

# Inteligência Ambiente: Tecnologias e Aplicações

Universidade do Minho

4º Ano de MIEI

## Questão Aula 1



Marco Matias Pereira Gonçalves A75480

# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>2</b>
<b>2</b>	<b><i>Dataset</i></b>	<b>3</b>
2.1	semDuplicados.csv . . . . .	3
<b>3</b>	<b>SBR</b>	<b>5</b>
3.1	Estações do Ano . . . . .	5
3.2	Linhas de Input . . . . .	6
3.2.1	Ar-condicionado . . . . .	6
3.3	Output . . . . .	6
<b>4</b>	<b>Conclusão</b>	<b>7</b>

# 1 Introdução

No âmbito da Unidade Curricular de Inteligência Ambiente: Tecnologias e Aplicações, foi proposto a resolução de uma Questão Aula de forma a obtermos uma avaliação contínua conforme os regulamentos desta Universidade. A Questão Aula é baseada no Rule-based Automotive Control System (ACS) que controla, gere e regula o comportamento de diversos dispositivos e sub-sistemas no veículo. Este assume diversos comportamentos de acordo com dados sensoriais. Deste modo, pretende-se o desenvolvimento de um sistema baseado em regras (SBR), tendo como base os dados do *dataset* fornecido, numa linguagem de programação à escolha, sendo que para tal foi utilizado o *python*. Para cada *input* dos sensores (neste caso linha do *dataset*), é necessário identificar a estação do ano, de modo a identificar a temperatura ideal. Tendo a temperatura, é necessário processar o output necessário. Para o funcionamento destes programas é preciso instalar as seguintes bibliotecas de python *pandas* *datetime* *metecalc*.

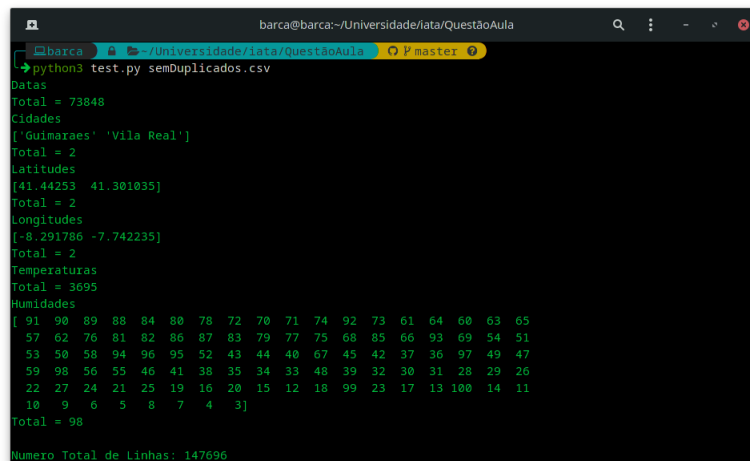
## 2 Dataset

### 2.1 semDuplicados.csv

Durante a apresentação da Questão Aula foi referido que os dados recebidos eram informações de sensores que estavam num carro. Inicialmente quando se procedeu à análise do *dataset* (AnexoTemperatura.csv) foi visível a existência de linhas repetidas, sendo que para remover essas linhas de informação desnecessária, foi necessário escrever um pequeno script em *python* que remove os duplicados (rmduplicates.py).

```
$ python3 rmduplicates.py AnexoTemperatura.csv semDuplicados.csv
```

Antes de começar a tratar os dados decidi fazer uma análise mais cuidada do *dataset*, para isso criei um novo script no qual chamei test.py. Este script vai a cada uma das colunas do *dataset* e selecciona apenas os valores únicos obtendo assim para cada coluna uma listas de todos os diferentes valores sem repetidos. Tendo estas listas ele imprime-as assim como o seu tamanho, mas como o tamanho das datas e das temperaturas eram muito grandes deixei em comentário para poder demonstrar os outputs de uma forma mais apelativa. Por fim este script ainda imprime o numero total de linhas do input fornecido.



```
barca@barca:~/Universidade/ata/QuestãoAula
python3 test.py semDuplicados.csv
Datas
Total = 73848
Cidades
['Guimaraes', 'Vila Real']
Total = 2
Latitudes
[41.44253, 41.301035]
Total = 2
Longitudes
[-8.291786, -7.742235]
Total = 2
Temperaturas
Total = 3695
Humidades
[ 91 90 89 88 84 80 78 72 70 71 74 92 73 61 64 60 63 65
 57 62 76 81 82 86 87 83 79 77 75 68 85 66 93 69 54 51
 53 50 58 94 96 95 52 43 44 40 67 45 42 37 36 97 49 47
 59 98 56 55 46 41 38 35 34 33 48 39 32 30 31 28 29 26
 22 27 24 21 25 19 16 20 15 12 18 99 23 17 13 100 14 11
 10 9 6 5 8 7 4 3]
Total = 98
Numero Total de Linhas: 147696
```

Figura 1: Output do test.py sobre semDuplicados.csv

Observando os outputs do test.py podemos observar que o "carro" está apenas em 2 pontos geográficos. Após uma breve análise do semDuplicados.csv reparei que temos informações 2 locais distintos com informações sobre a mesma hora. Como é fisicamente impossível 1 carro estar em 2 sítios ao mesmo tempo assumi que deveria separar a informação em 2 carros. Para isso criei um script chamado divider.py. Os argumentos são o input.csv, output1.csv, output2.csv, a coluna sobre a qual vamos dividir, e o valor respectivo da coluna.

```
$ python3 divider.py semDuplicados.csv Carro1.csv Carro2.csv lat 41.44253
```

Neste caso decidi dividir o semDuplicados.csv em 2 Carros pela latitude, como podemos verificar na figura 3.

```
barca@barca:~/Universidade/iata/QuestãoAula
python3 test.py AnexoTemperatura.csv
Dadas
Total = 73848
Cidades
['Guimaraes' 'Vila Real']
Total = 2
Latitudes
[41.44253 41.301035]
Total = 2
Longitudes
[-8.291786 -7.742235]
Total = 2
Temperaturas
Total = 3695
Humidades
[ 91 90 89 88 84 80 78 72 70 71 74 92 73 61 64 60 63 65
 57 62 76 81 82 86 87 83 79 77 75 68 85 66 93 69 54 51
 53 50 58 94 96 95 52 43 44 40 67 45 42 37 36 97 49 47
 59 98 56 55 46 41 38 35 34 33 48 39 32 30 31 28 29 26
 22 27 24 21 25 19 16 20 15 12 18 99 23 17 13 100 14 11
 10 9 6 5 8 7 4 3]
Total = 98
Numero Total de Linhas: 150750
```

Figura 2: Output do test.py sobre AnexoTemperatura.csv

```
barca@barca:~/Universidade/iata/QuestãoAula1
python3 AuxScripts/divider.py inputs/semDuplicados.csv inputs/Carro1.csv inputs/Carro2.csv lat 41.44253
Total de linhas lidas 147696
Total de linhas escritas no inputs/Carro1.csv 73848
Total de linhas escritas no inputs/Carro2.csv 73848
```

Figura 3: Usar o divider.py para criar os 2 Carros

```
barca@barca:~/Universidade/Iata/QuestãoAula1
python3 AuxScripts/test.py inputs/Carro1.csv
Dados
Total = 73849
Cidades
['Guimaraes' 'city_name']
Total = 2
Latitudes
['41.44253' 'lat']
Total = 2
Longitudes
['-8.291786' 'lon']
Total = 2
Temperaturas
Total = 3219
Humidades
['91' '90' '89' '88' '84' '80' '78' '72' '70' '71' '74' '92' '73' '61'
'64' '60' '63' '65' '57' '62' '76' '81' '82' '86' '87' '83' '79' '77'
'75' '68' '85' '66' '93' '69' '54' '51' '53' '50' '58' '94' '96' '95'
'52' '43' '44' '40' '67' '45' '42' '37' '36' '97' '49' '47' '59' '98'
'56' '55' '46' '41' '38' '35' '34' '33' '48' '39' '32' '30' '31' '28'
'29' '26' '22' '27' '24' '21' '25' '19' '16' '20' '15' '12' '18' '99'
'23' '17' '13' '100' '14' '11' '10' 'humidity']
Total = 92
Numero Total de Linhas: 295395
```

Figura 4: test.py sobre o Carro1

```
barca@barca:~/Universidade/Iata/QuestãoAula1
python3 AuxScripts/test.py inputs/Carro2.csv
Dados
Total = 73849
Cidades
['Vila Real' 'city_name']
Total = 2
Latitudes
['41.301035' 'lat']
Total = 2
Longitudes
['-7.742235000000001' 'lon']
Total = 2
Temperaturas
Total = 3664
Humidades
['92' '90' '91' '94' '93' '87' '80' '81' '65' '69' '71' '78' '84' '96'
'97' '85' '79' '74' '82' '76' '73' '77' '75' '72' '70' '83' '88' '86'
'95' '89' '99' '98' '59' '100' '62' '64' '61' '60' '67' '68' '66' '57'
'58' '63' '51' '53' '52' '48' '54' '47' '49' '46' '43' '39' '40' '45'
'50' '55' '56' '32' '33' '36' '34' '29' '22' '30' '38' '42' '35' '41'
'31' '44' '27' '37' '25' '24' '26' '17' '19' '21' '28' '18' '23' '20'
'16' '15' '12' '13' '14' '9' '11' '6' '5' '8' '7' '4' '3' 'humidity']
Total = 98
Numero Total de Linhas: 295395
```

Figura 5: test.py sobre o Carro2

## 3 SBR

### 3.1 Estações do Ano

Para definir o dia e hora em que começava cada uma das estações recorri este website <https://www.calendarr.com/portugal/estacoes-do-ano/> para definir os dias e a hora do ano em que cada uma das estações começa. Defini um função chamada de estacao que recebe um data e cria a data de inicio de cada estação no ano em que esta se encontra. Comparando as data devolve um string com a estação onde esta data se encontra. Tendo a estação defini uma função para devolver a temperatura ideal em que esta compra a a string que recebe e se esta representar o outono ou o inverno devolve 15<sup>o</sup> caso contrario apenas podem ser primavera e verão o que devolve como temperatura ideal 25<sup>o</sup>.

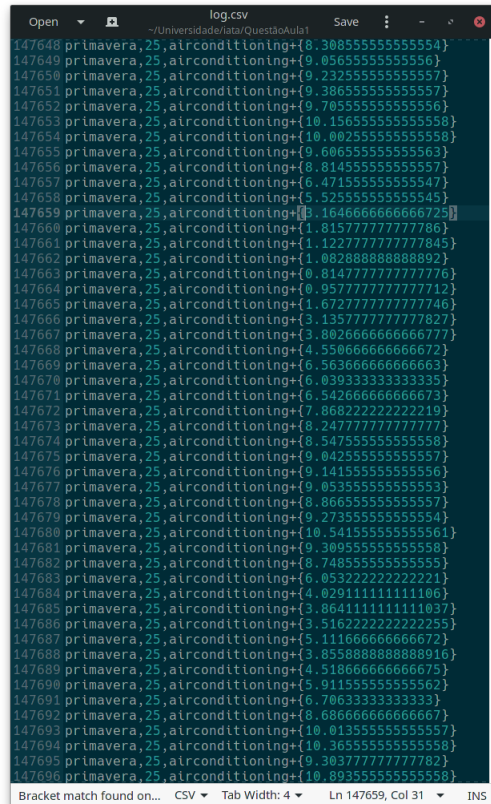
## 3.2 Linhas de Input

Para cada linha do nosso input file este programa lê a data, temperatura e humidade considere a localização e a cidade irrelevantes uma vez que os veículos estão sempre no mesmo sitio. Tendo os dados lidos este programa imprime num ficheiro csv a estação do ano, a temperatura ideal associada a esta estação, seguido do comando para o arcondicionado.

### 3.2.1 Ar-condicionado

Para o tratamento destes dados foram utilizados os valores da temperatura e da humidade para gerar um *Heat Index* com a ajuda da biblioteca *meteocalc*. O *Heat Index* (hi) combina a temperatura ambiente e a humidade para determinar a percepção humana da temperatura da temperatura ambiente do veículo. Esta temperatura que os humanos tem percepção é a utilizada para calcular a diferença para temperatura ideal. Com a diferença da temperatura ideal calculada ( $\text{diffTemp} = \text{tIdeal} - \text{hi.c}$ ) já obtemos a diferença de temperatura necessária para trabalhar com o ar-condicionado. Em que caso este ( $\text{diffTemp}$ ) seja maior que 0 a string relativa á regra deste subsistema muda para que o subsistema saiba que tem que aumentar a temperatura ('airconditioning+{' +  $\text{str}(\text{diffTemp})$  +'}) e caso contrario o sistema de regras avisa o subsistema que tem que diminuir a temperatura ('airconditioning{' +  $\text{str}(\text{diffTemp})$  +'})..

## 3.3 Output



```
log.csv
~/Universidade/ata/QuestãoAula1
147648 primavera,25,airconditioning+{8.30855555555554}
147649 primavera,25,airconditioning+{9.05655555555556}
147650 primavera,25,airconditioning+{9.23255555555557}
147651 primavera,25,airconditioning+{9.38655555555557}
147652 primavera,25,airconditioning+{9.70555555555556}
147653 primavera,25,airconditioning+{10.15655555555558}
147654 primavera,25,airconditioning+{10.00255555555558}
147655 primavera,25,airconditioning+{9.606555555555563}
147656 primavera,25,airconditioning+{8.81455555555557}
147657 primavera,25,airconditioning+{6.471555555555547}
147658 primavera,25,airconditioning+{5.525555555555545}
147659 primavera,25,airconditioning+{3.164666666666672}
147660 primavera,25,airconditioning+{1.815777777777786}
147661 primavera,25,airconditioning+{1.1227777777777845}
147662 primavera,25,airconditioning+{1.082888888888892}
147663 primavera,25,airconditioning+{0.814777777777776}
147664 primavera,25,airconditioning+{0.9577777777777712}
147665 primavera,25,airconditioning+{1.672777777777746}
147666 primavera,25,airconditioning+{3.1357777777777827}
147667 primavera,25,airconditioning+{3.802666666666677}
147668 primavera,25,airconditioning+{4.550666666666672}
147669 primavera,25,airconditioning+{6.563666666666663}
147670 primavera,25,airconditioning+{6.039333333333335}
147671 primavera,25,airconditioning+{6.542666666666673}
147672 primavera,25,airconditioning+{7.868222222222219}
147673 primavera,25,airconditioning+{8.24777777777777}
147674 primavera,25,airconditioning+{8.547555555555558}
147675 primavera,25,airconditioning+{9.042555555555557}
147676 primavera,25,airconditioning+{9.141555555555556}
147677 primavera,25,airconditioning+{9.053555555555553}
147678 primavera,25,airconditioning+{8.866555555555557}
147679 primavera,25,airconditioning+{9.273555555555554}
147680 primavera,25,airconditioning+{10.541555555555561}
147681 primavera,25,airconditioning+{9.309555555555558}
147682 primavera,25,airconditioning+{8.748555555555555}
147683 primavera,25,airconditioning+{6.053222222222221}
147684 primavera,25,airconditioning+{4.029111111111106}
147685 primavera,25,airconditioning+{3.8641111111111037}
147686 primavera,25,airconditioning+{3.516222222222255}
147687 primavera,25,airconditioning+{5.116666666666672}
147688 primavera,25,airconditioning+{3.8558888888888916}
147689 primavera,25,airconditioning+{4.518666666666675}
147690 primavera,25,airconditioning+{5.911555555555562}
147691 primavera,25,airconditioning+{6.706333333333333}
147692 primavera,25,airconditioning+{9.686666666666667}
147693 primavera,25,airconditioning+{10.013555555555557}
147694 primavera,25,airconditioning+{10.365555555555558}
147695 primavera,25,airconditioning+{9.303777777777782}
147696 primavera,25,airconditioning+{10.893555555555558}
```

Figura 6: Ficheiro log.csv

Cada vez que se executa o programa este escreve no fim do ficheiro de output o que executar das linhas lidas do ficheiro de input. Em cada uma das linhas ele escreve a Estação do ano, a temperatura ideal, e o comando para controlar o subsistema de refrigeração.

## 4 Conclusão

Assim, através deste sistema baseado em regras (**SRB**) consegui compreender um pouco melhor como funcionam as comunicações entre os diferentes subsistemas de um veículo. A recepção e utilização dos dados sensoriais de um veículo podem ser manipulados das mais variadas formas para que estes sejam aproveitados nas mais diversas aplicações. Durante este trabalho não foram utilizados os dados de localização, mas são dados muito interessantes. Caso o **ACS** Automotive Control System tenha acesso à Internet este pode identificar zonas como por exemplo casa ou mesmo quando este se começa a deslocar enviar um aviso podendo evitar assim roubos.