PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Instituto de Ciências Exatas e de Informática

Marcos Antonio Lommez Candido Ribeiro[1]

Lista Extra #3 - Mineração de Texto

Inteligência Artificial

[1]Aluno de Graduação em Ciência da Computação – tonilommez@hotmail.com

Links para o codigo feito nos exercicios:

Link para o código da questão 1
Link para o código da questão 2 usando multiclass
Link para o código da questão 2 usando multilearn

Questão 01

Nos endereços a seguir você encontrará alguns códigos que fazem o processamento e classificação de textos.

https://www.kaggle.com/code/leandrodoze/sentiment-analysis-in-portuguese/notebook
https://www.datacamp.com/tutorial/text-classification-python
https://reintech.io/blog/how-to-create-a-text-classification-model-with-python

O que você precisa fazer:
Experimentar estes códigos e se basear neles para resolver o problema da base de dados disponível no CANVAS: ReutersGrain-train.csv e ReutersGrain-test.csv
Esta base de dados é de classificação e classifica se o texto fala sobre grãos ou não.
Desta forma, você deverá usar qualquer algoritmo de aprendizado de máquina visto no semestre para resolver este problema de classificação dos textos.

Para a presente questão realizei o seguinte procedimento:

- **Import:** Import da base juntamente com separação entre X/y train/test.
- **Pré-processamento:** Tokenização dos valores, retirada de stop words, filtro de tokens para minusculo e radicalização das palavras para sua forma original.
- **Vetorização:** Transformação dos tokens em valores numéricos que podem ser aceitos por modelos de aprendizado de maquina convencionais.
- **Modelo:** Escolha do modelo de Support Vector Machine.
- **Grid Search:** C: 100, gamma: 0.1, kernal, sigmoid. Utilizando 5 folds.
- **Validação:** Dado o modelo treinado foi encontrado uma acuracia de 97,68%.

Questão 02

No endereço:

https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data

Você encontrará uma base de dados que faz análise de texto em 6 diferentes categorias:
toxic, severe_toxic, obscene, threat, insult, identity_hate.
Ou seja, este problema é considerado de classificação multirótulo. Diferentemente de problemas de classificação multiclasse, na classificação multirótulo cada instância pode pertencer a mais de uma classe. Resolva este problema de classificação e mostre o desempenho do classificador. Para classificação multirótulo, investigue a biblioteca Scikit-multilearn

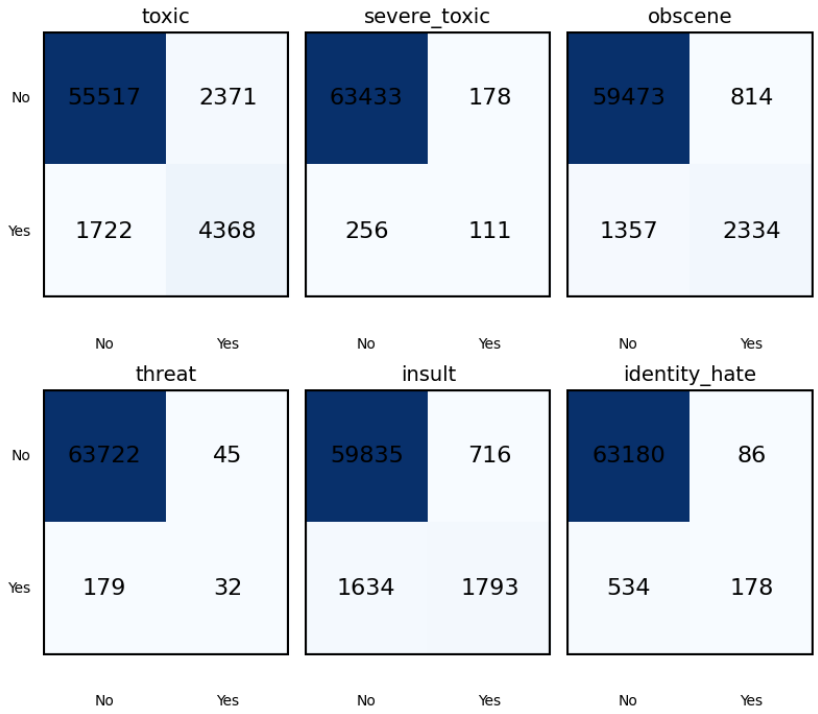
Para a presente questão realizei o seguinte procedimento:

- **Import:** Import da base juntamente com separação entre X/y train/test. Foi utilizado a recomendação feita para uso da base de excluir do treinamento valores marcados com -1 devido a sua chance de diminuir a qualidade dos modelos.
- **Pré-processamento:** Tokenização dos valores, retirada de stop words, filtro de tokens para minusculo e radicalização das palavras para sua forma original. Para a radicalização foi usado o SnowballStemmer devido ao tamanho massivo da base de dados, sendo o clássico Stemmer impossibilitado de usar devido a erros de stack overflow na recursão.
- **Vetorização:** Transformação dos tokens em valores numéricos que podem ser aceitos por modelos de aprendizado de maquina convencionais.
- **Estrategia de treinamento:** Para o presente trabalho foi realizado o treinamento utilizando um algoritmo de multilearn e também uma estratégia aplicada ao Random Forest onde seriam gerados 6 modelos, um para cada saida do problema, assim possibilidando a multi-classificação.

Multi-learn

Na estratégia de treinamento multilearn foi utilizado o modelo de Regressão Logística com Binary Relevance. Os hiper-parametros utilizados foram os default do modelo fornecido pelo sklearn.

O modelo chegou a uma acuracia de treinamento de **89,58%**



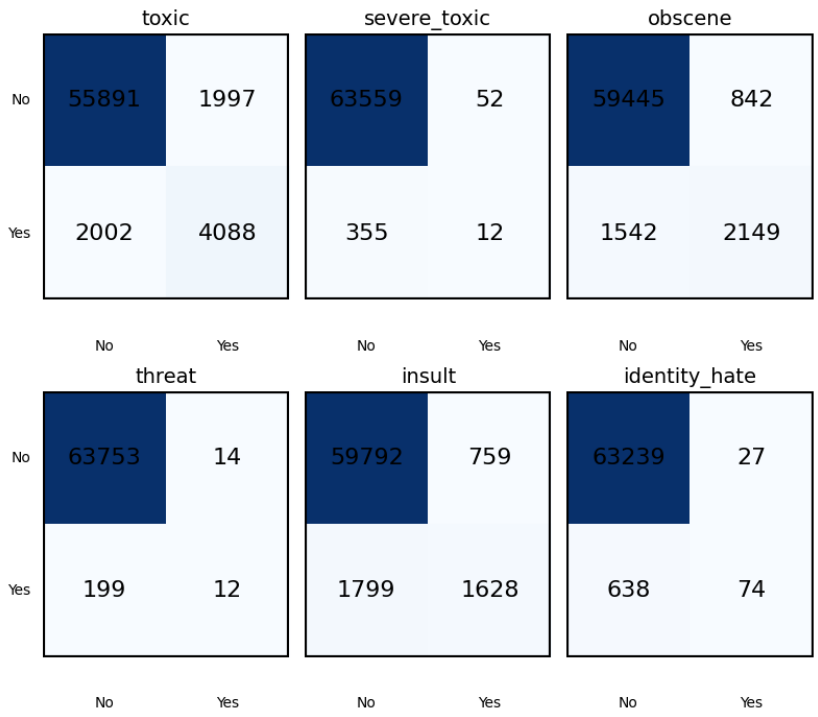
Assim como podemos ver pela matriz de confusão gerada para cada um dos atributos de saída, podemos perceber que pelo alto grau de desbalanceamento da base de dados o treinamento não obteve sucesso, embora para os atributos toxic e obscene chegou a métricas consideráveis, que acredito que podem ser consideravelmente melhoradas com a aplicação correta de técnicas de balanceamento.

Classe	Precisão	Recall	F1-Score
toxic	0.648167	0.717241	0.680957
severe_toxic	0.384083	0.302452	0.338415
obscene	0.741423	0.632349	0.682556
threat	0.415584	0.151659	0.222222
insult	0.714627	0.523198	0.604111
identity_hate	0.674242	0.250000	0.364754
Acurácia Geral	0.895870	0.895870	0.895870

Multi-class

Na estratégia de treinamento multi-class foi utilizado o modelo de Random Forest com a criação de uma floresta para cada valor target que desejamos classificar. Para isso foi utilizado a biblioteca OneVsRestClassifier do sklearn multiclass. Para cada instancia de Random Forest foi usado um numero de 100 estimadores.

O modelo chegou a uma acuracia de treinamento de **89,64%**



Podemos perceber claramente em comparação ao outro modelo que o desbalanceamento da base fez com que a acurácia subisse mesmo que em casas decimais, mesmo com o modelo sendo claramente pior para classificar os casos, pelo simples fato de ser tendencioso a responder "No"em todos os casos. Ainda sim podemos afirmar que o modelo ainda pode ser valido mesmo com sua tendencia em uma situação do mundo real para ajudar a filtrar uma quantidade massiva de mensagens.

Classe	Precisão	Recall	F1-Score
toxic	0.671816	0.671264	0.671540
severe_toxic	0.187500	0.032698	0.055684
obscene	0.718489	0.582227	0.643221
threat	0.461538	0.056872	0.101266
insult	0.682028	0.475051	0.560028
identity_hate	0.732673	0.103933	0.182042
Acurácia Geral	0.896402	0.896402	0.896402