Processamento de Big Data

Microsoft Malware Prediction

9 DE ABRIL DE 2023

Licenciatura em Ciência de Dados
Turma CDB2
Grupo 26
Francisco Rodrigues 105427
Simão Fonseca 105251

iscte INSTITUTO UNIVERSITÁRIO DE LISBOA

Índice

Business Understanding

Hoje em dia, com a rápida evolução das tecnologias, há um uso muito maior de equipamentos eletrónicos e ligados em rede sempre a produzir dados. Devido a isso existe também uma maior preocupação com a segurança dos dispositivos e dos dados.
Uma vez que um computador é infetado com um malware, os responsáveis desse malware podem causar muitos danos não tanto a indivíduos, mas sim a grandes empresas.
Com mais de um bilião de clientes (entre eles grandes empresas), a Microsoft investe tempo e dinheiro a tentar prevenir e resolver este problema da segurança.
Como uma das estratégias de combate, a Microsoft fez um desafio à comunidade de Data Science em 2019, para prever se um dispositivo seria brevemente infetado com malware. Para isso, organizou um concurso no Kaggle onde forneceu uma base de dados muito vasta e completa e encorajou a comunidade de Data Science para contruir um modelo de previsão de ocorrências de malware em equipamentos.

A competição e os dados encontram-se no seguinte link:
https://www.kaggle.com/competitions/microsoft-malware-prediction/overview

Este projeto consiste em prever a probabilidade de um dispositivo Windows estar infetado por um malware, baseado em diferentes propriedades desse dispositivo.
Estas propriedades e também as infeções dos dispositivos foram geradas pela combinação do heartbeat (programa que corre determinados scripts automaticamente quando se inicia ou reinicia o sistema) e relatórios de ameaças recolhidos pelo sistema e proteção do Windows, o *Windows Defender*. De acordo com a Microsoft e com o Kaggle, metodologia usada para criar este *dataset* foi desenhada para ir ao encontro de várias restrições impostas pelas empresas, não só em relação a questões de privacidade, mas também aos períodos em que o dispositivo estava ativo. Adicionalmente, estes dados não são representativos dos dispositivos de clientes da Microsoft, uma vez que a amostra pretende incluir uma proporção muito maior de equipamentos com malware.

Os dados utilizados no projeto foram obtidos a partir do ficheiro ***train.csv*** presente no link seguinte:

https://www.kaggle.com/competitions/microsoft-malware-prediction/data?select=test.csv

Uma vez que estes dados estão incluídos numa competição do Kaggle, é necessário iniciar sessão para ter acesso aos mesmos. Depois de ter a sessão iniciada, basta só selecionar o ficheiro test.csv e dar download aos dados.

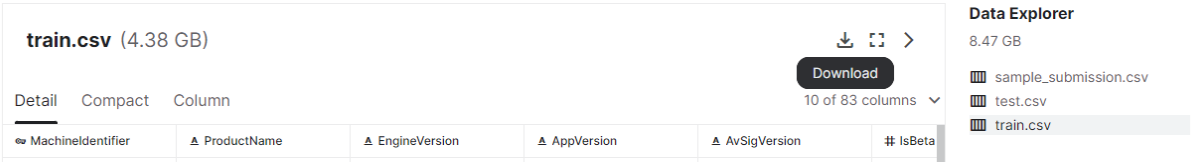

Figura 1 - Localização dos dados no website e como os descarregar

A base de dados tem 83 variáveis. Para observar as variáveis e os seus respetivos significados, o Notebook 1 e o site do Kaggle têm essas informações (devido ao elevado número de variáveis, não é possível colocá-las todas neste relatório).

As decisões mais relevantes tomadas foram: a exclusão de algumas colunas devido ao número de omissos e a imputação de omissos em outras colunas.

Na primeira decisão, foram apagadas todas as colunas que teriam mais de 10% de omissos na sua constituição. Esta decisão permitiu a exclusão de algumas colunas que não tinham informação relevante a acrescentar ao modelo.

Durante este processo de eliminação das variáveis com mais de 10% de omissos, foram eliminadas 9 colunas da base de dados.

Outra tomada de decisão que se revelou central no projeto, foi a imputação dos valores omissos. Esta decisão revelou-se importante, uma vez que foi possível usar variáveis que tinham alguns omissos que de outra forma não seria possível.

Lembrar que esta imputação foi realizada após a eliminação das 9 colunas acima referidas e que esta imputação não foi igual em todas as colunas da base de dados.

Em variáveis categóricas numéricas os valores foram substituídos pela moda.

Em variáveis quantitativas existiram dois critérios para a imputação: caso essa coluna apresentasse uma assimetria ou para a esquerda ou para a direita os valores omissos seriam substituídos pela mediana; caso não houvesse assimetria nas colunas, os valores omissos seriam substituídos pela média.

Ao longo deste projeto foram realizados vários testes: ora pelo aumento/redução do número de variáveis preditoras usadas, ora pelos modelos de classificação.

Em relação ao uso das variáveis preditoras, os primeiros modelos desenvolvidos contaram com o uso de 19 variáveis preditoras. Estas 19 variáveis consistiam apenas nas variáveis que não tinham valores omissos; o objetivo seria perceber o quão dependente os modelos estariam de outras variáveis (que tinham valores omissos) e também tentar perceber quão afetado sairia o modelo, caso se usassem apenas as variáveis completas (variáveis sem nenhuns omissos).

O segundo teste foi o uso de 34 variáveis preditoras. Aqui usou-se algumas das variáveis que tinham sido “vitimas” da imputação dos omissos, anteriormente realizada. Nesta fase evitou-se o uso de variáveis que poderiam estar um pouco correlacionadas.

Por fim o último teste, foi o uso de 41 variáveis preditoras. Aqui usou-se o máximo possível de variáveis preditoras e apenas, retiraram-se aquelas que estavam demasiado correlacionadas entre si. Esta foi a tentativa final, uma vez que foi este número de variáveis preditoras que os modelos finais usaram.

Em relação aos modelos, não existiu nada de relevante para apontar. A eficácia dos modelos quase não foi alterada com o aumento do número de variáveis. O único ponto que mudou foi o uso do modelo Naive Bayes.

Ou seja, quando os modelos ainda estavam a ser construídos com 19 variáveis, foi construído um modelo de Naive Bayes. Mas a partir do momento, em que se passou de 19 para 34, já não foi possível usar esse modelo, dado que, já existiam imensas variáveis preditoras e o modelo de Naive Bayes não consegue trabalhar com um número tão grande de variáveis.

Ambientes de Desenvolvimento

Para este projeto foram utilizados dois ambientes de desenvolvimento diferentes. Um deles foi o Ubuntu, que está instalado localmente no computador; e o outro foi a Amazon Web Services (AWS), que é um ambiente em *cloud*.

O Ubuntu é um sistema operacional open-source baseado no Debian, que é projetado para ser fácil de usar e configurar. É bastante utilizado em servidores e desktops, sendo conhecido pela sua segurança e estabilidade. O Ubuntu é gratuito para uso pessoal e comercial e é instalado localmente nos dispositivos.

Por outro lado, o Amazon Web Services (AWS) é uma plataforma de desenvolvimento em nuvem que oferece uma ampla gama de serviços, incluindo armazenamento, computação, bases de dados, rede, análise, inteligência artificial e muito mais. É um dos 3 maiores fornecedores de serviços de *cloud* em todo o mundo, oferecendo soluções escaláveis, seguras e confiáveis para empresas. O AWS cobra pelos serviços utilizados, mas com o acesso à AWS Academy (fornecido pelos docentes) foi possível utilizar algumas das suas capacidades.

Enquanto o Ubuntu é um sistema operacional, a AWS é uma plataforma de computação em *cloud*. Isso significa que o Ubuntu pode ser executado em servidores físicos ou virtuais, enquanto a AWS é executada numa infraestrutura em *cloud*. Além disso, o Ubuntu é um software open-source e gratuito, enquanto a AWS é uma solução comercial.

O projeto foi desenvolvido no Jupyter Lab. Ambos os ambientes permitem utilizar o Jupyter Lab e ambos os ambientes suportam também o pyspark, por isso foi possível comparar utilizando o mesmo código e até o mesmo ficheiro. O código foi então “corrido” e comparado em ambos os ambientes de desenvolvimento. De seguida, na análise e comparação dos resultados, irão também ser comparados os tempos de resposta de ambos os ambientes.

Análise e comparação de resultados

Para analisar melhor os modelos e os seus resultados, o segundo notebook (notebook das previsões) foi "corrido" em diferentes plataformas e com bases de dados de diferentes dimensões. Na tabela seguinte é possível observar então os resultados destes testes:

		20% dos dados		**100% dos dados**
		Jupyter Ubuntu	Jupyter AWS	Jupyter AWS
Random Forest	tempo de criação/treino	22,5 s	33,6 s	206,9 s
	eficácia	59,1%	59,0%	59,0%
	precisão	60,3%	60,5%	60,1%
Regressão Logística	tempo de criação/treino	30,1 s	45,2 s	218,4 s
	eficácia	59,1%	59,0%	59,0%
	precisão	60,9%	60,6%	60,6%
SVM	tempo de criação/treino	13,3 s	28,0 s	147,8 s
	eficácia	58,3%	58,1%	58,2%
	precisão	62,1%	61,7%	61,9%

Comparação dos Modelos

Numa analise inicial pode notar-se que dos 3 modelos, todos têm aproximadamente os valores de eficácia e precisão semelhantes. Podem ser distinguidos bastante pelo tempo de criação e treino do modelo. É possível observar então que o modelo mais rápido é o do método SVM, de seguida o da Regressão Logística, e o mais lento destes 3, o método de Random Forest.
Dos 3 modelos testados, tanto o modelo de Random Forest como o de Regressão Logística obtiveram o melhor valor de eficácia, com 59,1%. Já de precisão, o modelo com o melhor valor foi o SVM com uma precisão de 62,1%.

Comparação das Plataformas

Uma vez que neste projeto foi fornecido o acesso a uma plataforma de gestão de grandes volumes de dados, a AWS (Amazon Web Services), foi possível, não só correr os modelos noutro tipo de plataforma, mas também poder correr os modelos sobre uma base de dados muito maior que não seria suportada localmente nos nossos laptops.
Comparando as diferenças nos valores obtidos nas diferentes plataformas com a mesma quantidade de dados, é visível que os valores de eficácia e de precisão são bastante semelhantes. O que é bastante diferente são os tempos de criação/treino dos modelos. Constata-se que dentro do AWS, esses tempos são mais elevados do que localmente utilizando o Ubuntu.
Esta diferença pode-se dever a diferentes razões:
- o tamanho e a configuração da instance EC2 a que o notebook está associado;

- a conexão de rede entre a instance e o computador local;
- a complexidade do código;

Destas 3 razões anteriores, não faria sentido ser a terceira uma vez que, mesmo que o código seja muito complexo, é o mesmo em ambas as plataformas. De resto, não é possível concluir nada acerca das causas. Apenas se conclui que há, de facto, uma diferença grande entre os tempos de criação/treino do modelo.

Para se conseguir criar os modelos, foi necessário reduzir a base de dados em 20%. Isto porque a criação de modelos de previsão requer muita capacidade de memória e processamento do computador. Assim, à medida que se utilizou a *cloud* AWS, foi possível observar também que o código poderia ser adaptado de maneira a "correr" os modelos com a base de dados completa, e assim comparar como se comportavam os modelos e quais seriam os resultados obtidos.

Observando então os resultados deste teste final, é possível concluir que não compensa usar a base de dados na sua totalidade. Não só não trouxe vantagens em relação aos resultados da eficácia e precisão dos modelos (os valores são praticamente iguais) como o tempo de criação e teste dos modelos foi nitidamente maior. Estes resultados já eram expectáveis.

Era previsível que os tempos fossem mais elevados, uma vez que se trata de uma base de dados com 5 vezes mais dados do que as testadas anteriormente. Ainda assim foi interessante testar os modelos com a base de dados completa, para assim se observar que, os 20% utilizados anteriormente, estavam bem selecionados (uma vez que os resultados foram muito semelhantes).

Trade-off

Além disso, outro aspeto importante que deve ser considerado é o ***trade-off*** entre a precisão e velocidade de execução. Por vezes é possível que o modelo que apresentou melhores métricas de avaliação leve mais tempo para ser executado, o que pode ser pouco prático em algumas situações. Em alguns casos, em que o modelo de previsão tem de ser usado em tempo real, a velocidade de execução pode ser mais importante do que a precisão do modelo.

Neste caso específico, tento em conta o contexto do problema, apesar de ambos serem importantes, é mais importante valorizar a precisão e não tanto a velocidade de execução. Independentemente da situação, neste caso, é interessante observar que o modelo com maior precisão é também o modelo mais rápido. Assim conclui-se que, dos três modelos, o melhor é o SVM.

Notas referentes aos Notebooks

Para uma melhor organização, foram criados 4 notebooks.

O ficheiro ***Grupo26_Notebook1.ipynb*** é o notebook inicial. Criado e "corrido" no Ubuntu, este contém os passos iniciais do ML Pipeline, que inclui *"Problem formulation", "Collect and label data", "Evaluate data"* e *"Feature engineering"*. Neste notebook foram executados todos os passos até à criação dos dois ficheiros parquet. Um dos ficheiros parquet com 20% da base de dados (que foi utilizado para a criação e teste dos modelos) e outro parquet com 100% da base de dados (importante manter e também utilizado para alguns testes).

O ficheiro ***Grupo26_Notebook2.ipynb*** é o notebook relativo à criação, treino e teste dos modelos. Este notebook foi também "corrido" no Ubuntu, sendo o seu output relativo ao do computador local.

O ficheiro ***AWS_Grupo26_Notebook2.ipynb*** é um notebook igual ao anterior. A única diferença é o ambiente onde o mesmo é executado. Apesar do código ser igual, o output vai ser diferente uma vez que este é "corrido" na *cloud* AWS.

Por último, o ficheiro ***AWS_Grupo26_Notebook2_Total.ipynb*** é um notebook semelhante aos dois anteriores, onde consta o código referente aos modelos. A única diferença é que, em vez de se importar o parquet correspondente a 20% da base de dados, é importado o parquet relativo a 100% dos dados. Este ficheiro foi "corrido" na *cloud* AWS, por isso os outputs são referentes a esse ambiente.