TRABALHO FINAL DE MINERAÇÃO DE DADOS

Ember dataset

1

EMBER DATASET

O que é o ember dataset?

EMBER DATASET

EMBER: Endgame Malware Benchmark for Research.

Utilizado no campo da segurança cibernética para pesquisa e desenvolvimento de técnicas de detecção de malware.

Desenvolvido pela equipe da Endgame, uma empresa especializada em segurança cibernética.

EMBER DATASET

Seu objetivo é fornecer uma ampla variedade de amostras de arquivos executáveis em formato, incluindo tanto malwares quanto arquivos benignos.

Nele contém as features das amostras, tanto informações estáticas quanto dinâmicas.

O dataset consiste em uma coleção de arquivos JSON, em que cada linha contém um único objeto JSON.

EMBER DATASET

Cada objeto inclui os seguintes tipos de dados:

- O hash SHA-256 do arquivo original como um identificador único.
- Informações temporais aproximadas que estabelecem uma estimativa.
- Um rótulo, que pode ser 0 para benigno, 1 para malicioso ou -1 para não rotulado.
- Oito grupos de características brutas que incluem tanto valores analisados quanto histogramas independentes de formato.

EMBER DATASET

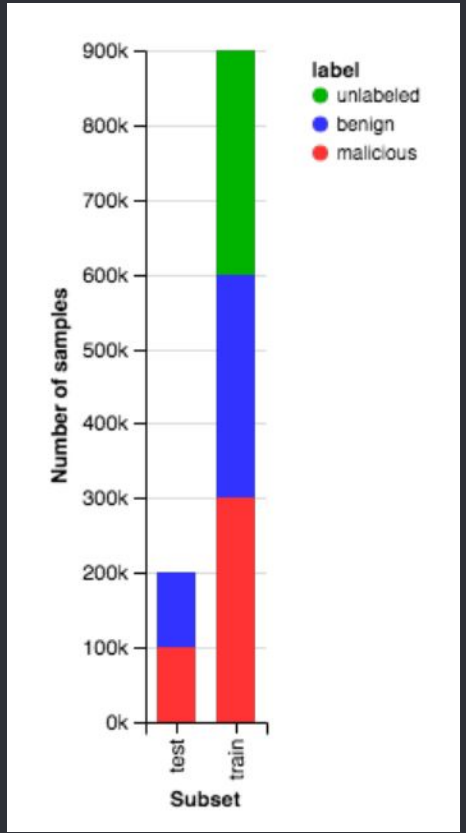

```
"sha256": "000185977be72c8b007ac347b73ceb1ba3e5e4dae4fe98d4f2ea92250f7f580e",
"appeared": "2017-01",
"label": -1,
"general": {
  "file_size": 33334,
  "vsize": 45056,
  "has_debug": 0,
  "exports": 0,
  "imports": 41,
  "has_relocations": 1,
  "has_resources": 0,
  "has_signature": 0,
  "has_tls": 0,
  "symbols": 0
},
"header": {
  "coff": {
    "timestamp": 1365446976,
    "machine": "I386",
    "characteristics": [ "LARGE_ADDRESS_AWARE", ..., "EXECUTABLE_IMAGE" ]
  },
```

```
"imports": {
  "KERNEL32.dll": [ "GetTickCount" ],
  ...
},
"exports": []
"section": {
  "entry": ".text",
  "sections": [
    {
      "name": ".text",
      "size": 3584,
      "entropy": 6.368472139761825,
      "vsize": 3270,
      "props": [ "CNT_CODE", "MEM_EXECUTE", "MEM_READ"]
    },
```

```
"histogram": [ 3818, 155, ..., 377 ],
"byteentropy": [0, 0, ... 2943 ],
"strings": {
  "numstrings": 170,
  "avlength": 8.170588235294117,
  "printabledist": [ 15, ... 6 ],
```

EMBER DATASET

- O dataset contém cerca de 1,1M de amostras, separadas em:
  - 900k para treino
    - 300k maliciosos
    - 300k benignos
    - 300k não rotulados
  - 200k para teste



EMBER DATASET

Porque escolher a base de dados EMBER para esse projeto de mineração:

- Pesquisa relacionado a meu projeto de IC
- Base de dados muito grande e consolidada
- Comparar com diversos outros resultados obtidos

2

Pré-processamento

Preparando os dados

Pré-processamento

Etapas do pré-processamento:

- Amostragem
- Filtragem
- Extração dos rótulos
- Seleção de um subconjunto de features

Amostragem

Como o dataset contém 1,1M de amostras, é muito custoso utilizá-lo completo.

Foi separado em 5 datasets de treino, contendo 180k cada e 1 de teste com 200k.

Cada conjunto foi testado separadamente.

Filtragem

No dataset original existem amostras rotuladas como -1, podendo ser tanto benignas quanto malignas.

Em nossa análise, esse tipo de amostra pode comprometer a sua qualidade, então foi necessário remover esses objetos.

Extração dos rótulos

Para realizar a mineração corretamente, é necessário extrair o label (0 ou 1) da base de dados, criando um novo conjunto.

Foi realizado em todos os conjuntos de dados, tanto nos de treino quanto no de teste.

Seleção de um subconjunto de features

No dataset original, existem algumas features que não são importantes para nossa mineração escolhida, como o sha256, md5, appeared, label, avclass.

Além dessas, existem características que nossa análise não é capaz de suportar, como strings.

Então foram removidos também as features header, section, imports, exports, datadirectories.

3

Mineração

Analisando os dados

Mineração

Como o nosso trabalho se resume em um problema de classificação, decidimos utilizar a Árvore de Decisão.

Árvore de Decisão é um bom um algoritmo de aprendizado de máquina supervisionado, além de ser muito utilizado devido à sua simplicidade e interpretabilidade.

Mineração

Para a criação dessa árvore, utilizamos a biblioteca Scikit-learn no python. Ela fornece uma ampla variedade de ferramentas e algoritmos para todas as etapas de uma mineração de dados.

Utilizamos a DecisionTreeClassifier, mais precisamente, que se resume em um algoritmo de Árvore de Decisão simples para classificação.

Mineração

Nela, é fornecido a base de dados e seus rótulos, criando a árvore. Depois, para a classificação é testada com a base de teste e seus respectivos rótulos.

```
# Organiza os dados como um array estilo numpy
X_train = np.array(X_train)
y_train = np.array(y_train)
X_test = np.array(X_test)
y_test = np.array(y_test)

# Cria a arvore de decizão vazia
clf = DecisionTreeClassifier()
num_features = len(X_train[0])

# Treina o classificador usando o conjunto de treinamento
clf.fit(X_train, y_train)

# Faz as previsões usando o conjunto de teste
y_pred = clf.predict(X_test)
```

4

Pós-processamento

Refinando os resultados

Pós-processamento

Após a análise dos dados, é necessário obter os resultados para verificar se o que foi produzido foi relevante.

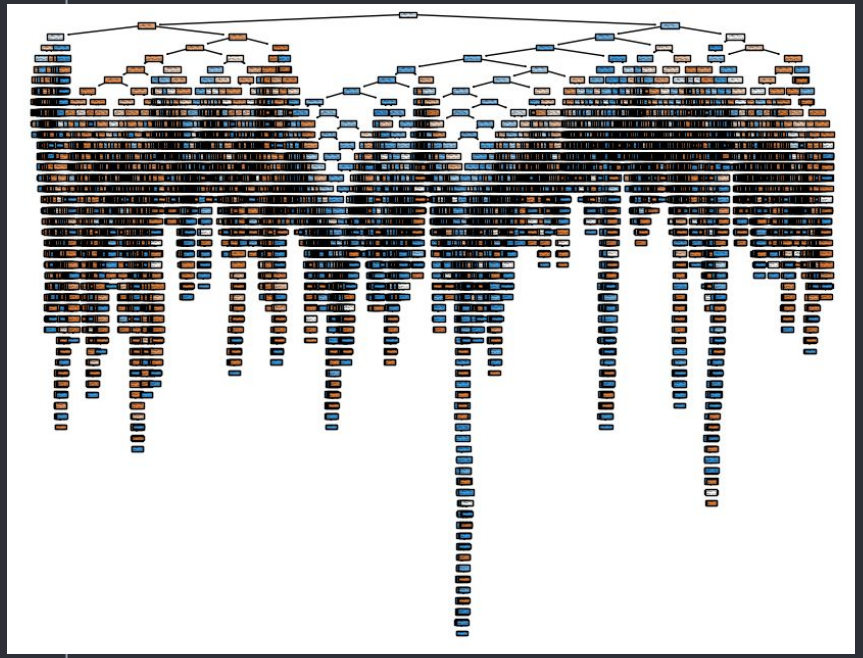
No nosso caso, foi calculado a acurácia dos dados (taxa geral de acertos do modelo).
Acurácia = (Verdadeiros Positivos + Verdadeiros Negativos) / (Total de Observações)

Pós-processamento

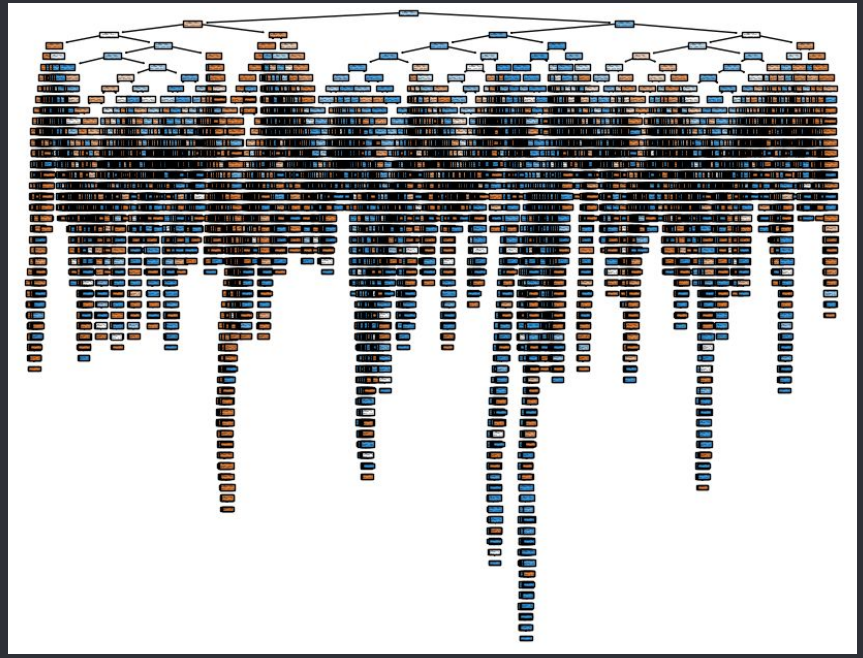
Também foi obtida a imagem da árvore de decisão, somente para obter uma noção de sua dimensionalidade.

Como foram obtidas árvores, diferentes (pois foram realizados alguns treinamentos), elas obtiveram resultados diferentes.

- Pós-processamento

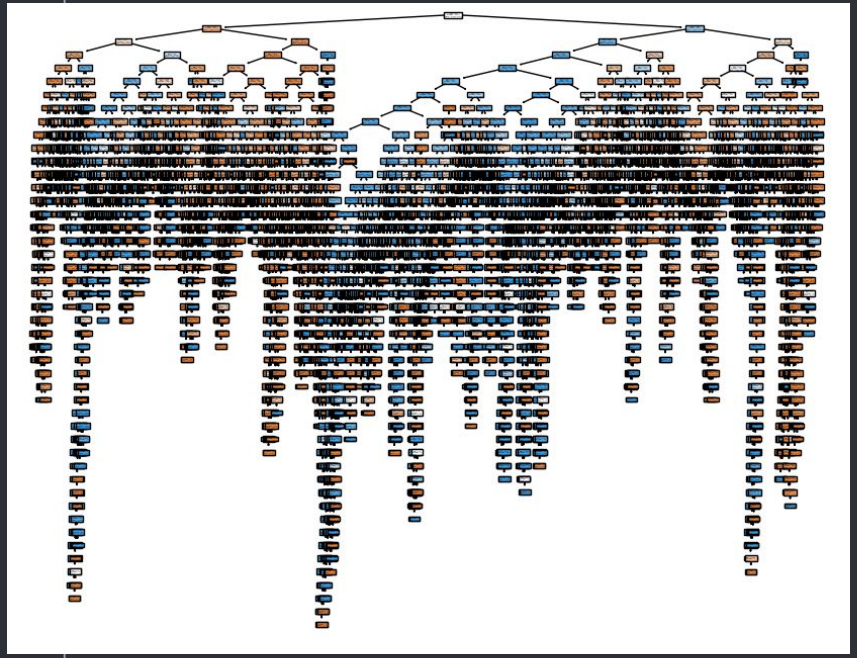


Acurácia: 0.7876

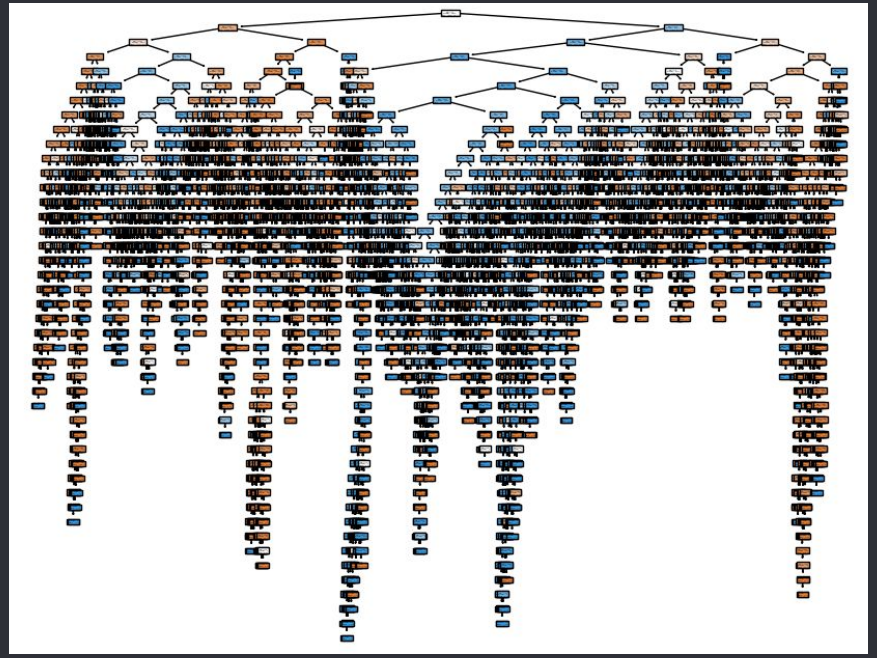


Acurácia: 0.8709

- Pós-processamento



Acurácia: 0.7993



Acurácia: 0.8173

5 Conclusão

Discussão do trabalho

Conclusão

Observando os dados obtidos, é possível perceber que mesmo retirando boa quantidade de features relevantes, o resultado ainda é satisfatório.

Certamente que comparando com outros trabalhos grandes ou com a própria análise do EMBER, nosso resultado foi muito inferior (0.99911), mas utilizamos apenas uma árvore de decisão simples.

Conclusão

Uma possível continuação para esse trabalho seria desenvolver algum método para utilizar as strings que foram removidas, ou ainda o label não rotulado (-1).

Outra ideia seria alterar o tipo de árvore usada, ou até trocar o classificador, como usar Naive Bayes, mas ainda mantendo a ideia de simplicidade e interpretabilidade .