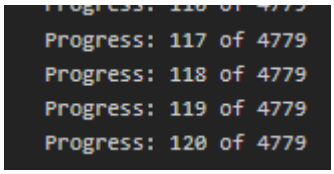
A Quantitative Study of Accuracy in System Call-Based Malware Detection

By Shankara Narayanan

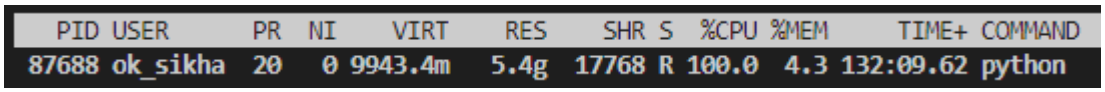
Problem with too much complexity

1. In our dataset, there are 83 unique system calls
2. There are 80 system calls that are associated mostly with malware than benign. So, let's call them "malware system calls"
3. For a 2-gram, total number of possible signatures are 80 * 80 = 6400
4. There is a total of 4779 hash values and their associated execution traces of varying length
5. Now, run each of the hash with all 6400 signatures to see how many of them are matching is the way we count the total matching signatures
6. Based on the number of matched signatures for each of the hash values, we will be able to decide on an appropriate alert threshold for a 2-gram model

The major impediment here is this take over **2 hours**



Even after running it on a high-performance computing facility from my college for



This impinges our approach of both generating the signatures and matching the signatures with the execution traces

Possible fixes

1. Pruning the number of signatures
2. Normalizing the execution traces to a fixed length / dropping hash values with extremely long execution traces

Applied fix: Normalizing the execution traces length

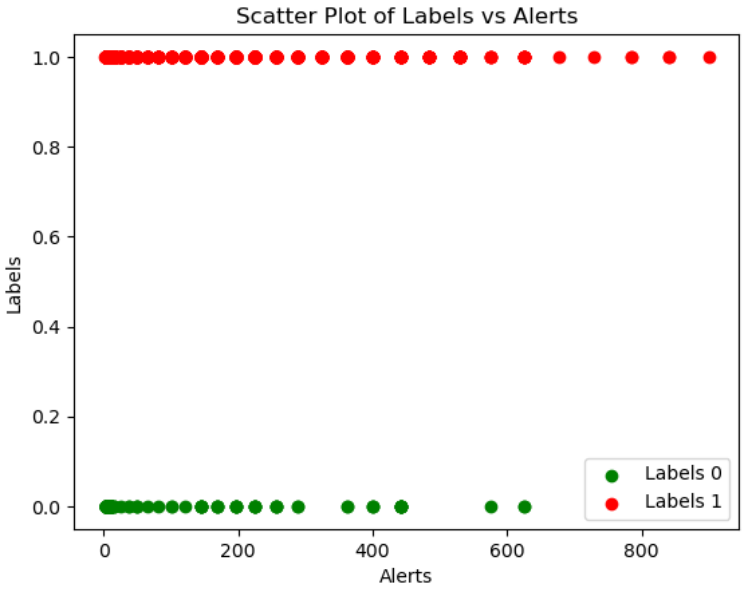
Originally the dataset contained 4779 hash values and their corresponding execution traces. Sorting the dataset based on the length of the execution traces revealed the existence of few outlier length execution traces (millions of calls) which made the training process way too slow. Removing the top 1200 hash values and using them for testing and using the retained 3579 hash values was done.

2-bag implementation

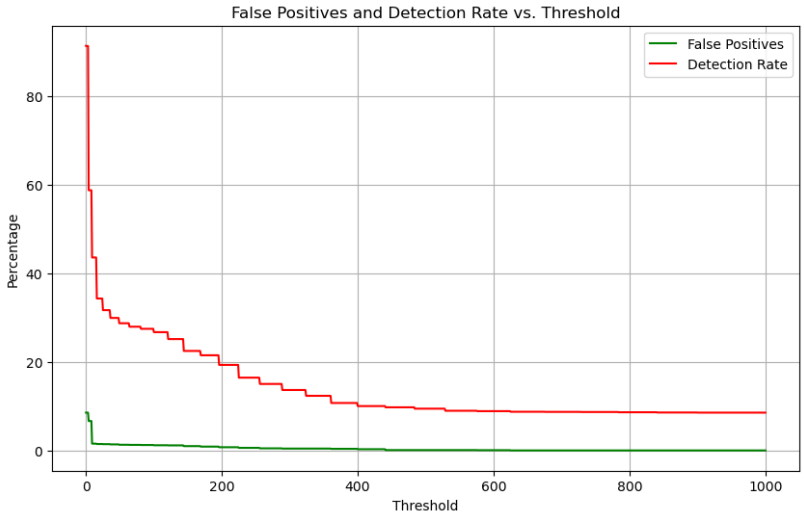
Without pruning, the signatures for 2 bags are combinations of 2 system calls => 80 * 80 = 6400. The matching of signatures and execution traces were done for 2 – bags and stored in the data frame as shown below

	hash	exec_trace	label	length	alerts
2862	02971EC76721DED496CC3D29B39DBADD.csv	['Process Start', 'Thread Create', 'QueryNorma...	1	13767	576
985	ac44e4ce372fa7cb8d72bb5b9e78db73.csv	['Thread Exit', 'Process Start', 'Thread Creat...	1	13765	576
1584	936319E934BCF1275D54FFFEDE2F5AF2.csv	['Thread Exit', 'ReadFile', 'ReadFile', 'Query...	1	13737	400
1414	55FF207D7C2824E86C8E3CE3C1E443B1.csv	['Thread Exit', 'ReadFile', 'QueryBasicInforma...	1	13734	400
1405	0F6DCFA0C985E03DEE906E6106B14625.csv	['QueryNormalizedNameInformationFile', 'QueryB...	1	13728	529
...	...	...	...	...	...
1532	5230284F982C86B94B559E082AE97AD0.csv	['Process Profiling', 'Thread Exit']	1	36	4
3644	b729751ba6d2e0a0d9bf52d74607ccd3.csv	['Process Profiling', 'Thread Exit']	1	36	4
4058	DD44F5D5E722D72622A261C0A586EEF0.csv	['ReadFile', 'Thread Exit']	1	27	4
188	WinRAR.csv	['ReadFile', 'Thread Exit']	0	27	4
208	SureCutsALot4.csv	['ReadFile', 'Thread Exit']	0	27	4

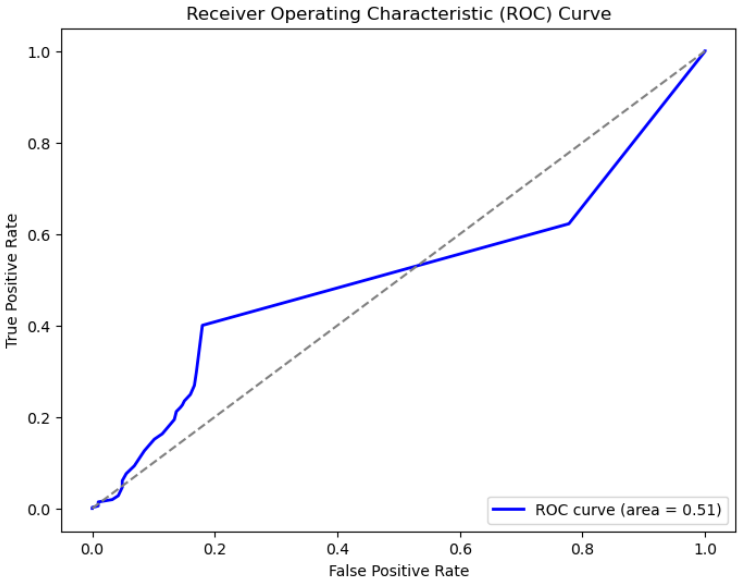
Now, analyzing the scatter of alerts vs the label of the hash value is as follows.



The scatter is clearly inseparable using visual analysis hence a variety of thresholds were chosen and the false positive rates along with the detection accuracy are plotted as a graph and shown below.



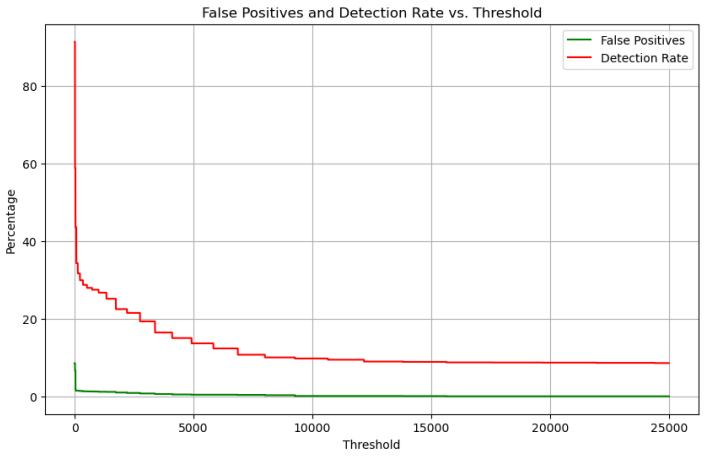
The trade-off incurred here is between the false positive rate and the detection accuracy. As expected, the accuracy is higher when the threshold is low and also the false positives are at an all-time high when the threshold is 0. Gradually as the threshold is increased, the detection accuracy plummets, and the false positive rates decrease too.



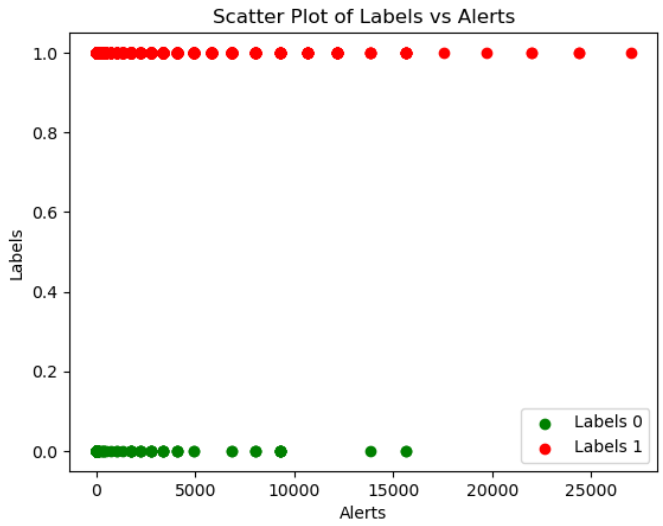
Also, the ROC curve was plotted which shows a poor performance of the model with only 51% of the area covered.

3 – Bag implementation

The number of signatures taken 3 at a time is 80 * 80 * 80 = 512,000. This shows how the number of signatures increases exponentially. Nevertheless, we carried out the signature matching without pruning to see how long the process would take. It took about 40 minutes to classify 3579 samples.



A Max threshold of 25000 alerts was captured hence the graph drawn till 25000 in the x–axis. Similar to the 2–bag's scatter plot, the scatter plot of 3-bag is also non-separable but some distinction is there on the basis of the number of alerts raised and the type of label.



Pruning technique

When we tried to implement 4 bags, the number of signatures generated was too high for even one pass through the longest execution trace (13000 system calls). Hence, pruning the number of signatures became an important step.

The pruning strategy is as follows,

1. Make a malware only data frame
2. Generate all possible signatures
3. Iterate through all possible signatures checking the 2 conditions
   a) Check if the signature covers at least 5 malware among all the malware
   b) Also, the covered malware should not be already covered by more than 20000 signatures
4. Only select such signatures and discard all other signatures

Applying pruning on the combination of 2 signatures and 3 signatures

Before pruning, total number of signatures = 80 * 80 = 6,400

After pruning, the total number of signatures = 1992

Pruning discarded **4,408** spurious signatures!

Optimizing this pruning technique with the help of lambda functions reduces the complexity from O(n^2) to near O(n). This leads to a massive computation boost from **~17 hours** to prune 512,000 signatures to meager **~51 minutes**.

After pruning, the total number of signatures = 55980

Pruning discarded **456,020** spurious signatures!

Now that our optimized pruning algorithm can prune 10000 signatures per minute,

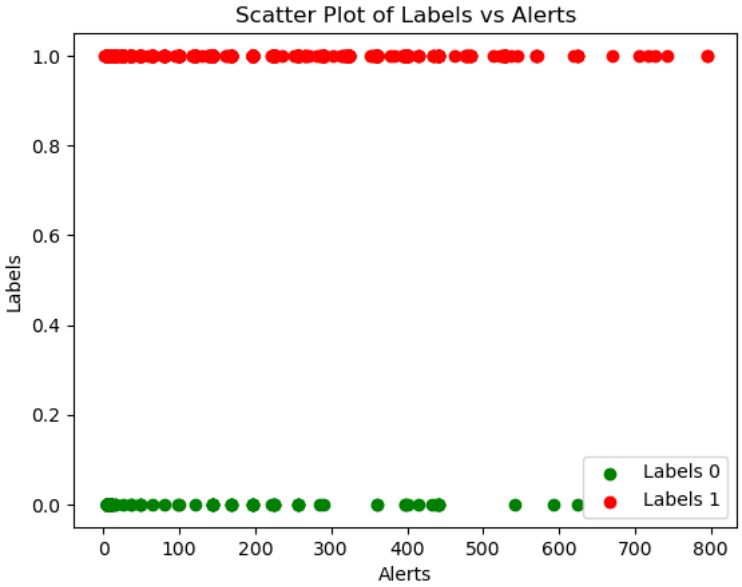
a) 2 combinations ➔ 80 * 80 = 6400 ➔ Pruned in **< 1 minute**
b) 3 combinations ➔ 80 * 80 * 80 = 512000 ➔ Pruned in ~ **51 minutes**
c) 4 combinations ➔ 80 * 80 * 80 * 80 = 40960000 ➔ Pruned in ~ **68 hours**

After 4 combinations, the pruning process becomes infeasible

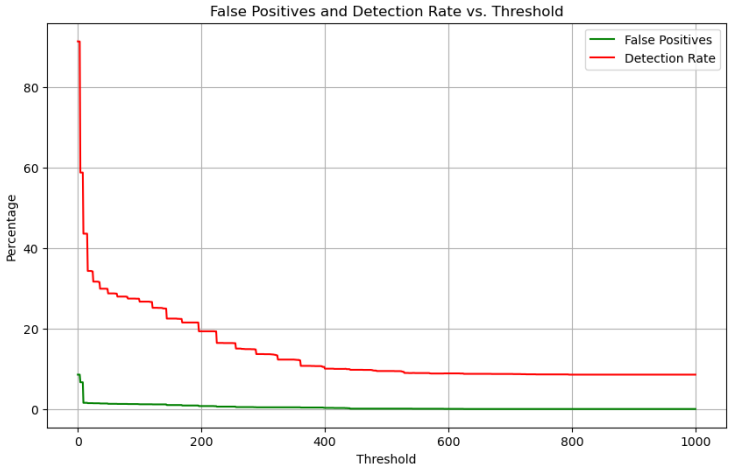
Bags implementation

2 – bag

After pruning 2 – signatures we have around 1992 signatures



From visual analysis, the distribution is still not easily distinguishable between 0 and 1 classes.

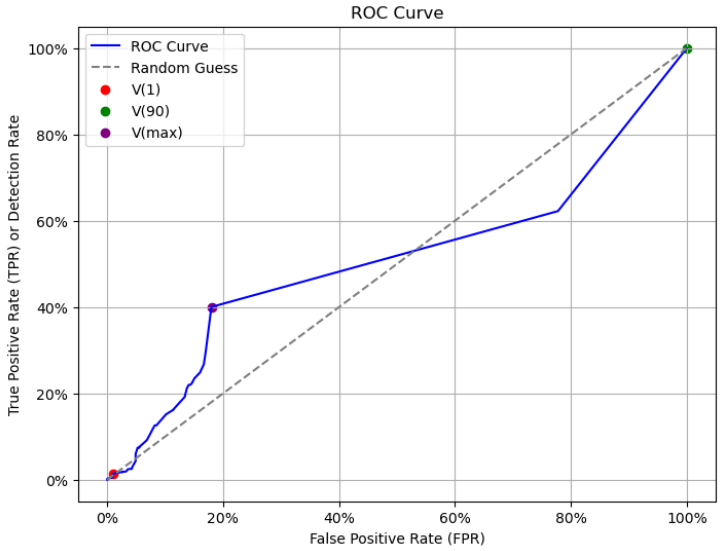


The maximum accuracy is around 95% with ~10% false positives. And as the threshold increases the accuracy decreases and false positive rates decrease too.

V1 ➔ is the detection rate at 1% False Positive Rate

V90 ➔ is the false positive rate at 90% Detection Rate

Vmax ➔ is the maximum area covered by the ROC curve



For a 2 – bag model,

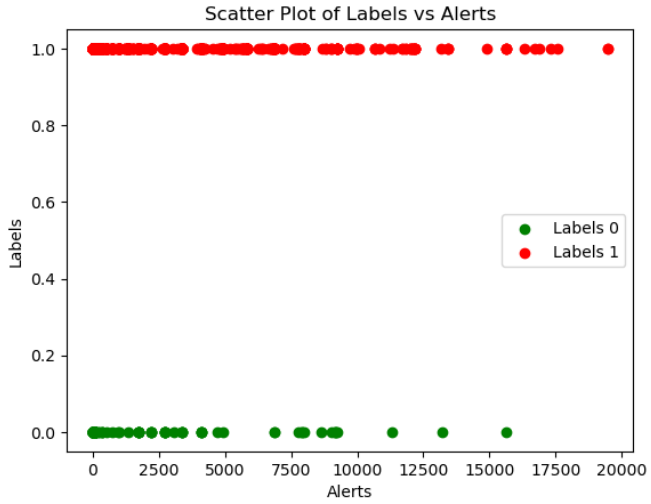
V1 = 1.45%

V90 = 100%

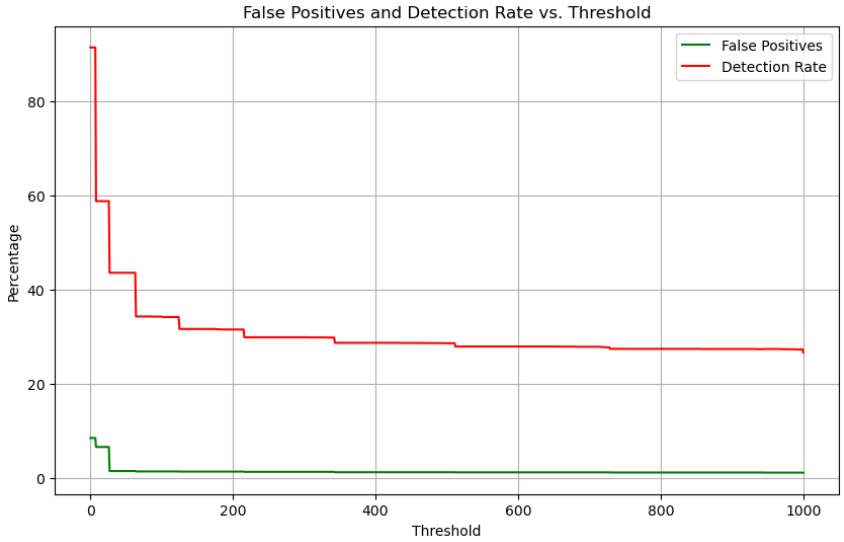
Vmax = 0.51

3 – bag

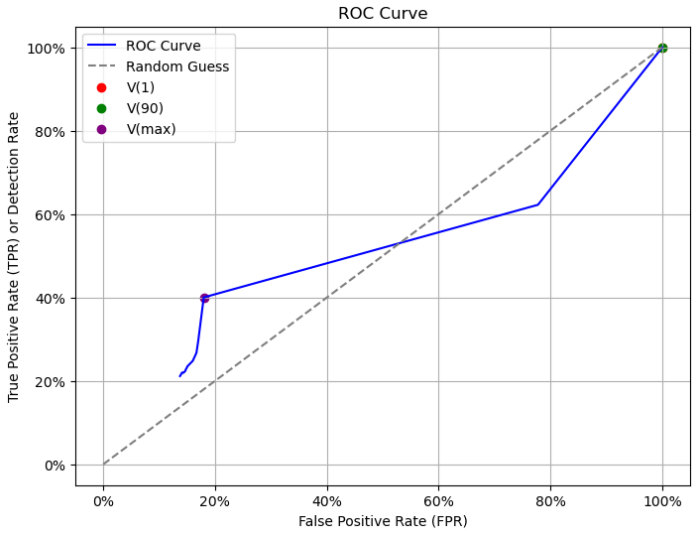
After pruning 3 signatures, there are **55980** are present



The classes are not separable but it is clear that **higher number of alerts are more likely to be malware.**



As the threshold increases, the detection rates drop sharply and so does the false positive rates.



For a 3–bag model

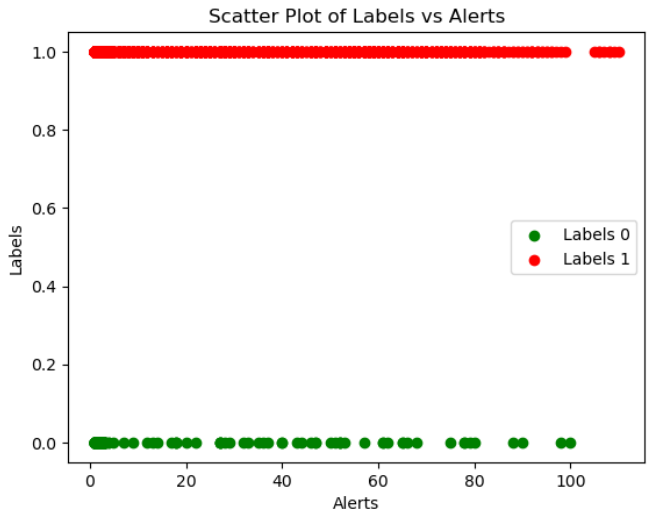
V1 = 100%

V90 = 100%

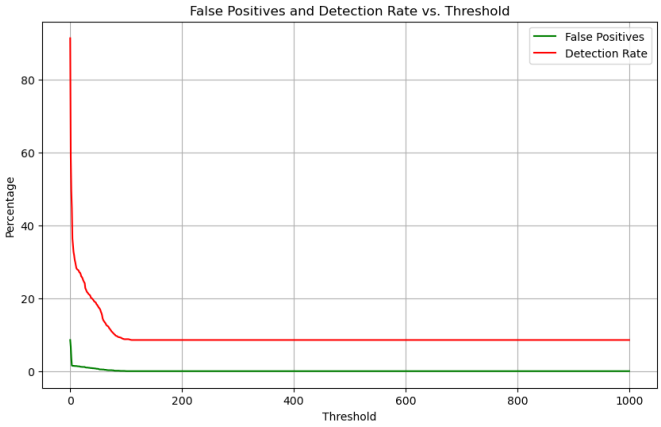
Vmax = 0.49

Grams implementation

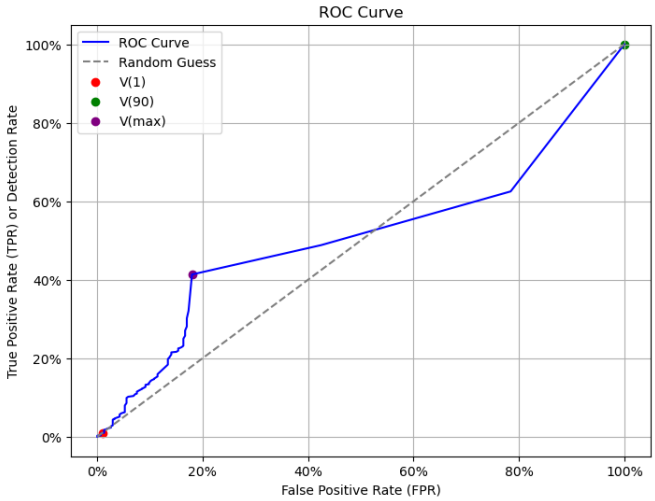
2 – gram



Nothing useful can be inferred from here as the number of alerts raised by the malware is spanning across the entire axis



It is clear that increasing the threshold decreases the detection rate but it is notable that the false positive rates drop immediately thus giving a **fair trade-off** between the detection rate and the threshold. The optimal threshold lies somewhere between **0 – 50 alerts.**



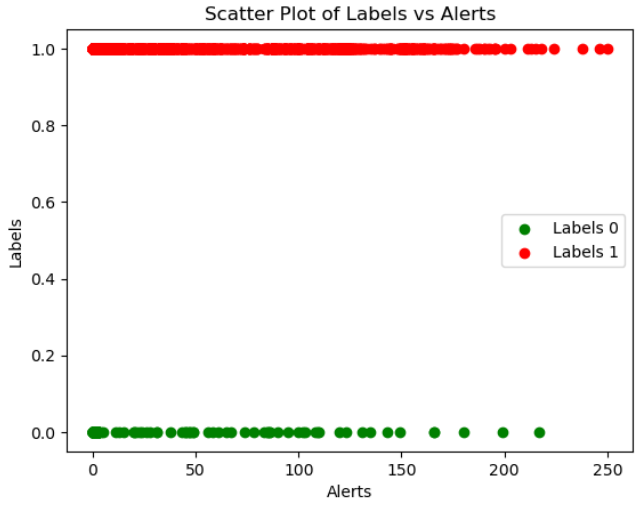
For 2–gram implementation

V1 = 0.91%

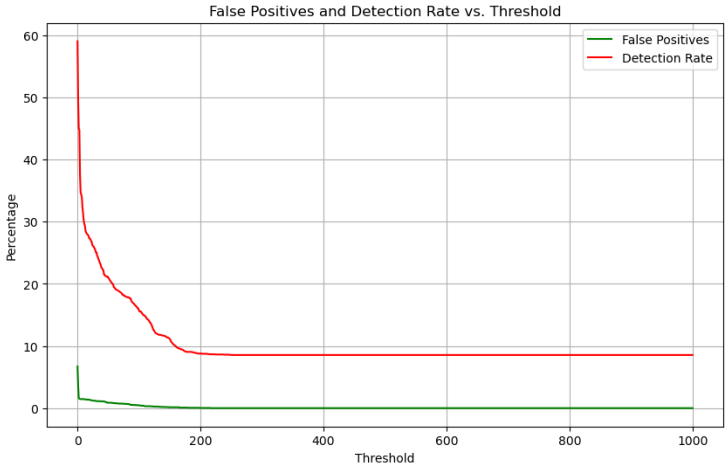
V90 = 100%

Vmax = 0.5%

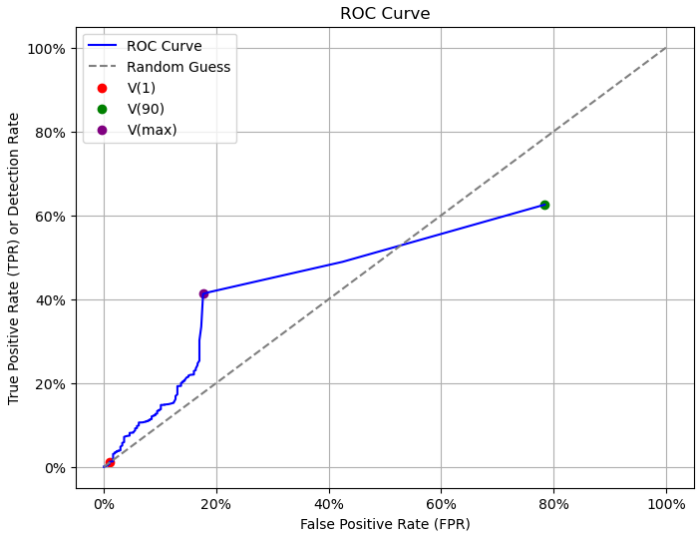
3-gram



The same trend as 3-bag, more alerts are correlated by more number of malware than benign samples.



Around ~**60% detection rate** for a significantly lesser false positive rate.



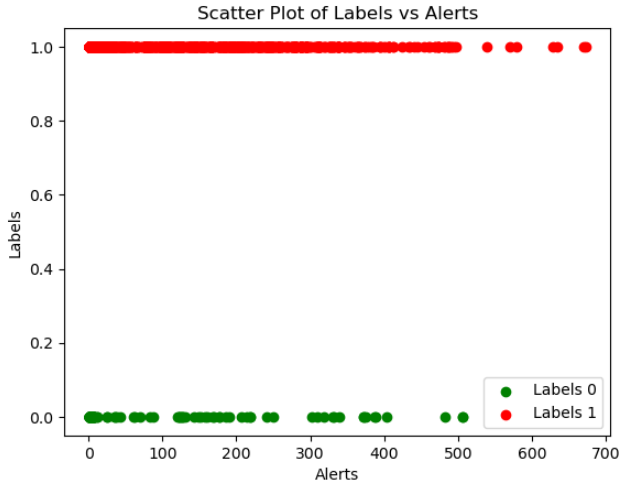
V1 = 1.19%

V90 =78.43%

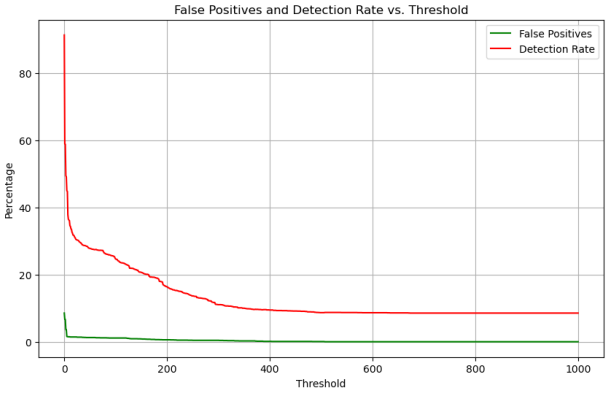
Vmax = 0.33

Tuples implementation

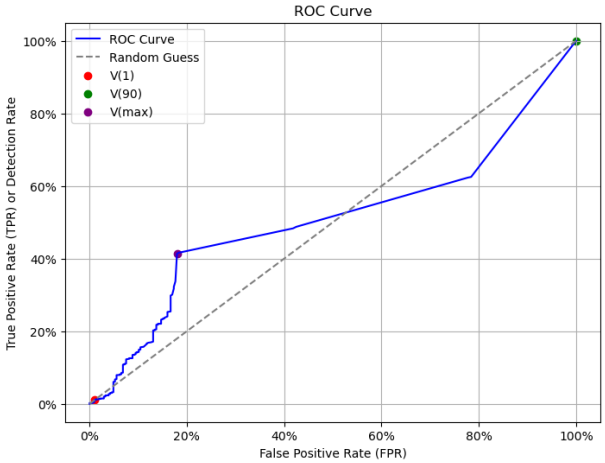
2-tuple



The classes are inseparable via visual inspection



For around **~80% detection rate**, there is a significantly lesser False Positive Rate.

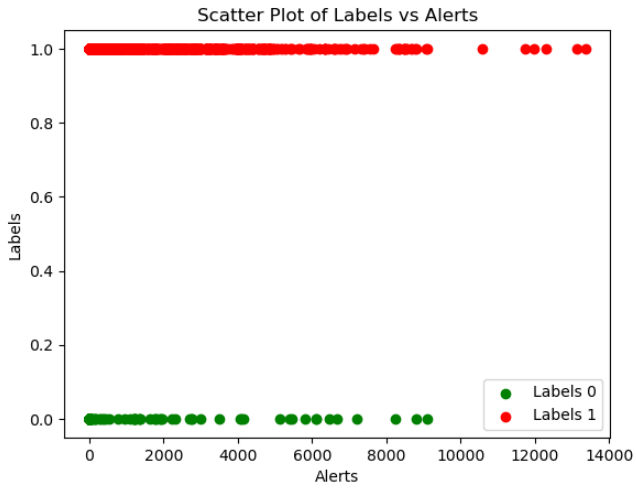


V1 = 2%

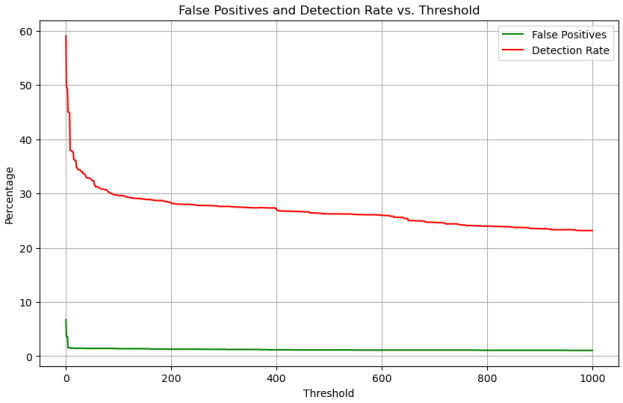
V90 = 100%

Vmax = 0.51

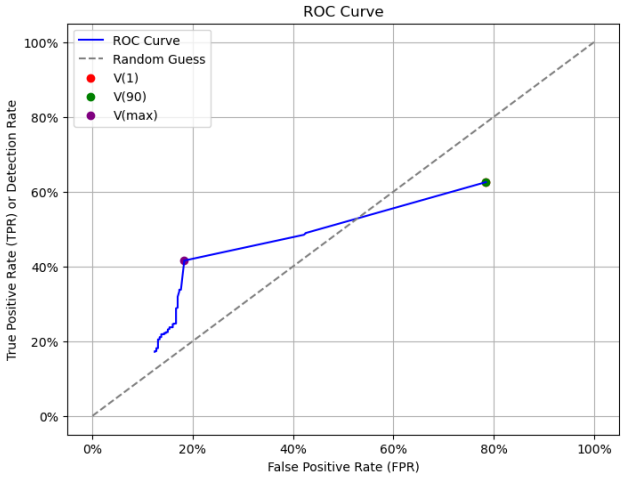
3-tuple



This is still inseparable yet the distinction is more pronounced in 3 – tuples than other 2 models



Detection rate across the threshold is fairly consistent which is a good sign.



V1 = 62.5%

V90 = 78%

Vmax = 0.32