Senior Labs
Processo Seletivo - Pesquisador
Análise de Spam

Letícia Amanda Cechinel

17 de agosto de 2020

Resumo

Esse documento contém a descrição do projeto técnico elaborado para o processo seletivo de pesquisador para o Senior Labs. O projeto foi desenvolvido em python.

1 Introdução

O desafio proposto pela Senior consiste em uma análise de spam em mensagens. Esse é um problema bastante conhecido e documentado na ciência de dados e inteligência artificial (IA), e pode ser abordado de maneiras e utilizando modelos diferentes. Cada abordagem depende do nível de dificuldade do problema e da maneira com que os dados estão estrurados. No caso do processamento de texto, uma abordagem bem comum dentro do universo da IA é o Natural Language Processing (NLP). Seja com modelos já treinados e bibliotecas disponíveis na literatura (nltk, spacy...) ou construindo a própria arquitetura de deeplearning que geralmente abrange as redes recursivas (RNN) utilizando camadas como a LSTM. Nessa abordagem do deeplearning, temos alguns exemplos disponíveis em sites como o kaggle [1]. Outra abordagem seria utilizar a vetorização [2][3], que é uma maneira mais simples de se usar NLP e a proposta é relativamente similar ao que já estava estruturado do dataset fornecido para o desafio.

2 Metodologia

Na etapa de análise exploratória, a primeira coisa a ser feita foi identificar a coluna da saída ou rótulo (IsSpam), e verificar qualquer correlação inicial com as outras colunas. Uma primeira abordagem foi utilizar a correlação de pearson disponível na biblioteca pandas (df.corr()) para ver se havia alguma correlação linear entre contagem de palavras comuns e a label "IsSpam", entre a quantidade de palavras, e entre a razão palavras comuns/total de palavras na mensagem. Também foi plotado um gráfico de barras para ver quais colunas tinham maior correlação linear com a label. Essa parte inicial não faz parte das questões da análise e portanto não consta no aplicativo, porém está no arquivo analises_extras.ipynb.

Em seguida, já entrando nas questões propostas pelo desafio, foram concatenadas todas as mensagens em um só texto para efetuarmos a contagem de palavras que

mais ocorreram. Para maior coesão nos resultados o texto foi transformado todo para low case, e foi disponibilizada uma etapa de tratar stopwords, algumas gírias, caracteres especiais e caracteres html. Esse tratamento é opcional e é aplicado pelo usuário no aplicativo, e também está opcional na análise disponível no jupyter notebook. Essa etapa foi realizada com auxílio do nltk.

Uma outra análise solicitada é a análise mensal de quantidade de mensagens (spam e não spam), alguns dados estatísticos (média, mediana, minimo, máximo...) da contagem de palavras das mensagens no mês, e o número de mensagens comuns por dia do mês. O primeiro ponto foi usar a biblioteca datetime para transformar a coluna "Date"no formato data e criar duas novas colunas, mês e dia, para facilitar a análise. Após essa etapa, foi usada a função set do pandas para pegar uma lista de meses únicos e usar como filtro para a função loc, também da biblioteca pandas, para retornar o dataframe filtrado por mês.

Com os dataframes filtrados por mês, foram retiradas as informações solicitadas, também utilizando o próprio pandas. Na análise de quantidades de spam e mensagem comum por mês, e quantidade de mensagem comum por dia, o resultado foi montado em gráficos, já para os dados estatísticos preferiu-se mostrar o resultado por meio de um dataframe.

Na modelagem, como foi fornecido um dataset já com uma contagem de palavras importantes ou comuns nas mensagens e que possivelmente poderiam ser indicadoras de um spam ou mensagem comum, optou-se por usar as colunas de contagem como entrada do modelo e testar os resultados. Na biblioteca scikit-learn temos muitos modelos de classificação mais simples disponíveis e, dependendo de como nossa base é estruturada, podem ter resultados tão satisfatórios quanto modelos de deeplearning mais complexos, porém com maior rapidez e acessibilidade.

Há uma etapa opcional de pré processamento com label encoder e balanceamento do dataset, pois temos muitos mais mensagens comuns do que spam, o que poderia piorar o modelo. Foram feitos testes com os modelos random forest classifier, decision tree classifier, support vector classifier, kneighbors classifier, adaboost classifier.

3 Resultados

3.1 Análise exploratória

Nas primeiras medições de correlações lineares, percebeu-se que a contagem de palavras comuns e a relação entre palavras comuns sobre o total de palavras não eram muitos correlacionadas com a flag IsSpam conforme Figura 1. Também da análise de colunas com maiores correlações lineares temos o gráfico da Figura 2.

Primeiros insights:

- existe uma relacao entre o common words e o IsSpam?
- existe uma relacao entre a taxa common_words/word_count e o IsSpam?

```
In [22]: from sklearn.preprocessing import LabelEncoder
         le = LabelEncoder()
         dados['IsSpam'] = le.fit_transform(dados['IsSpam'])
         dados['Common_Word_Count'].corr(dados['IsSpam'])

Out[22]: 0.26254371135682153

In [16]: dados["ratio"] = dados['Common_Word_Count']/dados['Word_Count']
         dados['ratio'].corr(dados['IsSpam'])

Out[16]: -0.01911609633905695
```

Figura 1: Correlação linear entre label e contagem de palavras.

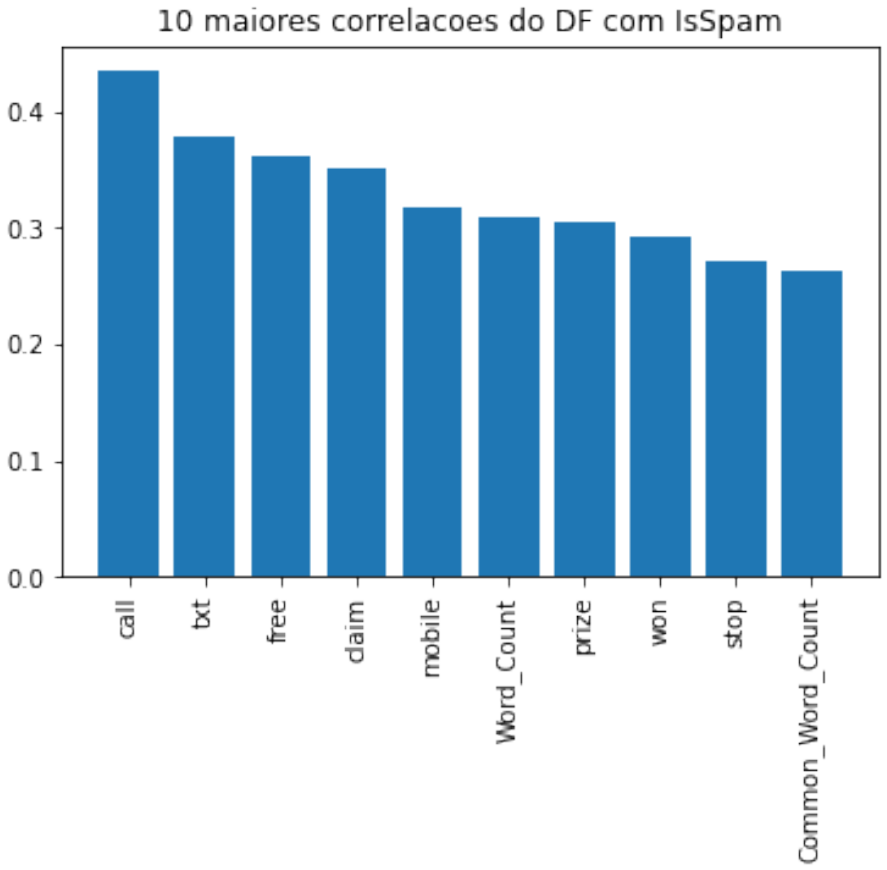

Figura 2: Gráfico de colunas de mais correlação linear com label.

Para a parte da análise efetivamente solicitada pela empresa, temos primeiramente as análises de contagem de mensagem comum e spam por mês. Foi observado que tínhamos apenas 3 meses de dados, e seus resultados são mostrados nas Figuras 3, 4 e 5.

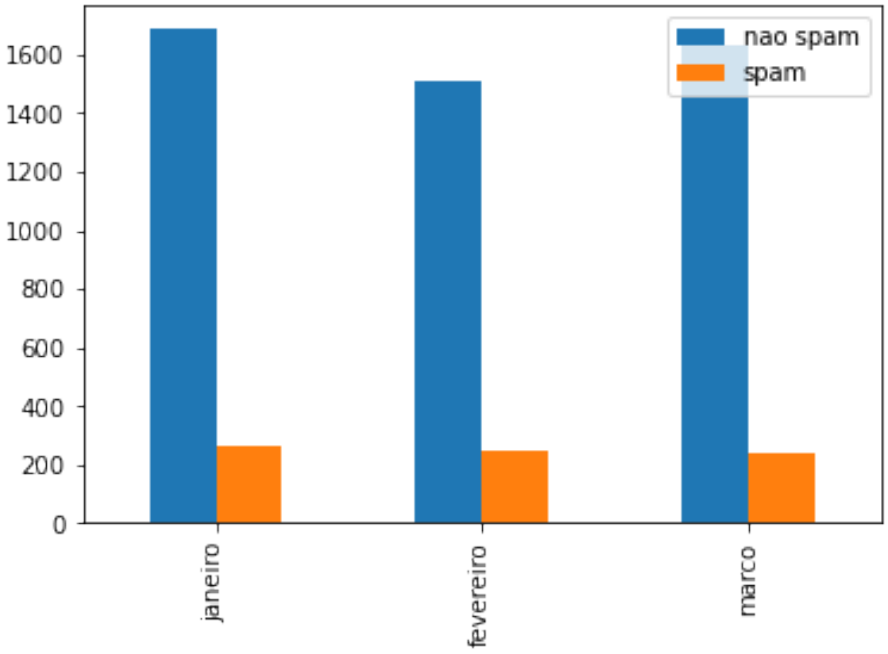

Figura 3: Contagem de mensagem comum e spam por mês.

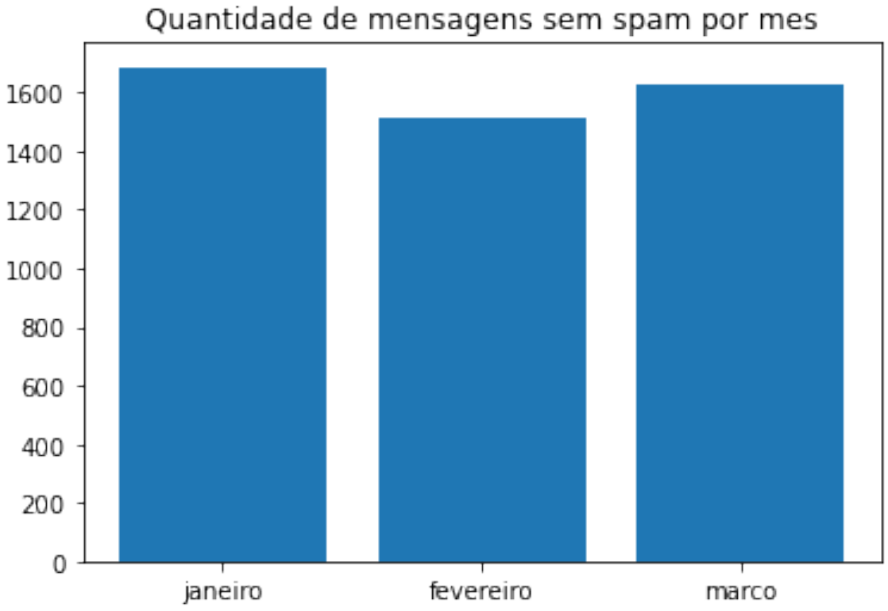

Figura 4: Contagem de apenas mensagem comum por mês.

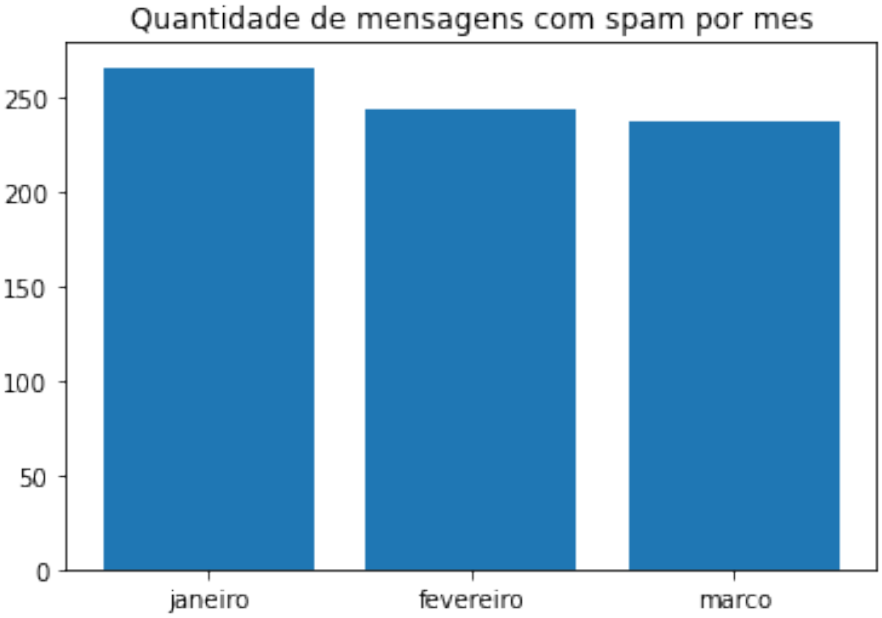

Figura 5: Contagem apenas spam por mês.

Também foi visto quando percentualmente cada mês representa do total de mensagens comuns, e do total de mensagens de spam, conforme as Figuras 6 e 7.

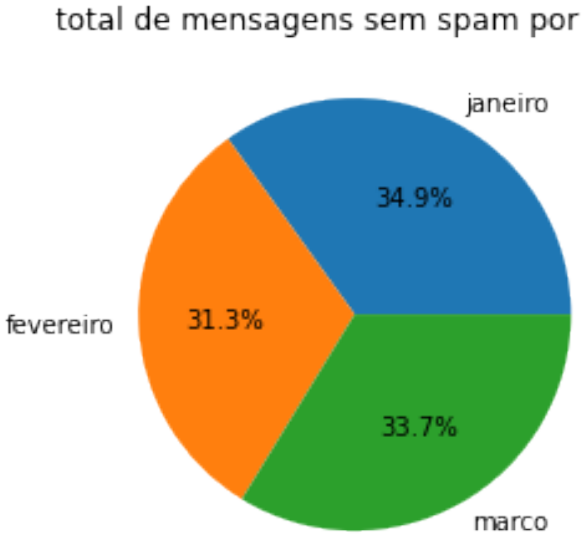

Figura 6: Percentual mensagem comum por mês.

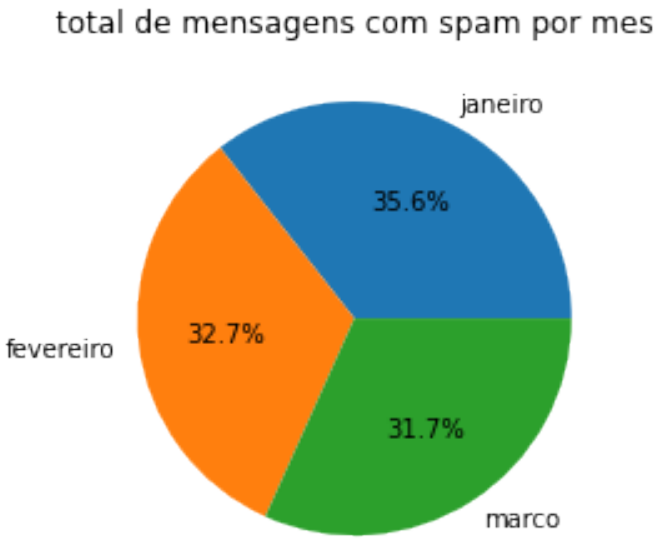

Figura 7: Percentual mensagem de spam por mês.

Na análise de dados de média, mediana, mínimo, máximo, desvio padrão e variância de contagem de palavras por mês, o resultado foi disponibilizado como um dataframe, conforme imagem da Figura 8.

	maximo de palavras	minimo palavras	media palavras	mediana palavras
fevereiro	100	2	16.0290	13
marco	115	2	16.2853	12
janeiro	190	2	16.3369	13

Figura 8: Dataframe dados estatisticos.

Finalizando a parte de análise exploratória temos os gráficos por dia do mês, mostrando a quantidade de mensagens comuns. A quantidade de spam também foi salva para os meses, podendo posteriormente ser implementada sua visualização. O exemplo de gráfico de mensagens diárias é mostrado na Figura 9 para Janeiro, porém foi implementado para cada mês. Vemos pela figura que, no caso de Janeiro, o dia com mais mensagens comuns foi o dia primeiro, também observamos que vários dias não têm mensagem comum.

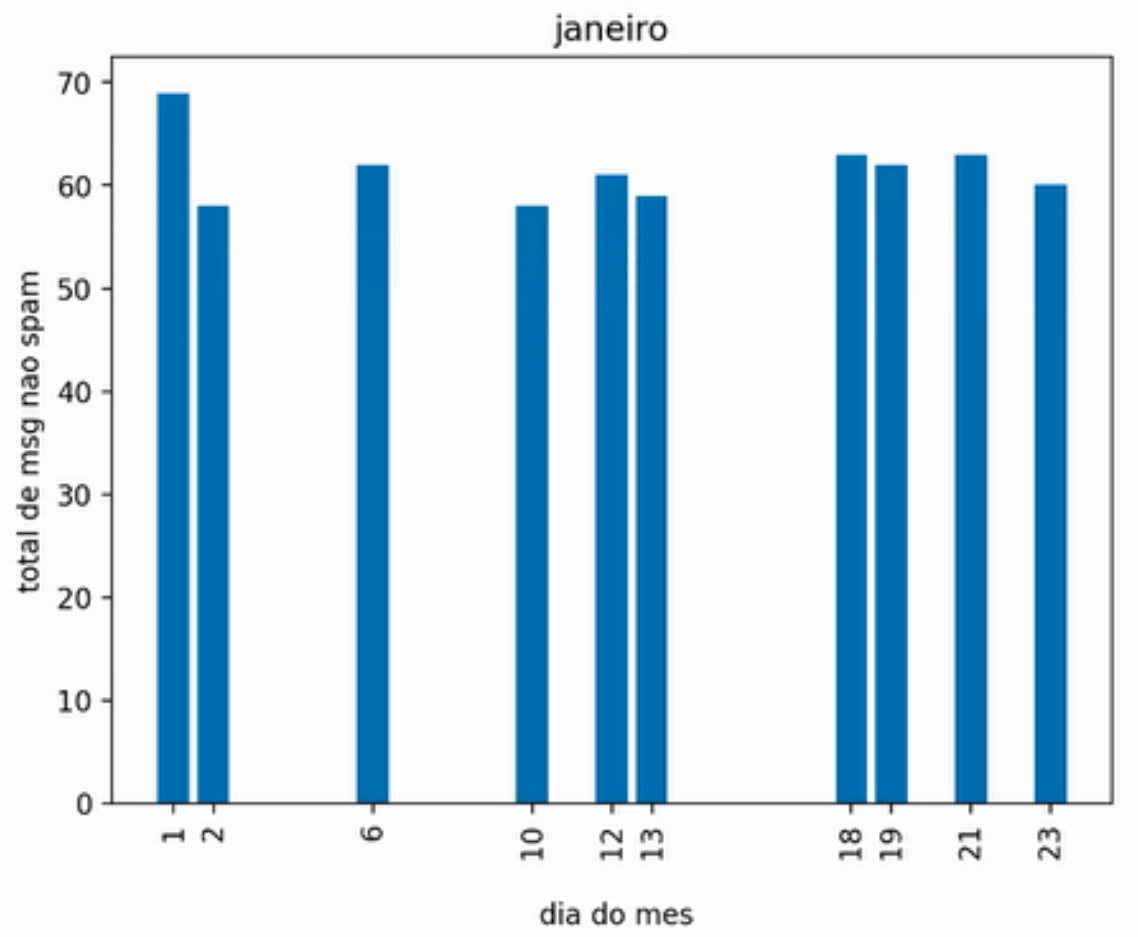

Figura 9: Mensagens comuns por dia do mês.

3.2 Modelagem

Conforme citado na metodologia, foram testados os modelos random forest classifier, decision tree classifier, support vector classifier, kneighbors classifier, adaboost classifier.

A formação dos dados consistiu da seguinte maneira: entradas como sendo todas as colunas exceto data, coluna textual e coluna de rotulagem (isSpam), e saída como a coluna isSpam após passar pelo pré processamento de label encoder.

Todos os modelos obtiveram performance acima dos 90% para o dataset original e acima de 80% para o dataset balanceado com a mesma quantidade de informações pra cada classe. Dessa maneira, também por uma questão de tempo, não se aprofundou no deeplearning, até pelos modelos mais simples apresentarem resultados promissores. Foi testada também a rede neural MLPClassifier do scikit, com diferentes funções de ativação e otimizadores. Apesar disso, o modelo se comportou semelhante aos modelos já citados anteriormente, e por ser mais complexo e demorado acabou não entrando no aplicativo final.

O melhor resultado foi de 97% para o dataset original utilizando o random forest classifier conforme mostrado na Figura 10, onde temos primeiro o classification report, em seguida o score de acerto e por fim a matriz de confusão.

```
RandomForestClassifier()

              precision    recall  f1-score   support

           0       0.97      0.99      0.98      1456
           1       0.96      0.81      0.88       217

    accuracy                           0.97      1673
   macro avg       0.96      0.90      0.93      1673
weighted avg       0.97      0.97      0.97      1673


0.9701135684399282

[[1448    8]
[  42  175]]
```

Figura 10: Resultado random forest classifier.

4 Conclusão

Como conclusão desse teste, gostaria de ponderar que os resultados para classificação parecem bastante satisfatórios. Com maior tempo dedicado a exploração de novos modelos (especialmente RNN) e nova formatação da base de dados, poderíamos aumentar ainda mais alguns pontos no score final. Uma opção seria criando novas features, utilizando ferramentas como vetorização com Tf-idf ou outras técnicas, e com testes de ferramentas de PCA e outras análises.

Percebe-se que, ao balancear, o resultado cai em mais de 5 pontos percentuais em sua precisão, e essa análise e posterior correção também poderia gerar uma melhoria.

Ainda assim, os resultados parecem um bom filtro de spam, e acredito que a análise exploratória tenha trazido bastante informações sobre a base de dados, ficando ainda mais interativa com a aplicação visual, que, embora não seja o foco da área de pesquisa, facilita bastante o acesso aos resultados e também ajuda no embasamento e na defesa de ideias ao apresentar para pessoas de dentro e fora da equipe.

Referências

[1] SIMPLE LSTM for text classification. Disponível em: <https://www.kaggle.com/kredy10/simple-lstm-for-text-classification>.

[2] NATURAL Language Processing (NLP) for Beginners. Disponível em: <https://www.kaggle.com/faressayah/natural-language-processing-nlp-for-beginners>.

[3] HOW To Design A Spam Filtering System with Machine Learning Algorithm. Disponível em: <https://towardsdatascience.com/email-spam-detection-1-2-b0e06a5c0472>.