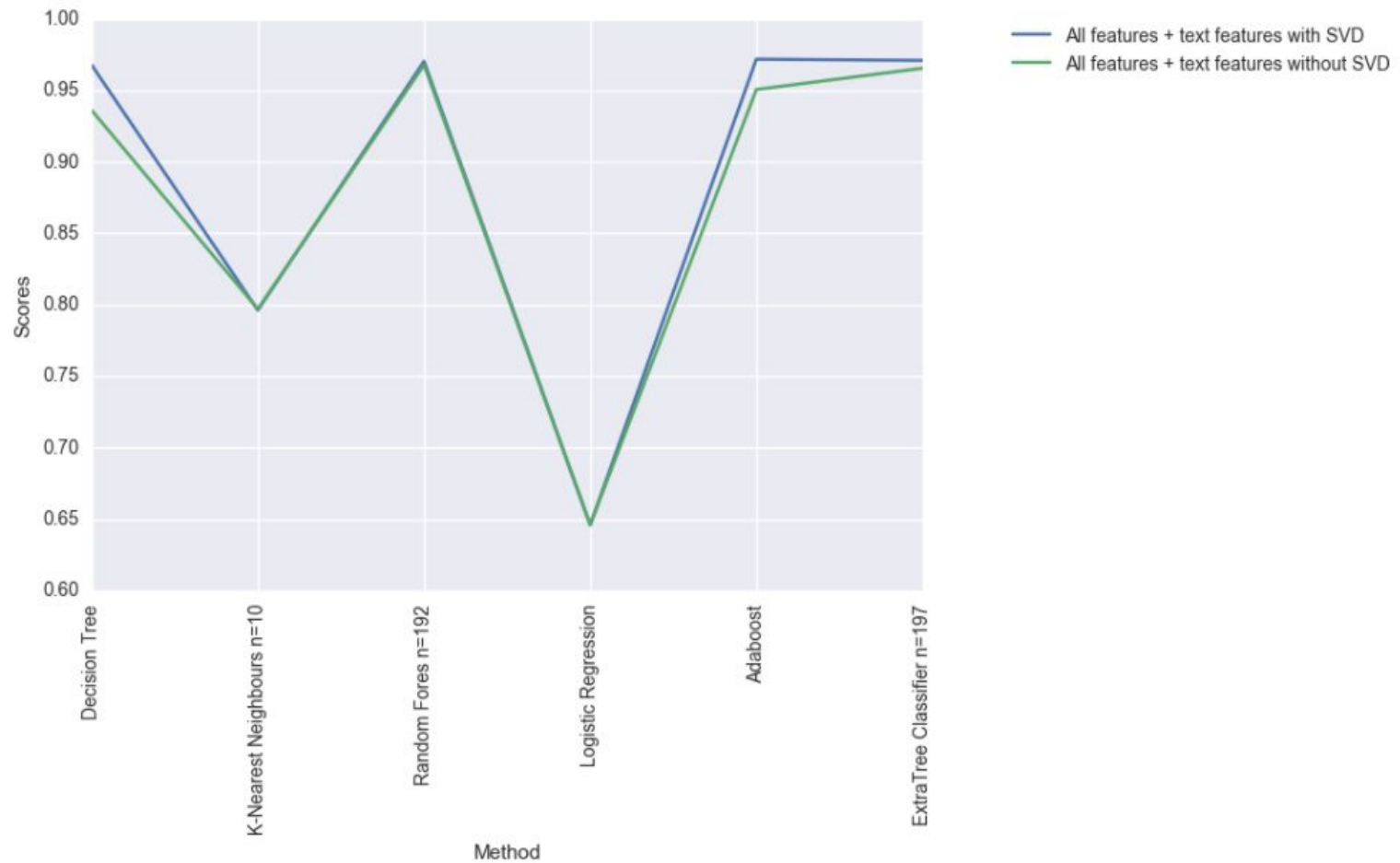
Creation and Evaluation of the Models

- Decision Tree Classifier
- K- nearest Neighbors
- Random Forest
- Logistic Regression
- AdaboostClassifier
- Extra Trees Classifier

Implement the Classifiers, Enhancing Accuracy and Tuning Parameters

- Tuning the parameters of the Random Forest Algorithm.
Using OOB(Out of Bag) Error Rate, we make a list number of maximum estimators, then we process in Random Forest Classifier with cross validation in order to make it comparable to other algorithms
- Implement the classifier using several different combinations of features along with Truncated SVD(which worked with sparse matrices efficiently)
- Using all the features (with and without SVD) gave comparatively better results both in the training and the test set

4.3 Accuracy Score



4.2 Kaggle Score

Method	Public Score	Private Score
All features+text with svd	0.97144	0.97084
All features+text without svd	0.97884	0.97820

A decorative network diagram in the top-left corner, featuring a complex web of interconnected nodes and lines. The nodes are represented by circles of varying sizes, some with concentric rings, and the lines are thin and grey. The diagram is partially cut off by the top and left edges of the slide.

5. **Learning**

5. Learning

- ❖ Learning to interpret the structure of JSON Data.
- ❖ Working with sparse data format and algorithms.
- ❖ One hot encoding is really good in transforming categorical features to a format that works better with classification and regression algorithms, however it can causing memory errors for analyzing large number of features.
- ❖ Selecting features is really important to prevent overfitting and increase the accuracy

A decorative network diagram in the top-left corner, consisting of various sized circles (nodes) connected by thin lines (edges). Some nodes are solid grey, while others are hollow with a grey outline. The network is dense and irregular.

**Thank you for
your attention**