Trabalho Final de Inteligência Artificial

Nomes: Arthur Sudbrack Ibarra, Felipe Grozse Nipper, Miguel Torres de Castro e Willian Magnum Albeche

1. Funcionamento

Para que a análise de sentimentos pudesse ser feita, o grupo optou por utilizar a linguagem de programação Python, em conjunto com a ferramenta Weka. Dessa forma, um script Python chamado *generator.py* foi criado, o qual é responsável por ler o arquivo fornecido pela professora com os dados de entrada já pré-processados e gerar arquivos .arff válidos que o Weka possa interpretar.

É possível informar ao script quantas palavras serão consideradas para o bag of words, bem como quais palavras devem ser desconsideradas. Inclusive, algumas palavras como "dell", "notebook" e "quer" de fato foram desconsideradas, visto que, após uma análise no Weka, o grupo percebeu que se tratavam de palavras neutras, que não agregavam para o treino do modelo, uma vez que podem ser usadas tanto em contextos positivos como negativos.

O uso do script se dá da seguinte forma:

| **python** generator.py <N_PALAVRAS> <NOME_DOS_ARQUIVOS> |

N_PALAVRAS = Quantas palavras serão usadas para o bag of words.

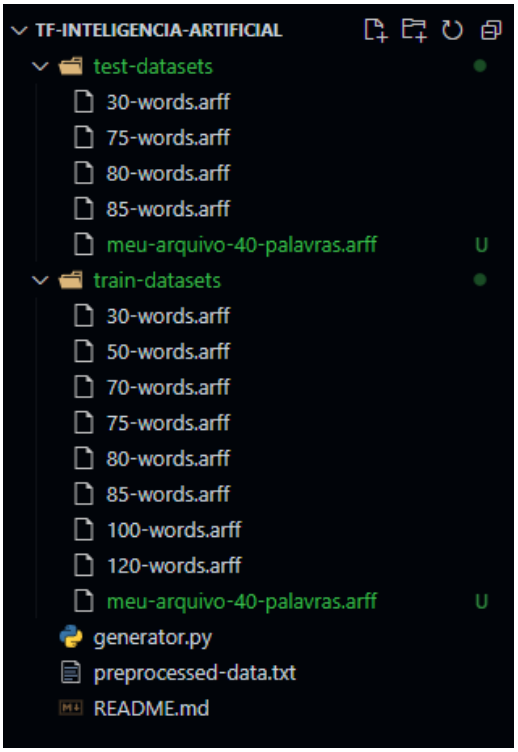
NOME_DOS_ARQUIVOS = Nome dos arquivos .arff que serão gerados (treino e teste).

O arquivo de treino somente inclui frases do dataset fornecido que contenham pelo menos 1 palavra do bag of words. Isso porque o grupo não viu sentido em treinar o modelo com linhas totalmente zeradas, já que estas não indicam/ensinam nada significativo ao modelo. Em contrapartida, no arquivo de teste, todas as palavras do dataset são incluídas, mesmo aquelas que já estavam presentes no arquivo de treino.

Exemplo de execução do algoritmo:

```
PS C:\Users\ibarrart\Documents\Faculdade\TF-Inteligencia-Artificial> python generator.py 40 meu-arquivo-40-palavras

Palavras sendo desconsideradas: ['dell', 'notebook', 'not', 'nov', 'quer', 'q', 'ter', 'ach', 'agor', 'vai', 'dess', 'inspiron', 'precis', 'cas', '%', "'s", "'tou"]

Arquivos gerados com sucesso:

train-datasets/meu-arquivo-40-palavras.arff: Arquivo de treino, não considera linhas totalmente zeradas.
test-datasets/meu-arquivo-40-palavras.arff: Arquivo de teste, considera todo o dataset.

PS C:\Users\ibarrart\Documents\Faculdade\TF-Inteligencia-Artificial>
```

Arquivos .arff gerados:



Exemplo de arquivo gerado com 30 palavras:

30-words.arff

test-datasets > 30-words.arff

```
@relation emotionAnalysis

@attribute compr {0,1}
@attribute http {0,1}
@attribute pra {0,1}
@attribute problem {0,1}
@attribute reclam {0,1}
@attribute aqu {0,1}
@attribute brasil {0,1}
@attribute reclameaqu {0,1}
@attribute comput {0,1}
@attribute nunc {0,1}
@attribute dellnobrasil {0,1}
@attribute dá {0,1}
@attribute car {0,1}
@attribute ano {0,1}
@attribute deu {0,1}
@attribute ta {0,1}
@attribute dellajud {0,1}
@attribute ness {0,1}
@attribute olh {0,1}
@attribute nao {0,1}
@attribute vou {0,1}
@attribute atual {0,1}
@attribute suport {0,1}
@attribute troc {0,1}
@attribute window {0,1}
@attribute tel {0,1}
@attribute 3 {0,1}
@attribute mand {0,1}
@attribute aind {0,1}
@attribute contr {0,1}
@attribute class {-1,1}

@data
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-1
0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-1
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-1
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-1
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1
0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1
1,1,0,0,1,0,0,1,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,-1
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-1
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,1
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-1
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-1
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1
```

2. Resultados

Após gerados os arquivos .arff, fornecemos eles à ferramenta Weka para que pudéssemos utilizar diferentes modelos de classificação e ver o desempenho destes com variadas configurações.

2.1. KNN (IBk)

Quantidade de palavras no bag of words: 80.

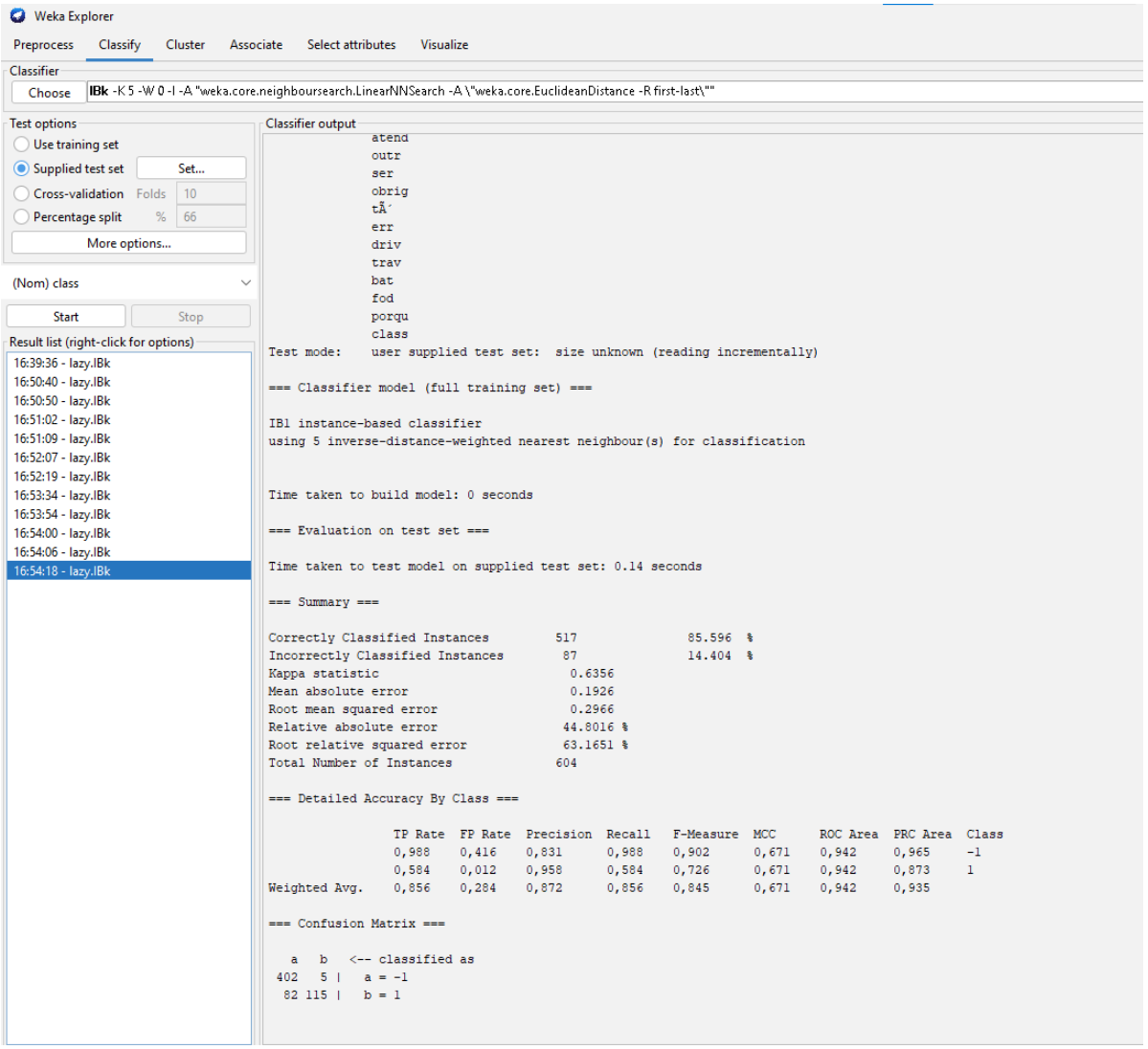
Palavras sendo desconsideradas: dell, notebook, not, nov, quer, q, ter, ach, agor, vai, dess, inspiron, precis, cas, %, 's, 'tou.

Tipo do teste: Conjunto de teste fornecido.

distanceWeighting: Weight by 1/distance.

Vizinhos próximos sendo considerados: 5.

Resultado:



Conforme pode ser observado na figura acima, o modelo acertou 85,596% das classificações, errando somente 14,404% dos palpites. O grupo notou que um dos fatores mais importantes para alcançar tal resultado foi a configuração **distanceWeighting**, que por padrão vem configurada como **No distance weighting**. Quando o campo foi mudado para **Weight by 1/distance**, os acertos do algoritmo aumentaram consideravelmente. Além disso, é de relevância comentar que para o caso em questão, diversos valores de números de vizinhos a se considerar (1, 2, 5…) foram testados, porém a taxa de acerto sempre manteve-se em 85,596%.

2.2. MultilayerPerceptron

Quantidade de palavras no bag of words: 80.

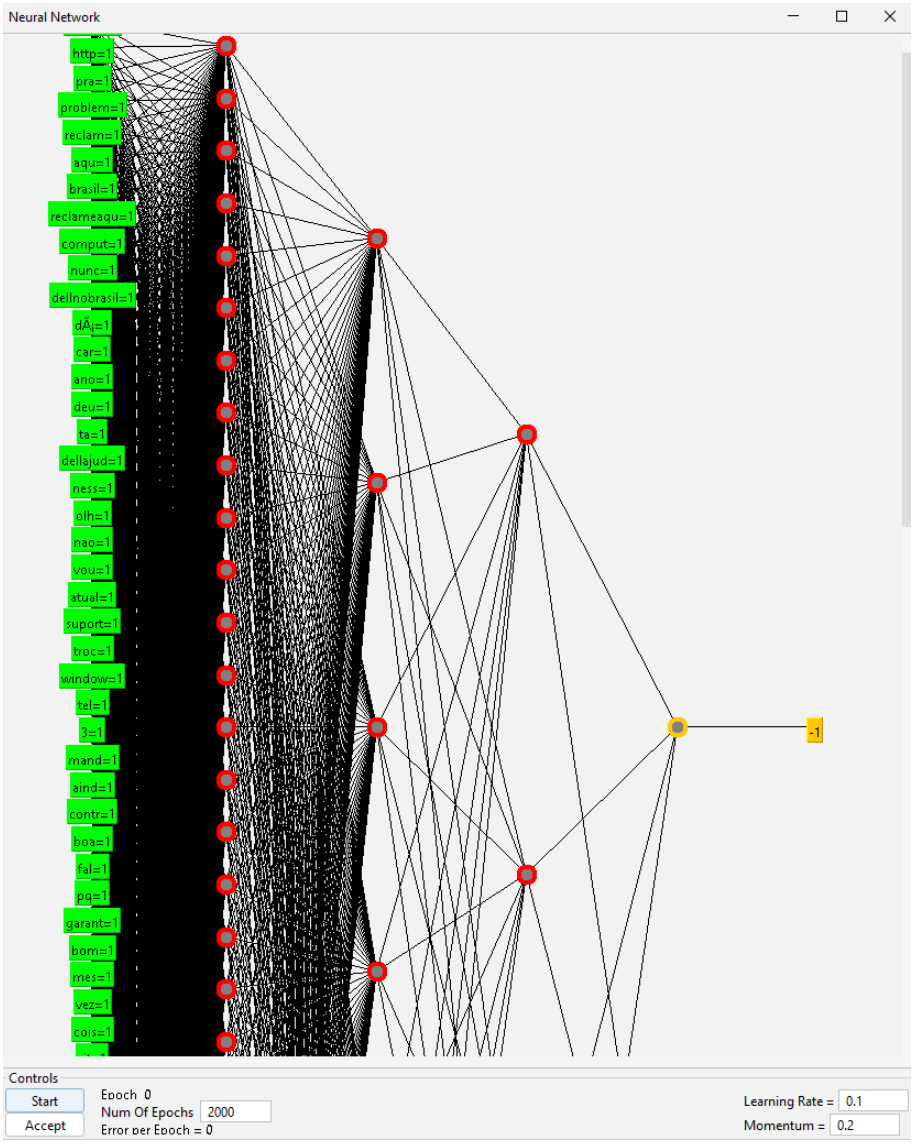
Palavras sendo desconsideradas: dell, notebook, not, nov, quer, q, ter, ach, agor, vai, dess, inspiron, precis, cas, %, 's, 'tou.

Tipo do teste: Conjunto de teste fornecido.

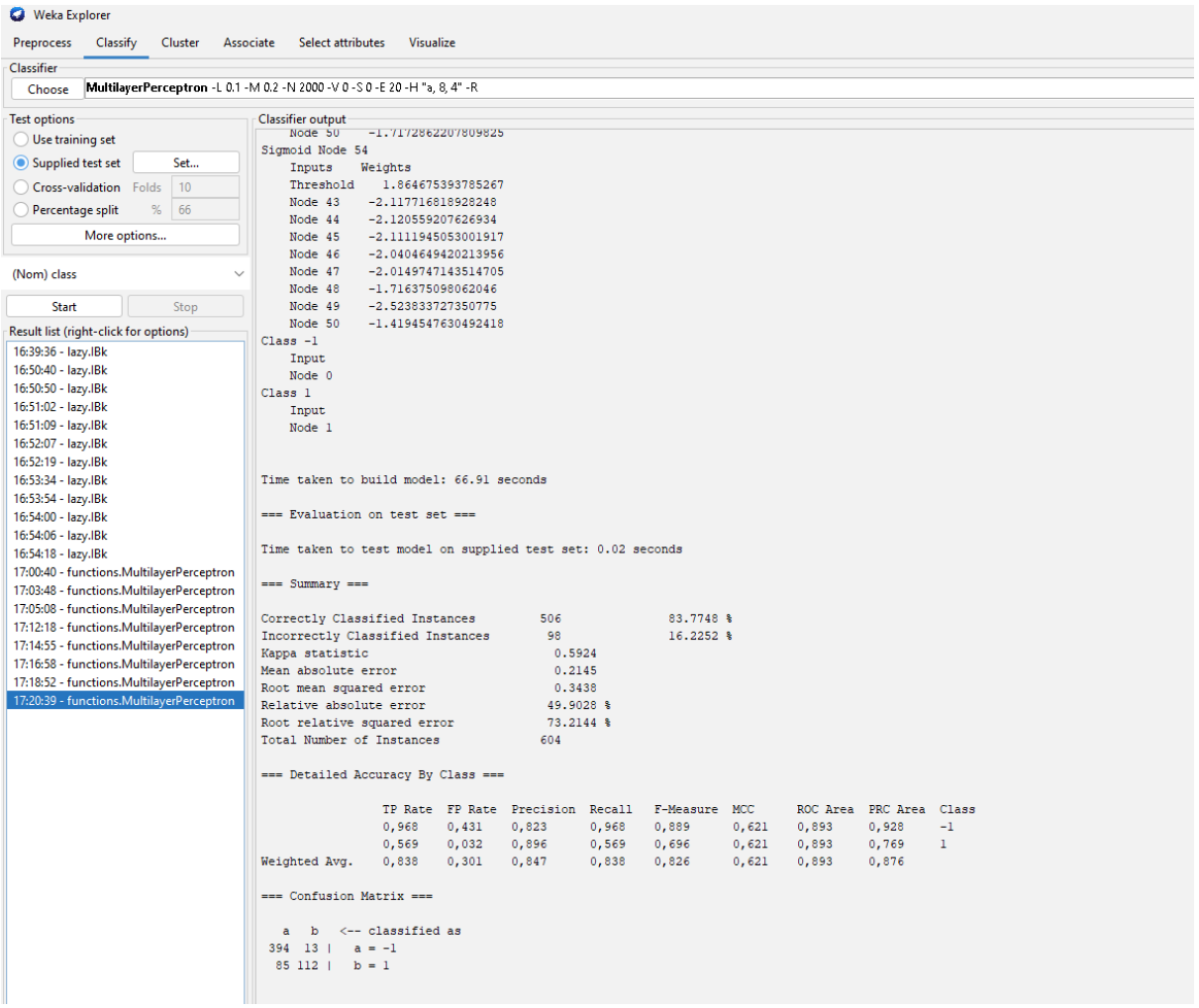
Quantidade de épocas: 2000.

Taxa de aprendizado: 0,1

Hidden layers: (8,4) 1 hidden layer com 8 neurônios e 1 hidden layer com 4 neurônios.



Resultado:



Conforme mostrado na imagem acima, o modelo de Multilayer Perceptron obteve uma taxa de acerto de 83,774% e uma taxa de erro de 16,225%, obtendo, portanto, um resultado um pouco inferior ao do modelo KNN. O grupo pôde perceber que a configuração das **hidden layers** impactou muito o resultado do modelo, visto que quando configurações diferentes foram testadas **(4,2)**, **(8,4,2)** e **(8,8,4)**, a taxa de acerto ficou perto de 67,5%, com ~15% de diferença em relação ao resultado final obtido com a configuração **(8,4)**.

2.3. RandomForest

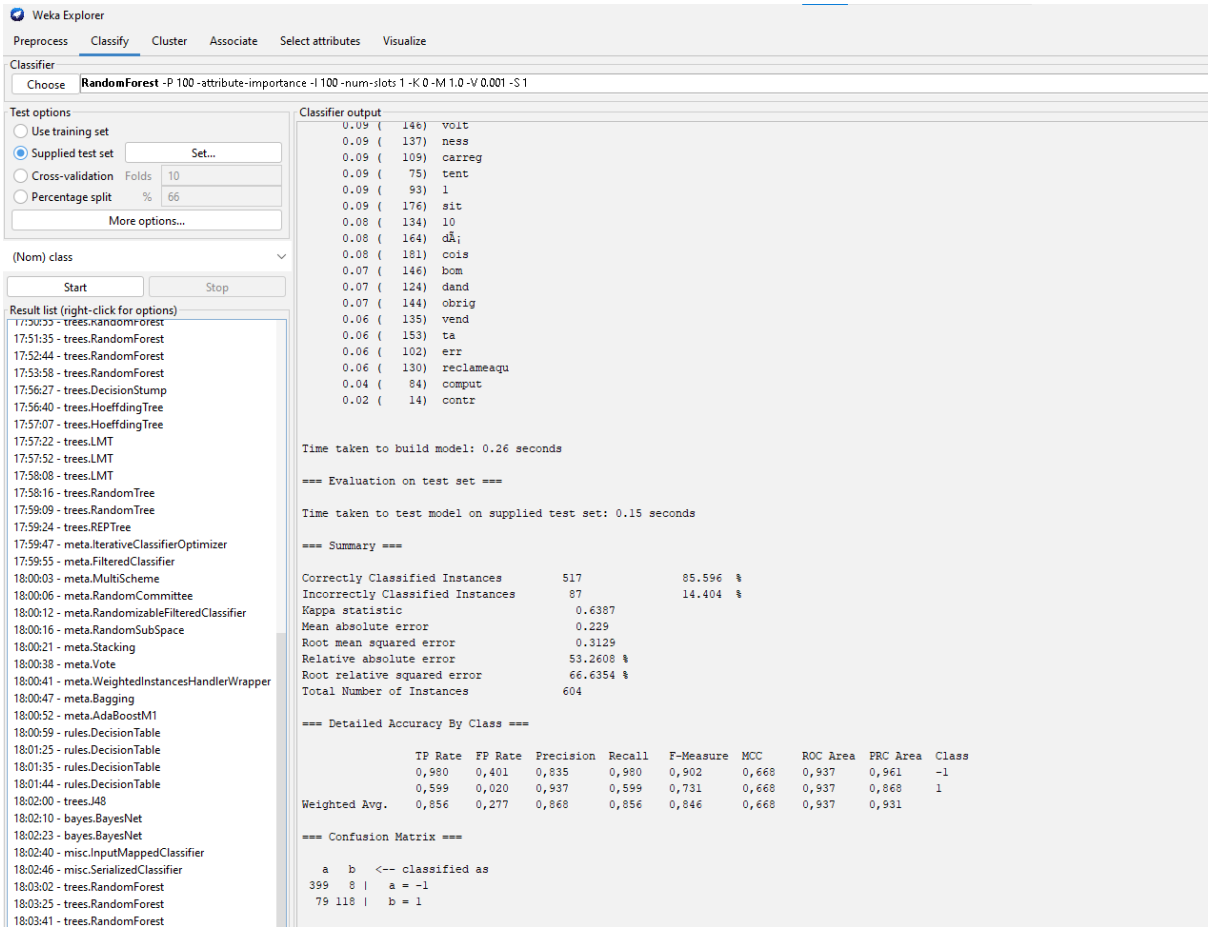
Quantidade de palavras no bag of words: 80.

Palavras sendo desconsideradas: dell, notebook, not, nov, quer, q, ter, ach, agor, vai, dess, inspiron, precis, cas, %, 's, 'tou.

Tipo do teste: Conjunto de teste fornecido.

Profundidade máxima: 0.

Resultado:



O último modelo testado foi o de RandomTree, o qual obteve uma taxa de acerto de 85,596% e uma taxa de erro de 14,404%, empatando com o modelo de KNN e superando o modelo de Multilayer Perceptron. A configuração observada pelo grupo como a mais importante para o sucesso do algoritmo foi a de **profundidade máxima**, que conforme tinha o seu valor incrementado, a acurácia do modelo

diminuía, porém quando a profundidade alcançava um determinado valor X, a acurácia do algoritmo voltava a aumentar até chegar em 85,595% novamente.

3. Conclusão:

Ao final dos testes realizados, foi possível perceber que as taxas de acerto dos três modelos testados foram razoáveis, porém o grupo acredita que não são taxas altas o suficiente para considerar os modelos confiáveis. Isso se dá, muito provavelmente, pelo dataset que foi utilizado para gerar os arquivos de treino, visto que ele era pequeno (apenas 604 instâncias), e que existiam muito mais amostras com classes negativas do que positivas Além disso, muitas das frases fornecidas, mesmo com um bag of words grande, não tinham nenhum termo do bag of words, resultando em linhas de dados zeradas como entradas durante a fase de teste, as quais acabavam prejudicando a taxa de acerto do modelo, uma vez que nessas situações o modelo acaba tendo de classificar a instância sem ter nenhuma informação relevante.