**Discorrer porque não posso alimentar minha base de Treinamento automaticamente usando o próprio classificador, aplicado a novos tweets.**

Não é possível alimentar a base de treinamento usando o próprio classificador. O classificador vai classificar os tweets e vai cometer erros, seja eles positivos falsos ou negativos falsos. Ao alimentar o classificador com os seus dados, os tweets errados vão treinar o algoritmo errado, aumentando a quantidade de erros que ele irá cometer. Como o intuito é diminuir os erros, alimentar a base de treinamento com tweets classificados pelo Naive-Bayes seria contraintuitivo.

**Sugerir e explicar melhorias reais no classificador com indicações concretas de como implementar (não é preciso codificar, mas indicar como fazer. Indique material de pesquisa sobre o assunto).**

O Naive-Bayes é um classificador com alto viés e baixa variância. Para diminuir esse viés, seria possível aplicar Boosting. Boosting se baseia nos erros feitos no modelo e, ao atribuir pesos maiores a esses erros, treina um novo modelo. Esse aprendizado incremental baseado nos erros leva a um modelo mais bem treinado para os evitar e diminuindo o seu viés. Ele não é um modelo em si, e sim um jeito de melhorar um classificador e propor um modelo mais eficaz. Um estudo feito pela Universidade de San Diego, na California, sustenta o uso de Boosting para melhorar a performance do classificador Naive-Bayes. Há bibliotecas no Python que já tem algoritmos de Boosting disponíveis, como o próprio sklearn que usamos em aula, de onde seria possível importar o AdaBoostClassifier, o mais popular algoritmo de Boosting.

Outra opção seria usar log para fazer a comparação das probabilidades. Como o Naive-Bayes trabalha com a comparação de probabilidades, isso não iria afetar a classificação do modelo. O log iria resolver um problema computacional que o Naive-Bayes pode encontrar. À medida que o vocabulário do modelo vai crescendo, as probabilidades vão de aproximando de zero. Isso acontece porque várias pequenas probabilidades multiplicadas para formar a probabilidade da frase vão gerar um número menor ainda. Dependendo do tamanho gerado, o computador pode não conseguir usar double nem long double para armazenar a probabilidade. Para evitar isso, o log pode ser implementado. Como, diferentemente de probabilidades, o log-probabilidade gera números entre menos infinito e 0, ele é mais facilmente computado. Para implementar isso, basta calcular ln P(palavra|vocabulário) ao invés de P(palavra|vocabulário).

**Propor diferentes cenários de uso para o classificador Naive-Bayes. Pense em outros cenários sem intersecção com este projeto.**

Considerando a quantidade de itens comprados e o valor final da compra, qual a probabilidade de o cliente ter pagado usando cartão ou dinheiro.

Classificar e-mails em spam vs ham, dado o total de e-mails.

**Referências:**

<http://www.cs.rhodes.edu/~kirlinp/courses/ai/f18/projects/proj3/naive-bayes-log-probs.pdf>

<https://acadgild.com/blog/naive-bayesian-model>

<http://pages.cs.wisc.edu/~dyer/cs540/handouts/elkan97boosting.pdf>

<https://blog.paperspace.com/adaboost-optimizer/>