MO444 - Projeto Final

Jardel Santos
Paulo Henrique Tadei
Anderson Rocha

1 - Introdução

Com a velocidade do crescimento do setor de tecnologia nos dias de hoje, a busca por estar preparado para novos desafios se torna cada vez mais importante para o sucesso no mundo corporativo.

Para uma empresa de Segurança da Informação isto não é diferente. Por estar sempre um passo atrás dos hackers mal intencionados, a característica que define a agilidade da empresa é poder encurtar ao máximo o tempo em que uma ameaça é desenvolvida e sua devida proteção também, para isso, conhecimento é fundamental e é nesse ponto que iremos atuar.

Neste projeto iremos abordar métodos alternativos para classificação de arquivos suspeitos de modo a utilizar informações do próprio arquivo para gerar as predições, ganhando agilidade na seleção dos arquivos que deverão ser analisados futuramente.

2 - Objetivos

Este trabalho tem como objetivo analisar uma base de dados contendo informações relacionadas a arquivos suspeitos e treinar um algoritmo de machine learning capaz de identificar se um arquivo pode ser uma ameaça do tipo malware ou não.

Dado um conjunto de características extraídas da base, visamos encontrar um padrão entre as informações e através delas realizar as predições com base nas semelhanças das mesmas.

3 - Base de dados

A base de dados completa é composta por 20000 arquivos coletados contendo apenas duas classes, malwares e não-malwares.

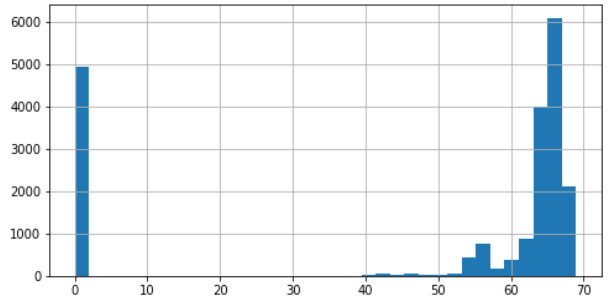
Esses arquivos foram coletados através de processos realizados em uma empresa de segurança. Alguns exemplos dos arquivos utilizados em nossa base são:

- Arquivos coletados por uma equipe técnica de suporte via acesso remoto em máquinas que sofreram fraudes bancárias;
- Arquivos coletados pela internet através de processos automatizados;
- Coletas realizadas através de APIs como no caso do "Virus Total", que disponibiliza uma série de informações como por exemplo a quantidade de antivírus que acusou o mesmo como um malware, informação que foi muito importante para nosso projeto.

3.1 - Determinando arquivos de vírus

Para determinar se o arquivo de dado é virus ou não, utilizamos o API chamado "Virus Total", cujo, como dito anteriormente, escaneia o arquivo em diversos antivírus e nos informa a proporção de antivírus que o classificou como malicioso (dado entre 0 e 100). Em nossa análise determinamos se um arquivo é malicioso para um limiar de 50%.

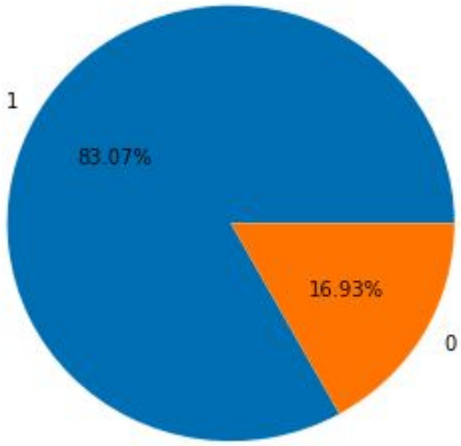
Gráfico 1 - Histograma Virus Total.



3.2 - Distribuição dos dados

O gráfico 2 mostra a distribuição de dados, onde pode-se ver uma proporção muito maior de arquivos de vírus. Isso se deve ao fato de os arquivos recebidos pela empresa de segurança, fonte de nossos dados, serem arquivos suspeitos e, consequentemente, na maioria dos casos acabam sendo maliciosos.

Gráfico 2 - Distribuição dos Dados (1 para arquivos de vírus, 0 para não vírus).



O desbalanceamento dos dados deve ser considerado e seu tratamento é mostrado no tópico **5.2**.

4 - Extração de Features

Para extrair as informações dos arquivos foram necessários a utilização de alguns programas auxiliares. Algumas informações foram geradas com base na análise estática dos arquivos da base de dados, como por exemplo data de modificação do arquivo ou quantidade de DLLs utilizada pelo arquivo.

Os programas auxiliares utilizados para criação de variáveis foram:

- Detect It Easy (DIE): utiliza os primeiros bytes do arquivo (magic number e cabeçalhos) para tentar descobrir qual o tipo do arquivo, independentemente de sua extensão. Também consegue informações de qual linguagem, compiladores e ligadores foram utilizados no binário,
- Static PE: são informações das sessões do formato executável (.exe), com ela é possível identificar funções importadas que o binário utiliza, funções exportadas no caso de ele ser uma DLL, permissões das sessões (leitura, escrita, execução), entre outras.
- Exiftool: informações de metadados, como versão do binário, tamanho do código do arquivo, se possui licença, descrição do arquivo, tipo do arquivo, data e hora da sua instalação, entre outros.

Além disso, foram criadas features de Malwares Strings Similarity (MSS), através da comparação de strings com exemplares de famílias de malwares conhecidas, portanto foi utilizado a comparação de cada string exposta no binário para cada string conhecida das famílias de malwares e com isso obtivemos uma porcentagem de similaridade do artefato com a família comparada, no nosso projeto utilizamos como base apenas 4 famílias de malwares conhecidas, fornecidas pela empresa. Para cada família, foi gerado uma feature, totalizando 4 features MSS.

Com a utilização dos programas auxiliares, após processamento e extração das variáveis [4], nosso modelo obteve as seguintes *features*:

Tabela 1 - Features geradas através de informações do binário.

Variável	Tipo	Descrição
HasUrls	booelano	se o arquivo faz requisições para url
DLL_COUNT	inteiro	número de DLLs utilizados pelo arquivo
Code Size	inteiro	tamanho do código fonte do arquivo
File Size	categórico	tipo de tamanho do arquivo(bytes, kB, MB ou large)
Library	categórico	biblioteca utilizada para criar arquivo (exemplo: python)
Compiler	categórico	compilador utilizado para criar arquivo
Linker	categórico	ligador utilizado para criar arquivo
CharSet	categórico	tipo de formato de caractere do arquivo (exemplo: ASCII)
FileType	categórico	tipo de arquivo (exemplo: executável)
HasCopyright	booleano	se arquivo contém licensa
HasDescription	booelano	se arquivo veio com descrição

MSS BawaneH	inteiro	comparação de strings com familia de virus BawaneH
MSS MultiEncryptWinDLL	inteiro	comparação de strings com familia de virus MultiEncryptWinDLL
MSS Sayonara	inteiro	comparação de strings com familia de virus Sayonara
MSS WhatsappSLDLL	inteiro	comparação de strings com familia de virus WhatsappSLDLL

5 - Classificação

Para a escolha de nosso modelo de classificação foram levados em conta o fato de possuirmos um número considerável de variáveis categóricas, além de dados numéricos. Outro fator decisivo foi a escolha de um modelo que possui formas para tratamento de dados desbalanceados.

Com os fatos mencionados acima, o modelo de classificação proposto em nosso projeto é o *Random Forest*.

5.1 - Random Forest

Random Forest se refere a um conjunto de classificadores em árvore de decisão [3], onde cada árvore é gerada a partir de um subconjunto das amostras de treino.

Dado um novo vetor *xi* , sua predição final é baseada na classe mais prevista pelas árvores do modelo, método conhecido como *ensemble*.

5.2 - Tratamento do desbalanceamento

Com os dados desbalanceados, foram utilizadas duas variações do Random Forest, vistas em [2]:

- **Balanced Random Forest:** as árvores de decisão da floresta são treinadas com amostras balanceadas. O balanceamento de cada amostra é feito pela diminuição de casos de vírus, que são escolhidos aleatoriamente para cada árvore de forma a manter a proporção de 1:1 com o caso minoritário de não vírus.
- **Weighted Random Forest:** forma de aumentar o custo do modelo ao classificar erroneamente a classe minoritária. Os pesos são criados de acordo com a proporção atual entre o número de casos das classes e são utilizados no critério de decisão da variável de divisão em cada nó, e também nos nós terminais na hora de escolher a qual classe pertence.

6 - Validação do modelo

Pelo fato de nosso dataset ser pequeno e também bastante desbalanceado, não serão utilizadas técnicas de validação cruzada em nosso modelo (de forma a evitar maior desbalanceamento), e apenas uma simples divisão dos dados disponíveis em 80% para treino e 20% para teste.

As métricas utilizadas para validação do modelo foram:

- ➔ Acurácia: quantidade de acertos do modelo, dividido pela quantidade total de dados.
- ➔ Precisão: para a predição de uma classe, indica a razão entre os dados previstos corretamentes e o total de dados classificados como sendo dessa classe.
- ➔ Revocação: para os dados pertencentes a uma classe, indica a razão entre os dados previstos corretamente e o total de dados pertencentes à classe
- ➔ F1-Score: média harmônica entre os resultados de precisão e revocação.
- ➔ Gráfico de ROC e AUC Score: O ROC(Receiver Operator Characteristic) é a representação gráfica dada pela taxa de falsos positivos versus positivos verdadeiros, enquanto o AUC (Area Under Curve) é o valor da área formada pelo ROC do modelo ao utilizar diferentes limiares de decisão (como a taxa de probabilidade para considerar os dados como vírus, em nosso caso).

7 - Resultados Obtidos

A seguir são dadas as tabelas, bem como os gráficos, das métricas obtidas para os dois modelos de Random Forest utilizadas em nosso trabalho [4]:

Tabela 2 - Métricas obtidas Weighted Random Forest.

Weighted Random Forest		
	Train	Test
ACC	0.8641	0.8183
Precision	0.9649	0.9295
Recall	0.8674	0.8482
F1 Score	0.9136	0.8870

Gráfico 3 - ROC obtido nos dados de teste, Weighted Random Forest.

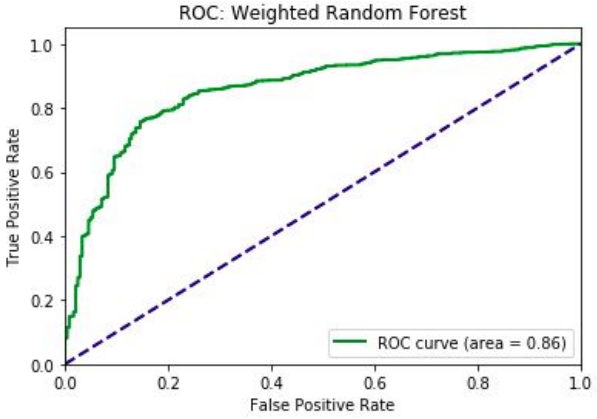
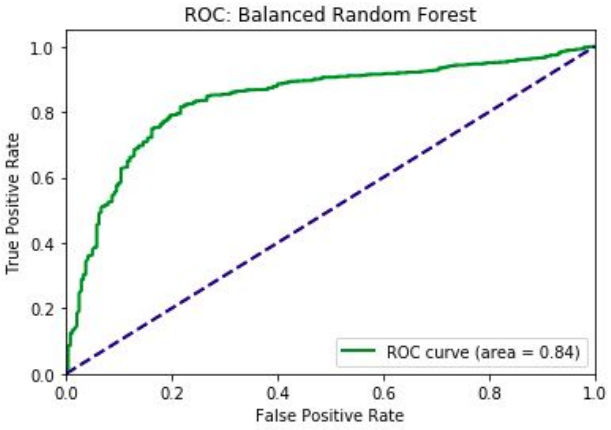


Tabela 3 - Métricas obtidas Balanced Random Forest.

Balanced Random Forest		
	Train	Test
ACC	0.8703	0.8138
Precision	0.9992	0.9414
Recall	0.8451	0.8241
F1 Score	0.9157	0.8789

Gráfico 4 - ROC obtido nos dados de teste, Balanced Random Forest.



Pelas métricas obtidas, vemos que para o problema proposto o método de Weighted Random Forest obteve no geral resultados poucos melhores.

Analisando a métrica de precisão ambos os modelos estão bem altos, mostrando que na maioria dos casos que eles chamam os arquivos de vírus eles realmente são. Para essa métrica também percebe-se que o Balanced Random Forest se comporta melhor.

Por outro lado, a métrica de revocação, talvez a mais importante para o problema proposto, é mais baixa, mostrando que os modelos ainda estão deixando passar despercebido alguns casos de arquivos maliciosos.

Para a curva de ROC nos dados de teste, também percebemos o Weighted Random Forest com uma AUC Score um pouco maior (0.86), nos levando a concluir ele como o melhor modelo.

8 - Conclusão

O projeto consistiu na criação de um modelo capaz de prever se os arquivos dados são maliciosos ou não, sendo um problema de classificação, onde a classe de vírus foi criada através da porcentagem de antivírus que acusaram o arquivo como malicioso, devendo ser maior que 50%.

A base de dados consiste em 20.000 arquivos e o modelo utilizado para a criação do classificador foi o Random Forest, pelo fato de possuirmos uma mistura de features numéricas e categóricas extraídas através de informações do conteúdo binário dos arquivos.

Ainda, considerando que os dados estavam desbalanceados (83% dos dados eram de vírus),

utilizou-se duas variações do Random Forest que tratam esse problema, sendo elas o Weighted Random Forest e o Balanced Random Forest.

Para validar os modelos, separou-se os dados em 80% para treino e 20% para teste, onde as métricas obtidas, como o AUC Score de 0.86 para o Weighted e 0.84 para o Balanced, no geral mostraram o Weighted Random Forest como um melhor modelo.

Como a base de dados é pequena, há uma grande possibilidade de os modelos gerados não estarem generalizando para tipos diferentes de vírus, sendo para projetos futuros a necessidade de uma maior quantidade de dados para a criação de um modelo mais robusto.

Além de um maior número de dados, também é interessante adicionar novas features com mais informações sobre os arquivos que possam ajudar nas predições, assim como algum método de seleção dessas features.

Nosso modelo utilizou-se de apenas 15 features, que em nosso caso consideramos as mais interessantes e um número suficiente para evitar um excesso de variáveis que possam aumentar a complexidade do modelo em relação ao pequeno número de dados da base.

9 - Referências Bibliográficas

[1] - Disciplina MO444 - Machine Learning and Pattern Recognition. Professor: Anderson Rocha.
[2] - Leo Breiman, Chao Chen. Using Random Forest to Learn Imbalanced Data.
[3] Leo Breiman. Random forests. Machine Learning, 2001.
[4] - GitHub com os códigos: https://github.com/ptadei/MO444A-Final-Assignment