The Net Wars: Attack of the Bots

Duarte Emanuel Ramos Meneses[2019216949]
Patricia Beatriz Silva Costa[2019213995]

Universidade de Coimbra, Coimbra, Portugal
{duartemeneses,patriciacosta}@student.dei.uc.pt

Abstract. Numa sociedade movida essencialmente pelas tecnologias, é importante que estas sejam seguras e que rapidamente consigam combater os ataques a que são alvo. Para isso, os mecanismos de Machine Learning são essenciais. É com isso em vista que este trabalho foi desenvolvido: encontrar o melhor algoritmo para classificar o tipo de ataque.

Keywords: Machine Learning · Ensemble · Random Forest · Neural Network · Classificação · Quality Metrics.

1 Introdução

Atualmente estamos inseridos num contexto em que dependemos das tecnologias para realizar as mais básicas tarefas do quotidiano. Muitas delas, inclusive, têm ligação à Internet, o que as torna um pouco vulneráveis a ataques. Deste modo, é imperativo saber lidar com cada tipo de ataque para que a tecnologia em questão não deixe de estar disponível.

A melhor tática para combater estas vulnerabilidades é conseguir identificar o mais cedo possível o tipo de ataque e agir conforme essa classificação. Para isso, existem inúmeros algoritmos de Inteligência Artificial que, dados alguns indicadores, conseguem classificar o tipo de ataque.

Foi com isso em mente que realizamos este trabalho. Testamos vários tipos de algoritmos, juntamente com outros tantos métodos de tratamento dos dados. Ao longo deste relatório iremos revelar os resultados obtidos bem como discutir cada um para percebermos qual a melhor opção para classificar ataques via Internet, dado o nosso dataset.

2 Feature Engineering

2.1 Desbalanceamento do dataset

Começamos por perceber o quão balanceado estava o dataset fornecido. O histograma seguinte mostra um claro desbalanceamento no tipo de ataques disponíveis nos dados.

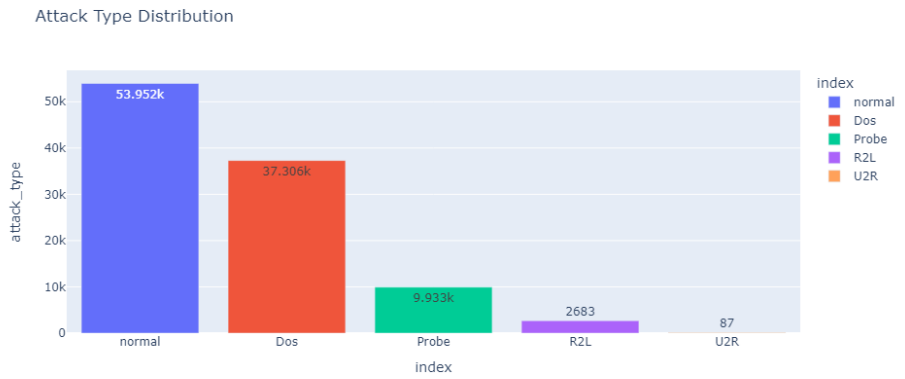

Fig. 1. Distribuição dos Tipos de Ataque (target)

Este é um problema uma vez que, se não for tratado devidamente, o modelo irá classificar quase todos os ataques como normal e nenhum como UDR ou R2L. Isto acontece pois, dada a distribuição dos dados, é muito mais provável o modelo acertar sendo ataque normal que nas restantes. Deste modo, se classificar todos os ataques como normal, consegue uma boa métrica, pelo que não ajusta o seu modelo convenientemente.

Existem inúmeras técnicas para contornar esta situação. Mais à frente iremos revelar os resultados obtidos em cada uma. Para já, mencionamos as que optamos por testar:

- **SMOTE:** gera dados sintéticos para as classes minoritárias a partir dos dados de input, de forma a equilibrar a quantidade de valores de cada classe
- **Pesos:** dá maior importância às classes minoritárias para que estas sejam mais notadas aquando da execução do modelo

2.2 Outliers

Decidimos também analisar a quantidade de outliers existente no dataset. Tínhamos à nossa disposição dois métodos comummente utilizados: Standard Deviation e Interquartil Range. No entanto, o primeiro é mais adequado a dados com distribuições normais. Verificamos essa questão no nosso dataset e percebemos que este não seguia uma distribuição normal, pelo que optamos pelo método Interquartil Range.

Com os outliers detetados, restavam duas opções: eliminar ou substituir por um valor aceitável (média dos valores dessa coluna). Se eliminassemos cada entrada que continha um outlier em pelo menos uma coluna, reduziríamos o dataset a, aproximadamente, 5%. Isso levaria a que perdessemos quase toda a informação, pelo que não consideramos ser uma boa opção.

Deste modo, optamos por tratar os outliers substituindo-os pela média dos valores da coluna em questão.

2.3 Variáveis categóricas

Os modelos de machine learning apenas funcionam para variáveis numéricas. Deste modo, utilizamos duas técnicas para transformar os atributos categóricos existentes no dataset em numéricos:

- **OneHotEncoding:** cria uma feature binária para cada valor da feature original
- **LabelEncoder:** substitui cada valor categórico por um valor numérico

Os resultados revelaram-se semelhantes entre si.

Quanto à target *attack_type*, cada um dos valores foi transformado num valor numérico numa escala de 0 a 4.

Table 1. Class - Label

Normal - 0	Dos - 1	R2L - 2	UR2 - 3	Probe - 4

2.4 Transformação

Neste capítulo tínhamos duas opções: normalização min-max e standardização. No que toca à normalização, esta não se revelou muito eficaz pois tínhamos dados com uma escala muito diversa. Mesmo tratando dos outliers, a quantidade de valores próximos de 0 na maior parte dos atributos faz com que a normalização se torne ineficaz pois ficam valores muito baixos para compensar os mais elevados.

De igual modo, também a standardização não se mostrou muito eficaz na melhoria dos resultados.

2.5 Redução da dimensionalidade

Optamos por não reduzir a dimensionalidade através de PCA pois não consideramos necessário.

No entanto, removemos a coluna correspondente ao *Service* pois apresentava demasiados valores e, ao utilizar a função dummies, o dataset ficava com demasiadas colunas. Aliado a isto, como os dados de test e de train não contêm os mesmos valores, os conjuntos não teriam o mesmo tamanho.

Decidimos retirar também o *SampleID* pois apenas é necessário para a submissão no Kaggle e aparece unicamente no conjunto de teste.

Optamos ainda por remover o *Attack_type* quando estamos a tratar do conjunto de treino pois este atributo diz respeito à *target*.

3 Métricas de qualidade

Existem inúmeras métricas de qualidade disponíveis e prontas a utilizar de modo a avaliar a qualidade de um modelo de machine learning. A mais lógica e fácil de utilizar seria a accuracy. No entanto, esta métrica, sozinha, acarreta diversos problemas devido ao desbalanceamento dos dados. Por exemplo, se em 100 amostras tivermos 99 de X e 1 de Y e o modelo estiver programado para dizer

sempre que a amostra é do tipo X, tem uma accuracy de 99%. No entanto, o modelo é péssimo pois não tem em atenção os dados de input.

Posto isto, optamos por utilizar:

- **Accuracy:** indica a quantidade de previsões corretas
- **Precision:** indica a quantidade de previsões corretas do tipo X, tendo em conta todas as previsões que deram X
- **Recall:** responde à pergunta "Que percentagem dos casos que deviam ser X acertou?"
- **F1-Score:** responde à pergunta "Que percentagem de previsões classificadas como X são mesmo X?"
- Para a Precision, Recall e F1-score utilizamos ainda:
  - Macro Avg: média dos resultados de cada métrica
  - Weighted Avg: média dos resultados de cada métrica, tendo em conta os pesos de cada classe

Deste modo, analisando os resultados provenientes destas métricas, pensamos estar aptos para decidir se um determinado modelo teve, ou não, um bom desempenho.

4 Modelos e Experimentação

Tínhamos vários algoritmos de Machine Learning à nossa disposição para testar. Decidimos experimentar vários, com várias configurações. De seguida, segue um resumo de todos aqueles que testamos:

- Ensemble: Logistic Regression, K-Nearest Neighbours, Decision Tree, Random Forest
- Random Forest
- Balanced Random Forest
- Artificial Neural Networks

Em todos os casos, optamos por dividir o nosso dataset em 20% para conjunto de teste e os restantes 80% para treino. Defininimos o parâmetro *stratify* de acordo com a target de modo a que cada conjunto (train e test) contenha aproximadamente a mesma percentagem de amostras de cada classe tendo em conta o conjunto completo.

Sempre que um método por nós utilizado tinha o parâmetro de *random_state*, definimo-lo como sendo 42. Este é um parâmetro que indica a capacidade do método de, aleatoriamente, arranjar os dados. Definindo com 42, por mais execuções que façamos, os resultados serão sempre os mesmos.

4.1 Ensemble

O Ensemble é um método de Machine Learning que combina diversos algoritmos para se atingir uma melhor predição. Basicamente, através da junção de todos os resultados, entende-se que os erros individuais são minimizados. O Ensemble é, assim, considerado uma técnica poderosa de Machine Learning.

Com isto em vista, decidimos implementar um ensemble paralelo (Voting Classifier), isto é, todos os algoritmos são executados sem interferência uns dos outros e só no fim se combinam os resultados.

O Voting suporta dois tipos:

- **Hard:** a classe de output prevista é a classe com a maioria dos votos
- **Soft:** a classe de output é a previsão baseada na média das probabilidades dadas a essa classe

Optamos pelo método soft. Utilizamos ainda a técnica *GridSearchCV* para podermos determinar o melhor modelo. Esta escolhe de entre todos os modelos testados no ensemble e o próprio voting. Parametrizamos o cross-validation a 5 (default) e o refit (métrica a ter em conta) como sendo a accuracy.

Começamos por testar o Ensemble com os dados quase originais. Apenas substituímos a target por valores numéricos (escala de 0 a 4), retiramos a coluna Service e alteramos todas as variáveis categóricas utilizando o One Hot Encoding. Todos os algoritmos estavam em modo default, à exceção do Logistic Regression que estava com o máximo de iterações fixado em 1000. Com isto, os resultados foram os seguintes:

Table 2. Quality metrics dos Classification Reports dos algoritmos presentes no Ensemble

	LR			KNN			DT			RF			Vt			
Class	Pr	Re	F1	Pr	Re	F1	Pr	Re	F1	Pr	Re	F1	Pr	Re	F1	Support
0	0.83	0.90	0.86	0.99	0.99	0.99	0.99	0.99	0.99	0.99	1.00	0.99	0.99	1.00	0.99	10791
1	0.75	0.90	0.82	0.99	1.00	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	7461
2	0.00	0.00	0.00	0.89	0.91	0.90	0.91	0.94	0.93	0.96	0.93	0.94	0.97	0.93	0.95	537
3	0.00	0.00	0.00	0.33	0.12	0.17	0.57	0.47	0.52	1.00	0.29	0.45	0.75	0.35	0.48	17
4	0.52	0.06	0.10	0.96	0.93	0.94	0.99	0.99	0.99	0.99	1.00	0.99	0.99	0.99	0.99	1987
Acc			0.80			0.98			0.99			0.99			0.99	20793
M	0.42	0.37	0.36	0.83	0.79	0.80	0.89	0.88	0.88	0.99	0.84	0.88	0.94	0.85	0.88	20793
W	0.75	0.80	0.75	0.98	0.98	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	20793

Legenda: LR - Logistic Regression / KNN - K-Nearest Neighbours / DT - Decision Tree / RF - Random Forest / Vt - Voting / Pr - Precision / Re - Recall / F1 - F1-score / Acc - Accuracy / M - Macro Avg / W - Weighted Avg

Analisando os resultados, percebemos claramente o que já tínhamos reparado anteriormente: a quantidade de ataques do tipo 3 é muito inferior aos restantes, pelo que que o dataset não é balanceado.

O melhor algoritmo foi, evidentemente, o proveniente do Voting (combinação de todos os modelos). No entanto, se nos singirmos apenas aos modelos individuais, concluímos que os melhores algoritmos foram a Decision Tree e o Random Forest: tanto nas métricas individuais de cada classe como nas gerais foram os que apresentaram melhores valores.

Após um trabalho de pesquisa, e ao perceber que um dos grandes problemas recaía no desbalanceamento do dataset, encontramos um método de Random Forest que balanceia os dados. Deste modo, optamos por experimentar mais este modelo.

Antes de o fazer, decidimos ainda analisar o desempenho deste ensemble standardizando os dados, utilizando LabelEncoder em vez do One Hot Encoding e removendo colunas que consideramos serem redundantes. No entanto, os resultados não evoluiram.

4.2 Random Forest

Começamos por correr o Random Forest com os dados quase originais. Apenas trocamos a target por valores numéricos (escala de 0 a 4), retiramos a coluna Service e trocamos todas as variáveis categóricas utilizando o One Hot Encoding.

Na função de Random Forest, foram utilizados como parâmetros o *n_estimators* (a 150), que significa o número de árvores presentes no modelo.

Pelos resultados da tabela 3, percebemos que o algoritmo continuou a ter um ótimo desempenho. Apenas na classe 3 não obteve tão bons resultados, tendo apenas acertado em 35% daqueles que devia.

Tal como já tínhamos detetado na secção anterior, o problema passava sobretudo pelo desbalanceamento dos dados. Com isto, decidimos experimentar o algoritmo Balanced Random Forest. Este é um método semelhante ao Random Forest só que utiliza porções de igual quantidade de todas as classes, o que acaba por balancear os dados.

Pelos resultados, fica evidente que este método não se revelou tão eficaz como o esperado. Podemos explicar isto uma vez que o algoritmo Balanced Random Forest utiliza no máximo, em cada classe, o número de amostras da classe minoritária, o que reduz, neste caso, o dataset significativamente.

Posto isto, optamos por experimentar Random Forest com outras técnicas de balanceamento dos dados, como o SMOTE.

Vendo a tabela 3, percebemos que o método se revelou melhor que qualquer outro já testado anteriormente. As métricas foram em tudo melhores, inclusive, na questão da classe 3.

Table 3. Quality metrics dos Classification Reports de Random Forest, Random Forest com Smote e Balanced Random Forest

	RF			SRF			BRF			
Class	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Support
0	0.99	1.00	0.99	1.00	0.99	1.00	0.98	0.87	0.92	10791
1	1.00	1.00	1.00	1.00	1.00	1.00	0.99	0.96	0.97	7461
2	0.96	0.93	0.94	0.93	0.99	0.96	0.36	0.95	0.52	537
3	1.00	0.35	0.52	0.85	1.00	0.92	0.04	1.00	0.08	17
4	0.99	0.99	0.99	0.99	1.00	1.00	0.90	0.96	0.93	1987
Accuracy			0.99			1.00			0.91	20793
Macro Avg	0.99	0.85	0.89	0.95	1.00	0.97	0.65	0.95	0.69	20793
Weighted Avg	0.99	0.99	0.99	1.00	1.00	1.00	0.96	0.91	0.93	20793

4.3 Artificial Neural Networks

Uma rede neuronal é um método de machine learning que tem como objetivo dotar máquinas da capacidade de aprender de forma semelhante aos humanos. É um processo de deep learning que utiliza nós interligados entre si, semelhante ao cérebro humano.

O ANN tem dois parâmetros em que a sua configuração é crucial: batch e epoch. Por conseguinte, de forma a encontrar a melhor combinação de valores nestes dois parâmetros, experimentamos com o valor default (32), com o 5, o 10, o 15 e com o 20. Os melhores resultados foram obtidos com batch 32 e epoch 20. Estes foram os valores utilizados em todas as experiências seguintes.

Em todas as experiências, utilizamos 3 camadas na rede neuronal (sendo a última de output). Cada uma tinha 32, 16 e 5 neurónios, respetivamente.

Foram utilizados outros parâmetros como o *input_dim=50* que diz respeito ao número de features presentes no dataset de input, o *kernel_initializer='uniform'* que representa o valor dos pesos de inicialização e o *activation='relu'* que explicita a função de ativação para o cálculo dentro de cada neurónio. No caso da camada de output, são usados o *optimizer='adam'* que ajuda a encontrar os valores ótimos de cada peso na rede, o *loss='sparse_categorical_crossentropy'* que representa a função de perda e o *activation='softmax'*.

Numa tentativa inicial de experimentar a ANN, utilizamos o SMOTE para balanceamento. No entanto, este mostrou-se pouco eficaz pois, durante a classificação, eram previstos vários ataques da classe 3 (mais de 1000). Isso não faria sentido uma vez que esta é uma classe minoritária. Decidimos então que o SMOTE não seria uma boa técnica a utilizar nesta rede neuronal.

Posto isto, decidimos experimentar ANN com os dados quase originais. Apenas trocamos a target por valores numéricos (escala de 0 a 4), retiramos a coluna Service e trocamos todas as variáveis categóricas utilizando o One Hot Encoding.

Pelos resultados (tabela 4), percebemos que este algoritmo não se revelou tão eficaz como outros que já apresentamos. O problema do desbalanceamento dos dados é evidente uma vez que as métricas da classe 3 são todas 0.

Para contornar esta situação, e visto que a técnica de SMOTE já se tinha mostrado ineficaz, surgiu a ideia de aplicarmos pesos a cada classe, isto é, um valor representativo da quantidade de cada uma. Quanto menos valores uma classe tiver, maior deve ser esse peso.

Tendo os dados exatamente como utilizamos no ANN simples, começamos por atribuir manualmente os pesos (Pesos 1). Calculamos a percentagem de valores de cada classe em relação ao total e atribuímos essas percentagens aos pesos: a classe com menos valores ficou com a percentagem mais elevada, a classe com mais valores ficou com a percentagem menor,...

Pelos resultados, percebemos que os pesos distribuídos desta forma não se revelaram muito mais eficazes que o ANN por si só. Deste modo, decidimos atribuir os pesos de forma mais automatizada (Pesos 2). Sendo que a classe com menos valores deve ser aquela com maior peso, optamos por calcular esse valor a partir da diferença da unidade e a razão entre o número de elementos da classe e o total.

Analisando os resultados da tabela abaixo percebemos que este se revelou um método mais eficaz mas continuou a não superar o Random Forest com SMOTE.

Table 4. Quality metrics dos Classification Reports de ANN, ANN com Método de Pesos 1 e ANN com método de Pesos 2

	ANN			ANN - Pesos 1			ANN - Pesos 2			
Class	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Support
0	0.96	0.99	0.97	1.00	0.45	0.62	0.94	0.98	0.96	10791
1	0.99	0.96	0.98	0.88	0.95	0.91	0.99	0.95	0.97	7461
2	0.89	0.80	0.84	0.15	0.96	0.25	0.78	0.81	0.79	537
3	0.00	0.00	0.00	0.00	0.29	0.01	0.00	0.00	0.00	17
4	0.98	0.97	0.97	0.64	0.98	0.77	0.97	0.92	0.95	1987
Accuracy			0.97			0.69			0.96	20793
Macro Avg	0.76	0.74	0.75	0.53	0.73	0.51	0.74	0.73	0.73	20793
Weighted Avg	0.97	0.97	0.97	0.90	0.69	0.73	0.96	0.96	0.96	20793

4.4 Melhor modelo

Tendo percebido que o melhor modelo experimentando era Random Forest com SMOTE, tentamos melhorar as suas métricas de qualidade, realizando outro tipo de feature engineering nos dados de entrada.

Experimentamos substituir os outliers pela média dos valores da coluna em questão, removê-los, standardizar os dados e ainda substituir o One Hot Encoding pelo Label Encoder. Todas estas experiências foram realizadas em conjunto e em separado. No entanto, nenhuma apresentou melhores resultados do que os já revelados anteriormente.

5 Conclusão

Numa sociedade movida essencialmente pelas tecnologias, é importante que estas estejam sempre disponíveis. Por esta razão, os ataques a que são alvo devem ser rapidamente combatidos. A Inteligência Artificial tem um papel fundamental nessa matéria, sobretudo os algoritmos de Machine Learning, uma vez que dados certos parâmetros, consegue identificar o tipo de ataque a que uma tecnologia está a ser alvo. Conseguindo identificar o tipo de ataque, torna-se mais fácil combatê-lo.

Este trabalho foi desenvolvido com o intuito de encontrar o melhor modelo de machine learning para classificar os tipos de ataque dada informação de input. Após várias experiências realizadas e tendo em conta o dataset que nos foi fornecido, aquele que revelou ser mais eficaz foi o Random Forest aplicado juntamente com SMOTE.

References

1. How to use Artificial Neural Networks for classification in python?, https://thinkingneuron.com/how-to-use-artificial-neural-networks-for-classification-in-python/, Acedido a 6 Abr 2023
2. Keras documentation: Probabilistic losses, https://keras.io/api/losses/probabilistic_losses/, Acedido a 6 Abr 2023
3. NumPy documentation — NumPy v1.24 Manual, https://numpy.org/doc/stable/index.html, Acedido a 6 Abr 2023
4. PhD, N. D. (2016, 4 de fevereiro), Ensemble Methods: Elegant Techniques to Produce Improved Machine Learning Results https://www.toptal.com/machine-learning/ensemble-methods-machine-learning, Acedido a 6 Abr 2023
5. Nelson, D. (2020, 22 de janeiro), Ensemble/Voting Classification in Python with Scikit-Lear, https://stackabuse.com/ensemble-voting-classification-in-python-with-scikit-learn/, Acedido a 6 Abr 2023
6. A Comprehensive Guide to Ensemble Learning Methods, https://www.projectpro.io/article/a-comprehensive-guide-to-ensemble-learning-methods/432, Acedido a 6 Abr 2023
7. Korstanje, J. (2021, 29 de agosto), https://towardsdatascience.com/smote-fdce2f605729, Acedido a 6 Abr 2023
8. How to Handle Imbalanced Classes in Machine Learning, https://elitedatascience.com/imbalanced-classes, Acedido a 6 Abr 2023
9. Huh, K. (2021, 13 de fevereiro), Surviving In a Random Forest with Imbalanced Datasets, https://medium.com/sfu-cspmp/surviving-in-a-random-forest-with-imbalanced-datasets-b98b963d52eb, Acedido a 6 Abr 2023
10. TensorFlow, https://www.tensorflow.org/?hl=pt-br, Acedido a 6 Abr 2023
11. Keras documentation: The Sequential model, https://keras.io/guides/sequential_model/, Acedido a 6 Abr 2023
12. Huilgol, P. (2019, 24 de agosto), Accuracy vs. F1-Score, https://medium.com/analytics-vidhya/accuracy-vs-f1-score-6258237beca2, Acedido a 6 Abr 2023
13. Understanding the Classification report through sklearn – Muthukrishnan, https://muthu.co/understanding-the-classification-report-in-sklearn/, Acedido a 6 Abr 2023
14. How to Interpret the Classification Report in sklearn, https://www.statology.org/sklearn-classification-report/, Acedido a 6 Abr 2023
15. Pramoditha, R. (2022, 30 de abril), Why do we set a random state in machine learning models?, https://towardsdatascience.com/why-do-we-set-a-random-state-in-machine-learning-models-bb2dc68d8431, Acedido a 6 Abr 2023
16. Chatterjee, S. (2018, 26 de maio), Deep learning unbalanced training data?Solve it like this, https://towardsdatascience.com/deep-learning-unbalanced-training-data-solve-it-like-this-6c528e9efea6, Acedido a 6 Abr 2023
17. Korstanje, J. (2019, 13 de dezembro), 6 ways to test for a Normal Distribution - which one to use?, https://towardsdatascience.com/6-ways-to-test-for-a-normal-distribution-which-one-to-use-9dcf47d8fa93, Acedido a 6 Abr 2023
18. 7 Techniques to Handle Imbalanced Data - KDnuggets, https://www.kdnuggets.com/2017/06/7-techniques-handle-imbalanced-data.html, Acedido a 6 Abr 2023

19. How to Develop Voting Ensembles With Python - MachineLearningMastery.com, https://machinelearningmastery.com/voting-ensembles-with-python/, Acedido a 6 Abr 2023
20. Micro, Macro & Weighted Averages of F1 Score, Clearly Explained, https://towardsdatascience.com/micro-macro-weighted-averages-of-f1-score-clearly-explained-b603420b292f, Acedido a 6 Abr 2023
21. ML — Voting Classifier using Sklearn - GeeksforGeeks, https://www.geeksforgeeks.org/ml-voting-classifier-using-sklearn/, Acedido a 6 Abr 2023