Aprendizado de Máquina no Contexto de Detecção de Phishing em Blogs

Vinícius Brandão Crepschi
Graduação em Ciência da Computação
UFSCar
Sorocaba - Brasil
vibracre@gmail.com

***Resumo*—Este trabalho apresenta um desafio de aprendizado de máquina aplicado ao contexto de detecção de fraudes em blogs. O problema conhecido como *phishing* é uma forma de fraude no meio virtual que visa obter informações pessoais dos usuários através de sites e conteúdo falsos. Com o objetivo de distinguir blogs com conteúdo fraudulento de blogs comuns, serão utilizadas os principais modelos de aprendizado de máquina e, também técnicas de pré-processamento e otimização.**

***Keywords*-classificação; phishing; spam;**

I. Introdução

Para um serviço de hospedagem de sites e blogs, é de suma importância a detecção e prevenção de fraudes. Uma empresa que presta este serviço deve garantir que os sites hospedados não distribuem conteúdo fraudulento ou enganoso. É possível detectar este tipo de conteúdo através de elementos HTML da página, conteúdo textual, ou, através dos comentários nas publicações. Ao longo deste artigo serão exploradas técnicas de aprendizado de máquina a fim de encontrar, em um conjunto de dados, amostras que representam fraudes.

II. Dados e Pré-processamento

Foram fornecidos três bases de dados, com quantidade de atributos diferente e não identificados. A única informação fornecida sobre a base é quanto a sua origem. Pra fins de conveniência, estes conjuntos serão chamados *set1*, *set2* e *set3*, sendo eles:

- set1: Extraído de conteúdo textual dos blogs;
- set2: Extraído a partir da estrutura de relacionamento (por meio de links) do Blog alvo com outros Blogs hospedados;
- set3: Extraído a partir do código HTML do Blog.

Diante dos conjuntos de dados, foram feitas análises as seguintes análises para observar e explorar os conjuntos de dados fornecidos:

- Gráficos de Boxplot para observar a dispersão dos dados;
- Contagem de valores NaN (set1: 2360, set2: 1848, set3: 1576);
- Observar as primeiras amostras
- Exibir os dados estatísticos por coluna, tais como: Média, desvio padrão, valores máximos e mínimos, quartis.

A. Pré-processamento

Nesta seção serão justificadas as escolhas e processos realizadas durante a etapa de pré-processamento dos dados.

1) Montagem dos Conjuntos de Treino: A primeira etapa executada foi o carregamento da base de dados e os três arquivos *.csv* com os índices correspondentes aos três conjuntos supracitados (set1, 2 e 3). Nos três *sets* foi removida a coluna *Id* e também adicionada uma coluna *Class* com os rótulos fornecidos. Além disso, foram removidas as amostras rotuladas como 0, incertas se o conteúdo possui ou não *phishing*. Os rótulos possuem o seguinte significado:

- 1: A amostra contém conteúdo *phishing*;
- 0: O especialista não teve certeza se contém *phishing* ou não;
- -1: A amostra não contém conteúdo *phishing*;

2) Valores NaN: Após a formatação dos conjuntos a serem utilizados, foram corrigidas as amostras com valores indefinidos (NaN). A técnica utilizada foi substituir estes valores pela média dos valores daquela coluna. Em seguida foi realizada a normalização dos três *sets* por atributo. Por fim, exibimos os dados estatísticos com as bases agora normalizadas.

3) Análise PCA: Em seguida foi feita uma Análise de Componentes Principais, reduzindo a dimensionalidade dos conjuntos para 2 dimensões, de forma que foi possível plotar gráficos de dispersão e observar os dados dispostos em um plano bidimensional. Podemos observar estes gráficos a seguir nas figuras 1, 2 e 3.

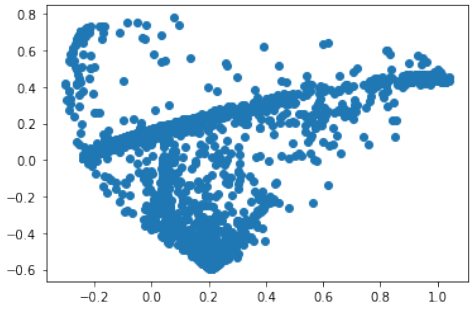

Figura 1. set1 PCA 2D

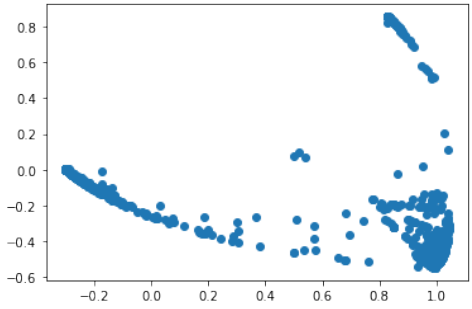

Figura 2. set2 PCA 2D

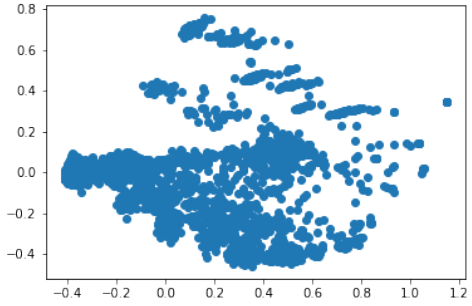

Figura 3. set3 PCA 2D

4) Feature Selection: Com os *sets* normalizados, foi executado um algoritmo de seleção de *features* com objetivo de identificar e remover atributos que não possuíam influência na classificação, ou seja grau de impacto = 0. Para cada conjunto foram removidos a seguinte quantidade de atributos:

- set1: Removidos 12 atributos;
- set2: Removidos 0 atributos;
- set3: Removido 1 atributo.

5) Distribuição das Classes: Foi feita uma análise sobre a distribuição das classes com o objetivo de determinar se o conjunto estava desbalanceado. Uma vez que estamos tratando do domínio de fraudes na internet, é comum que os casos de fraude sejam minoria dentre todos os blogs observados neste trabalho. A figura 4 mostra que a classe correspondente aos blogs com conteúdo *phishing* eram minoria, compondo 6,31% das bases de treino. Isto deverá ser lidado com nas etapas posteriores, onde será necessário utilizar técnicas como *Upsampling* ou o uso de parâmetros que atribuem pesos às classes de acordo com a sua distribuição.

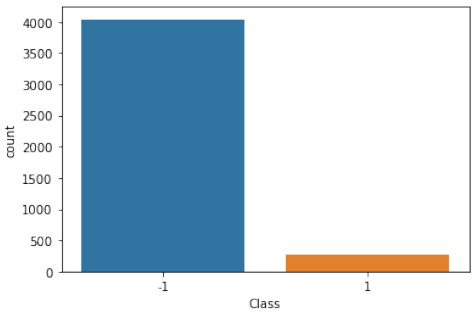

Figura 4. Distribuição de Classes

III. Protocolo Experimental

Para a realização do ajuste de parâmetros dos modelos testados, foi utilizado uma busca em grade com os principais parâmetros disponíveis. A busca em grade testa exaustivamente todas as combinações entre os parâmetros fornecidos e escolhe a melhor combinação para uma métrica específica. Uma vez que lidamos com um conjunto de dados desbalanceado, onde a maioria das amostras representam a classe 'não-phishing', a métrica acurácia não representa bem a realidade, pois os modelos ficam tendenciosos a escolher a classe majoritária e, mesmo errando a classe minoritária, a acurácia aparece alta. A métrica escolhida para otimizar com a busca em grade foi a área sob a curva ROC.

A ROC, ou Curva Característica de Operação do Receptor mostra o quão bem o modelo criado consegue distinguir entre duas classes. A curva traça a relação entre a taxa de verdadeiro-positivos e de falso-positivos. A AUC, ou *area under curve* representa o grau da medida de separabilidade entre as classes. O AUC varia de 0 até 1 e, quanto mais próximo de 1, melhor a capacidade do modelo distinguir uma classe da outra.

Além do AUC, foi avaliado o desempenho dos classificadores nas seguintes métricas:

- Acurácia - Porcentagem de amostras classificadas corretamente;
- BLU (*Blocked Legit Users*) - Porcentagem de blogs legítimos incorretamente classificados como *phishing*;
- PC (*Phishing Caught*) - Porcentagem de blogs com conteúdo *phishing* corretamente classificados como tal.
- Macro F1 - Média dos valores F1 de cada classe. O Macro F1 dará importância igual para as duas classes,

de forma que, caso o classificador só tenha boa performance em uma classe, o Macro F1 será baixo.

A. Parâmetros da busca em grade

Para cada modelo, foram utilizados os seguintes parâmetros para o processo de ajuste via busca em grade. Com exceção do Naive Bayes, que não possui parâmetros para busca.

A busca foi realizada com Nesse caso, a busca foi realizada com um conjunto de treino correspondente a 80% dos dados e teste correspondente a 20% dos dados. Além disso, a busca em grade do Scikit realiza validação cruzada com padrão de 5 *folds*.

Para todos os modelos foi aplicado a técnica de *Upsampling*, que consiste em replicar amostras da classe minoritária a fim de balancear as classes. Naqueles modelos em que estava disponível a aplicação de pesos para as classes de acordo com sua distribuição, este também foi usado.

1) k-Vizinhos Próximos: Para o algoritmo de k-Vizinhos Próximos, foram usados os seguintes parâmetros via busca em grade:

- Número de vizinhos - [3, 5, 7, 14, 28]
- Pesos - Por distância

A busca em grade pelo melhor valor de K teve como resultado 28 para os conjuntos de dados. Além disso foi empregado também o parâmetro para utilizar busca por distância.

2) Naive Bayes: Para o modelo Naive Bayes não existem parâmetros para a busca em grade. A técnica utilizada para obter melhores resultados foi o *upsampling*, que consiste em replicar amostras da classe minoritária até que se obtenha um balanceamento na base de dados.

3) Regressão Logística: Para o algoritmo de Regressão Logística, foi feito *upsampling* e foram usados os seguintes parâmetros via busca em grade:

- Constante inversa de regularização C - [500, 1000, 3000, 5000]
- Penalidade - regularização L1 ou L2

A busca em grade pelo melhor valor de C teve como resultado 5000 para os *sets* 1, 2, 3. Todos foram treinados com penalidade L2. Além disso, foi usado também um parâmetro que atribui pesos às classes de acordo com a sua distribuição (class_weight='balanced').

4) Rede Neural Artificial: Para o modelo da rede neural artificial, foram testados, com 2000 iterações, os parâmetros via busca em grade:

- Alfa de Regularização - $[10^-1, 10^-2, 10^-3]$
- Tamanho da camada oculta (nós) - $[input/3, input/2, input]$ (onde input significa o tamanho da entrada)

A busca em grade pelos parâmetros obteve como resultado, para cada conjunto, os seguintes resultados:

- set1 - Alpha = 0.01; Tamanho da camada oculta - 39
- set2 - Alpha = 0.1; Tamanho da camada oculta - 73
- set3 - Alpha = 0.1; Tamanho da camada oculta - 148

5) SVM Linear: Para o SVM Linear foram feito busca em grade para o valor de C (parâmetro de regularização) e com parâmetro de pesos para as classes como 'balanceado'. Os resultados para cada conjunto foram:

- set1 - C = 1000
- set2 - C = 1000
- set3 - C = 1000

IV. Resultados

Nesta seção serão apresentados os resultados obtidos para cada modelo utilizando o conjunto de dados que apresentou melhor desempenho. A tabela a seguir mostra, para cada modelo, as métricas de Acurácia, BLU (Blocked Legitimate Users), PC (Phishing Caught) e Macro F1. Em negrito destaca-se o modelo que obteve melhor desempenho na competição do Kaggle com score: 81.17%.

Modelo (conjunto usado)	Acurácia	BLU	PC	mF1
kNN (set2)	71	25	58.6	53
Regressão Logística (set3)	78.2	25.2	62	58
Naive Bayes (set3)	85	12	38	56
SVM Linear (set2)	74	25	62	53
Rede Neural (set2)	85	11	36	58

V. Estratégia Final

Para os envios na competição online no Kaggle, foi utilizado como estratégia final a Regressão Logística. Para isto foi treinado um novo classificador, desta vez utilizando a base completa como treinamento (set3), realizando também o *upsampling*. Através desta abordagem foi possível obter um resultado de 81.17% no placar público. Utilizando o conjunto 2 (set2) foi possível obter desempenho similar, porém inferior, de 74%.

Diversos outros modelos foram testados na competição mas a regressão foi a que rendeu o melhor resultado e, assim, foi usada como estratégia final.

VI. Conclusão

Através do trabalho realizado, foi possível sumarizar todas as técnicas aprendidas ao longo do semestre e aplicá-las num contexto real. É interessante também destacar a dificuldade de lidar com uma base de dados onde não se sabe sobre os atributos. Este fato foi de grande valia para o aprendizado, uma vez que, fez com que as análises fossem mais genéricas, visto que não havia informações semânticas que pudessem guiar ou enviesar a pesquisa.

Outra dificuldade foi lidar com o desbalanceamento das classes e ajustar os modelos para que detectassem corretamente os casos de *Phishing* sem errar demais no julgamento dos casos que não fraudulentos.

Ao final deste trabalho, é possível dizer que o curso realizado ao longo do semestre foi proveitoso e cumpriu com o propósito de ensinar as principais téncicas e modelos de aprendizado de máquina.

REFERÊNCIAS

[1] *Understanding ROC Curves with Python* disponível em: https://stackabuse.com/understanding-roc-curves-with-python/

[2] *Practical Machine Learning* Ryan A. Mardani disponível em: https://towardsdatascience.com/practical-machine-learning-tutorial-part-1-data-exploratory-analysis-c13d39b8f33b

[3] *Grid Search Explained* Ajitesh Kumar disponível em: https://vitalflux.com/grid-search-explained-python-sklearn-examples/

[4] *SciKit API Reference* disponível em: https://scikit-learn.org/stable/modules/classes.html