Trabalho Final de Inteligência Artificial

Nomes: Arthur Sudbrack Ibarra, Felipe Grozse Nipper de Oliveira, 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 "inspiron" 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.

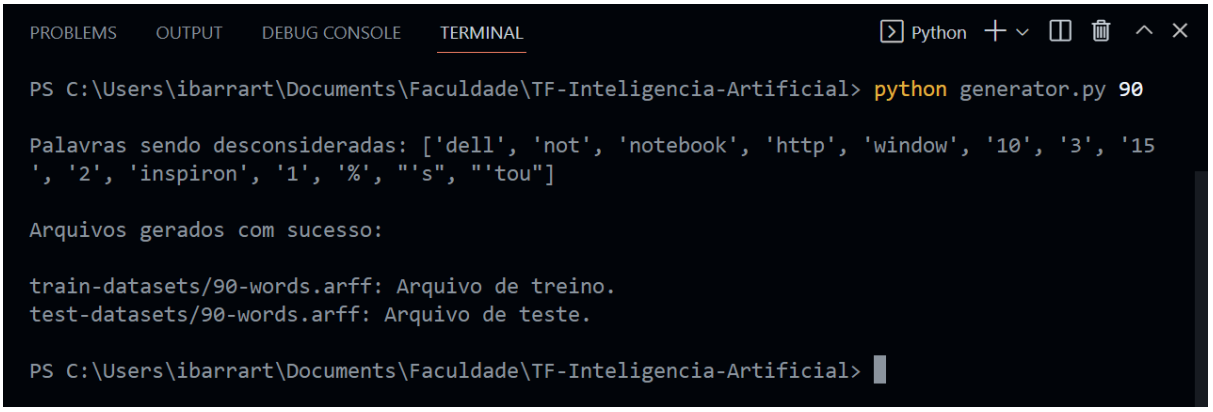
O uso do script se dá da seguinte forma:

python generator.py <N_PALAVRAS>

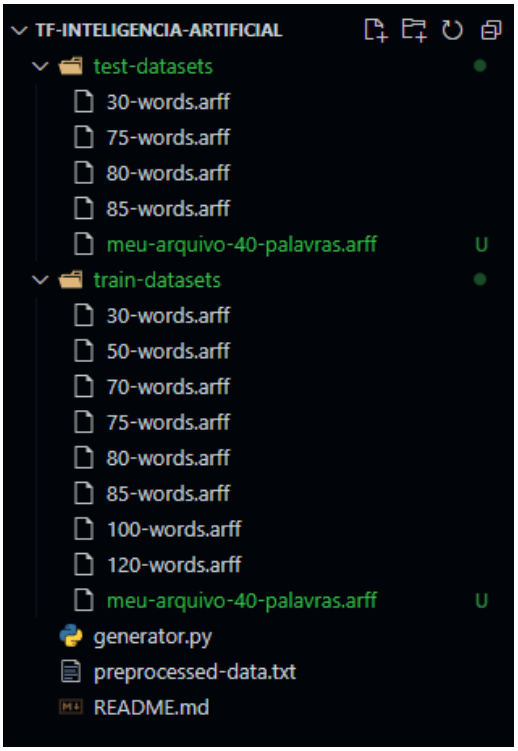
N_PALAVRAS = Quantas palavras serão usadas para o bag of words.

70% dos dados são usados para treino e 30% para teste, sendo que a distribuição dos dados é feita de forma aleatória. De 70% dos dados, 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, tal restrição não existe.

Exemplo de execução do algoritmo:



Exemplo de arquivos ARFF sendo gerados:



Exemplo de arquivo de treino gerado com 30 palavras:

generator.py M | 30-words.arff U

train-datasets > 30-words.arff

```
@relation emotionAnalysis

@attribute compr {0,1}
@attribute pra {0,1}
@attribute nov {0,1}
@attribute problem {0,1}
@attribute reclam {0,1}
@attribute aqu {0,1}
@attribute quer {0,1}
@attribute brasil {0,1}
@attribute comput {0,1}
@attribute reclameaqu {0,1}
@attribute dellnobrasil {0,1}
@attribute nunc {0,1}
@attribute dá {0,1}
@attribute car {0,1}
@attribute q {0,1}
@attribute ta {0,1}
@attribute ano {0,1}
@attribute olh {0,1}
@attribute dellajud {0,1}
@attribute ach {0,1}
@attribute ness {0,1}
@attribute troc {0,1}
@attribute vou {0,1}
@attribute deu {0,1}
@attribute nao {0,1}
@attribute atual {0,1}
@attribute boa {0,1}
@attribute lind {0,1}
@attribute agor {0,1}
@attribute dess {0,1}
@attribute class {-1,1}

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

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)

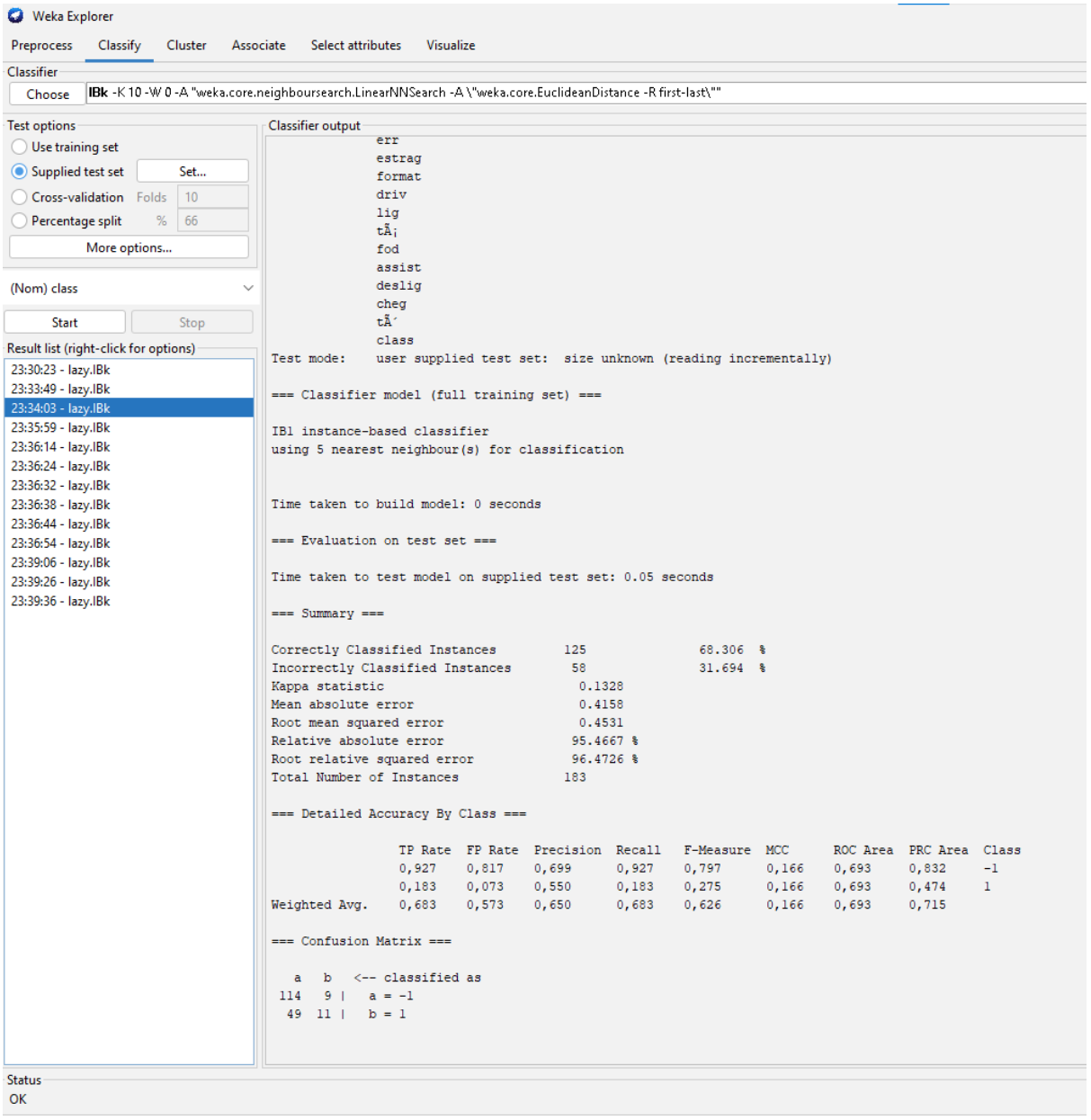
Quantidade de palavras no bag of words: 85.

Palavras sendo desconsideradas: ['dell', 'not', 'notebook', 'http', 'window', '10', '3', 'inspiron', '1', '%', "'s", "'tou"]

Tipo do teste: Conjunto de teste fornecido.

Vizinhos próximos sendo considerados: 5.

Resultado:



Conforme pode ser observado na figura acima, o modelo acertou 68,306% das classificações, errando 31,694% dos palpites. Outros testes foram feitos com bag of words de tamanhos diferentes e conjuntos de treino e teste distintos, porém a taxa de acerto do modelo manteve-se na faixa de 64-68%. Ademais, o grupo notou que analisando os **5** vizinhos mais próximos para definir a classe de um elemento, o modelo obteve um resultado melhor do que quando esse valor era mais elevado, como **10**, ou mais baixo, como **2**.

2.2. MultilayerPerceptron

Quantidade de palavras no bag of words: 85.

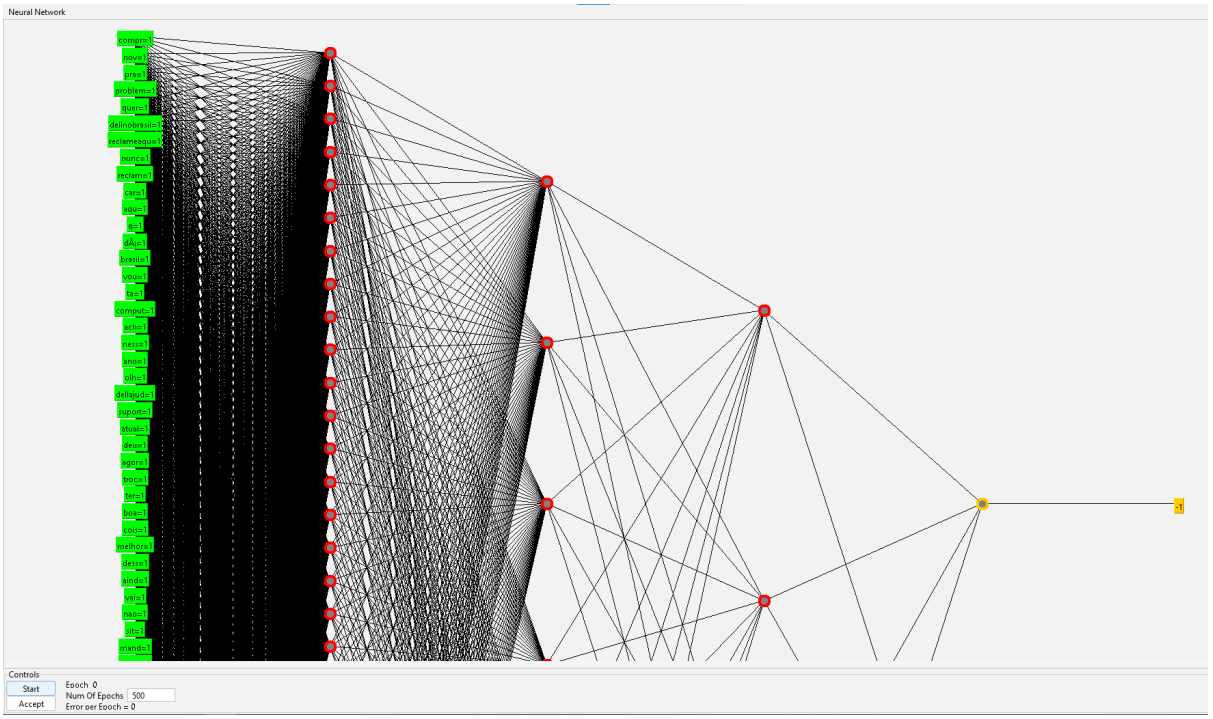
Palavras sendo desconsideradas: ['dell', 'not', 'notebook', 'http', 'window', '10', '3', 'inspiron', '1', '%', "'s", "'tou"]

Tipo do teste: Conjunto de teste fornecido.

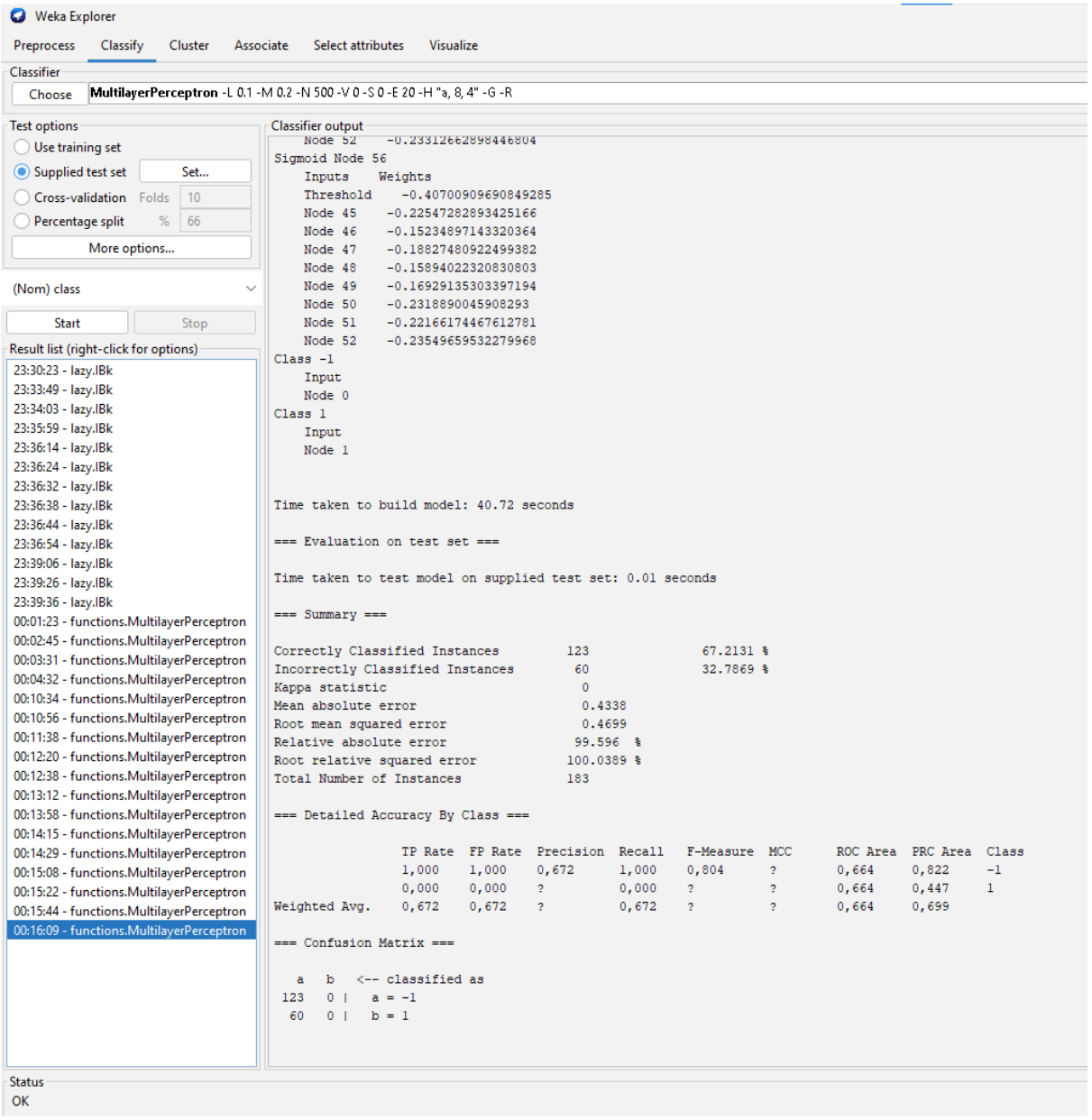
Quantidade de épocas: 2000.

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 67,213% e uma taxa de erro de 32,786%, 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 só existia 1 hidden layer, a taxa de instâncias classificadas corretamente foi inferior à 50%. Além disso, um fator curioso de comentar é o fato de que o valor da **taxa de aprendizado** não influenciou nos resultados finais do modelo, o que chamou a atenção do grupo, dado que uma taxa de aprendizado pequena implica em saltos de ajuste de peso menores e em um refinamento mais preciso do modelo.

Todavia, quando a taxa de aprendizado foi configurada para **0,01** (baixo) e para **1** (alto), a taxa de acerto do modelo manteve-se a mesma.

2.3. RandomForest

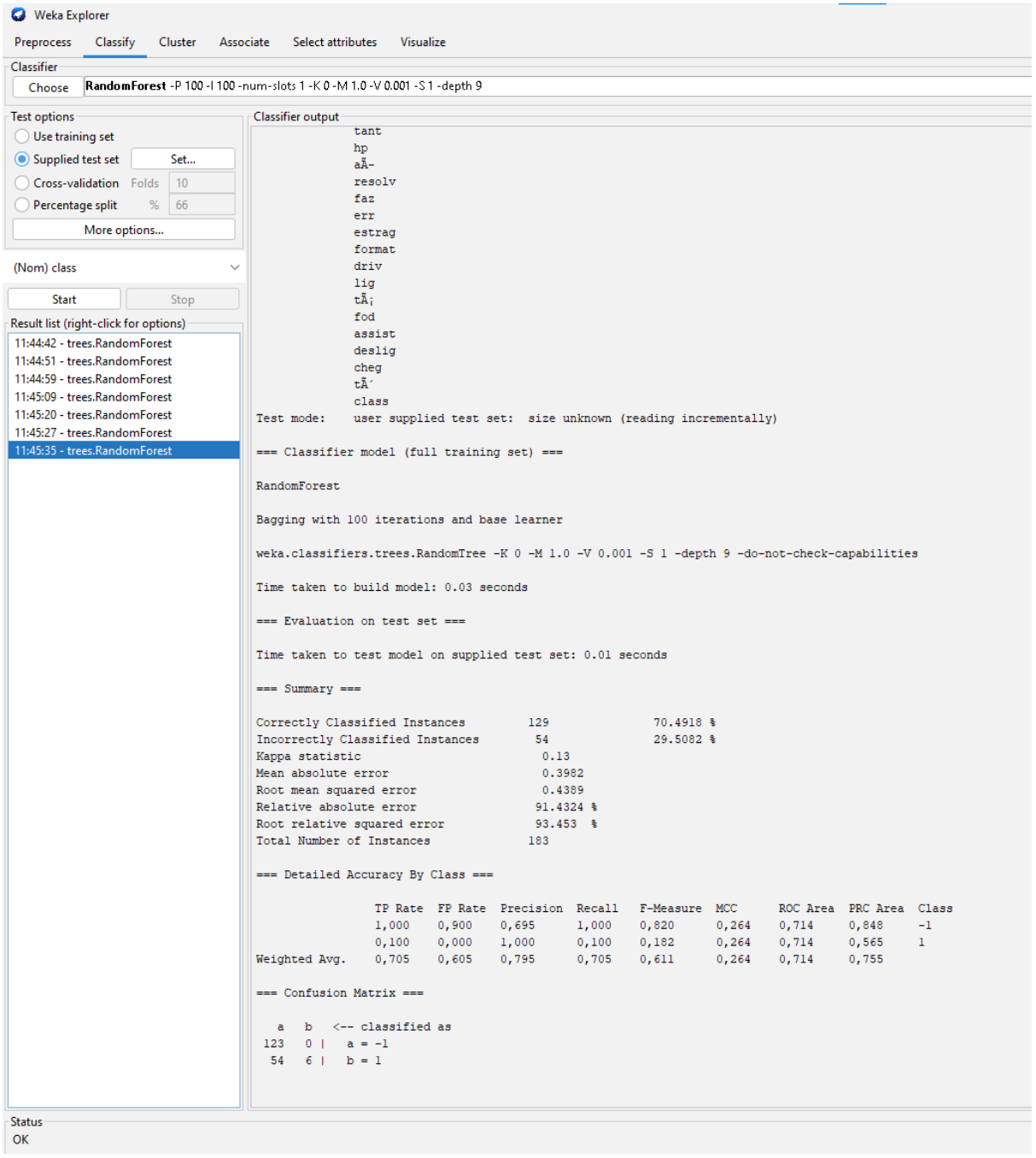
Quantidade de palavras no bag of words: 85.

Palavras sendo desconsideradas: ['dell', 'not', 'notebook', 'http', 'window', '10', '3', 'inspiron', '1', '%', "'s", "'tou"]

Tipo do teste: Conjunto de teste fornecido.

Profundidade máxima: 9.

Resultado:



O último modelo testado foi o de RandomForest, o qual obteve uma taxa de acerto de 70,491% e uma taxa de erro de 29,508%, superando por pouco tanto o modelo de KNN quanto o modelo de Multilayer Perceptron. A configuração observada pelo grupo como a mais importante para o sucesso do algoritmo foi a de **profundidade máxima**, sendo que o valor ideal foi de **9**. Se o valor da profundidade máxima era < 9, a taxa de acerto do modelo caía para < 70%; para valores próximos de 9, como 10 e 11, o resultado mantinha-se em 70,491%; mas quando o valor da profundidade

máxima se distanciava de 9 para cima, as taxas de acerto do modelo voltavam a cair.

3. Conclusão

Ao final dos testes realizados, foi possível perceber que as taxas de acerto dos três modelos testados não foram altas, girando em torno de 67%-70%, não sendo, portanto, suficientes para considerarmos os modelos confiáveis. Isso se dá, muito provavelmente, pelo dataset que foi utilizado para gerar os arquivos de treino e teste, visto que ele era pequeno (604 frases), 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.