SMALL EFFECT SIZES IN MALWARE DETECTION? MAKE HARDER TRAIN/TEST SPLITS!

- TIRTH PATEL (UMBC)
- FRED LU (BOOZ ALLEN HAMILTON)
- DR. EDWARD RAFF (BOOZ ALLEN HAMILTON)
- DR. CHARLES NICHOLAS (UMBC)
- DR. CYNTHIA MATUSZEK (UMBC)
- JAMES HOLT (LABORATORY FOR PHYSICAL SCIENCES)

Agenda

01 Challenges and Motivation

02 Methodology

03 Results

04 Conclusion

05 Limitations and Future Work

Industry vs Academia Data Challenges

Academics can’t easily test their classifiers on large-scale datasets

Disparity between malware dataset availability in industry vs academia

Testing on small-scale datasets can lead to overfitting

May not enable a researcher to distinguish minute differences in two models’ accuracies

The SOREL and EMBER datasets provide access to benign file metadata

But there are no public datasets with large amounts of benign files

Objectives

Creation of improved train/test datasets for evaluating malware detection.

Configure the “difficulty” of a train/test split

Enable smaller test sets that can robustly evaluate differences in classifier performance

Approach Summary

Robust Classifiers

A strong malware detector should be able to generalize, identifying unseen data

Key Insight

Can configure train/test split “difficulty” by carefully selecting which families go in the train/test splits

Bias and Data Leakage

The families in the train and test splits are exclusive, mitigating sources of overfitting

Dataset Sources

- We malware samples from 184 different malware families.
  - Top families in VirusShare, labeled using AVClass

- 10,000 total files per family
  - 8,000 train / 2,000 test

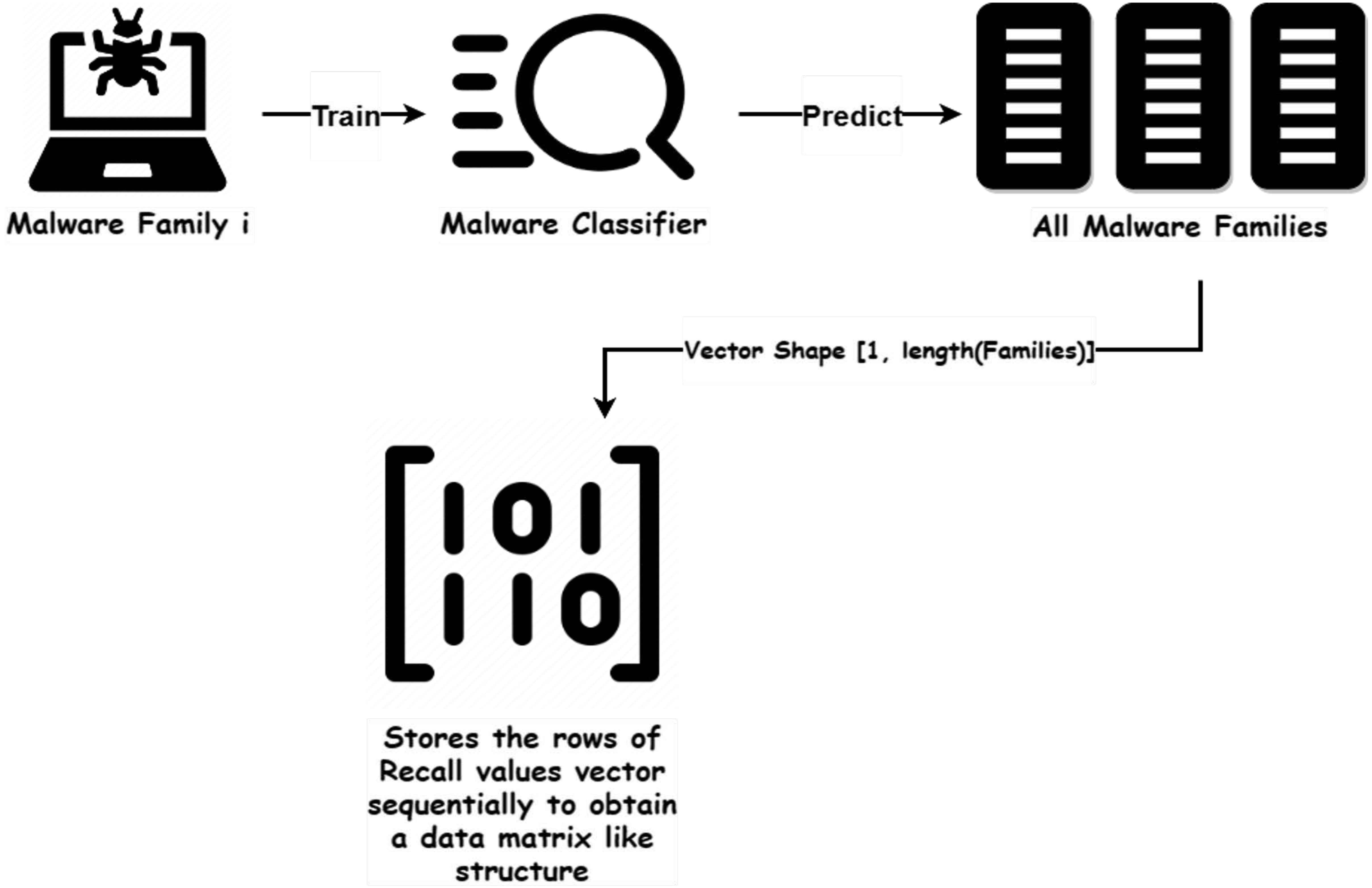
- Malware was collected from the VirusShare corpus

- Benignware was collected from the EMBER dataset

Description of Dataset

Files	**Training**	**Testing**	**Total**	**Source**
Malicious	1,472,000	368,000	1,840,000	VirusShare
Benign	300,000	100,000	400,000	EMBER
Total	1,772,000	468,000	2,240,000	

Our Approach

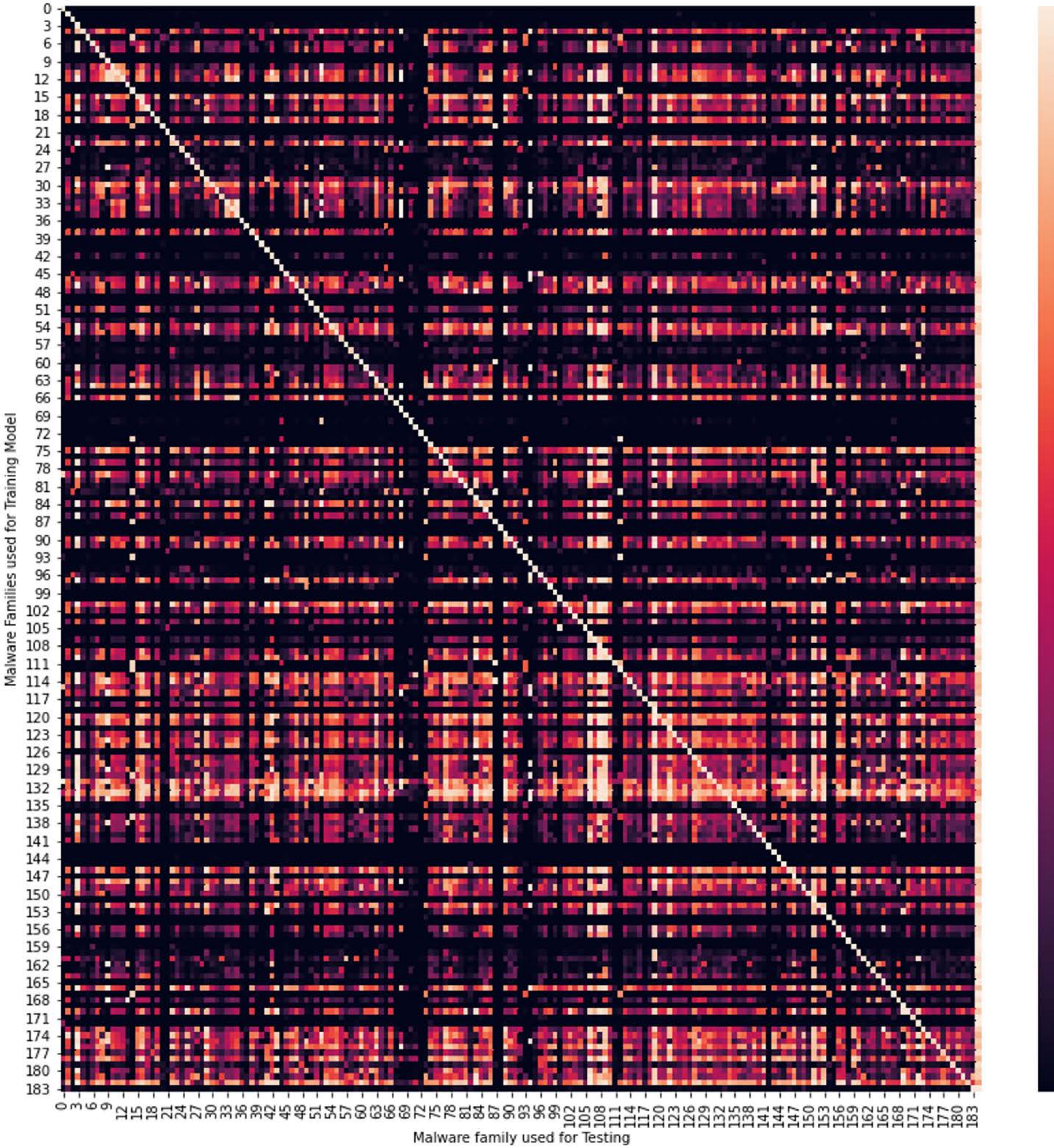


Stores the rows of Recall values vector sequentially to obtain a data matrix like structure

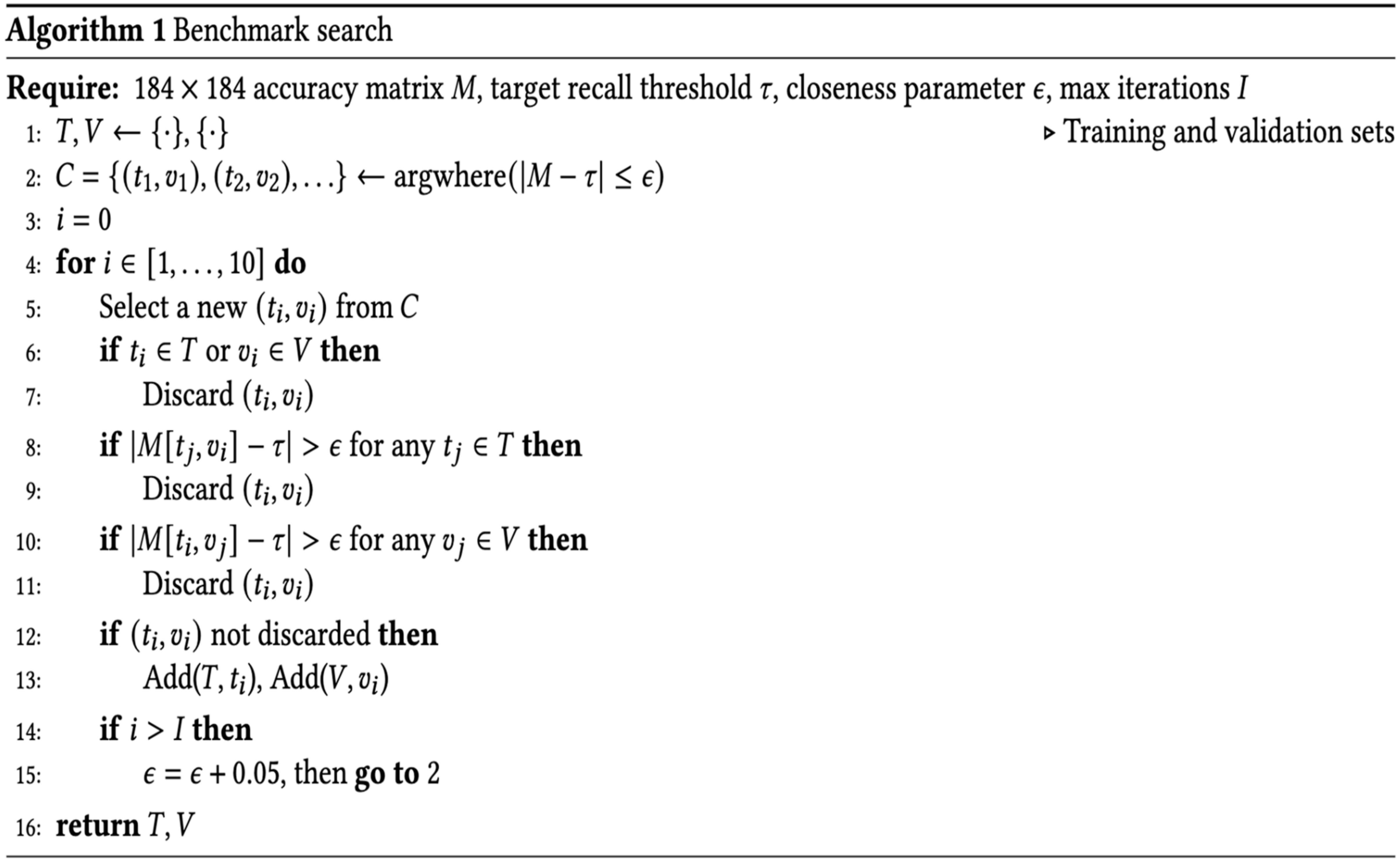
184 X 184 Data Matrix

- R[i,j] represents the recall value obtained when a malware classifier trained on family i is used to predict samples on family j.

- The matrix reveals families which are globally “easy” or “hard” to predict



Benchmark Search Algorithm



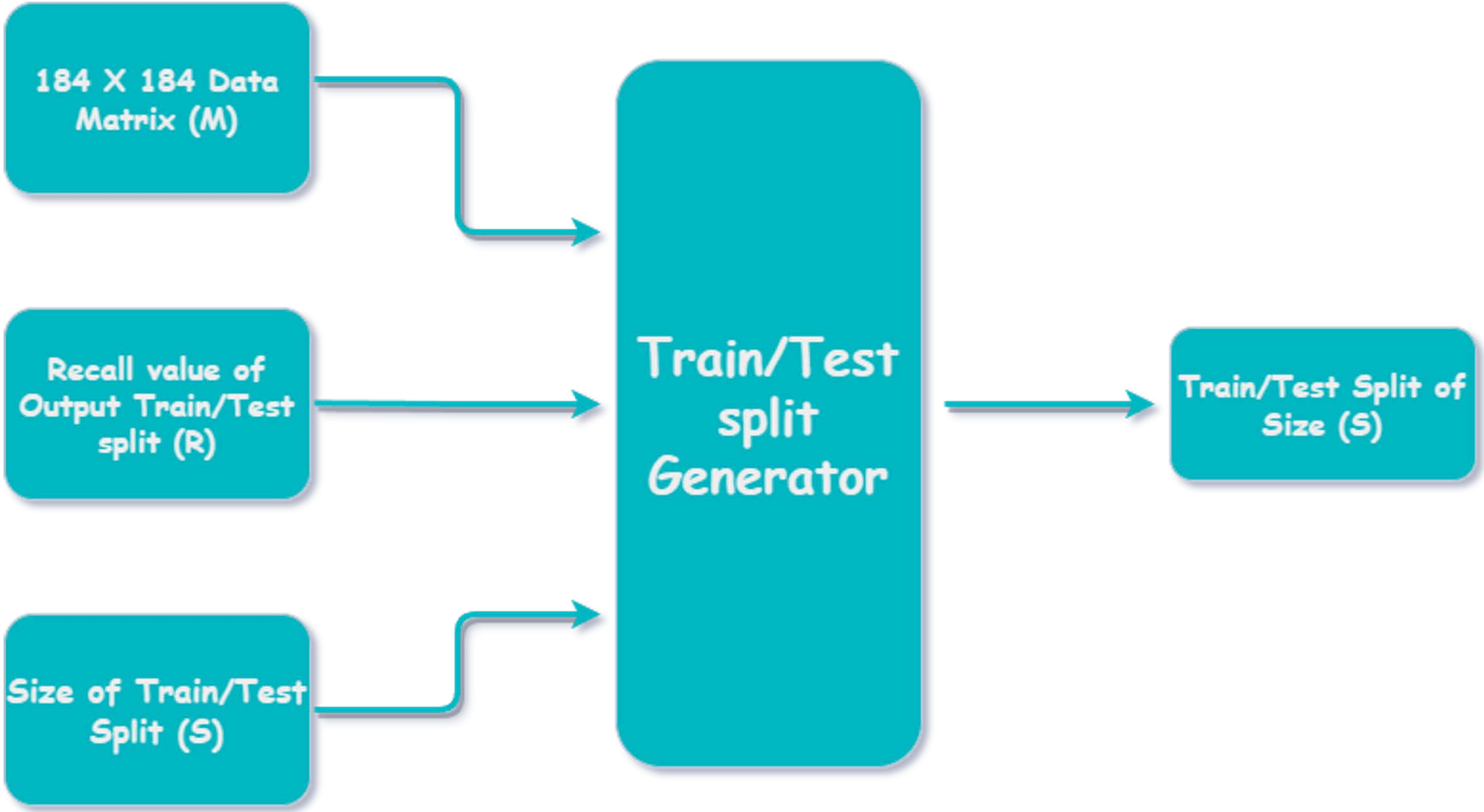
Algorithm 1 Benchmark search

```
Require: $184 \times 184$ accuracy matrix $M$, target recall threshold $\tau$, closeness parameter $\epsilon$, max iterations $I$
 1: $T, V \leftarrow \{\cdot\}, \{\cdot\}$                    ▷ Training and validation sets
 2: $C = \{(t_1, v_1), (t_2, v_2), \ldots\} \leftarrow \text{argwhere}(|M - \tau| \leq \epsilon)$
 3: $i = 0$
 4: for $i \in [1, \ldots, 10]$ do
 5:     Select a new $(t_i, v_i)$ from $C$
 6:     if $t_i \in T$ or $v_i \in V$ then
 7:         Discard $(t_i, v_i)$
 8:     if $|M[t_j, v_i] - \tau| > \epsilon$ for any $t_j \in T$ then
 9:         Discard $(t_i, v_i)$
10:     if $|M[t_i, v_j] - \tau| > \epsilon$ for any $v_j \in V$ then
11:         Discard $(t_i, v_i)$
12:     if $(t_i, v_i)$ not discarded then
13:         Add$(T, t_i)$, Add$(V, v_i)$
14:     if $i > I$ then
15:         $\epsilon = \epsilon + 0.05$, then go to 2
16: return $T, V$
```

- **Inputs**
  a. Malware detection data M (e.g., Malconv 184x184 matrix)
  b. Target recall threshold τ
  c. A small threshold ϵ for the difference between actual recall and target recall
  d. Number of iterations I (set to 1000)

- **Procedure**
  a. Start with the Malconv 184x184 data M
  b. Identify elements in M that are ϵ-close to the target recall τ.
  c. Randomly sample pairs of training families T and testing families V corresponding to the identified elements

- **Output**
  a. Training Set of Malware families T
  b. Testing set of malware families V



184 X 184 Data Matrix (M)
Recall value of Output Train/Test split (R)
Size of Train/Test Split (S)
Train/Test split Generator
Train/Test Split of Size (S)

Train/Test Splits

Each train/test split consists of two sets of malware families

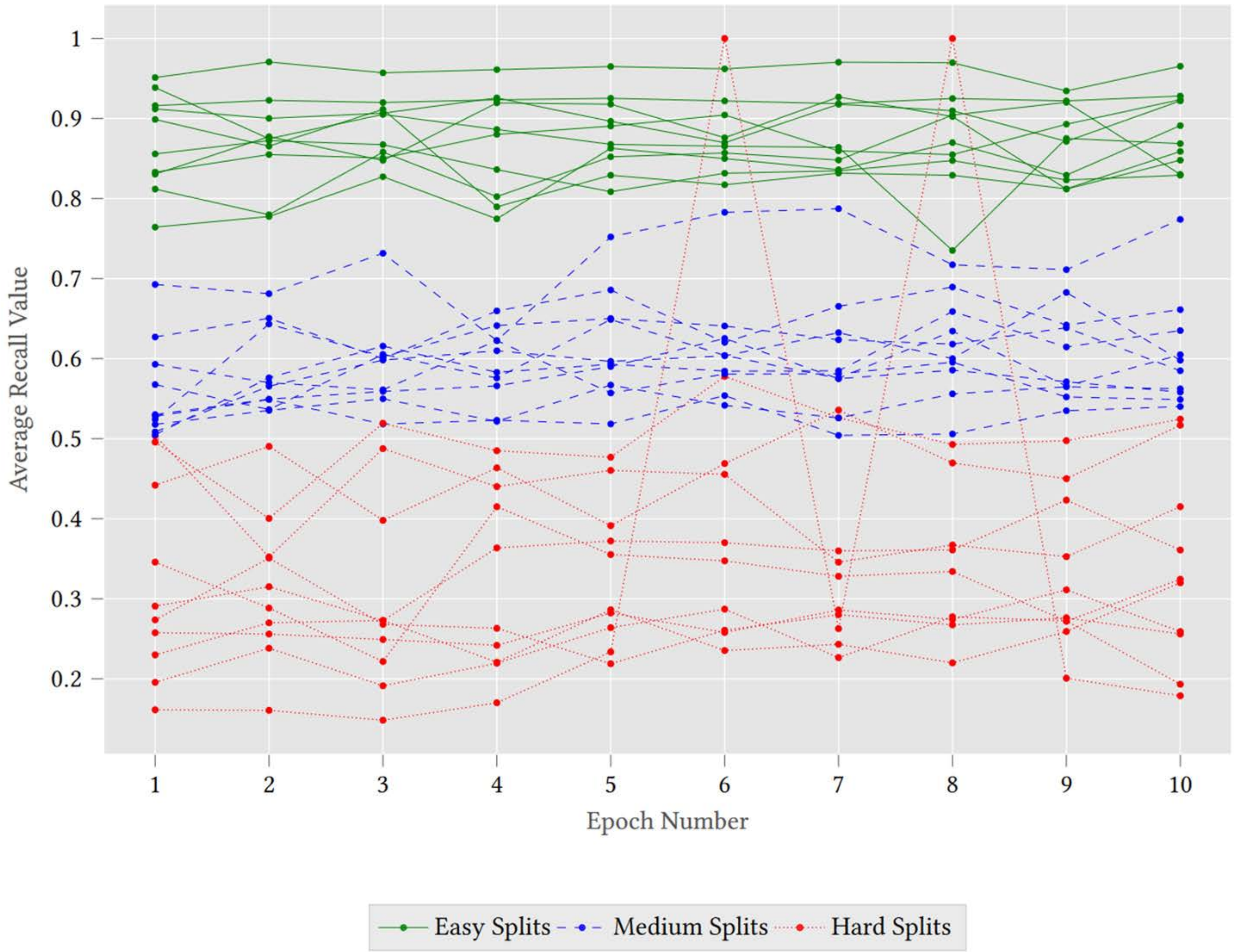
Train/test splits are divided into three categories based on difficulty:

- Easy: predicted recall ~0.9
- Medium: predicted recall ~0.5
- Hard has predicted recall ~0.25

Results

	Modified Train/Test Splits			
Algorithm	Normal	Easy	Medium	Hard
Byte n-grams	94.87	79.48	66.06	58.52
MalConv	91.14	85.88	63.81	44.73
MalConv GCT	93.29	83.43	61.51	33.49
XGBoost	99.64	99.08	90.80	72.80

MalConv GCT Train/Test Split Results



Conclusion

It is possible to configure the difficulty of malware classification by selecting the families in a train/test split

We showed consistency in split difficulty for all four types of malware classifiers

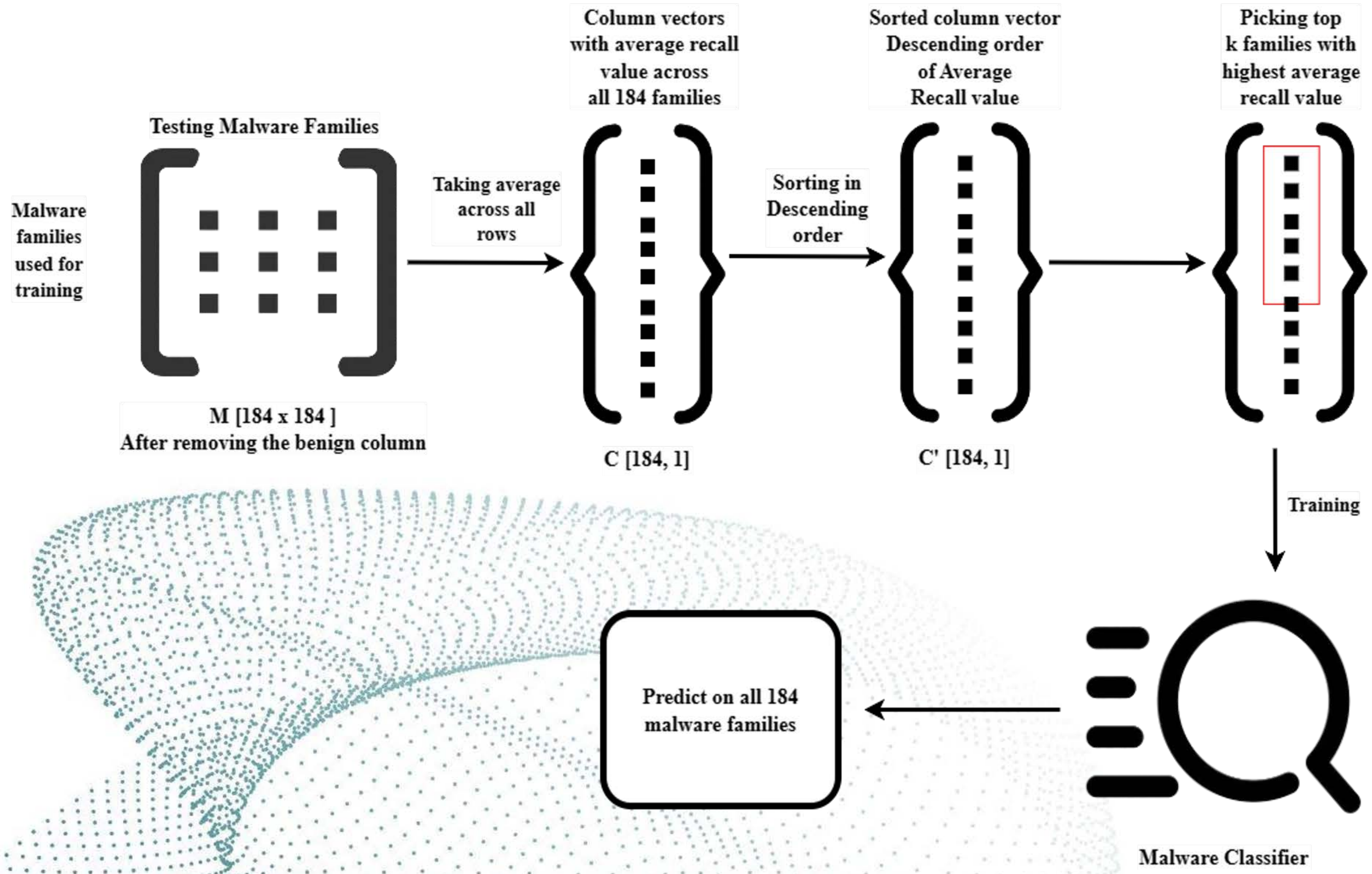
Can distill a small but challenging test set which can distinguish between the performances of two classifiers

Limitations and Future Work

- Tens of thousands of malware families exist, and our evaluation was limited to 184 common ones
- Bias from the classification algorithm selected for generating train/test splits
- Further investigation of unexpected performance for certain combinations of malware families

Failed Approaches For Generating Train/Test Splits

Top K families Pick

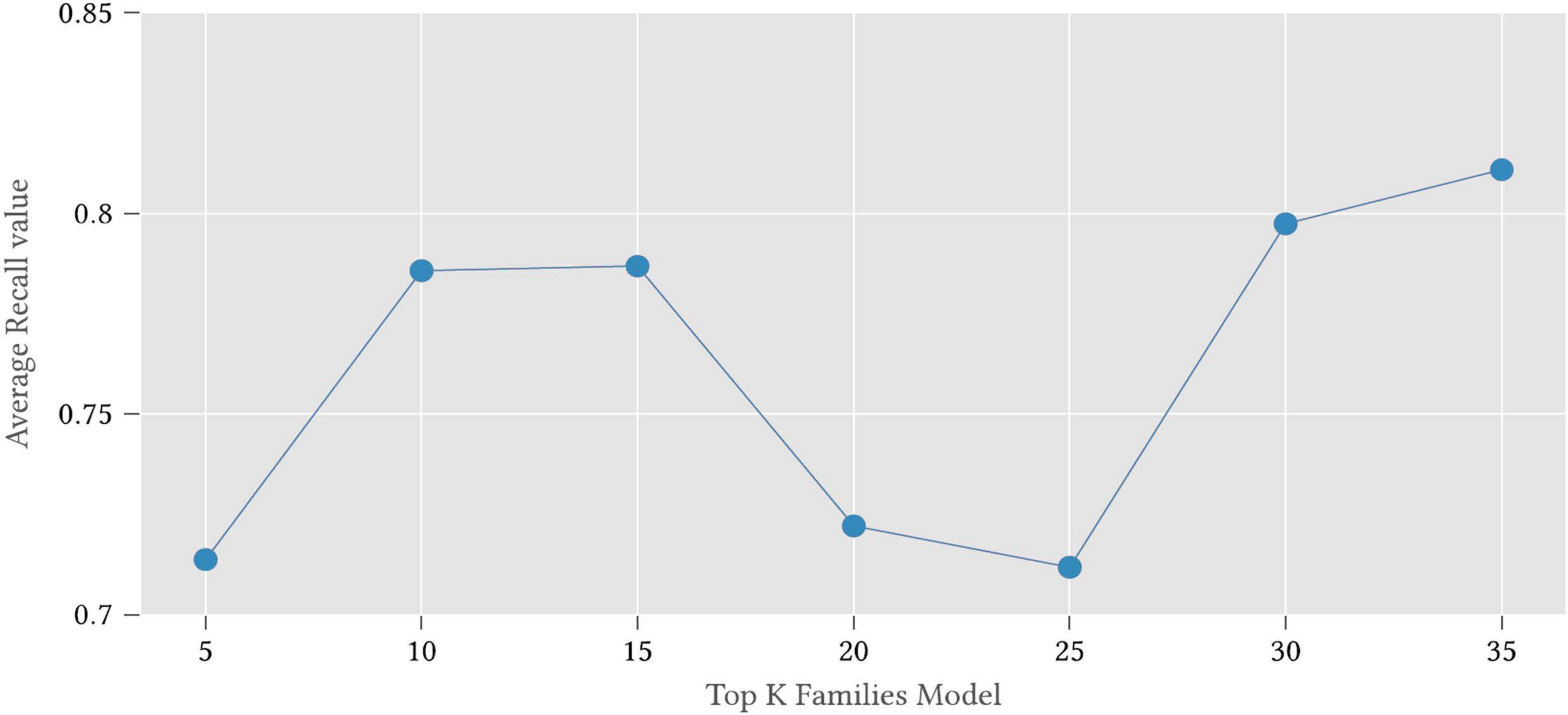

Testing Malware Families
Malware families used for training
M [184 x 184]
After removing the benign column
Taking average across all rows
Column vectors with average recall value across all 184 families
C [184, 1]
Sorting in Descending order
Sorted column vector Descending order of Average Recall value
C' [184, 1]
Picking top k families with highest average recall value
Training
Malware Classifier
Predict on all 184 malware families

- **Inputs**
  a. Malware detection data M (e.g., Malconv 184x184 matrix)
  b. No. Malware families *K*

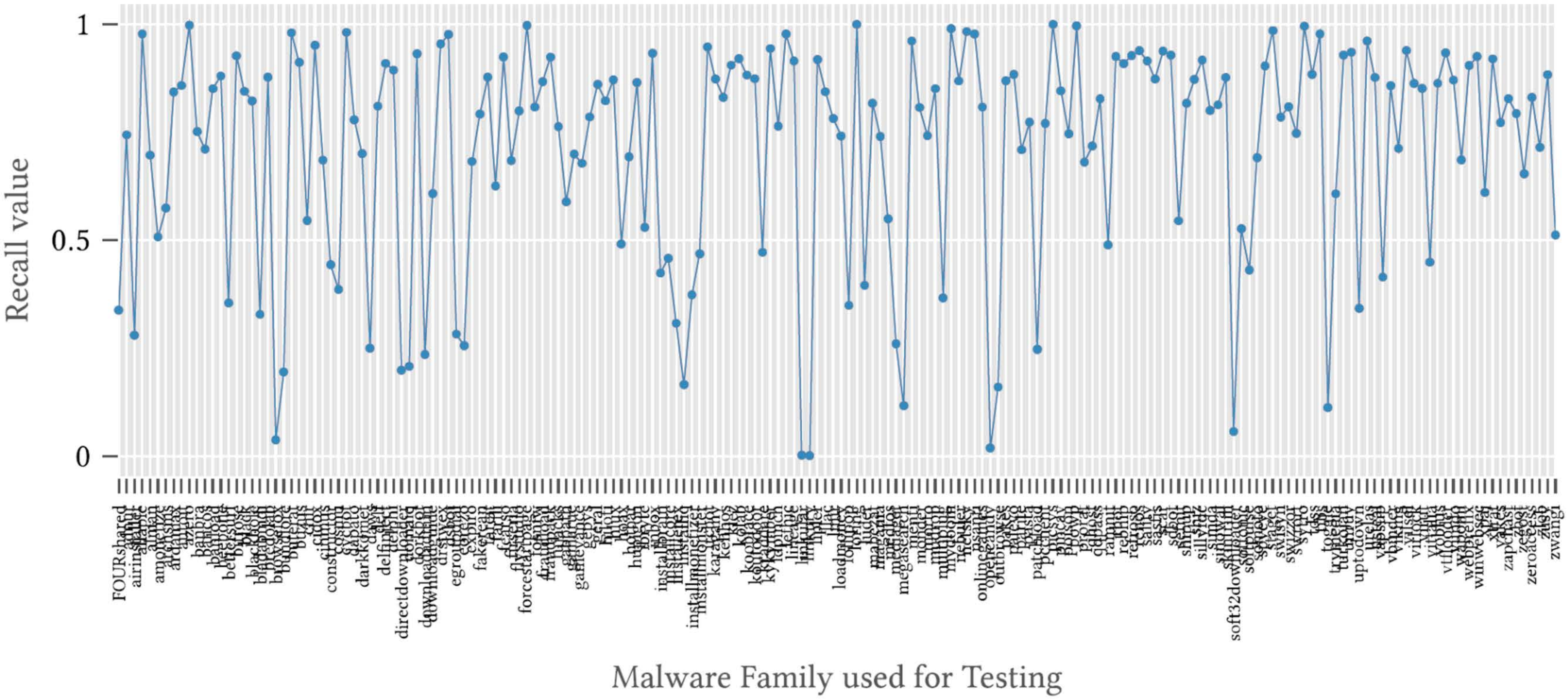
- **Procedure**
  a. Start with the Malconv 184x184 data M
  b. Take Average across all rows.
  c. Sort in Descending order
  d. Pick top K families with highest recall values

- **Output**
  a. Training Set of Malware families *T* of size *K*

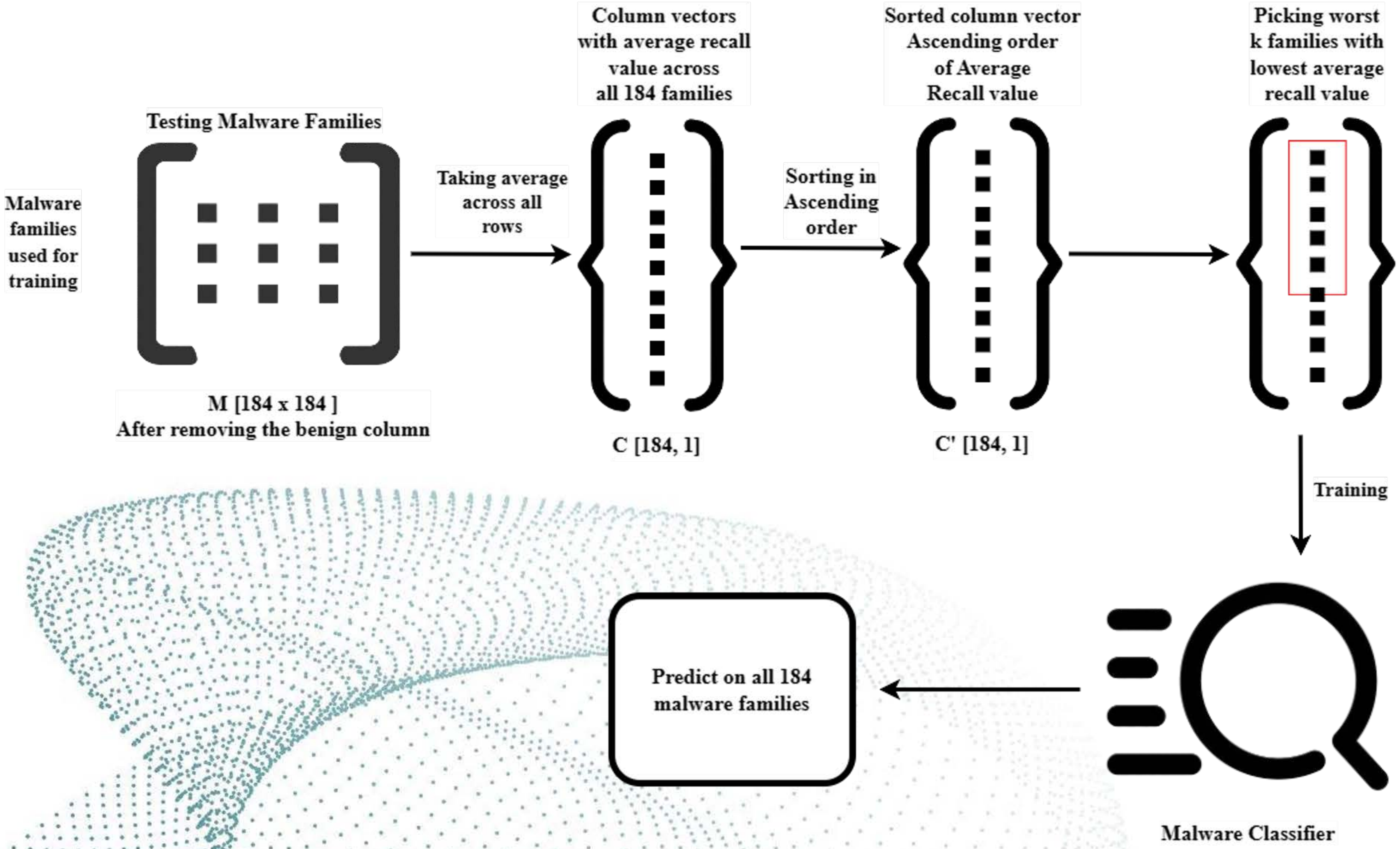
Top K families Results



Top 5 families Results



Worst K families Pick

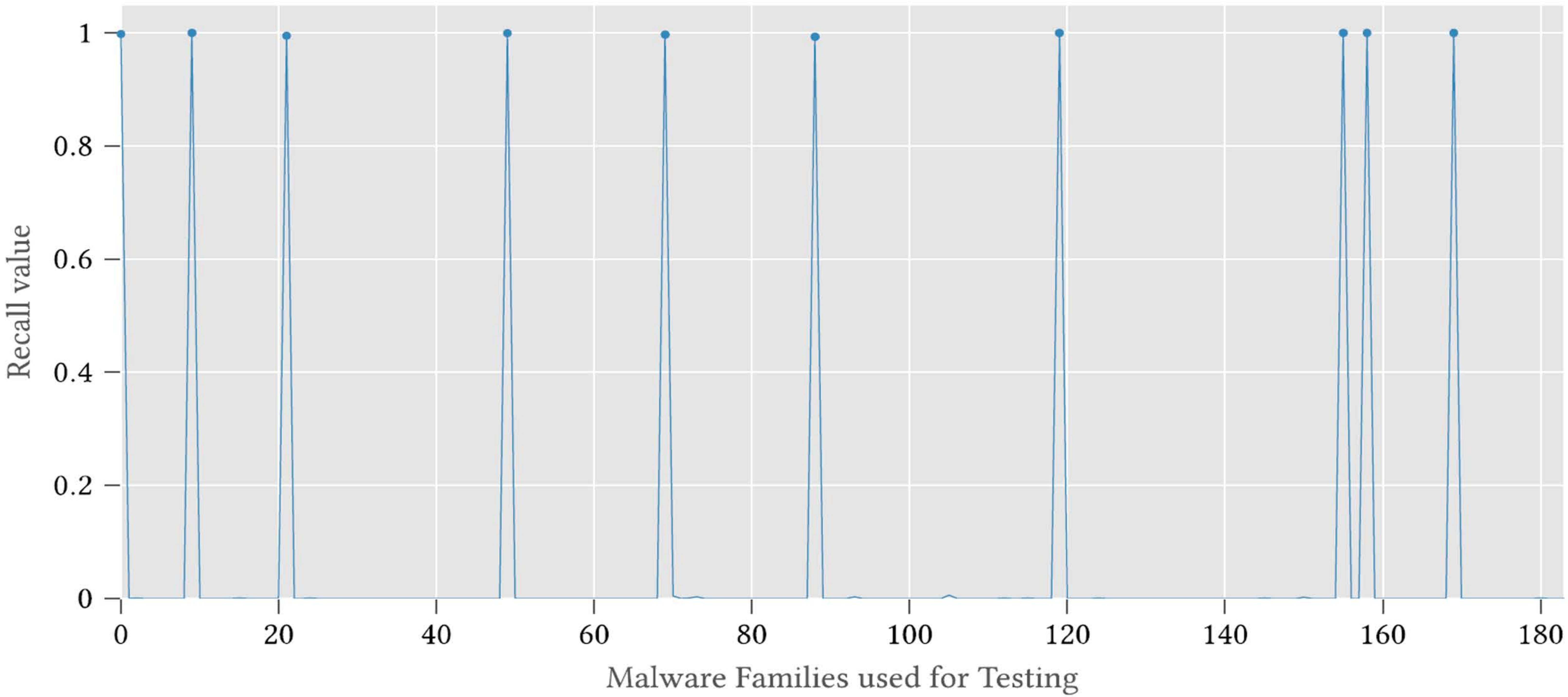

Column vectors with average recall value across all 184 families
Sorted column vector Ascending order of Average Recall value
Picking worst k families with lowest average recall value
Testing Malware Families
Malware families used for training
Taking average across all rows
Sorting in Ascending order
M [184 x 184]
After removing the benign column
C [184, 1]
C' [184, 1]
Training
Predict on all 184 malware families
Malware Classifier

- **Inputs**
  a. Malware detection data M (e.g., Malconv 184x184 matrix)
  b. No. Malware families *K*

- **Procedure**
  a. Start with the Malconv 184x184 data M
  b. Take Average across all rows.
  c. Sort in Ascending order
  d. Pick worst K families with lowest recall values

- **Output**
  a. Training Set of Malware families *T* of size *K*

Worst 10 families Results



THANK YOU

Questions?