Universidade do Porto

Faculdade de Engenharia

FEUP

Aplicação de C5.0 ao diagnóstico de Parkinson

Relatório Final

Inteligência Artificial

3º ano do Mestrado Integrado em Engenharia Informática e Computação

Grupo E4_3:

Henrique Ferrolho – ei12079 – henriqueferrolho@gmail.com

João Pereira – ei12023 – pereiraffjoao1993@gmail.com

31 de Maio de 2015

Índice

1. Objectivo

Este projecto, desenvolvido no âmbito da unidade curricular de Inteligência Artificial, consiste na aplicação do algoritmo C5.0 para o diagnóstico da doença de *Parkinson*.

Inicialmente são utilizados exemplos (*data sets*) na derivação das regras de classificação para os elementos de um domínio em análise. Os elementos de um domínio definem-se por um conjunto de variáveis que irão formar os parâmetros das regras a derivar após o processo de aprendizagem. As variáveis podem ser identificadas consoante produzem ou não uma melhor definição do domínio em análise.

O programa final deve, através do algoritmo C5.0 e com base num conjunto de dados, previamente, disponibilizados, determinar uma Árvore de Decisão que traduz as regras de classificação desse conjunto de dados. O conjunto de dados é, posteriormente, cuidadosamente analisado de forma a verificar a eventual necessidade de pré-processamento. O modelo obtido deve poder depois ser utilizado na classificação de novos casos.

O objectivo deste projecto é a determinação da árvore de decisão que traduz essas regras no diagnóstico de Parkinson.

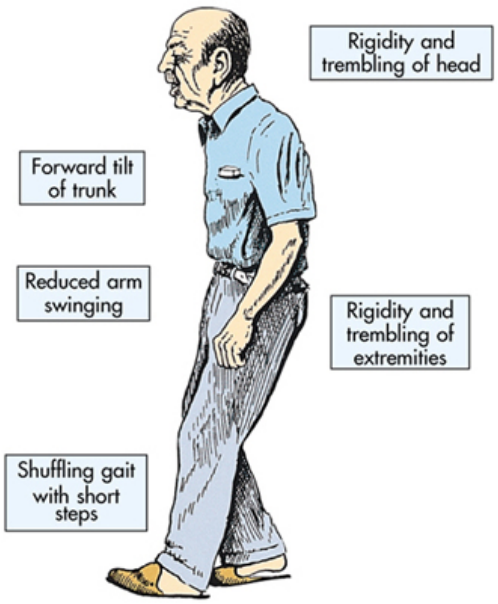
2. Especificação

2.1. Análise

Detalhes do tema

Com o término do desenvolvimento do projecto, aplicou-se uma ferramenta capaz de construir uma árvore de decisão para o diagnóstico de *Parkinson*.

Antes de mais convém abordarmos o que é a doença de *Parkinson*. *Parkinson* é uma doença com desordem degenerativa do sistema nervoso central.



No início do curso da doença, os sintomas mais óbvios relacionavam-se com o movimento; entre eles destacam-se tremores, rigidez, lentidão de movimentos e dificuldade em andar.

Posteriormente, podem surgir problemas comportamentais e mentais, como a demência, que ocorrem normalmente em estágios mais avançados da doença. A depressão é o sintoma psiquiátrico mais comum.

Ilustração de cenários

Um exemplo da aplicação da ferramenta neste contexto é o auxílio aos médicos no diagnóstico da doença. O médico poderá usar a aplicação para confirmar o seu parecer sobre o estado de saúde de um paciente, ou para tirar alguma dúvida no diagnóstico feito pelo próprio.
Outro exemplo é a hipótese de qualquer pessoa poder auto diagnosticar-se com o seu próprio smartphone, através de uma aplicação que use esta ferramenta.

Training data set

Vários tipos de artefactos sonoros de todos os indivíduos foram gravados (26 amostras de voz, incluindo vogais, números, palavras e frases curtas). Posteriormente, um grupo de 26 características baseadas em frequências linear e temporal foram extraídas de cada amostra de voz. A UPDRS (*Unified Parkinson's Disease Rating Scale*) de cada paciente (que é determinada por um médico especialista) também está disponível na base de dados.

Test data set

Depois de o conjunto de dados de treino ter sido recolhido, continuou-se a recolher dados para um novo conjunto de dados independente, a partir da análise de mais indivíduos com doença de Parkinson, avaliados pelo mesmo especialista. Durante a recolha deste conjunto de dados, 28

pacientes com doença de Parkinson foram convidados a suster as vogais 'a' e 'o' três vezes, um total de 168 gravações. Os mesmos 26 recursos foram extraídos das amostras de voz desse conjunto de dados. Este conjunto de dados pode ser usado como um conjunto independente de teste para validar os resultados obtidos no conjunto de treino.

2.2. Abordagem

Técnicas

Boosting

Uma inovação incorporada no C5.0, comparando com o C4.5, é o boosting adaptativo, baseado no trabalho de Rob Schapire e Yoav Freund. A ideia é gerar vários classificadores em vez de apenas um. Quando um caso novo precisa de ser classificado, cada classificador vota na classe onde prevê que esse caso pertence. Posteriormente os votos são contados para determinar a classe final.

Isto é possível da seguinte forma:
Uma primeira árvore de decisão é gerada a partir dos casos de treino (tal como se fez anteriormente). Geralmente, este classificador classifica erradamente alguns dos casos de treino.
Durante o processo de criação do segundo classificador, é prestada mais atenção a esses casos de forma a tentar que, desta vez, sejam classificados correctamente pela nova árvore de decisão. Consequentemente, o segundo classificador costuma ser diferente do primeiro. Contudo, o segundo classificador também cometerá algumas classificações erróneas, e estas tornam-se o foco de atenção durante a contrução do terceiro classificador.
Este processo continua durante o número pré determinado de iterações, mas pára se o classificador mais recente for extremamente preciso, ou se for extremamente impreciso.

Winnowing

O algoritmo C5.0 consegue também prever quais os atributos que são relevantes na classificação e quais não o são. Esta técnica é conhecida como ***winnowing*** e é especialmente utilizada para lidar com datasets de grandes dimensões.

Algoritmos e sua breve explicação

Input: Exemplo, Atributo alvo, Atributo

Output: Árvore de decisão

Algoritmo:

- Verificar a classe base
- Construir uma *DT* (decision tree) usando o *training data*
- Descobrir o atributo com maior **ganho de informação**

- Para cada atributo t*i* ∈ D, aplicar a *DT* para determinar a sua classe uma vez que a aplicação de um dado tuplo para uma *DT* é relativamente simples.

Casos base:

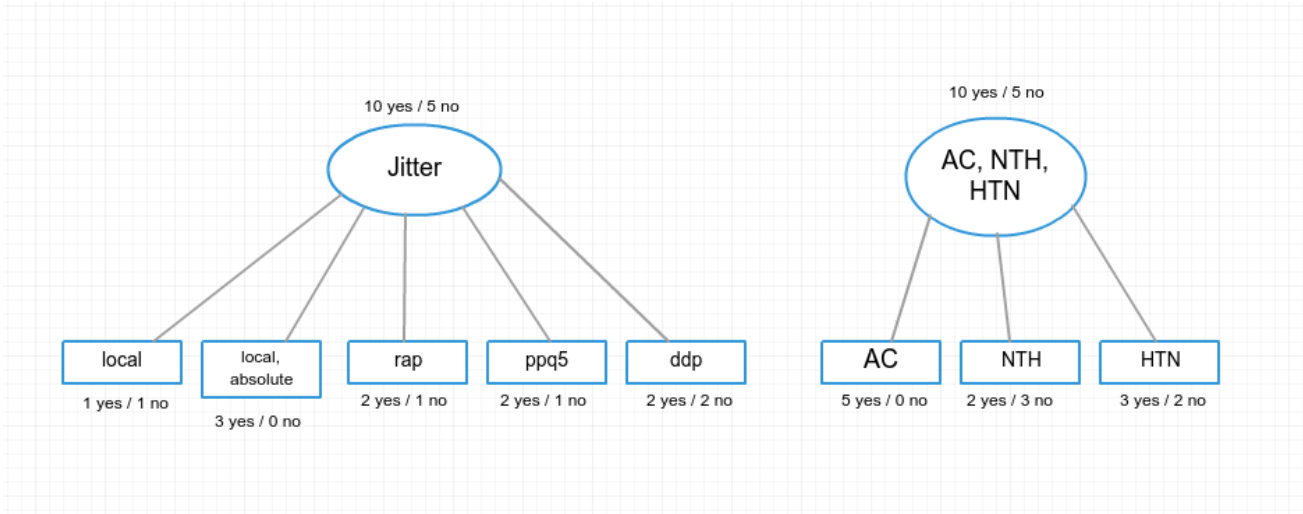
1. Todos os exemplos do conjunto de treino pertencem à mesma classe, isto é, uma folha da árvore marcada com essa classe é retornada;
2. O conjunto de treino estando vazio é retornada uma folha caraterizada por insuficiência;
3. A lista de atributos estando vazia é devolvida uma folha contendo a classe mais frequente ou a disjunção de todas as classes.

Métricas

A grande dificuldade deste algoritmo consiste em descobrir qual o atributo pelo qual se vai exercer a decisão. Este atributo pode ser o inicial ou um intermédio, e o método pelo qual se escolhe está presente na métrica que o algoritmo contempla.

A métrica utilizado pelo algoritmo C5.0 é o **ganho de informação** (*information gain*). Esta métrica consiste em obter o maior número de elementos dos conjuntos puros. Os conjuntos puros são aqueles em que os resultados balançam todos para o mesmo lado, isto é, ou apontam todos para o **sim** ou apontam todos para o **não**. A **pureza** de um conjunto é obtida medindo a **incerteza** de uma classe e esta é dada a partir do cálculo da *entropia* da sua divisão.

Vejamos o seguinte exemplo:



Estamos no início da árvore e ainda não houve nenhuma divisão dos dados. Podemos escolher entre dois atributos: Jitter ou AC, NTH, HTN. A escolha incidirá sobre o atributo que origina o maior ***information gain***. No exemplo dado, existem dois subconjuntos puros em ambos os conjuntos. Neste caso a escolha iria recair sobre o conjunto que apresentasse maior ganho.

A fórmula do ganho é a seguinte:

$$Gain(S,\ A)\ =\ H(S)\ -\ \sum_{V\ \in Values(A)} \frac{|Sv|}{|S|} H(Sv)$$

V - valores possíveis de A

S - conjunto de exemplos {X}

Sv - subconjunto onde Xa = V

H(S) é o cálculo da entropia:

$$H(S)\ =\ -\ p_+ * log_2 p_+\ -\ p_- * log_2 p_-$$

S - subconjunto de exemplos de treino

p_+/p_- - percentagem de exemplos positivos/negativos em S

A entropia devolverá o número de *bits* necessários para dizer se X é positivo ou negativo.

Na imagem acima anexada, *local, absolute (Jitter)* resultaria em 0 *bits* pois é um caso de pureza a 100%, porém no caso de *ddp (Jitter)* resultaria em 1 bit pois o nível de pureza é de 50 % (impuro).

Exemplo de fórmula final (para conjunto do Jitter):

$$Gain(S,\ Jitter)\ =$$
$$H(Jitter)\ -\frac{2}{15} * H(Jitter_{local}) - \frac{3}{15} * H(Jitter_{local,absolute}) - \frac{3}{15} * H(Jitter_{rap}) - \frac{3}{15} * H(Jitter_{ppq5}) - \frac{4}{15} * H(Jitter_{ddp})$$

Heurísticas

A heurística de seleção de teste original baseada em ganho de informação não foi satisfatória. Haviam problemas como os atributos com um grande número de resultados serem favoráveis em relação aos atributos com um número menor e alguns atributos dividiam os dados em um grande número de classes únicas (*singleton*), como por exemplo os *id's*. Estas classes, por serem únicas, possuíam o seu nível de pureza ao máximo (100%).

Com isto foi necessário criar algo que contrariasse estes problemas e assim foi criado o ***gain_ratio***. Este cálculo é realizado posteriormente ao ***gain*** e utiliza outra fórmula para prever os problemas acima mencionados:

$$Split(S,\ A)\ =\ -\ \sum_{V\ \in\ Values(A)} \frac{|Sv|}{|S|} log \frac{|Sv|}{|S|}$$

A - atributo candidato

V - possíveis valores de A

S - Conjunto de exemplos {X}

Sv - subconjunto onde Xa = V

O cálculo final para contornar os atributos *singleton* é:

$$GainRatio(S,\ A)\ =\ \frac{Gain(S,\ A)}{Split(S,\ A)}$$

Esta é a heurística utilizada no algoritmo C5.0.

3. Desenvolvimento

3.1 Ferramentas e Ambiente de Desenvolvimento

O projecto foi desenvolvido no sistema operativo Ubuntu, versão 15.04. A linguagem de programação utilizada pelo grupo foi C++, usando o Eclipse Luna como IDE.
O grupo utilizou uma ferramenta de *data mining* sofisticada, o algoritmo C5.0, que permite descobrir padrões que delineiam categorias, construindo classificadores que permitirão fazer previsões sobre um domínio em análise.

3.2 Detalhes Relevantes

Neste ponto o grupo ir debater o porquê da escolha do algoritmo C5.0 em relação ao anteriormente previsto, o C4.5.
No que diz respeito a **velocidade**, o algoritmo C5.0 é bem mais rápido que o C4.5. A diferença das suas velocidades chega a atingir algumas ordens de grandeza.
O C5.0 é mais eficiente no uso de **memória** do que o C4.5, pois geralmente usa menos memória na construção das regras.
Os conjuntos de regras do algoritmo C5.0 têm **taxas de erro** mais baixas que o C4.5 no que diz respeito a casos "invisíveis".
Os seus conjuntos de regras apresentem a mesma **precisão** nas previsões, porém o conjunto de regras do C5.0 é menor, da mesma maneira que os resultados entre ambos os algoritmos são semelhantes mas as **árvores de decisão** apresentadas pelo C5.0 são razoavelmente menores, sendo deste modo mais simples e coeso.
Neste algoritmo é possível a utilização de ***boosting*** para construir várias árvores de decisão, tentando melhorar as anteriores. Desta forma, é possível fazer previsões mais precisas.
O C5.0 é possível ponderar (***weighting***) diferentes atributos e tipos de classificação incorretas.
A ferramenta permite ainda resolver a questão do ***label noise***, que ocorre quando os valores dos atributos de duas instâncias são exatamente iguais, mas diferem no resultado final.

4. Experiências

4.1. Objetivos

Construção de classificadores

A forma geral do comando para executar o C5.0 em Unix é a seguinte: c5.0 -f filestem [options]
Este comando invoca o C5.0 com a opção -f para identificar o nome da aplicação (**filestem**).
Existem várias opções que afectam o tipo de classificador que o C5.0 produz, bem como a forma como esse classificador é construído. Muitas das opções têm valores por defeito que devem ser satisfatórios para a maioria das aplicações.

Avaliação

Os classificadores construídos pelo C5.0 são avaliados com os dados de treino, a partir dos quais foram gerados, e também com dados de teste, não vistos previamente, se estiverem disponíveis num ficheiro à parte.

Os resultados da árvore de decisão dos casos em parkinson.data são:

Árvore de decisão	
Tamanho	Erros
55	168 (16.2%)

A coluna da esquerda mostra o tamanho de folhas da árvore que não estão vazias. A coluna da direita mostra o número e a percentagem de casos mal classificados.
Neste caso, a árvore tem 55 folhas e classifica erradamente 168 dos 1040 casos, uma taxa de erro de 16.2%.

Quando não há mais de 20 classes, a performance dos casos de treino é ainda analisada numa matriz de confusão que aponta que tipo de erros foram feitos.

(a)	**(b)**	**<- classifcado como**
396	124	**(a): class 0**
44	476	**(b): class 1**

Neste caso, a árvore de decisão classifica erradamente:

- **124** casos saudáveis (classificou com doença de Parkinson)

- **44** casos com a doença de Parkinson (classificou como casos saudáveis)

Para algumas aplicações, especialmente aquelas com muitos atributos, pode ser util saber quanto é que cada atributo contribui para o classificador. Isso é indicado depois da matriz de confusão:

Attribute usage:
100% htn
96% jitter ppq5
76% shimmer apq5
75% standard deviation
71% median pitch
69% shimmer apq11
39% fraction of locally unvoiced frames
34% jitter rap
31% maximum pitch
28% number of voice breaks
25% mean pitch
23% minimum pitch
23% standard deviation of period
21% shimmer local
16% ac
8% shimmer local db
7% number of pulses
4% mean period
4% jitter local absolute
3% shimmer apq3
3% jitter ddp
1% nth

A percentagem associada a cada atributo corresponde à percentagem de casos de treino em parkinson.data para os quais o valor desse atributo é conhecido e é usado para prever uma classe.
Por exemplo: a segunda entrada significa que a árvore de decisão usa um valor conhecido de *jitter ppq5* ao classificar **96%** dos casos de treino.
Os atributos para os quais esta percentagem é inferior a **1%** não aparecem nesta lista.

Se o ficheiro **parkinson.test** existir (ficheiro que contém os casos de teste), a performance do classificador é sumarizada num formato semelhante ao dos casos de treino.

Árvore de decisão

Tamanho	Erros
55	74 (44.0%)

(a)	(b)	<- classifcado como
0	0	**(a): class 0**
74	94	**(b): class 1**

Um classificador genérico muito simples prevê que cada caso novo pertence à classe mais comum dos dados de treino.

Boosting

A flag -t x comanda ao C5.0 para construir até x classificadores da forma que acabou de ser descrita. Existe a flag -b, que é exatamente o mesmo que usar a flag -t 10.

Ensaios sobre conjuntos de dados numerosos, grandes e pequenos, mostraram que um boosting com uma média de 10 classificadores reduz a taxa de erro dos casos de teste em cerca de 25%.

Neste caso, o comando `c5.0 -f parkinson -b` origina sete árvores de decisão. O resumo da performance individual e conjunta destas árvores nos casos de teste é:

	Árvore de decisão	
Ensaio	**Tamanho**	**Erros**
0	55	156 (15.0%)
1	24	309 (29.7%)
2	25	259 (24.9%)
3	4	485 (46.6%)
4	18	359 (34.5%)
5	2	480 (46.2%)
6	7	401 (38.6%)
boost		152 (14.6%)

(a)	(b)	<- classifcado como
459	50	**(a): class 0**
102	429	**(b): class 1**

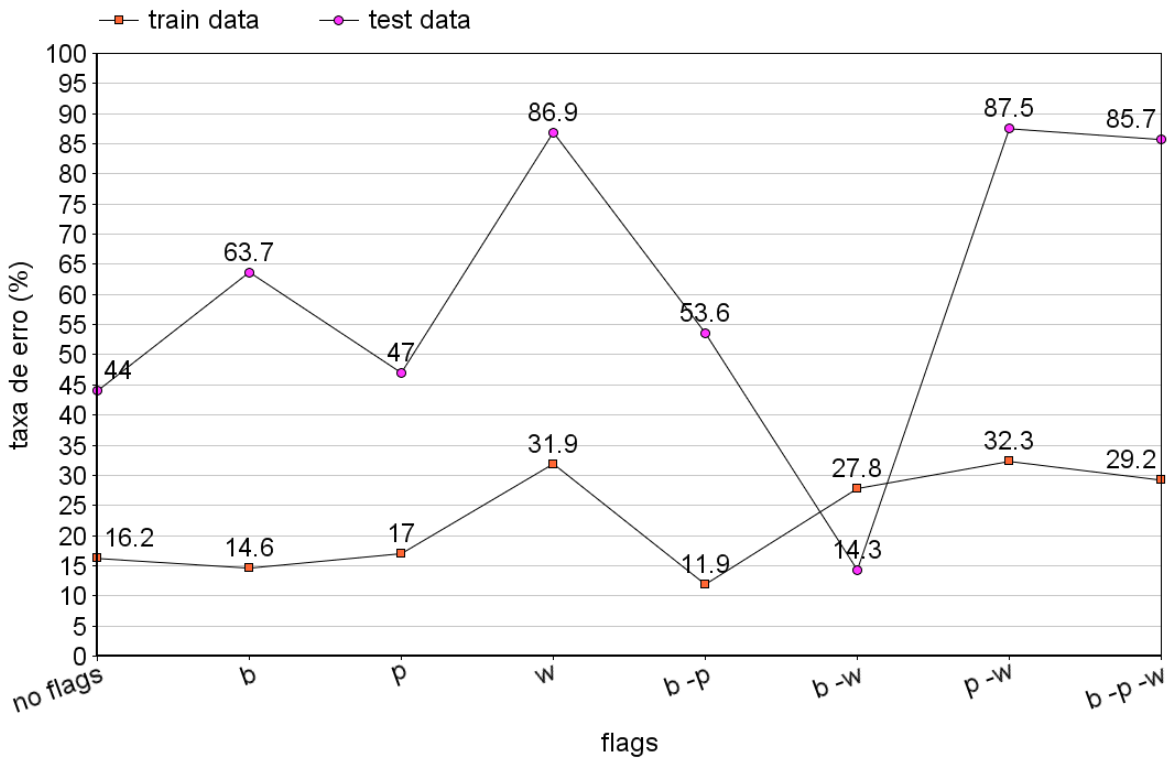
A performance de cada classificador construído em cada etapa é resumida em linhas separadas. A linha com a etiqueta boost apresenta o resultado da votação de todos os classificadores.

A árvore de decisão construída na primeira etapa é idêntica àquela produzida sem a flag -b. Algumas das árvores subsequentes, geradas a prestar mais atenção a certos casos, têm taxas de erro superiores. Contudo, quando as árvores se combinam para votar, a previsão final tem uma taxa de erro mais baixa.

4.2. Resultados

Os resultados obtidos podem ser consultados na tabela e gráfico seguintes:

	no flags	**-b**	**-p**	**-w**	**-b -p**	**-b -w**	**-p -w**	**-b -p -w**
train data	16.2	14.6	17	31.9	11.9	27.8	32.3	29.2
test data	44	63.7	47	86.9	53.6	14.3	87.5	85.7



A tabela seguinte contém os mesmos dados que a anterior, mas está ordenada da melhor para a pior taxa de erro para o conjunto dos dados de treino. A melhor taxa de erro é obtida com o uso de **boosting** e de **prunning**.

	-b -p	**-b**	**no flags**	**-p**	**-b -w**	**-b -p -w**	**-w**	**-p -w**
train data	11.9	14.6	16.2	17	27.8	29.2	31.9	32.3
test data	53.6	63.7	44	47	14.3	85.7	86.9	87.5

A tabela seguinte apresenta, novamente, os mesmos dados que as anteriores. Contudo, os dados estão ordenados da melhor para a pior taxa de erro para o conjunto dos dados de teste.
A taxa de erro mais baixa é obtida quando se usa em conjunto **boosting** com **winnowing**.

	-b -w	**no flags**	**-p**	**-b -p**	**-b**	**-b -p -w**	**-w**	**-p -w**

train data	27.8	16.2	17	11.9	14.6	29.2	31.9	32.3
test data	14.3	44	47	53.6	63.7	85.7	86.9	87.5

Opção de poda avançada

O algoritmo C5.0 constrói árvores de decisão em duas fases: primeiramente, uma árvore grande é construída para classificar todos os dados, e posteriormente essa árvore é podada para remover partes que possam ter uma taxa de erro relativamente elevada. Este processo de poda é aplicado em primeiro lugar a cada sub árvore para decidir se essa sub árvore deve ser substituída por uma folha ou por um sub ramo, e finalmente analisa-se a performance da árvore como um todo.
A flag **-g** desativa esta segunda fase de poda, e resulta geralmente em árvores de decisão maiores.
Em algumas aplicações, pode ser benéfico desativar esta poda usando a flag.

De seguida encontram-se os resultados obtidos com o uso desta flag isolada, e juntamente com as flags que nos testes em cima descritos obtiveram o melhor resultado nos dados de teste.

	-g	**-b -g -w**
train data	15.3	29.5
test data	44	12.5

5. Conclusões

Com o término do projecto o grupo pode concluir que houve um progresso assinalável no que diz respeito a algoritmos de tratamento de dados, nomeadamente os que abordam árvores de decisão.
Uma das vantagens que o grupo assinalou durante o desenvolvimento do projecto é que as árvores de decisão são de simples interpretação, conduzindo a um fácil entendimento do seu funcionamento.

A abordagem profunda do algoritmo C5.0 permitiu ao grupo não só a interiorização das suas componentes e da sua lógica de funcionamento mas também contribuiu para um estudo dos seus precedentes: caso do algoritmo C4.5 e ID3.

Este projecto permitiu ainda reconhecer a aplicabilidade e utilidade deste tipo de ferramentas no quotidiano e o seu potencial para diversas áreas, como neste caso a medicina.

7. Recursos

7.1. Bibliografia

1. Aplicação de ID3 ou C4.5 ao Diagnóstico de Parkinson
2. A. S. Galathiya, A. P. Ganatra and C. K. Bhensdadia: Improved Decision Tree Induction Algorithm with Feature Selection, Cross Validation, Model Complexity and Reduced Error Pruning
3. Quinlan, Ross: C5.0: An Informal Tutorial
4. Russel, Stuart; Norvig, Peter: Artificial Intelligence: A Modern Approach. Terceira Edição, Prentice-Hall - 2010
5. Oliveira, Eugénio: Apontamentos das aulas teóricas de Inteligência Artificial, Ano Lectivo 2014/2015

7.2. Software

1. Ubuntu 15.04
2. Eclipse Luna, C++ IDE

7.3. Recursos

1. LibreOffice Writer
2. GitHub
3. UCI Machine Learning Repository - Parkinson Speech Dataset with Multiple Types of Sound Recordings Data Set

7.4. Percentagem de Trabalho dos elementos do grupo

1. Henrique Ferrolho - 50 %
2. João Pereira - 50 %

8. Apêndice

Flags

De seguida encontra-se uma tabela com as flags suportadas pelo C5.0.

-f aplicação	selecionar a aplicação
-t ensaios	usar boosting com o número de ensaios especificado
-b	usar boosting com 10 ensaios
-w	“joeirar” atributos antes de construir o classificador
-p	usar limiares suaves
-g	não usar poda global da árvore
-X folds	executar uma validação cruzada