Aplicação de ID3 ou C4.5 para a predição de espécies de anuros

Mestrado Integrado em Engenharia Informática e Computação

Inteligência Artificial

Turma 5:
Daniel Pereira Machado - 201506365
José Pedro Dias de Almeida Machado - 201504779
Sofia Catarina Bahamonde Alves - 201504570

Faculdade de Engenharia da Universidade do Porto
Rua Roberto Frias, sn, 4200-465 Porto, Portugal

20 de Maio de 2018

Conteúdo

1 Objetivo

O objetivo deste projeto é a determinação da árvore de decisão que traduz essas regras na predição de espécies de anuros a partir dos seus chamamentos.

Os anuros constituem uma ordem de animais pertencentes à classe *Amphibia*, que inclui os sapos e as rãs, sendo que cada espécie produz um sinal sonoro muito específico, que permite distingui-la de qualquer outra espécie. Um biólogo especializado é capaz de fazer essa distinção, mas essa abordagem torna-se demasiada lenta e propensa a erros, por isso a nossa solução passa por usar algoritmos de *Deep Learning*.

Desta forma, foi criada uma aplicação com capacidade de receber um conjunto de dados referentes a chamamentos de anuros e classificar cada um dos sons relativamente à espécie, género e família a que pertence. Este diagnóstico será feito tendo em conta os datasets fornecidos. A partir dos quais serão inferidas as regras necessárias para que posteriormente seja possível uma tradução destas regras para uma árvore de decisão, que por sua vez será utilizada para classificar cada chamamento.

2 Especificação

2.1 Análise Detalhada do Tema

Os anuros constituem uma ordem de animais pertencentes à classe *Amphibia*, que inclui os sapos e as rãs, os anuros estão muito relacionados com o ecossistema onde vivem e são muitas vezes usados pelos biólogos como indicadores de stress ecológico (está demonstrado que existe uma relação direta entre alterações climáticas e a mortalidade em espécies de anuros).

O chamamento dos anuros é feito fazendo passar ar através da sua laringe na garganta e este é amplificado por um ou mais sacos vocais (membranas de pele debaixo da garganta ou no canto da boca), o principal motivo dos chamamentos destes anfíbios é atrair um parceiro para acasalar. Por estes chamamentos serem exclusivos para cada espécie permitem que funcionem como meio de identificação dos indivíduos permitindo a sua classificação em termos de família, género e espécie.

Um especialista humano pode ser usado para classificar anuros através dos seus chamamentos, mas essa abordagem é lenta e propensa a erros, existem também métodos que recorrem a hardware, mas este hardware tem custos demasiado elevados portanto a solução passa por usar algoritmos de *Deep Learning*. Como tal a automatização da classificação dos anuros através da análise do seu chamamento ajuda os biólogos a analisar a sua atividade em larga escala.

2.2 Ilustração de Cenários

O mais antigo sistema de classificação remonta a Aristóteles que dividiu os organismos vivos em plantas e animais. Desde então os sistemas de classificação têm vindo a ser melhorados e cada vez mais detalhados. Atualmente o sistema utilizado faz uma divisão por domínio, reino, filo ou divisão, classe, ordem, família, género e espécie. A taxonomia introduz um conjunto de critérios muito específicos que permite dividir os diferentes indivíduos consoantes as suas caraterísticas.

Esta classificação permite relacionar as diferentes espécies com os seus antepassados bem como os diferentes indivíduos atuais. Deste modo além de contribuir para os conhecimentos de evolução das espécies contribui para os estudos que envolvem as atual biodiversidade: a forma como interagem com o ambiente, as suas propriedades curativas. No caso dos anuros, os interesses consistem em: prever catástrofes naturais, nomeadamente terramotos; produção de remédios através dos seus venenos; produção de colas através das secreções presentes nas suas patas adesivas.

2.3 Explicação de Datasets

A base de dados é composta por 60 ficheiros, todos eles gravações de áudio. Estas gravações correspondem a chamamentos de anuros, sendo comtempladas 4 famílias, 8 géneros e 10 espécies distintas. Estas gravações foram protagonizadas no campus Universidade Federal da Amazónia, Manaus, Mata Atlântica (Brasil) assim como e, Córdoba (Argentina). Todas as gravações obtidas foram guardadas no formato wav, com frequência 441.kHz e resolução 32bits, sendo assim possível analisar sinais até 22kHz. De todas as sílabas extraídas foram calculados 22 MFCCs usando 44 filtros triangulares.

Espécies

- AdenomeraAndre - 672
- AdenomeraHylaedacta - 3478
- Ameeregatrivittata - 542
- HylaMinuta - 310
- HypsiboasCinerascens - 472
- HypsiboasCordobae - 1121
- LeptodactylusFuscus - 270
- OsteocephalusOopha - 114
- Rhinellagranulosa - 68
- ScinaxRuber - 148

Géneros

- Adenomera - 4150
- Ameerega - 542
- Dendropsophus - 310
- Hypsiboas - 1593
- Leptodactylus - 270
- Osteocephalus - 114
- Rhinella - 68
- Scinax - 148

Famílias

- Bufonidae - 68
- Dendrobatidae - 542
- Hylidae - 2165
- Leptodactylidae - 4420

2.4 Abordagem

2.4.1 Processo utilizado na captação de dados

MFCC's(at Mel-Frequency Cepstral Coefficients) são das tecnologias mais utilizadas no reconhecimento de discursos humanos, visto que os chamamentos dos anuros consistem em curtos sons, chamados sílabas, estas sílabas conseguem portanto ser equiparadas a fonemas no caso dos humanos. No primeiro passo desta abordagem as sílabas são extraídas do áudio original, de seguida são escolhidos os parâmetros de forma a que após a extração MFCC cada sílaba seja representada por uma matriz quadrada como representado na Figura 1.

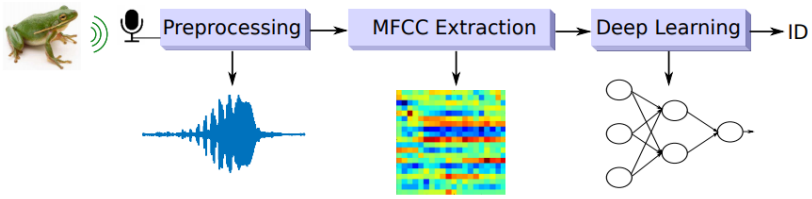

Figura 1: Processo utilizado no tratamento de dados

2.4.2 Pré-processamento de Dados

A fim de ser possível a extração de MFCCs foi necessário converter as gravações iniciais em ondas passíveis de serem analisadas, como tal, foi utilizada entropia espectral e um cluster binário. Desta forma é possível detetar as frames de áudio pertencentes a cada sílaba.

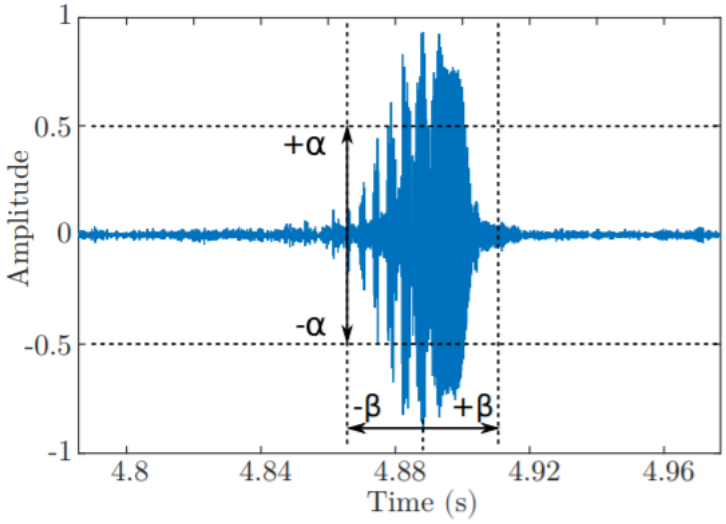

Figura 2: Pré-processamento de uma amostra

2.4.3 Extração de MFCC

Mel-Frequency Cepstrum (MFC) é uma representação do poder espectral de um som, baseado numa transformação linear numa escala de frequência não linear. Os Mel-frequency cepstral coefficients (MFCCs) são os coeficientes, que em conjunto formam um MFC. Neste contexto correspondem às frames extraídas a partir das gravações após o pré-processamento. A fim de calcular os MFCCs o áudio foi dividido em pequenos frames, sendo cada um deles tratado através de uma transformação de Fourier. Houve um cuidado especial, pois as gravações originais foram obtidas in loco, em condições naturais e apresentam algum ruído de fundo.

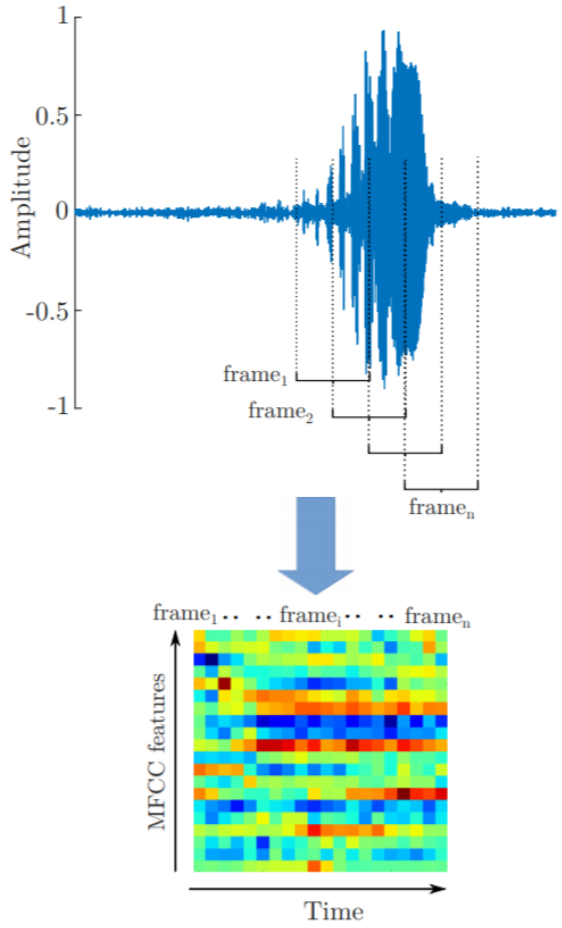

Figura 3: Conversão para MFCC

2.4.4 Árvores de decisão

As árvores de decisão são modelos que apresentam uma notória utilização na área da inferência indutiva. Este modelo representa funções como árvores de decisão. Estas árvores são treinadas de acordo com um conjunto de treino (exemplos previamente classificados) e posteriormente, outros exemplos são classificados de acordo com essa mesma árvore. Para a construção destas árvores são utilizados algoritmos como o ID3, ASSISTANT e C4.5. Neste projeto pretende-se aplicar árvores de decisão, nomeadamente com os algoritmos ID3 e C4.5, para a predição de espécies de anuros. Vai-se usar parte dos dados previamente obtidos para construir a árvore de decisão, de modo a identificar cada tipo de anuro. Posteriormente, a outra parte dos dados vão conduzir a uma determinada espécie de anuro, o que vai permitir validar ou não a árvore em questão.

Neste trabalho foi utilizado o algoritmo C4.5 visto ser uma melhoria do algoritmo ID3 e por ser possível tirar conclusões intermédias. Este algoritmo foi utilizado de forma a gerar de forma sistemática uma árvore de decisão. Aprendizagem supervisionada é uma tarefa de *machine learning* que infere um função a partir de um conjunto de dados de treino categorizados que é este o casos visto que temos 22 MFCC's para cada sílaba e o output esperado a classificação em família género e espécie de um dado indivíduo.

2.4.5 Análise de implementação

Depois de terem sido obtidos os conjuntos de dados de treino e teste, é construído um classificador, que pode ser utilizado com as mais variadíssimas opções, no geral, entendemos que, os valores predefinidos produzem bons resultados, para a poda o weka baseia-se em três características :

- Não continuar a dividir os nós se eles ficarem demasiado pequenos, o valor predefinido é 2;
- Construir a árvore completa, e seguidamente, proceder das folhas para cima, aplicando testes estatísticos em cada estado, para decidir se a poda se deve realizar, o valor predefinido para o factor de confiança é de 0.25(valores mais baixos levam a uma poda maior).
- Por vezes, é bom selecionar uma sub-árvore e substituí-la com uma árvore filha, isto denomina-se subtreeRaising que está ativado por default no weka. Este parâmetro aumenta a complexidade do algoritmo, tornando-o um pouco mais lento, no entanto, achámos que faria sentido manter esta opção, uma vez que produz, teoricamente, melhores resultados.

É também possível utilizar a poda no modo "reduced error pruning", que é um dos modos mais simples de realizar a poda, consiste em começando nas folhas, cada nó é substituído com a classe mais popular. Se a precisão da predição não for afetada a alteração mantém-se.

Após a construção do classificador são inferidas as regras baseadas no cálculo da entropia e do ganho para cada um dos atributos:

Entropia:

$$H(S) = \sum_{i=1}^{n}(-p_i * \log_2(p_i)) \tag{1}$$

Onde:

S : Conjunto de exemplos de treino.
p_i : Percentagem de elementos de uma determinada classe.
$\sum_{i=1}^{n}$: Somatório de todas as classes

Ganho:

$$Gain(S, A) = H(S) - \sum_{j=1}^{n}(p_j * E(p_j)) \tag{2}$$

Onde:

S : Conjunto de exemplos de treino.
A : Atributo
p_i : Percentagem de elementos em que o atributo(A) influência a classe em questão.
$\sum_{i=1}^{n}$: Somatório de todas as classes

A árvore de decisão final obtida, terá os atributos com maior valor de ganho e menor valor de entropia mais próximos da raiz, uma vez que, quanto menor for o nível de desordem, mais pertinentes serão esses atributos e mais conclusões serão inferidas diretamente por estes. A partir deste momento são avaliados os valores obtidos para cada atributo do dataset de teste, sendo comparados com os valores iniciais de modo a permitir o cálculo da fiabilidade da árvore de decisão obtida.

3 Desenvolvimento

3.1 Ferramentas Utilizadas

Para esta fase de desenvolvimento o grupo optou pela utilização da linguagem de programação Java, uma linguagem bastante familiar e produtiva para os elementos do grupo, foi também utilizada a API Weka para Java, e para ambiente de desenvolvimento, foi utilizado o Eclipse Oxygen. Para a interface foi utilizado Swing.

3.2 Pré-processamento do DataSet

A fim de ser possível utilizar os MFCCs foi necessário tratar os dados previamente fornecidos no ficheiro textitFrogs_MFCC.csv . O inteiro que se encontrava sempre presente no início de cada gravação foi eliminado, assim como o record_ID que se encontra no final da mesma porque na entrega intercalar estávamos a obter resultados de 100% em termos de indivíduos bem classificados por causa deste atributo, porque para cada indivíduo havia diversas gravações ambas com o mesmo record_ID e então isto erradamente estava a ser usado como atributo.

Na nossa implementação optamos por utilizar um software de *machine learning* desenvolvido pela Universidade de Waikato, Nova Zelândia. Este sotfware possui um mecanismo bastante intuitivo para tratar ficheiros *.arff*, sendo assim, decidimos converter o ficheiro original - *.csv* para *.arff*. Além disso foi necessário dividir o ficheiro em três ficheiros diferentes, um para cada parte da classificação: espécie, género e família. Esta divisão foi feita porque apenas é permitido um classificador de cada vez neste tipo de estruturas e então se os outros atributos não fossem eliminados iriam interferir na classficação.

3.3 Estrutura da Aplicação

O nosso trabalho é composto por três fases distintas:

- Seleção do tipo de classificação a aplicar (família, género ou espécie)
- Configuração do filtro a aplicar aos dados e à Árvore
- Configuração de opções para a Árvore
- Treino e teste da Árvore

3.3.1 Seleção do Tipo de Classificação

No contrutor da árvore é logo criado um objeto *Data Source* que vai permitir distinguir o tipo de classificação que se pretende usar: familia, género ou espécie. Como cada um dos tipo se encontra num ficheiro de dados diferente, a aplicação vai fazer *load* dos dados respetivos.

```
1     DataSource source ;
2
3     if(classifier_attr.equals("Family"))
4         source=new DataSource("class_familia.arff");
5     else if(classifier_attr.equals("Species"))
6         source = new DataSource("class_especie.arff");
7     else
8         source=new DataSource("class_genero.arff");
9
10      data = source.getDataSet();
```

Seleção do Tipo de Classificação

3.3.2 Configuração do Filtro para os Dados

Além do tipo de classificação é necessário determinar a percentagem de dados que será utilizada para treino da árvore, sendo que os dados são divididos em N partes e é escolhida aleatoriamente a parte para teste. O código referente a esta segunda fase é o seguinte:

```
1     // use StratifiedRemoveFolds to randomly split the data
2     StratifiedRemoveFolds filter = new StratifiedRemoveFolds();
3
4     // set options for creating the subset of data
5     String[] options = new String[6];
6
7     options[0] = "-N";                  // indicate we want to set
          the number of folds
8     options[1] = Integer.toString(division);  // split the data
          into n random folds
9     options[2] = "-F";                  // indicate we want to
          select a specific fold
10    options[3] = Integer.toString(1);   // select the first fold
11    options[4] = "-S";                  // indicate we want to set
          the random seed
12    options[5] = Integer.toString(1);   // set the random seed to 1
13
14    filter.setOptions(options);         // set the filter options
15    filter.setInputFormat(data);        // prepare the filter for
          the data format
16    filter.setInvertSelection(false);   // do not invert the
          selection
17
18    // apply filter for test data here
19    Instances test = Filter.useFilter(data, filter);
20
21    //  prepare and apply filter for training data here
22    filter.setInvertSelection(true);
23    Instances train = Filter.useFilter(data, filter);
```

Configuração do Filtro para os Dados

3.3.3 Configuração de Opções para a Árvore

Ainda na parte das configurações a árvore deve-se determinar o tipo de pruning a ser utilizado. Podendo este ser o standard, sem pruning, com pruning reduzido ou ainda com uma determinada percentagem de confiança. Esta última não deve ultrapassar o valor de 0.5.

```
1      // invert the selection to get other data
2      tree = new J48();
3      tree.setUnpruned(prun);
4      ArrayList<String> tree_options= new ArrayList();
5      if(reduced_error_prunning && !prun) {
6           tree_options.add("-R");
7           System.out.println("reduced error");
8      }
9      else if(confidence&& !prun && confidence_perc>0 &&
           confidence_perc <=0.5 ) {
10          System.out.println("confidence");
11          tree_options.add("-C");
12          String temp=""+confidence_perc;
13          tree_options.add(temp);
14     }
15
16     tree.setOptions(tree_options.toArray(new String[0]));
```

Configuração de Opções para a Árvore

3.3.4 Treino e teste da Rede

Por último, com todos os filtros aplicados e o ficheiro de dados selecionado, a biblioteca *weka.classifiers* permite-nos obter a árvore de treino e de teste com alguma facilidade:

```
1      tree.buildClassifier(train);
2      // evaluate classifier and print some statistics
3      eval = new Evaluation(train);
```

Treino e Teste da Árvore de Decisão

3.4 Detalhes Relevantes da Implementação

Foi criada um interface gráfica em *Swing* de modo a melhorar a experiência do utilizador e a facilitar a escolha do conjunto de requisitos para gerar a árvore de decisão. Nesta interface é possível escolher os parâmetros com que o algoritmo C4.5 será executado. Um dos parâmetros é o *pruning*: o algoritmo pode ser executado sem *pruning*, com *pruning* reduzido ou com *pruning* standard. Além disso é possível escolher o classificador que será usado para gerar a árvore: família, género ou espécie. Por último, é possível escolher qual a percentagem de dados que será utilizada para treinar a árvore de decisão.

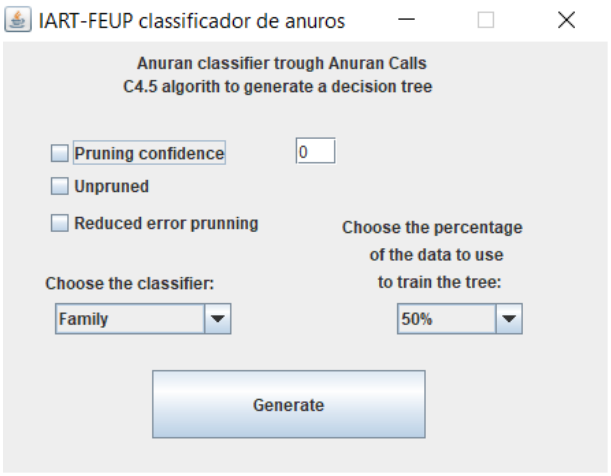

Figura 4: Interface - Requisitos

Após os dados serem inseridos na interface a nossa aplicação gera um árvore de decisão onde é possível classificar qualquer chamamento de anuro. Cada nó possui duas folhas que, à medida que se vai avançando na árvore levam a uma classificação cada vez mais apurada relativamente ao classificador escolhido. Quando deixam de haver folhas significa que se chegou ao resultado final

Figura 5: Interface - Árvore de Decisão

Além de árvore de decisão, a nossa aplicação também gera uma nova janela onde são apresentados os resultados relativos a essa mesma árvore. Esta janela mostra o número de nós e de folhas geradas assim como a classificação que a árvore gerou para cada um dos exemplos de teste. Com este exemplos é

possível averiguar a fiabilidade da árvore em questão, ou seja, a percentagem de instâncias que foi corretamente classificada.

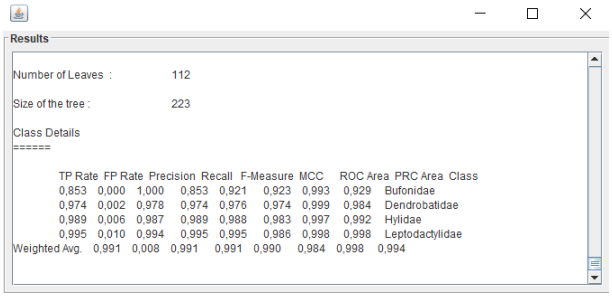

Figura 6: Interface - Resultados

4 Experiências

4.1 Experiência 1 : Classificação de famílias com diferentes percentagens de dados para treino

	Sem Pruning	Pruning Standard	Pruning Reduzido
Folhas	123	112	60
Número de Nós	245	223	119
Profundidade Máxima	14	13	14
Fiabilidade	99,2774	99,055	97,443

Tabela 1: Resultados para classificação de famílias com 50% dos dados para treino

	Sem Pruning	Pruning Standard	Pruning Reduzido
Folhas	123	112	60
Número de Nós	245	223	119
Profundidade Máxima	14	13	14
Fiabilidade	99,1106	99,1106	97,3874

Tabela 2: Resultados para classificação de famílias com 75% dos dados para treino

	Sem Pruning	Pruning Standard	Pruning Reduzido
Folhas	123	112	60
Número de Nós	245	223	119
Profundidade Máxima	14	13	14
Fiabilidade	99,375	99,5	98,125

Tabela 3: Resultados para classificação de famílias com 90% dos dados para treino

4.2 Experiência 2 : Classificação de géneros com diferentes percentagens de dados para treino

	Sem Pruning	Pruning Standard	Pruning Reduzido
Folhas	150	143	84
Número de Nós	299	285	167
Profundidade Máxima	14	14	12
Fiabilidade	99,3052	99,1106	96,9983

Tabela 4: Resultados para classificação de géneros com 50% dos dados para treino

	Sem Pruning	Pruning Standard	Pruning Reduzido
Folhas	150	143	84
Número de Nós	299	285	167
Profundidade Máxima	14	14	12
Fiabilidade	99,1106	99,8883	98,6648

Tabela 5: Resultados para classificação de géneros com 75% dos dados para treino

	Sem Pruning	Pruning Standard	Pruning Reduzido
Folhas	150	143	84
Número de Nós	299	285	167
Profundidade Máxima	14	14	12
Fiabilidade	98,875	98,625	97,125

Tabela 6: Resultados para classificação de géneros com 90% dos dados para treino

4.3 Experiência 3 : Classificação de espécies com diferentes percentagens de dados para treino

	Sem Pruning	Pruning Standard	Pruning Reduzido
Folhas	155	146	84
Número de Nós	309	291	167
Profundidade Máxima	14	12	12
Fiabilidade	99,1662	98,9994	96,776

Tabela 7: Resultados para classificação de espécies com 50% dos dados para treino

	Sem Pruning	Pruning Standard	Pruning Reduzido
Folhas	155	146	84
Número de Nós	309	291	167
Profundidade Máxima	14	12	12
Fiabilidade	99,333	99,2774	97,1651

Tabela 8: Resultados para classificação de espécies com 75% dos dados para treino

	Sem Pruning	Pruning Standard	Pruning Reduzido
Folhas	155	146	84
Número de Nós	309	291	167
Profundidade Máxima	14	12	12
Fiabilidade	99	99	97

Tabela 9: Resultados para classificação de espécies com 90% dos dados para treino

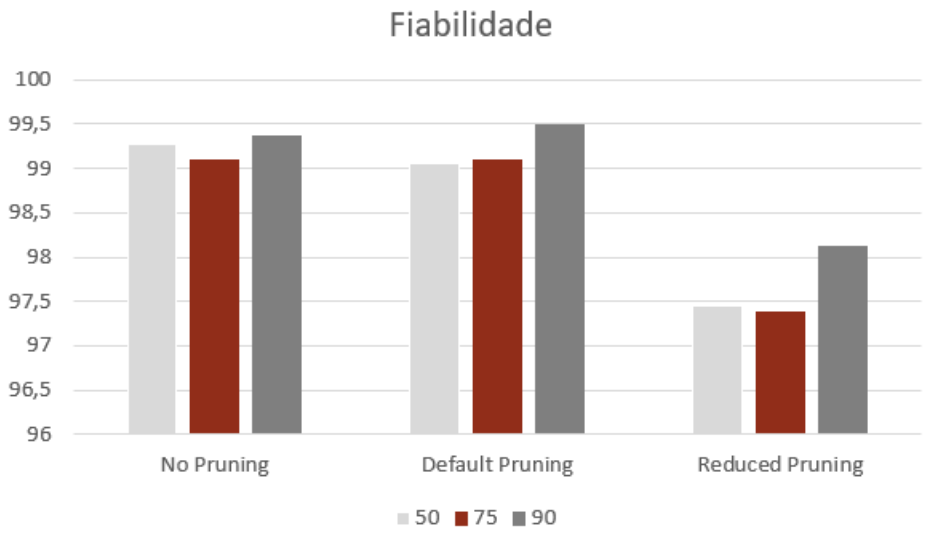

Figura 7: Gráfico da fiabilidade - família

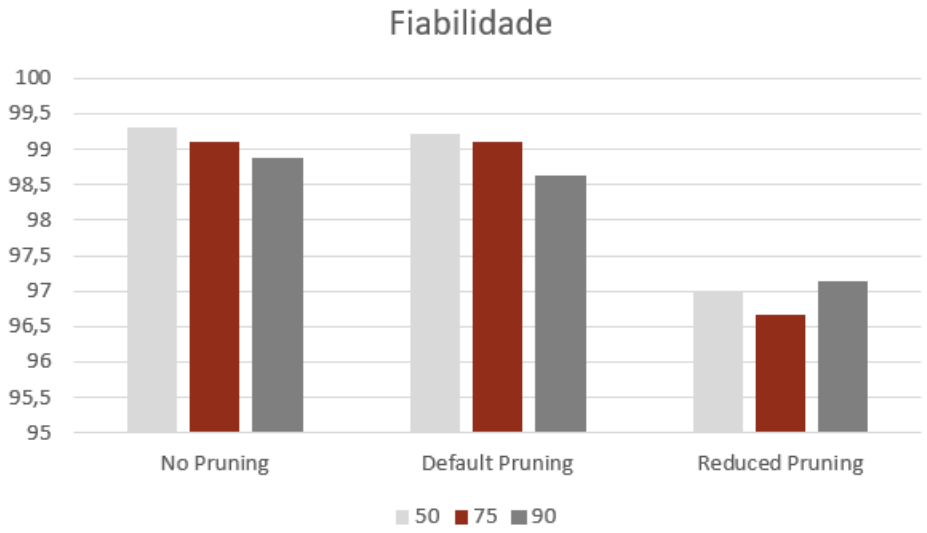

Figura 8: Gráfico da fiabilidade - género

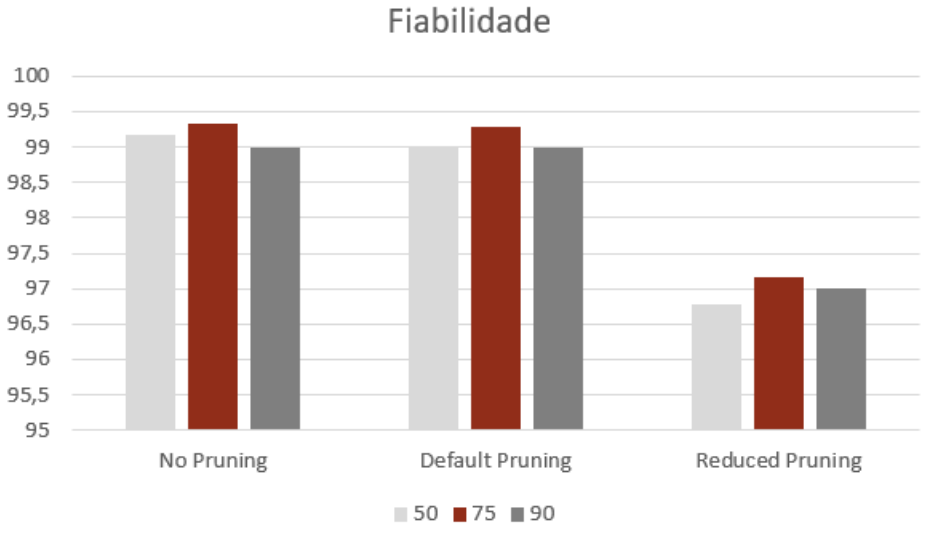

Figura 9: Gráfico da fiabilidade - espécie

5 Conclusões

É possível observar que em qualquer uma das experiências os dados relativos ao número de folhas, nós e profundidade máxima se mantém constante em cada uma das tabelas de *pruning* apesar de a percentagem de dados para treino ser alterada. Também é possível observar que em todas as experiências estes valores diminuem em função do *pruning*. Não havendo qualquer tipo de *pruning* os valores são de forma geral os mais elevados, seguindo os valores para o *pruning standard*, e por último, os valores mais baixos para *pruning* reduzido.

O *pruning* tem como objetivo reduzir o tamanho da àrvore de decisão, nomeadamente nas grandes árvores, pois estas correm o risco de sofrer *overfitting*, ou seja, o conjunto de teste ficar demasiado adequado ao conjunto de teste, e não a um conjunto mais genérico. Desta forma o *pruning* tenta reduzir o tamanho da árvore eliminando os nós que não acrescentam nova informação à árvore. Analisando os resultados obtidos, podemos concluir que o *pruning* surtiu efeito, quanto maior o índice de pruning aplicado, maior a redução no tamanho da árvore.

Relativamente à fiabilidade de cada experiência. É possível observar que, de uma forma geral, a fiabilidade é reduzida com o aumento do *pruning*. Mas este fenómeno já era esperado e bastante fácil de perceber. A fiabilidade traduz a percentagem de classificações corretamente efetuadas. Sem pruning a árvore de decisão vai-se assemelhar muito ao conjunto de treino, logo a sua fiabilidade também vai ser maior. Ao conseguirmos uma árvore mais genérica é óbvio que a fiabilidade para os dados usados seja menor.

Para sumarizar, consideramos que o *pruning* se revela bastante útil neste tipo de experiências, pois reduz o tamanho da árvore de decisão permitindo uma melhor classificação. Além disso, consegue manter a fiabilidade bastante elevada.

6 Melhorias

Uma possível melhoria seria usar transformar a árvore de decisão em algo mais dinâmico e de fácil compreesão para qualquer utilizador. Devido à elevada quantidade de nós e folhas torna-se complicado identificar cada espécie / família / género.

7 Recursos

7.1 Bibliografia

Referências

[1] Sonia Singh, Priyanka Gupta. *Comparative Study ID3, CART and C4.5 Decision Tree Algorithm: A Surve,.*, 2014.

[2] ID3 algorithm, Wikipedia
`https://en.wikipedia.orgwikiID3_algorithm`

[3] C4.5 algorithm, Wikipedia
`https://en.wikipedia.orgwikiC4.5_algorithm`

[4] Artigo do InescTec
https://repositorio.inesctec.ptbitstream12345678943091P-00K-S9H.pdf

[5] Slides da Unidade Curricular
https://paginas.fe.up.pteolIA1617APONTAMENTOS6_ASA.pdf

[6] Share Latex
https://www.sharelatex.com/learn

[7] Info Escola
https://www.infoescola.com/biologia/taxonomia/

[8] Weka 3: Data Mining Software in Java
https://www.cs.waikato.ac.nz/ml/weka/ https://www.cs.waikato.ac.nz/ml/weka/

7.2 Software

- Software - Weka 3.9
- Bibliotecas Weka
  - weka.classifiers.Evaluation
  - weka.classifiers.trees.J48
  - weka.core.Instances
  - weka.core.converters.ConverterUtils.DataSource
  - weka.filters.Filter
  - weka.filters.supervised.instance.StratifiedRemoveFolds
- Linguagem de programação - Java
- Interface Gráfica para Java - Swing
- IDE - Eclipse Oxygen

7.3 Percentagem de Trabalho Efetivo de Cada Elemento do Grupo.

- Daniel Pereira Machado - 33.3%
- José Pedro Dias de Almeida Machado - 33.3%
- Sofia Catarina Bahamonde Alves - 33.3%

8 Apêndice

8.1 Manual Do Utilizador

1. Criar um novo projeto Java no IDE Ecplise
2. Usar a localização da pasta IART para iniciar o projeto
3. Adicionar o ficheiro weka.jar através do Java Build Path que se encontra nas propriedades do projeto
4. Correr a aplicação