Análise dos algoritmos
Pedro Henrique de Brito Canabarro

1. Usando a trees.*RandomForest* temos:

| | | |
|---|---|---|
| Correctly Classified Instances | 11 | 68.75 % |
| Incorrectly Classified Instances | 5 | 31.25 % |

=== Confusion Matrix ===

```
 a b c d e   <-- classified as
 6 0 0 0 0 | a = mamífero
 0 2 1 0 0 | b = réptil
 0 0 3 0 0 | c = peixe
 2 0 0 0 0 | d = anfíbio
 2 0 0 0 0 | e = ave
```

O algoritmo cria múltiplas árvores de decisão, cada uma treinada em subconjuntos aleatórios dos dados e atributos. A decisão final é feita por votação da maioria ou média. O modelo está agrupando classes como "ave" e "anfíbio" dentro de "mamífero", o que mostra que esse modelo não é altamente recomendado.

2. Usando a *functions.SMO* temos:

| | | |
|---|---|---|
| Correctly Classified Instances | 16 | 100 % |
| Incorrectly Classified Instances | 0 | 0 % |

=== Confusion Matrix ===

```
 a b c d e   <-- classified as
 6 0 0 0 0 | a = mamífero
 0 3 0 0 0 | b = réptil
 0 0 3 0 0 | c = peixe
 0 0 0 2 0 | d = anfíbio
 0 0 0 0 2 | e = ave
```

O algoritmo encontra o hiperplano que maximiza a margem entre classes no espaço de atributos. O SMO (Sequential Minimal Optimization) é um algoritmo eficiente para treinar SVMs.

3. Usando o *functions.MultilayerPerceptron* temos:

Correctly Classified Instances 16 100 %
Incorrectly Classified Instances 0 0 %

=== Confusion Matrix ===

```
 a b c d e   <-- classified as
 6 0 0 0 0 | a = mamífero
 0 3 0 0 0 | b = réptil
 0 0 3 0 0 | c = peixe
 0 0 0 2 0 | d = anfíbio
 0 0 0 0 2 | e = ave
```

O algoritmo é composto por camadas (entrada, ocultas e saída) de neurônios artificiais. Utiliza retropropagação para ajustar os pesos e minimizar o erro.

## Relatório adicionando mais um parâmetro.

No arquivo *vertebrados.arff* atribuímos o parâmetro ‘gila’ no nome mas no *@data* não adicionamos as características completas do animal. Foi criado um arquivo *vetebrados1.arff* com a mesma base do antigo mas no *@data* foi removido todos os outros 16 animais e adicionado somente o monstro de gila com suas características completas dizendo que a classe dele era ‘mamífero’. Testando no *Weka* usando a *random.tree*, treinando e comparando as bases de dados ele mostrou isso:

```
 a b c d e   <-- classified as
 1 0 0 0 0 | a = mamífero
 0 0 0 0 0 | b = réptil
 0 0 0 0 0 | c = peixe
 0 0 0 0 0 | d = anfíbio
 0 0 0 0 0 | e = ave
```

Mas utilizando outros algoritmos como o *randomForest* e o *MultilayerPerceptron*, o resultado mudou para esse:

```
 a b c d e   <-- classified as
 0 1 0 0 0 | a = mamífero
 0 0 0 0 0 | b = réptil
 0 0 0 0 0 | c = peixe
 0 0 0 0 0 | d = anfíbio
 0 0 0 0 0 | e = ave
```

Conclui-se que o *randomTree* não tem assertividade diante da classe do monstro de gila, sendo que em outros algoritmos, o mesmo monstro de gila onde no código-fonte foi designado como mamífero, exibiu o resultado dizendo que ele é um réptil de acordo com suas características.