Universidade do Minho

Escola de Engenharia

Licenciatura em Engenharia Informática

Unidade Curricular de

Sistemas de Representação de Conhecimento e Raciocínio

Ano Letivo de 2014/2015

Trabalho de grupo – 3º Exercício

Grupo nº 5

Daniel Caldas a67691

José Cortez a67716

Susana Mendes a63464

Xavier Rodrigues a67676

Maio, 2015

Resumo

Com o presente relatório pretendemos expor sucintamente as decisões tomadas ao longo do estudo e análise do comportamento de redes neuronais com base num problema específico. Iremos apresentar o caso de estudo, seguido de uma pequena análise do problema. Posteriormente, vamos expor a solução desenvolvida com o detalhe necessário. Finalmente, será feita uma análise dos resultados e apresentaremos as conclusões deste segundo exercício.

Área de Aplicação: Inteligência Artificial.

Índice

1 Introdução

1.1. Apresentação do Caso de Estudo

Neste trabalho prático é pedido que sejam analisados dados referentes ao estudo de fadiga mental inferidos através do comportamento de indivíduos aquando do uso dos seus computadores. A amostra que nos foi fornecido contém dados biométricos como a velocidade de pressão do teclado ou a velocidade de movimentação do rato. O desafio será então utilizar redes neuronais de modo a tentar calcular o estado de fadiga mental (numa escala de 1 a 7) em função desses parâmetros de entrada fornecidos no *dataset*.

1.2. Objetivos

O nosso objetivo para este exercício será, então, usar redes neuronais para fazer a análise dos dados e conseguirmos tirar conclusões acerca do uso adequado dessas mesmas redes, isto é, qual a topologia de rede que melhores resultados obteve e tentar perceber o porquê desses resultados terem surgido e fazer diferentes análises comparativas.

1.3. Estrutura do Relatório

O presente relatório encontra-se estruturado conforme os seguintes pontos:

- Introdução;
- Preliminares:
    - Estudo prévio;
- Abordagem ao problema:
    - Análise do *dataset;*
    - Decisões face à obtenção da solução
    - Packages de redes neuronais utilizados;
- Apresentação de resultados:
    - Deteção do nível de fadiga;
    - Deteção de existência ou ausência de fadiga;
    - Redefinição de uma escala para deteção de fadiga e respetivos resultados;
    - Testes e análise de resultados:
        - Testes inicias efetuados com o neuralnet;

        - Análise dos parâmetros de entrada, e efetuação de novos testes;
        - Testes com diferentes redes neuronais:
            - Package NNET;
            - Package MLP;
            - Package PCA NNET;
        - Análise comparativa da performance das diferentes redes;
- Conclusões.

2. Preliminares

2.1 Estudo prévio

A introdução a esta parte da matéria passou, primeiro, pela exposição de alguns conceitos teóricos relativamente a redes neuronais e como a filosofia destas se baseia nos neurónios do ser humano, ou seja, existe uma tentativa de replicação daquilo que acontece nas trocas de informação dentro do corpo do ser humano, para uma rede digital, ou computacional.

Inicialmente, usamos a ferramenta ***JustNN,*** mas esta treinava e explorava o conceito de redes neuronais de forma bastante superficial, pelo que de modo a ser aprofundado o conhecimento nesta matéria foram introduzidas outras ferramentas mais poderosas.

Numa das aulas práticas foi feita uma introdução da linguagem R. Esta mesma foi usada nas aulas seguintes, juntamente com o *package* de redes neuronais *neuralnet* para o treino de redes neuronais capazes de tomar decisões relativamente a problemas específicos.

3. Abordagem ao problema

3.1 Análise do *dataset*

Inicialmente por forma a conseguirmos perceber com que tipo de dados estávamos a lidar, a sua dimensão, significado e frequência foi feita uma análise detalhadas do ficheiro fornecido e estudados os parâmetros de entrada e saída para podermos também ter uma noção da distribuição dos valores.

De modo a evitar conglomerados (clusters) nos quais os dados possam estar viciados, foram reorganizados os dados do dataset de forma aleatória através do Excel.

Para o fim mencionado traçamos então alguns gráficos e calculámos a frequência média dos parâmetros e o seu desvio padrão.

3.1.1 Parâmetros de entrada

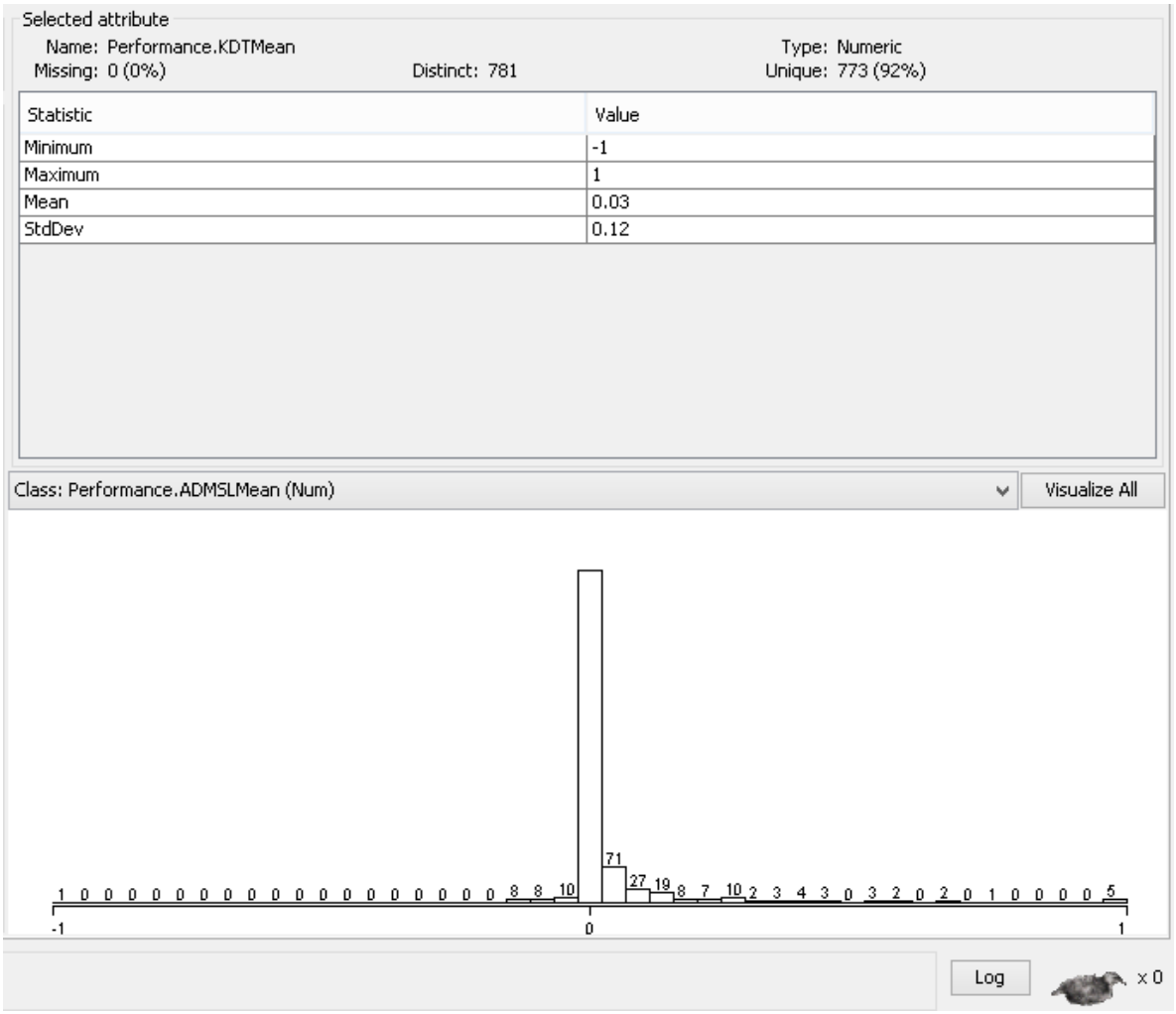

Figura 1 - Biométricas KDTMean.

Esta biométrica aparentemente é constante, isto pode ser-nos útil pois futuramente poderemos retirar este parâmetro de entrada da nossa rede e verificar se o mesmo de facto têm alguma influência no resultado produzido.

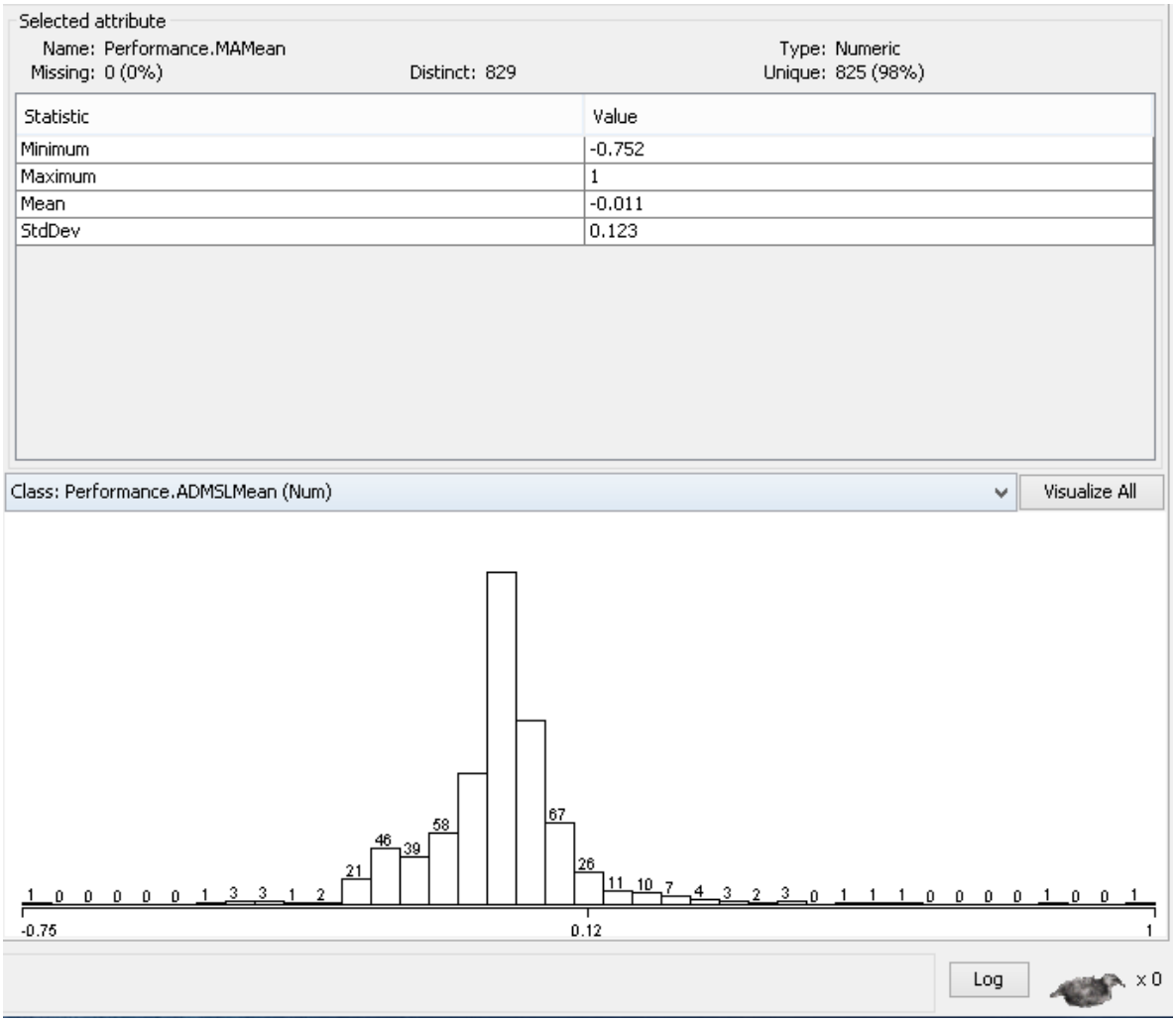

Figura 2 – Biométrica MAMean.

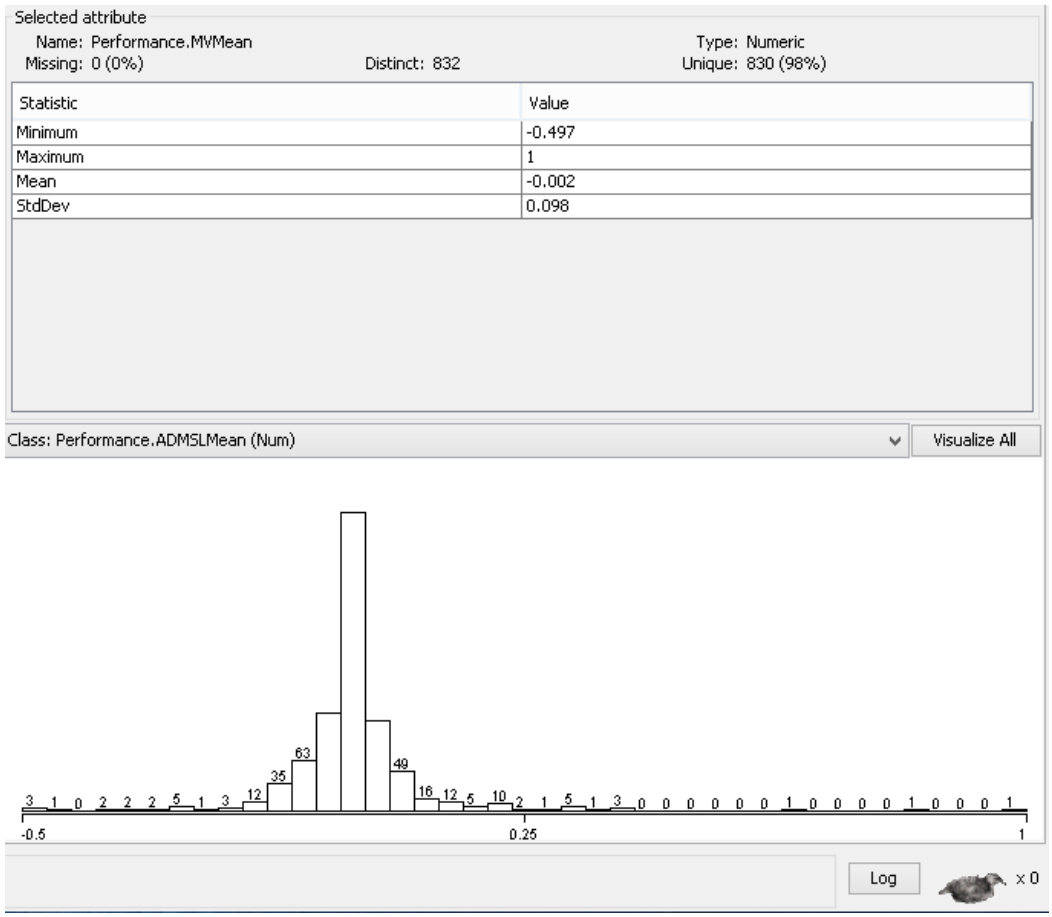

Figura 3 - Biométrica MVMean.

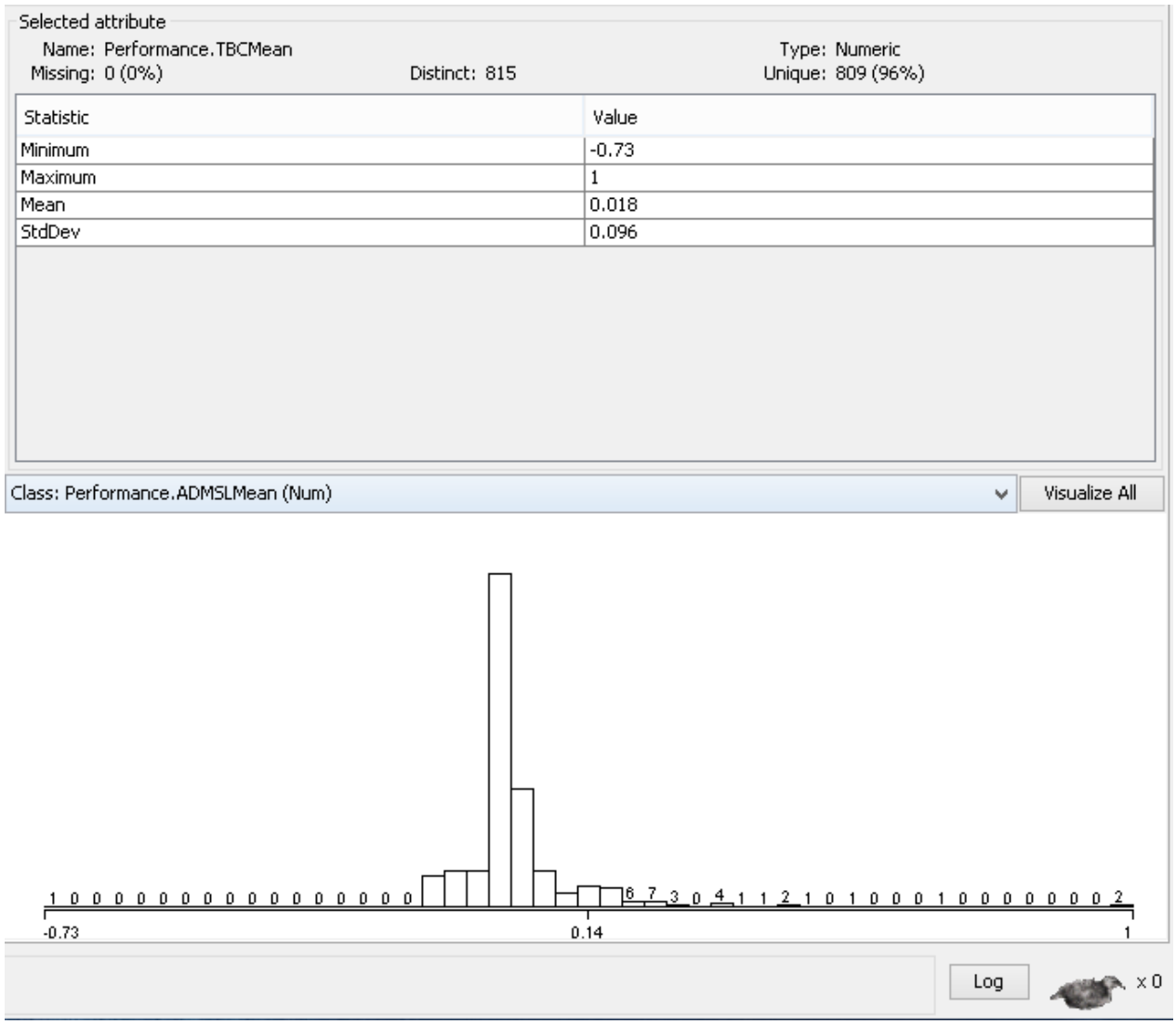

Figura 4 - Biométrica TBCMean.

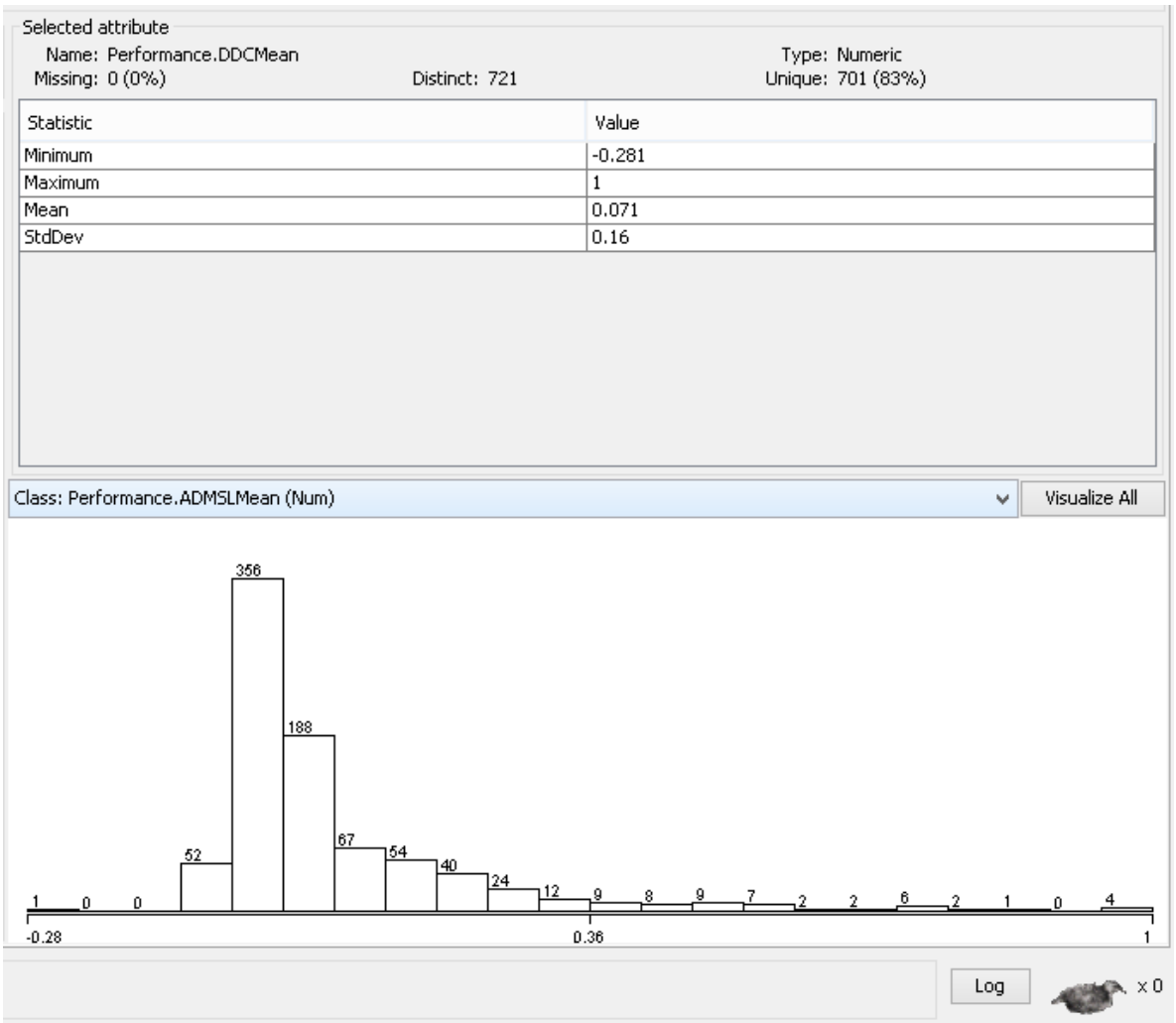

Figura 5 - Biométrica DDCMean.

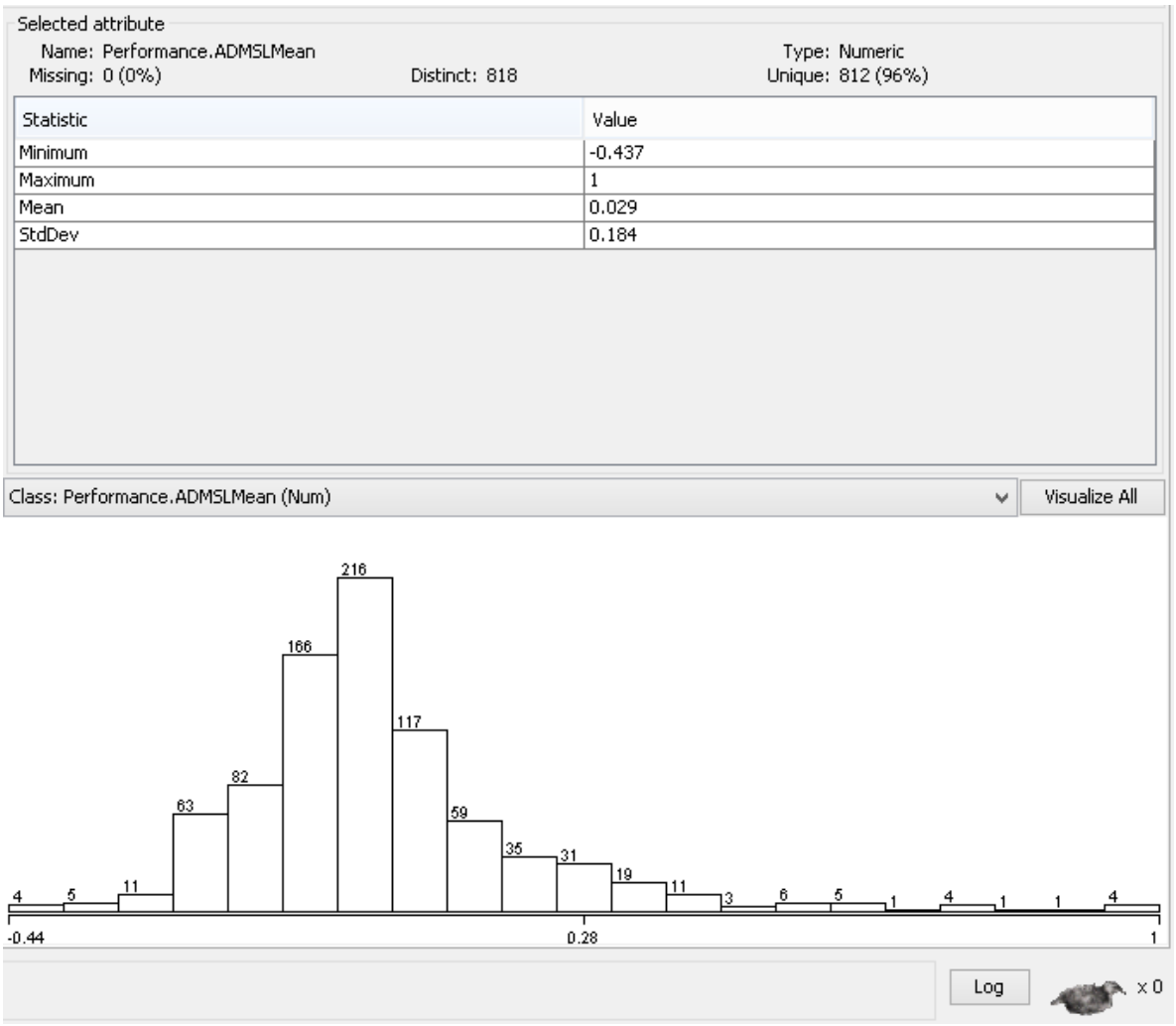

Figura 6 - Biométrica ADMSLMean.

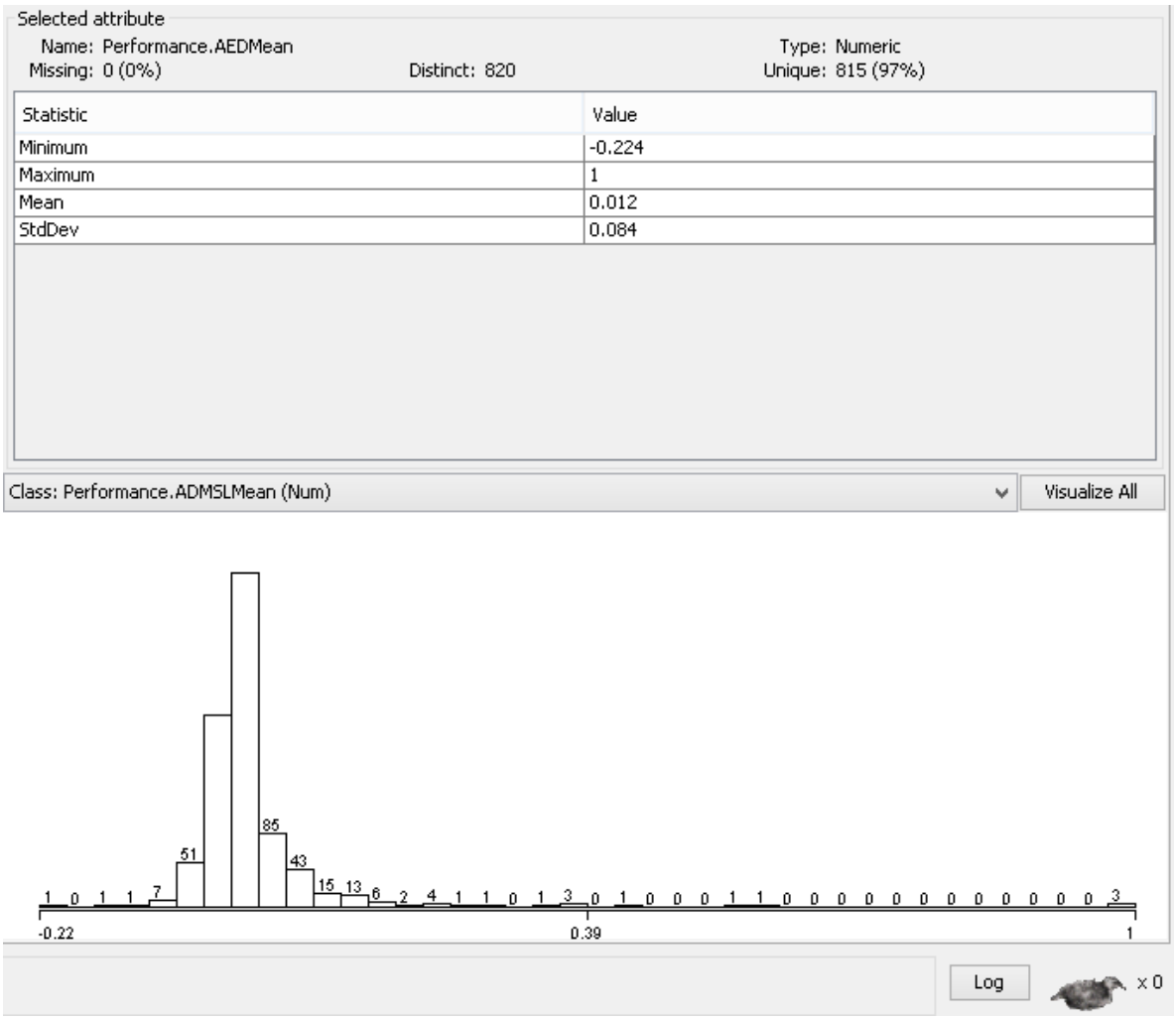

Figura 7 – Biométrica AEDMean.

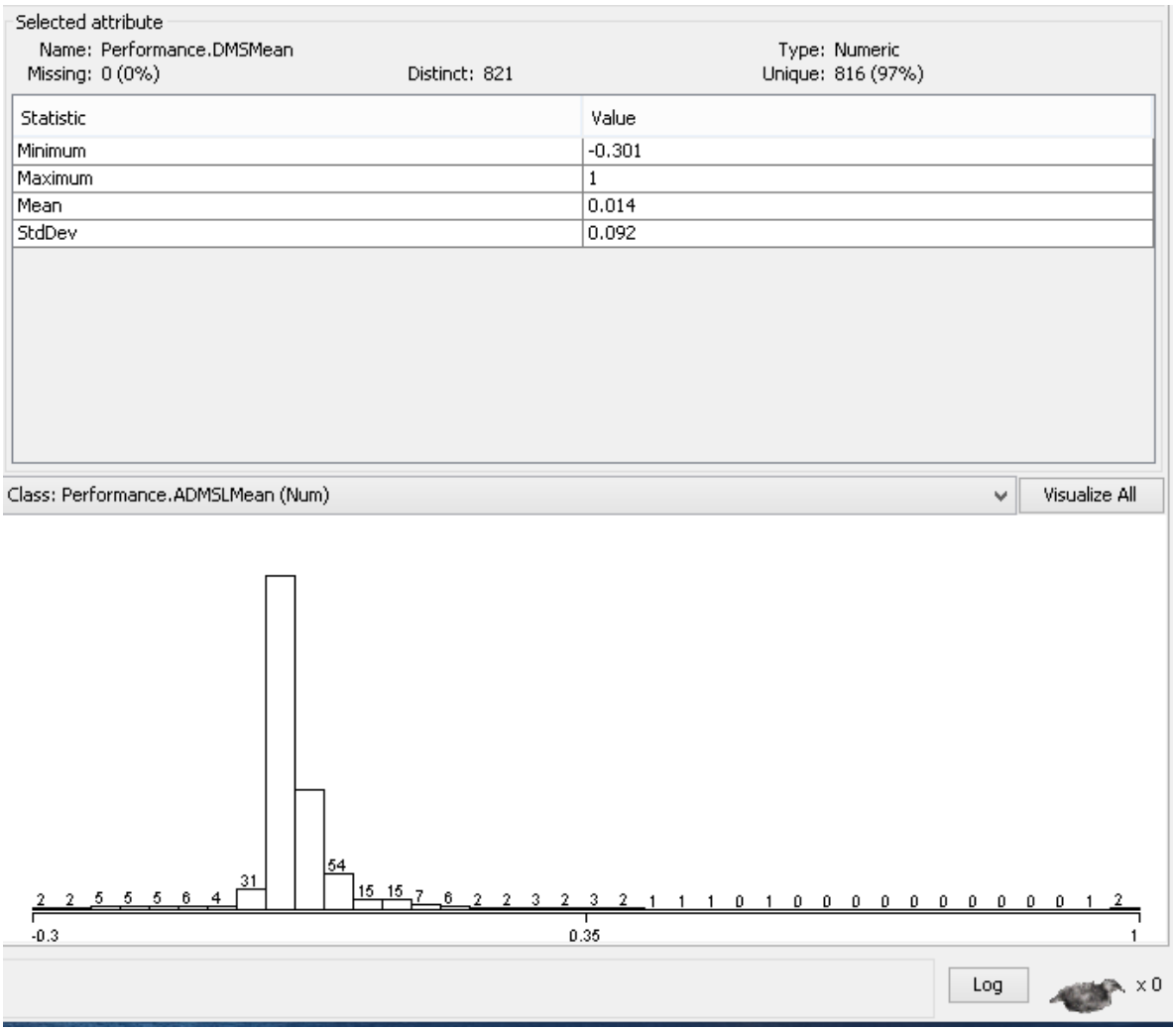

Figura 8 - Biométrica DMSMean.

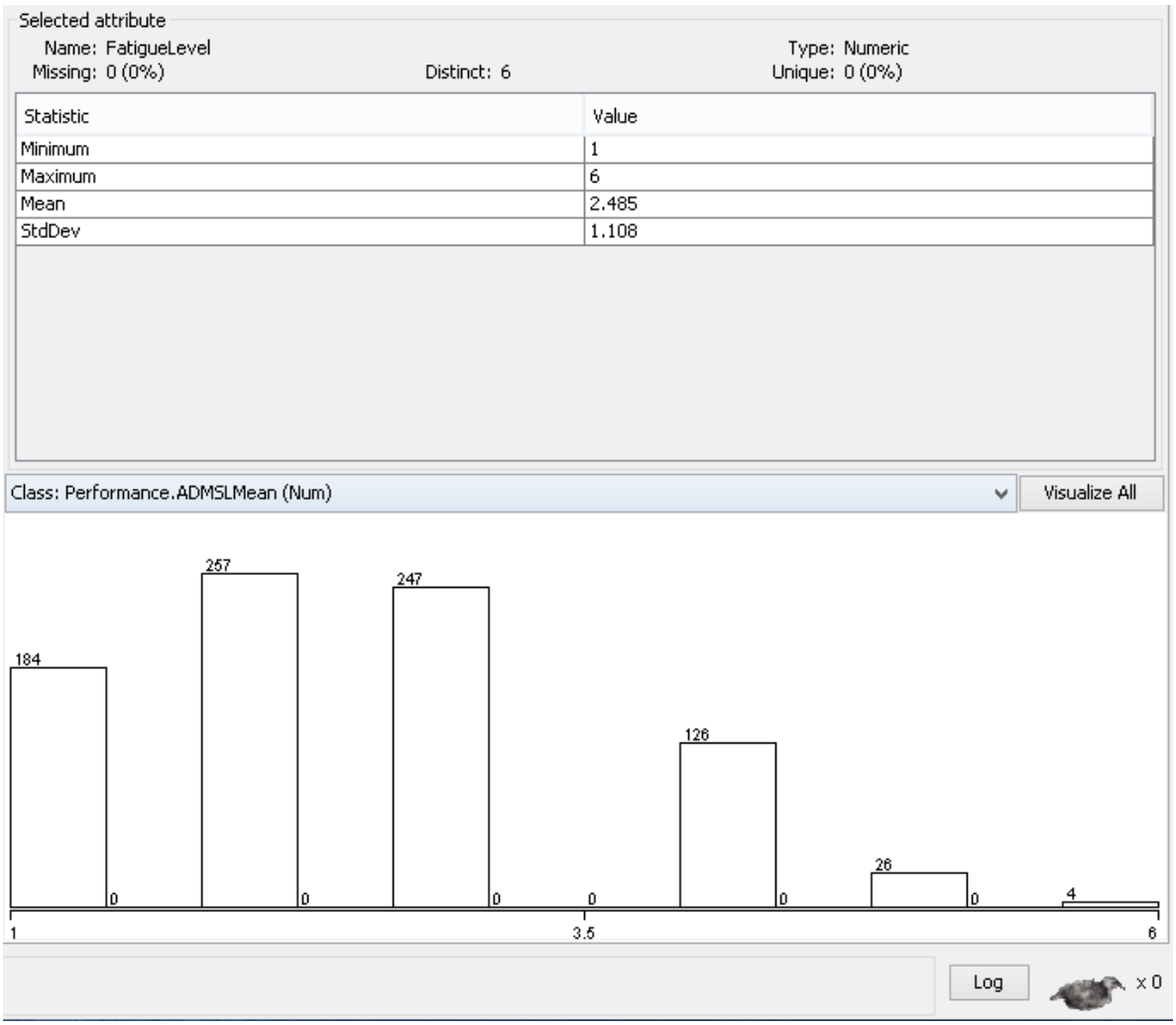

Figura 9 - FatigueLevel (Nível de fadiga).

Como podemos observar, relativamente aos níveis de fadiga existem algumas observações importantes que automaticamente podemos fazer apenas olhando para a distribuição amostra:

- Os níveis de fadiga da amostra estão compreendidos no intervalo de 1 a 6, não existindo nenhum caso com fadiga 7;

- Na maioria dos casos da amostra os indivíduos apresentam nível de fadiga 2;

Este tipo de inferências iniciais permitiu-nos em fases posterior ter uma melhor perceção do porquê dos resultados obtidos.

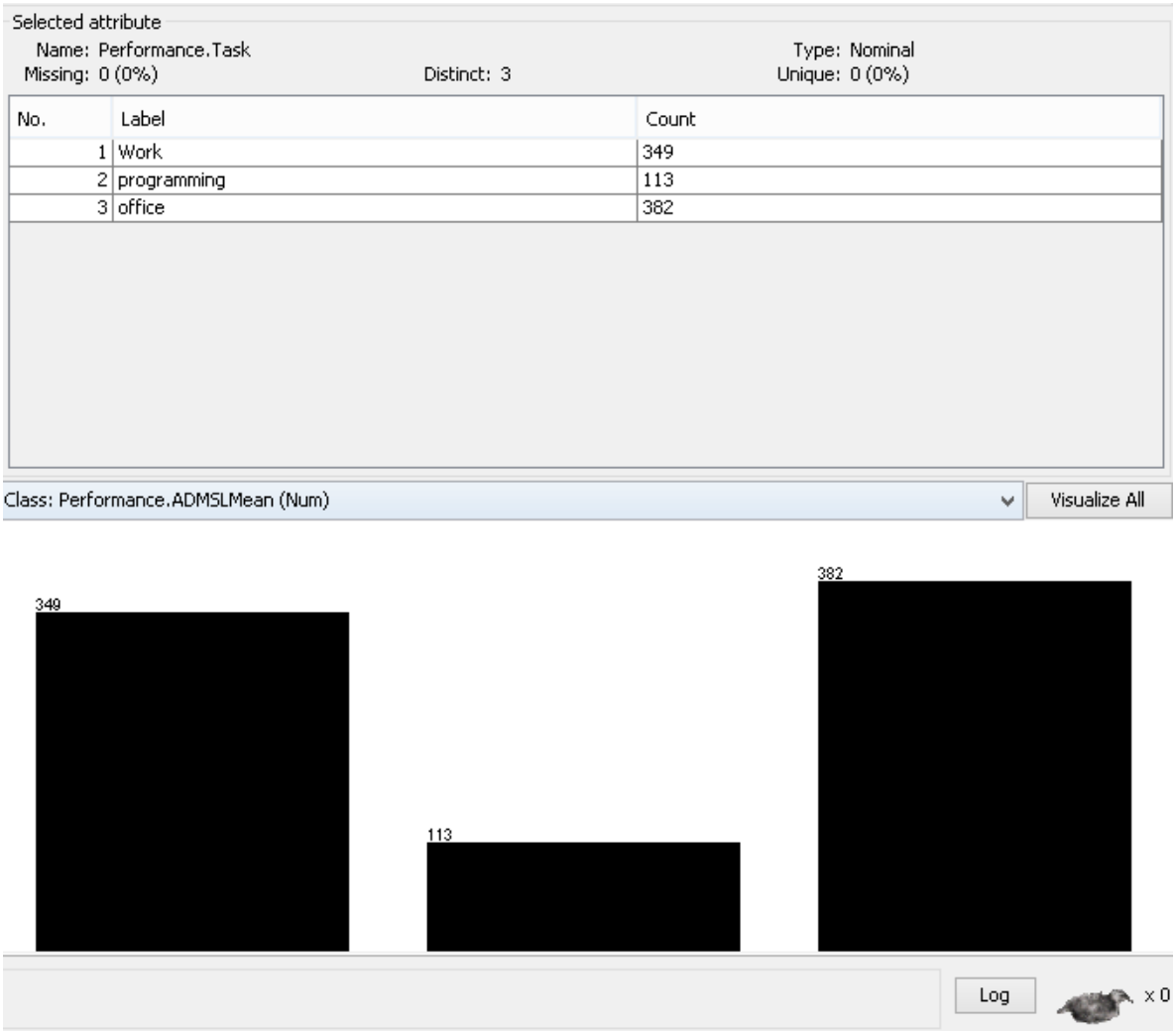

Figura 10 - Task (Tarefa)

Relativamente às tarefas neste *dataset*, encontram-se três tipos de tarefa diferentes com a distribuição que podemos observar na figura 10.

3.2 Decisões tomadas face à obtenção da solução

Após a análise ao dataset, foram tomadas decisões face aos dados que iriam ser utilizados tanto para treino como para teste:

- A rede neuronal será treinada com todas as entradas do dataset
- Vão ser escolhidas várias pequenas e médias porções do dataset. À porção inicial, que é guardada, vão sendo concatenados outros pedaços de dados. Eventualmente, serão formados vários blocos de dimensões diferentes, de modo a poder testar com rigor a rede

neuronal, conseguindo analisar o comportamento da rede à medida que o volume dos dados de teste aumenta.

```
teste <- dataset[15:20,]
teste2 <- dataset[550:560,]
teste3 <- dataset[440:450,]
teste4 <- dataset[236:348,]
teste5 <- dataset[60:90,]
teste6 <- dataset[700:800,]

total <- rbind(teste,teste2)
total_2 <- rbind (total, teste3)
total_3 <- rbind (total_2, teste4)
total_4 <- rbind (total_3, teste5)
total_5 <- rbind (total_4, teste6)
```

Figura 11 - Instruções relativas à formações dos vários blocos de teste.

- De modo a testar e comparar as várias redes neuronais, foi escolhido o seguinte formato de testes:

Camadas	**Nodos Intermédios**	**Nº Casos**	***Threshold***	**RMSE (Resultado)**
1 2	3 6 (3,3) (6,5)	17	0.1	
		28		
		141		
		172		
		273		

Ou seja, na tabela em cima encontra-se **um guião** (ou template) de como iremos proceder aos testes das redes treinadas. No que toca ao número de neurónios e número de camadas intermédias, não existe um modelo pré-definido para a escolha que irá otimizar os nossos resultados finais, no entanto há quem sugira metodologias para determinarmos estes valores:

"A rule of thumb is for the size of this [hidden] layer to be somewhere between the input layer size ... and the output layer size ..." *(Blum, 1992, p. 60*).

"How large should the hidden layer be? One rule of thumb is that it should never be more than twice as large as the input layer." (Berry and Linoff, 1997, p. 323).

Como medida de performance das nossas redes calculámos o **RMSE** (*Root Mean Square Error*).

3.3 Packages de redes neuronais utilizados

3.3.1 Neuralnet

Este é o *package* que temos vindo a utilizar nas aulas práticas da UC. O *package neuralnet* utiliza o método de *backpropagation* para treinar a rede. Este é um dos métodos mais comuns e é considerado um método supervisionado de aprendizagem onde são usados *datasets* de teste onde se espera obter um certo valor final, pois usamos um *dataset*. Este *package* é flexível no sentido em que permite personalizar facilmente o critério de paragem a ter sido em conta bem como definir a topologia da rede.

3.3.2 NNet

Este é talvez um dos *packages* mais simples de se utilizar. No entanto ficamos limitados a uma única camada o que pode não ser de todo um problema, estudos indicam que em norma apenas é necessária uma camada intermédia numa rede neuronal por forma a obter melhores resultados, pois redes com maior número de camadas intermédias para além de aumentarem a complexidade dos cálculos podem não fornecer um melhor modelo. Outra desvantagem do *nnet* é não podermos visualizar os nodos uma vez criada a rede (isto nativamente).

3.3.3 MLP (Multilayer Perceptron)

Outro package de redes neuronais que utilizamos.

3.3.4 PCANNet

Estas redes têm aplicações desde as ciências da computação até ao processamento de sinais na biomédica. O objetivo destas redes é encontrar um conjunto ortogonal de componentes que minimizam o erro da amostra reconstruída. As redes PCAs tentam minimizar o erro dada uma amostra inicial reduzindo os fatores inicialmente considerados.

4. Apresentação de resultados

4.1 Requisitos básicos

As três seguintes secções representam aquilo que são os requisitos mínimos deste exercício.

4.1.1 Deteção do nível de fadiga

Foi criado um script para leitura do dataset, treino da rede neuronal e testes sobre esta.

```
4  dataset <- read.csv("D:\\Dropbox\\SRCR\\Exercicio3\\Ex3_aleatorio.csv",header=TRUE,sep=";",dec=".")
5
6  teste <- dataset[15:20,]
7  teste2 <- dataset[550:560,]
8  teste3 <- dataset[440:450,]
9  teste4 <- dataset[236:348,]
10 teste5 <- dataset[60:90,]
11 teste6 <- dataset[700:800,]
12
13 total <- rbind(teste,teste2)
14 total_2 <- rbind (total, teste3)
15 total_3 <- rbind (total_2, teste4)
16 total_4 <- rbind (total_3, teste5)
17 total_5 <- rbind (total_4, teste6)
18
19 # treinar rede neuronal
20 fadiganet <- neuralnet(FatigueLevel ~ Performance.KDTMean + Performance.MAMean
21                        +Performance.MVMean+Performance.TBCMean+Performance.DDCMean+Performance.DMSMean
22                        +Performance.AEDMean+Performance.ADMSLMean+Performance.Task, dataset,
23                        hidden = c(6), threshold = 0.1)
24
25
26 # imprimir resultados
27 plot(fadiganet)
28
29 temp_teste  <-subset (teste3, select = c(Performance.KDTMean,Performance.MAMean,
30                                          Performance.MVMean,Performance.TBCMean, Performance.DDCMean,
31                                          Performance.DMSMean ,Performance.AEDMean, Performance.ADMSLMean,
32                                          Performance.Task))
33
34 fadiganet.results <- compute(fadiganet, temp_teste)
35
36 res <- data.frame(actual = teste3$FatigueLevel, prediction = fadiganet.results$net.result)
37
```

Figura 12- Script em R com o funcionamento da rede

	row.names	actual	prediction
1	15	2	1.524974640
2	16	1	1.654565403
3	17	2	3.651332576
4	18	2	2.537601367
5	19	1	1.656490196
6	20	3	2.695023616

Figura 13 - Níveis de fadiga previstos pela rede neuronal

4.1.2 Identificar existência ou ausência de fadiga

De forma a determinarmos, com base na escala do *Dr. Layne Perelli,* se um determinado indivíduo apresentar ou não fadiga, foi feita uma análise superficial da mesma e consideramos que existe fadiga a partir do nível 5 (inclusive).

TABLE 2. HYPOTHESIS OF RELATIONSHIP BETWEEN SUBJECTIVE FATIGUE REPORT AND OPERATIONAL PERFORMANCE CAPABILITY

Fatigue Class	Subjective Fatigue Checkcard (SAM Form 136)	Crew Status Check (SAM Form 202)	Predicted Effect of Fatigue on Performance
IV	20 - 12	1 - 3	Sufficiently alert. No performance impairment due to fatigue.
III	11 - 8	4	Mild fatigue. Performance impairment possible but not significant. Treat as class IV.
II	7 - 4	5 - 6	Moderate to severe fatigue. Some performance impairment probably occurring. Flying duty permissible but not recommended.
I	3 - 0	7	Severe fatigue. Performance definitely impaired. Flying duty not recommended. Safety of flight in jeopardy.

Figura 14- Tabela original da escala de fadiga.

Foi, então, espelhada esta característica no script de R.

```
#### Nova coluna (Fatigado ou não)
ex <- (fadiganet.results$net.result)

dd <- matrix(,nrow(ex))

tmp <- match(rownames(ex), rownames(dd))
st <- cbind( ex, dd[tmp,] )


for (i in 1:nrow(ex)) {
  if (ex[i,]<5) st[i,2]<-0 else  st[i,2]<-1 }

colnames(st) <- c("FatigueLevel","BooleanValue")
########
```

Figura 15 - Pedaço do script em R.

Consequentemente, é possível averiguar, duma maneira simples e prática, a existência ou ausência de fadiga:

	row.names	FatigueLevel	BooleanValue
1	440	1.473346498	0
2	441	2.129789602	0
3	442	2.827317349	0
4	443	2.949509104	0
5	444	3.176639466	0
6	445	3.071163783	0
7	446	3.573999284	0
8	447	1.403713694	0
9	448	2.547343991	0
10	449	1.329760609	0
11	450	2.170962200	0

Figura 16 - Exemplo de output do script em R.

4.1.3 Criação de uma escala para classificação de fadiga

Uma forma de otimizar os resultados da rede, por forma a obtermos resultados mais fidedignos, dados pelos nodos de saída da mesma, é olhar para o nosso problema, para a nossa amostra e repensar a escala de classificação de fadiga mental.

Será que são mesmo necessários os 7 níveis de fadiga? Serão demasiados? Isto é, será que estamos a pedir à rede demasiado detalhe a nível da reposta fornecendo-lhe um número adequado de parâmetros de entrada?

O objetivo será a que a rede nos consiga dar respostas com precisão, com um nível de erro baixo. Portanto, a calibração do número de parâmetros de entrada bem como a influência que os mesmos produzem na resposta final é um fator muito importante.

Com base na escala de fadiga inicialmente proposta o grupo aplicou uma granularidade maior ao nível da resposta final, isto é, definimos intervalos de fadiga para os quais um indivíduo se irá encontrar numa certa classe de fadiga.

Classes de fadiga:

Classe 1 – Fadiga severa, pode influenciar fortemente a performance do indivíduo na execução de uma tarefa; (**Fadiga forte**)

Classe 2 - Fadiga moderada, pode ou não ter impacto no indivíduo; (**Fadiga Moderada**)

Classe 3 – Fadiga a não ser considerada, pequena probabilidade de interferência; (**Fadiga Insignificante**)

Classe 4 – A fadiga não interfere em nada na performance do indivíduo; (**Sem fadiga**)

Estas classes de fadiga poderão não fornecer o detalhe da escala anterior mas dar-nos-ão uma noção mais generalista sobre o estado de fadiga de um determinado indivíduo. Podemos dizer como já foi referido que analisando as respostas da rede não obteremos o nível de detalhe na resposta como obteríamos com a escala de 1 a 7 mas o objetivo aqui será minimizar o erro, ou seja o desvio dos resultados da rede mediante o resultado real.

	row.names	actual	prediction
1	550	3	3.667535150
2	551	3	3.752792733
3	552	4	3.919539938
4	553	3	3.751782320
5	554	4	3.832059320
6	555	4	3.939546016
7	556	4	3.951962526
8	557	4	3.673655833
9	558	3	3.742097741
10	559	4	3.529990972
11	560	4	3.532622124

Figura 17 - Resultados da aplicação da nova escala.

Mais à frente neste relatório iremos efetuar testes de performance desta rede comparativamente à inicial com a escala origina (*secção 5.2 pág. 31*)

5. Testes e análise de resultados

Nesta secção do relatório, iremos proceder à demonstração dos testes efetuados assim como explicitar os resultados obtidos. Estes testes seguem o guião (ou template) que está descrito na secção nº 3.2.

Para que melhor se perceba estrutura dos nossos testes e análise de resultados, criámos um esquema, onde se pode visualizar quais as topologias testadas e comparadas entre os diferentes pacotes de redes neuronais entre outras.

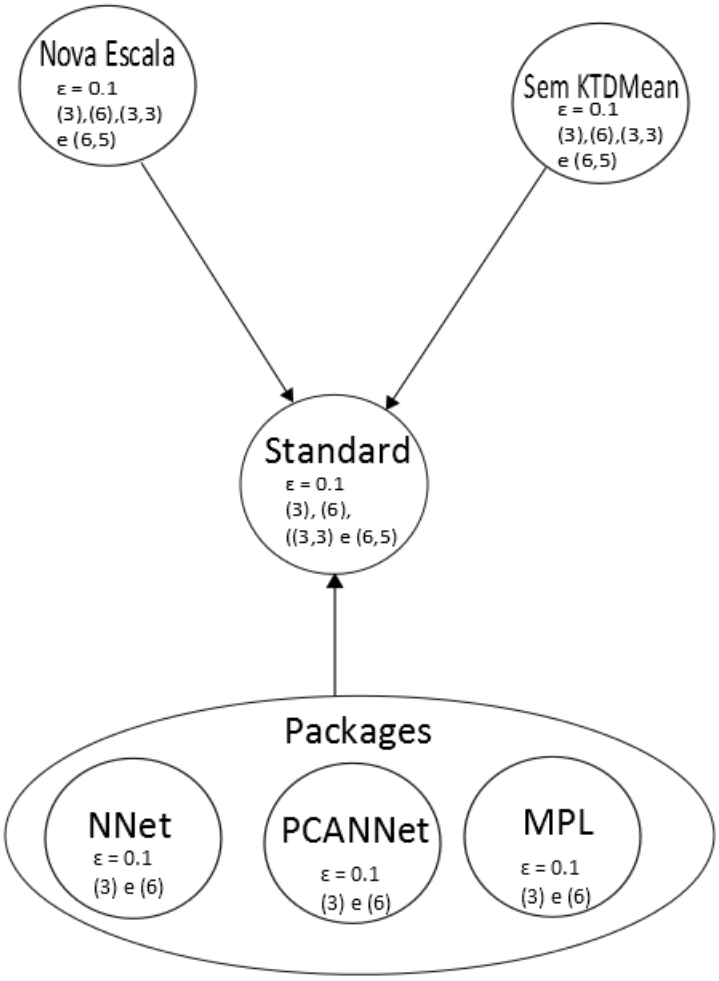

Figura 18 - Esquematização dos testes efetuados.

5.1 Testes iniciais com o neuralnet

Teste 1				
Camadas	**Nodos Intermédios**	**Nº Casos**	***Threshold***	**RMSE**
2	(6,5)	17	0.1	0.6934
		28		0.9588
		141		0.8924
		172		0.8825
		273		0.8851

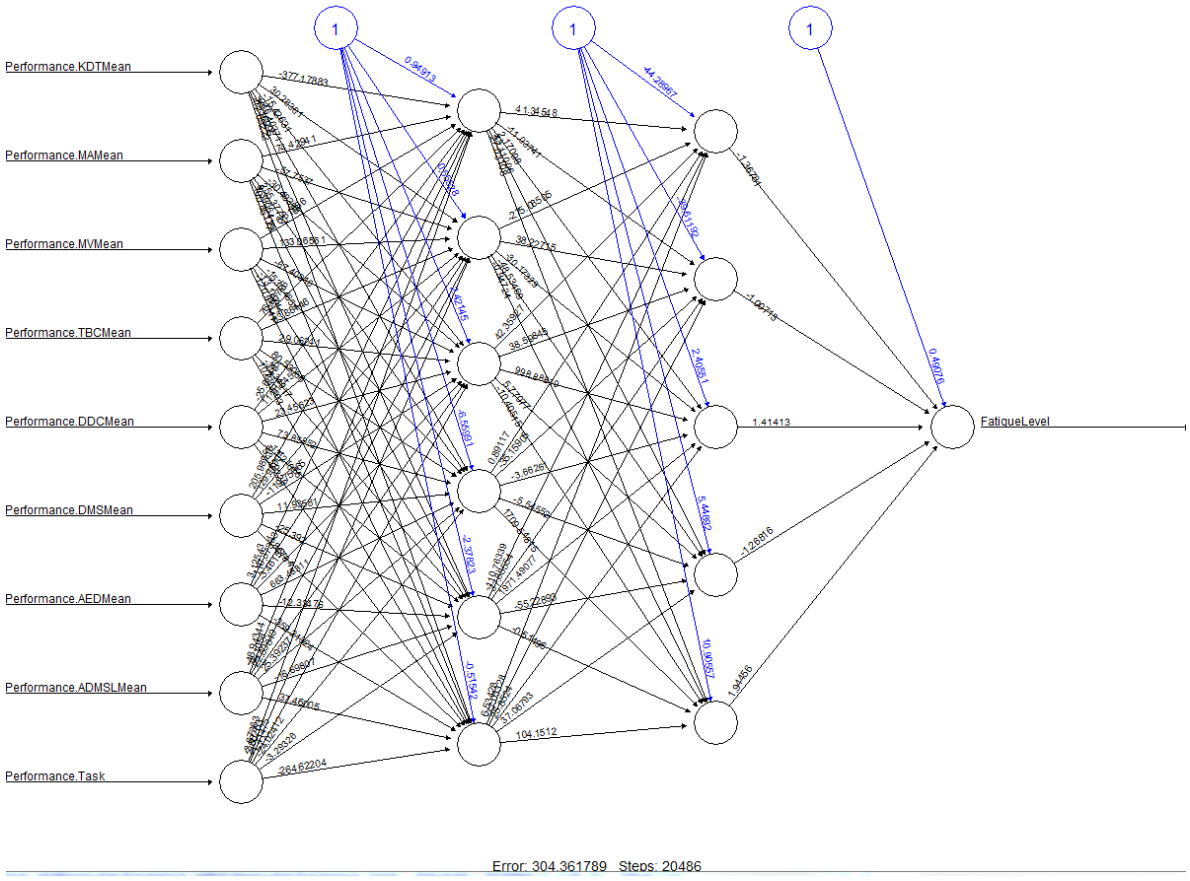

Figura 19 - Representação gráfica da rede relativa ao Teste 1.

Teste 2				
Camadas	**Nodos Intermédios**	**Nº Casos**	***Threshold***	**RMSE**
1	6	17	0.1	0.7209
		28		0.7761
		141		0.9515
		172		0.9337
		273		0.9501

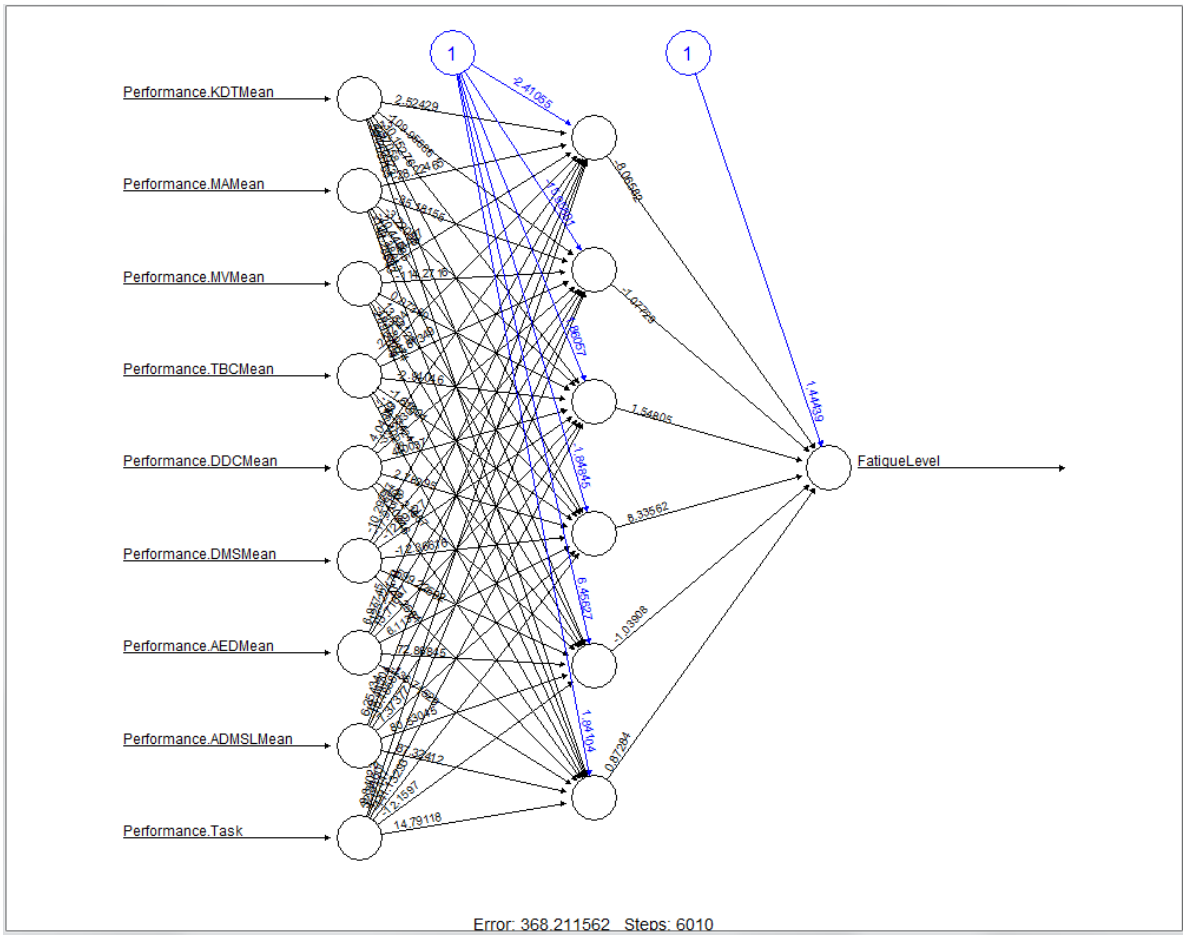

Figura 20 - Representação gráfica da rede relativa ao Teste 2..

Teste 3				
Camadas	**Nodos Intermédios**	**Nº Casos**	***Threshold***	**RMSE**
1	3	17	0.1	0.8670
		28		0.9405
		141		1.027
		172		1.0071
		273		1.0317

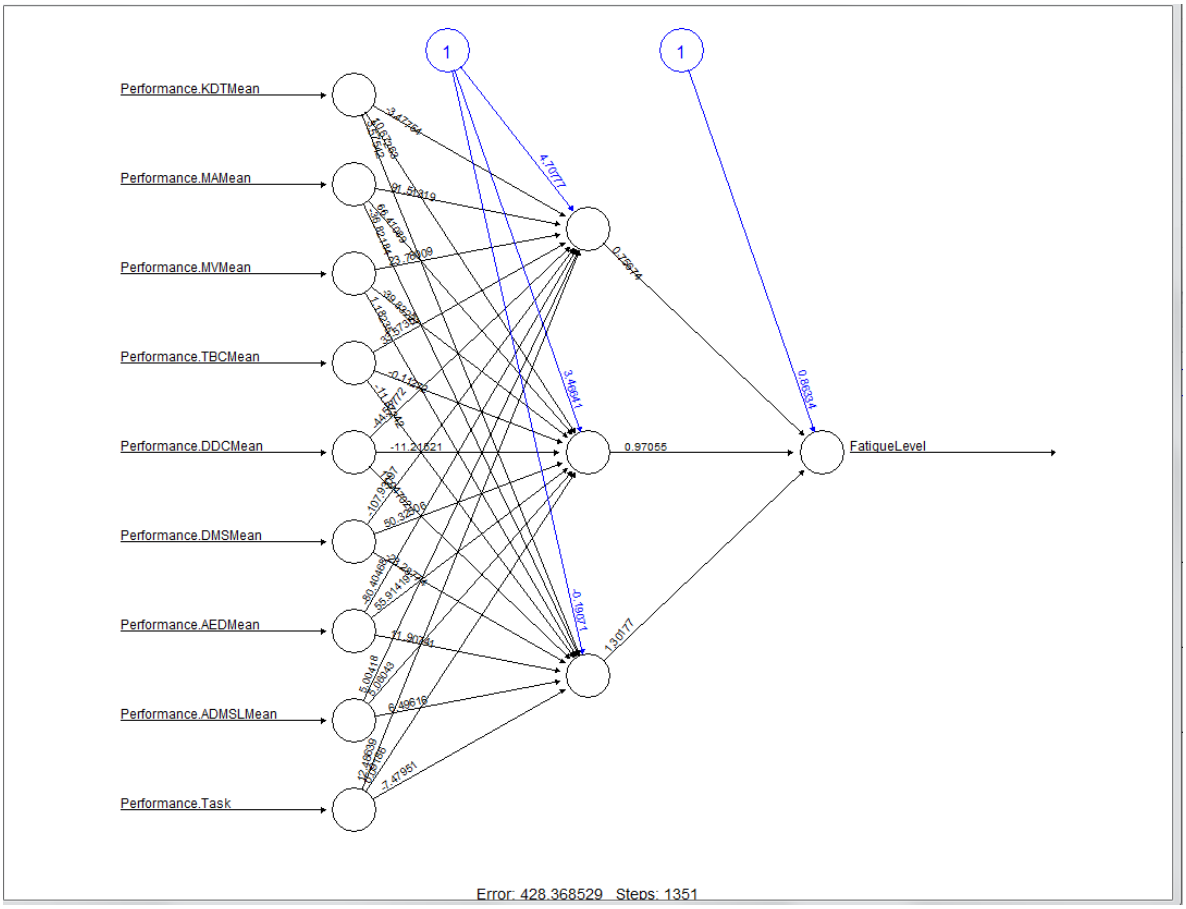

Figura 21- Representação gráfica da rede relativa ao Teste 3.

Teste 4				
Camadas	**Nodos Intermédios**	**Nº Casos**	***Threshold***	**RMSE**
2	(3,3)	17	0.1	0.8620
		28		0.9017
		141		1.0152
		172		0.9799
		273		0.9823

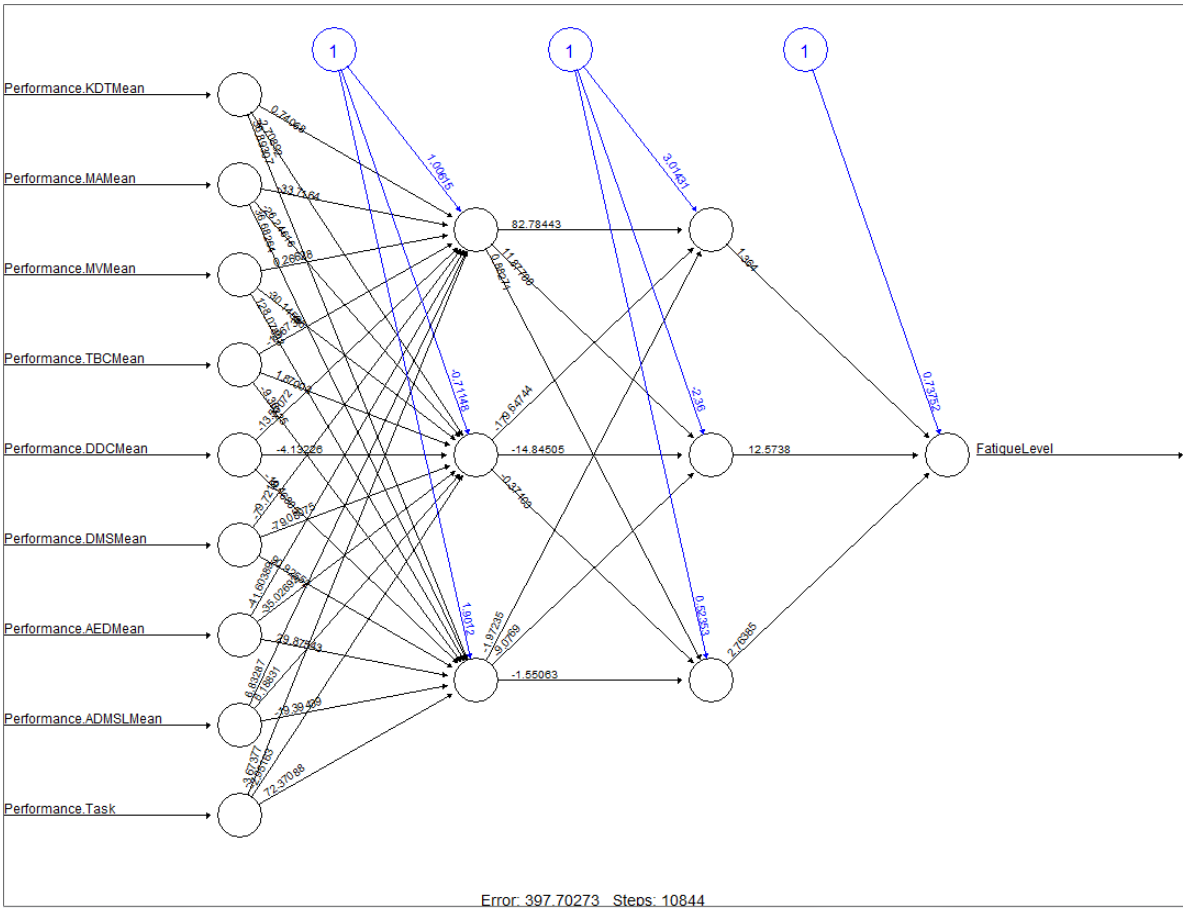

Figura 22 - Representação gráfica da rede relativa ao Teste 4.

5.2 Teste da rede com a criação de uma escala por classes

Como referenciado na secção (*4.1.3 pág. 24*), iremos agora proceder à exposição dos resultados obtidos quando treinamos a rede com a escala redefinida. Será de esperar que o erro sofra melhorias minimamente satisfatórias, pois sendo a nossa escala é mais pequena, é normal que exista uma melhor convergência da rede para o resultado real.

Camadas	**Nodos Intermédios**	**Nº Casos**	***Threshold***	**Steps**	**RMSE (com a nova escala)**	**RSME (com a escala inicial)**
2	(3,3)	17	0.1	184	0.4157	
		28			0.4503	
		141			0.4733	
		172			0.4622	
		273			**0.4682**	**0.9823**
1	(3)	17	0.1	7527	0.2884	
		28			0.4136	
		141			0.4371	
		172			0.4265	
		273			**0.4373**	**1.0317**
2	(6,5)	17	0.1	9127	0.4244	
		28			0.4753	
		141			0.4391	
		172			0.4348	
		273			**0.4185**	**0.8851**
1	(6)	17	0.1	3592	0.4488	
		28			0.4616	
		141			0.4481	
		172			0.4461	
		273			**0.4427**	**0.9501**

5.3 Remoção do parâmetro de entrada KDTMean

No seguimento dos requisitos básicos, seguem alguns testes e resultados extras que obtivemos/analisamos, com o intuito de aprofundar a abordagem a este exercício, este é um deles onde fizemos uma reavaliação dos parâmetros de entrada da nossa amostra. Após efetuada a análise estatística do *dataset* fornecido, uma das principais deduções que conseguimos retirar quase de imediato foi a existência de um parâmetro aparentemente constante. Independentemente do individuo, esse parâmetro apresentava valores iguais (ou semelhantes), como tal surge a pertinente questão que é, será que esse parâmetro influencia o resultados final da mesma forma que os restantes que possuem distribuição mais dispersas?

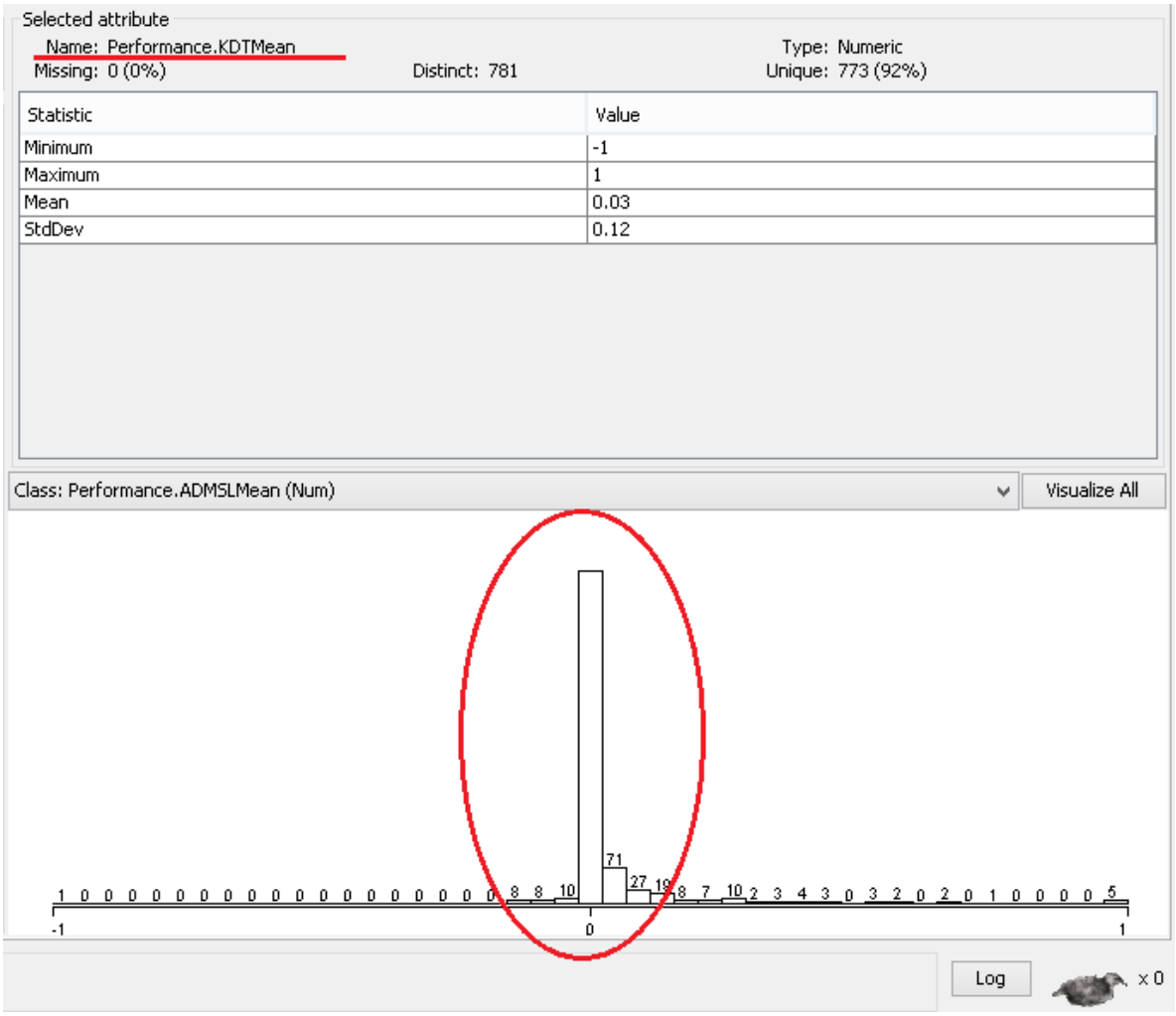

Figura 23- Recorde-se o gráfico da análise estatística do dataset, referente ao parâmetro KDTMean

Portanto com o script que desenvolvemos para os testes iniciais, apenas agora tivemos de retirar esse parâmetro constante.

5.2.1 Resultados obtidos

Camadas	**Nodos Intermédios**	**Nº Casos**	***Threshold***	**Steps**	**RMSE**	**RMSE (Com o KDTMean)**
2	(3,3)	17	0.1	9240	0.9961	
		28			1.0435	
		141			1.0263	
		172			0.9908	
		273			**0.9875**	**0.9823**
1	(3)	17	0.1	4632	0.9381	
		28			0.9671	
		141			1.0489	
		172			1.0209	
		273			**1.0254**	**1.0317**
2	(6,5)	17	0.1	3597	0.7707	
		28			0.9782	
		141			0.9836	
		172			0.9590	
		273			**0.9392**	**0.8851**
1	(6)	17	0.1	16963	0.7921	
		28			0.8540	
		141			0.9074	
		172			0.8843	
		273			**0.8978**	**0.9501**

Como podemos observar, a remoção do parâmetro **Performance.KDTMean** não altera significativamente o valor do erro comparativamente aos resultados obtidos anteriormente. Isto se definirmos como uma margem significativa de alteração do erro um valor 0.01 no caso da rede que fez uma melhor aproximação (2 camadas 3 neurónios por camada).

5.3 Testes com diferentes redes neuronais

Nesta secção do relatório vamos expor os nossos resultados e observações da utilização de outras redes neuronais, na mesma com recurso à linguagem R. Uma descrição teórica (superficial) já foi feita das redes que decidimos utilizar, na secção 4. Página 16 deste relatório. No entanto, apenas referindo, estas são as diferentes três redes neuronais que pretendemos testar e posteriormente analisar resultados:

- **Nnet**
- **MLP (Multilayer Perceptron)**
- **PCANNet**

Será utilizado o package Caret de modo a utilizar todas estas redes neuronais diferentes:

```
# treinar rede neuronal
fadiganet = nnet(FatigueLevel~., data=trainset,size=6,maxit=10000,decay=.001)

source_url('https://gist.githubusercontent.com/fawda123/7471137/raw/466c1474d0a505ff044412703516c34f1a4684a5/nnet_plot_update.r')
plot.nnet(fadiganet)

my.grid <- expand.grid(.size = 6, .decay = c(0.1))
fnet <- train(FatigueLevel ~ Performance.KDTMean + Performance.MAMean+Performance.MVMean+Performance.TBCMean+Performance.DDCMean+Performance.DMSMean+
                Performance.AEDMean+Performance.ADMSLMean+Performance.Task, data = trainset,
                method = "pcaNNet", maxit = 1000, tuneGrid = my.grid, trace = F, linout = 1)

plot.nnet(fnet)
plot(fnet)


fnet.predict <- predict(fnet, newdata = total)
prestige.rmse <- sqrt(mean((fnet.predict - total$FatigueLevel)^2))
```

Figura 24 – Porção do script utilizada para utilizar redes segundo o método PCANNet

5.3.1 Nnet

Camadas	Nodos Intermédios	Nº Casos	*Threshold*	RMSE	RMSE (neuralnet)
1	(3)	17	0.1	0.9536	0.8670
		28		0.9698	0.9405
		141		1.0463	1.027
		172		1.0255	1.0071
		273		1.0387	1.0317
1	(6)	17	0.1	0.8614	0.7209
		28		0.9095	0.7761
		141		1.0011	0.9515
		172		0.9814	0.9337
		273		1.0023	0.9501

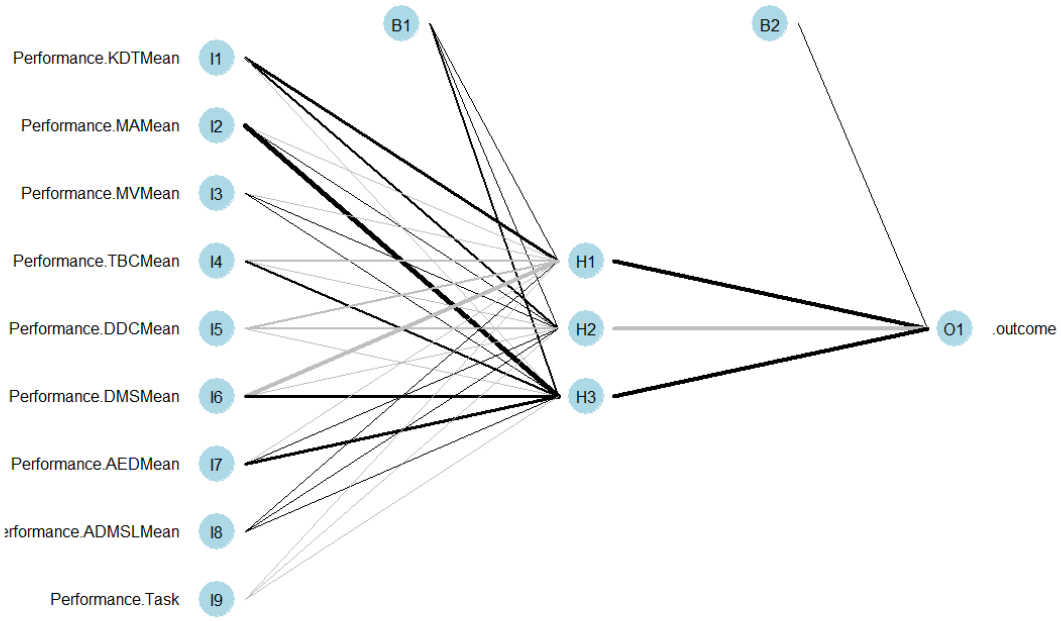

Figura 25 - Representação gráfica da rede neuronal com 3 nodos na camada intermédia.

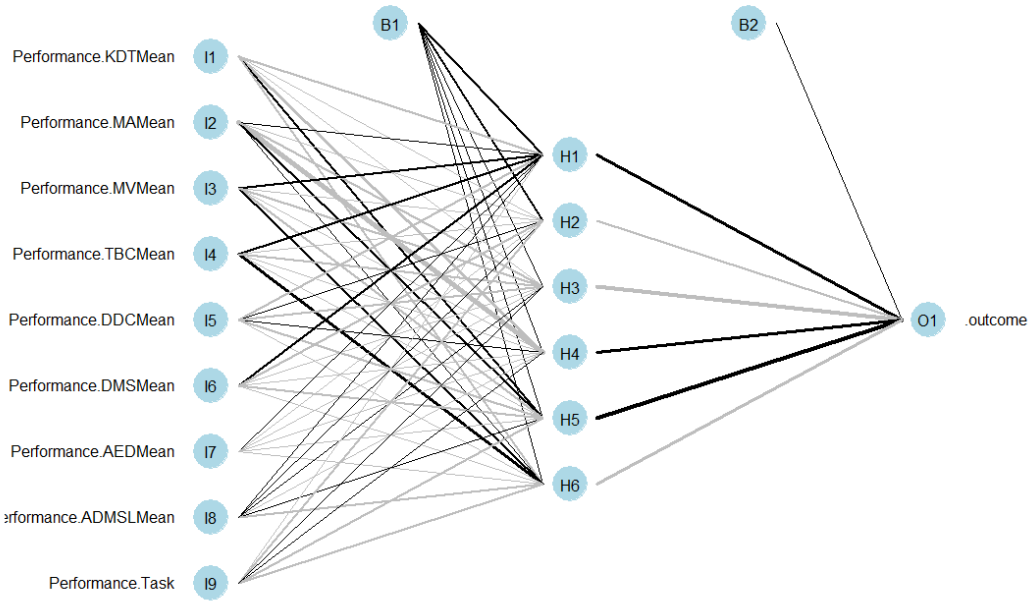

Figura 26 - Representação gráfica da rede neuronal com 6 nodos na camada intermédia.

Nativamente, não seria possível fazer o plot duma rede nnet. No entanto, através duma função encontrada, foi possível atingir tal efeito.

```
source_url('https://gist.githubusercontent.com/fawda123/7471137/raw/466c1474d0a505ff044412703516c34f1a4684a5/nnet_plot_update.r')
plot.nnet(fadiganet)
```

Figura 27 - Porção do script utilizada para fazer plot da rede nnet

5.3.2 MLP

Camadas	Nodos Intermédios	Nº Casos	*Threshold*	RMSE	RMSE (neuralnet)
1	(3)	17	0.1	1.2672	0.8670
		28		1.4168	0.9405
		141		1.2350	1.027
		172		1.2419	1.0071
		273		1.2528	1.0317
1	(6)	17	0.1	1.2146	0.7209
		28		1.3595	0.7761
		141		1.1902	0.9515
		172		1.1928	0.9337
		273		1.2069	0.9501

5.3.3 PCA Nnet

Camadas	Nodos Intermédios	Nº Casos	Threshold	RMSE	RMSE (neuralnet)
1	(3)	17	0.1	0.8731	0.8670
		28		0.8738	0.9405
		141		1.0240	1.027
		172		1.0002	1.0071
		273		1.0005	1.0317
1	(6)	17	0.1	0.9161	0.7209
		28		0.9800	0.7761
		141		0.9954	0.9515
		172		0.9743	0.9337
		273		0.9782	0.9501

5.4 Análise comparativa do desempenho das redes

Após recolha dos dados, e de modo a podermos fazer uma melhor análise comparativa das diferentes redes entre si, construímos os gráficos que de seguida apresentamos.

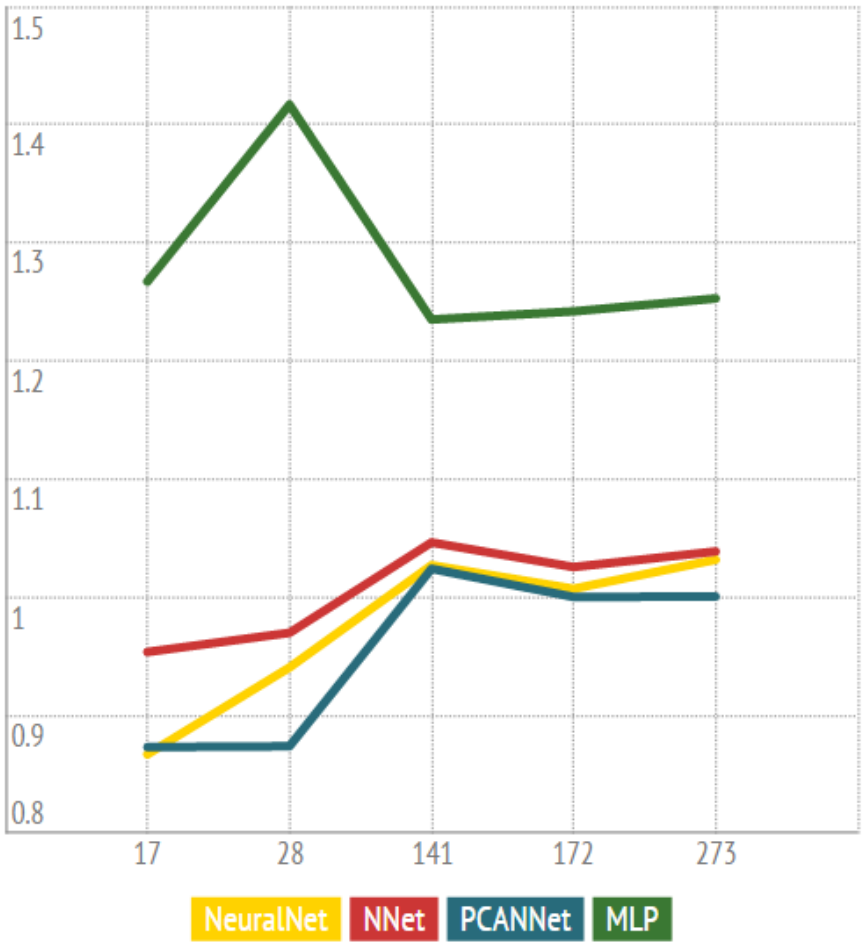

Figura 28 - Gráfico de comparação para redes com 1 camada intermédia com 3 neurónios.

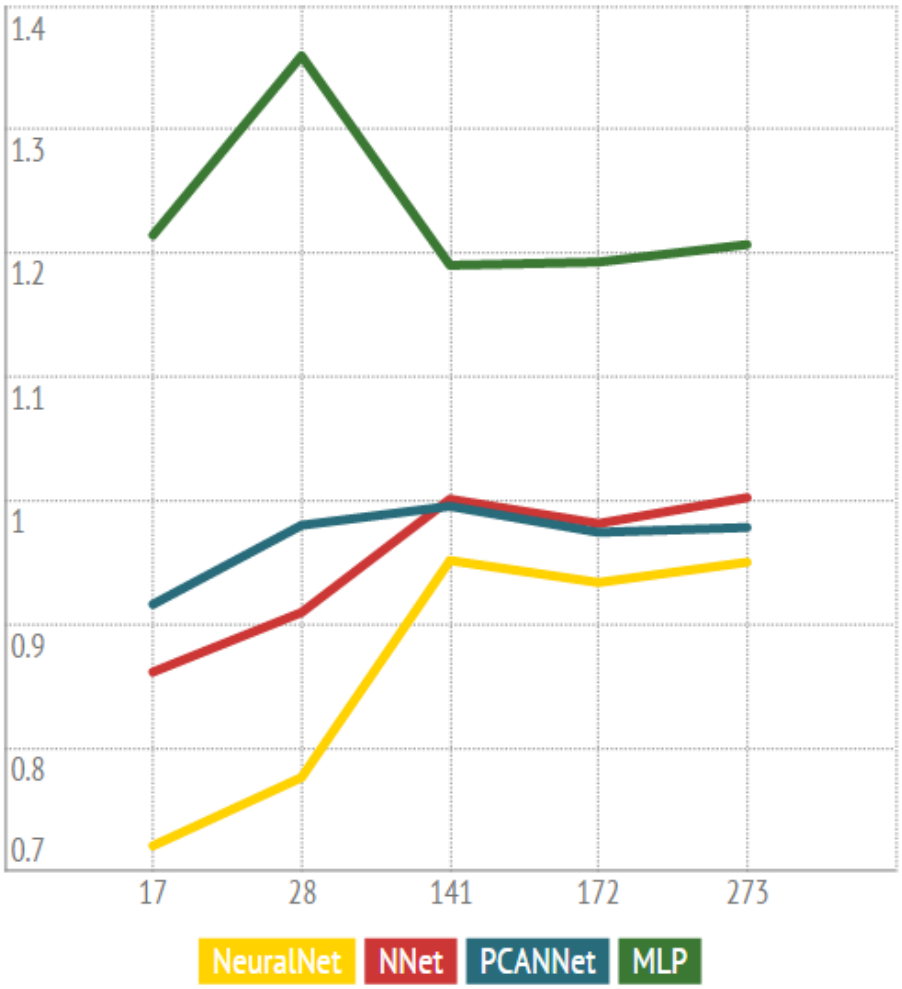

Figura 29 - Gráfico de comparação para redes com 1 camada intermédia com 6 neurónios.

Análise Comparativa:

Comparando a performance dos diferentes packages, usando como medida de desempenho o **RMSE,** pode-se imediatamente perceber que a rede **MLP** obteve maus resultados relativamente às restantes.

Justificando agora o uso de diferentes números de casos nos testes das redes, observamos que entre os 17 casos e os 141 casos tanto para redes com 3 neurónios como para redes com 6 neurónios, o erro obtido aumenta e chegando aos 141 casos, podemos observar que o comportamento da rede estabiliza (à exceção da **MLP**).

Com uma camada intermédia de 3 neurónios a rede que apresenta menor erro é a PCANNet. Quando temos uma rede com 6 neurónios na camada intermédia verifica-se que a neuralnet apresenta melhores resultados. Das 3 redes que apresentam

comportamentos semelhantes a Nnet é a que demonstra piores resultados devido à sua simplicidade.

De um modo geral também podemos concluir que as redes com 6 neurónios nas camadas intermédias se comportam melhor que as redes com 3 neurónios.

6. Conclusões

A quando da introdução desta matéria, não podemos deixar de nos sentir algo céticos em relação a esta. No entanto, à medida que as aulas foram sendo lecionadas e, sobretudo, com a realização do trabalho prático, a nossa opinião mudou, pois fomos ganhando visão para transportar este novo conceito para cenários reais, agarrando a noção da utilidade que pode ter. Nomeadamente em atribuição de montantes de crédito, precisão de tempo, ações da bolsa etc...

7. Anexos

7.1 ex3.R

```
library("neuralnet")
library("hydroGOF")

dataset <- read.csv("D:\\Dropbox\\SRCR\\Exercicio3\\Ex3_aleatorio_novaEscala.csv",header=TRUE,sep=";",dec=".")

teste <- dataset[15:20,]
teste2 <- dataset[550:560,]
teste3 <- dataset[440:450,]
teste4 <- dataset[236:348,]
teste5 <- dataset[60:90,]
teste6 <- dataset[700:800,]

total <- rbind(teste,teste2)
total_2 <- rbind (total, teste3)
total_3 <- rbind (total_2, teste4)
total_4 <- rbind (total_3, teste5)
total_5 <- rbind (total_4, teste6)

# treinar rede neuronal
fadiganet <- neuralnet(FatigueLevel ~ Performance.KDTMean + Performance.MAMean+Performance.MVMean

+Performance.TBCMean+Performance.DDCMean+Performance.DMSMean+Performance.AEDMean+Performance.ADMSLMean
                                      +Performance.Task, dataset, hidden = c(6), threshold = 0.1)

plot(fadiganet)

temp_teste  <-subset (total_5, select = c(Performance.KDTMean,Performance.MAMean,Performance.MVMean,Performance.TBCMean,
                                Performance.DDCMean,Performance.DMSMean ,Performance.AEDMean, Performance.ADMSLMean, Performance.Task))

fadiganet.results <- compute(fadiganet, temp_teste)

res <- data.frame(actual = total_5$FatigueLevel, prediction = fadiganet.results$net.result)

#### Nova coluna (Fatigado ou não)
ex <- (fadiganet.results$net.result)
```

```

dd <- matrix(,nrow(ex))

tmp <- match(rownames(ex), rownames(dd))
st <- cbind( ex, dd[tmp,] )


for (i in 1:nrow(ex)) {
  if (ex[i,]<5) st[i,2]<-0 else  st[i,2]<-1 }
########

rmse(c(total_5$FatigueLevel),c(res$prediction))
```

7.2 ex3_caret.R

```r
#install.packages("devtools")
#install.packages("nnet")
#install.packages("ROCR")
#install.packages("caret")
#install.packages("car")
#install.packages("mlpWeightDecay")
#install.packages("scales")
library("neuralnet")
library("hydroGOF")
library("nnet")
library("ROCR")
library("car")
library("caret")
library("devtools")

# ler ficheiro 5.1,3.5,1.4,0.2,Iris-setosa
dataset <- read.csv("D:\\Dropbox\\SRCR\\Exercicio3\\Ex3_aleatorio_2.csv"
                    ,header=TRUE,sep=",",dec=".")
#trainset <- read.csv("D:\\Dropbox\\SRCR\\Exercicio3\\nnet\\iris.txt",header=TRUE,sep=",",dec=".")


teste <- dataset[15:20,]
teste2 <- dataset[550:560,]
teste3 <- dataset[440:450,]
teste4 <- dataset[236:348,]
teste5 <- dataset[60:90,]
teste6 <- dataset[700:800,]

total <- rbind(teste,teste2)
total_2 <- rbind (total, teste3)
total_3 <- rbind (total_2, teste4)
total_4 <- rbind (total_3, teste5)
total_5 <- rbind (total_4, teste6)


# treinar rede neuronal
fadiganet = nnet(FatigueLevel~., data=trainset,size=6,maxit=10000,decay=.001)

source_url('https://gist.githubusercontent.com/fawda123/7471137/raw/466c1474d0a505ff044412703516c34f1a4684a5/nnet_plot_update.r')
plot.nnet(fadiganet)

my.grid <- expand.grid(.size = 6, .decay = c(0.1))
fnet <- train(FatigueLevel ~ Performance.KDTMean +
Performance.MAMean+Performance.MVMean+Performance.TBCMean+Performance.DDCMean+Performance.DMSMean+
                      Performance.AEDMean+Performance.ADMSLMean+Performance.Task, data = trainset,
                      method = "pcaNNet", maxit = 1000, tuneGrid = my.grid, trace = F, linout = 1)

plot.nnet(fnet)
plot(fnet)


fnet.predict <- predict(fnet, newdata = total)
prestige.rmse <- sqrt(mean((fnet.predict - total$FatigueLevel)^2))
```

8. Referências Bibliográficas

[1] - Yampolskiy, Roman V., and Venu Govindaraju. "Behavioural biometrics: a survey and classification."International Journal of Biometrics 1.1 (2008): 81-113.

[2] - Perelli, Layne P. Fatigue Stressors in Simulated Long-Duration Flight. Effects on Performance, Information Processing, Subjective Fatigue, and Physiological Cost. No. SAM-TR-80-49. SCHOOL OF AEROSPACE MEDICINE BROOKS AFB TX, 1980.

[3] - Pimenta A., Carneiro D., Novais P., Neves J., Detection of Distraction and Fatigue in Groups through the Analysis of Interaction Patterns with Computers, Intelligent Distributed Computing VIII, Springer-Verlag - Studies in Computational Intelligence, David Camacho, Lars Braubach, Salvatore Venticinque and Costin Badica (Eds) Vol. 570, pp 29-39, ISBN: 978-3-319-10421-8, 2014.

[4] - Pimenta A., Carneiro D., Novais P., Neves J., Monitoring Mental Fatigue through the Analysis of Keyboard and Mouse Interaction Patterns, Hybrid Artificial Intelligent Systems - 8th International Conference HAIS 2013, Jeng-Shyang Pan, Marios M. Polycarpou, Michał Woźniak, André C. P. L. F. de Carvalho, Héctor Quintián, Emilio Corchado (eds), Lecture Notes in Computer Science, Vol 8073, ISBN 978-3-642-40845-8, pp 222-231, 2013.

[5] - https://gist.githubusercontent.com/fawda123/7471137/raw/466c1474d0a505ff044412703516c34f1a4684a5/nnet_plot_update.r