



Entrega de Encomendas

Agentes e Inteligência Artificial Distribuída
Mestrado Integrado em Engenharia Informática e Computação

Maria João Viana - up201604751

Descrição do Problema de Análise de Dados

Com este projeto pretende-se alocar encomendas a camiões rentabilizando o transporte das encomendas para a empresa que as distribui. Podem existir várias encomendas com diferentes prioridades entre elas, e camiões com diferentes capacidades. Estes camiões deverão transportar encomendas podendo estes passar por diferentes pontos para descarregar diferentes encomendas em vários pontos de entrega. Assim, as encomendas serão alocadas aos camiões de modo que a empresa dispense menos recursos, ficando o transporte mais barato.

Os diferentes camiões deverão cooperar de forma a que os clientes recebam as encomendas com o menor preço possível. Outro cenário possível de entrega será o facto de diferentes camiões quererem entregar as mesmas encomendas, e aí a cooperação transforma-se em competição.

Problema 1 (classificação): Como é que o número de camiões e as suas variáveis e, número de encomendas e as suas variáveis afetam a **capacidade de um camião entregar uma encomenda?**

Problema 2 (regressão): Como é que o número de camiões e as suas variáveis e, número de encomendas e as suas variáveis afetam o **custo de entrega de uma encomenda?**

**EXPERIÊNCIAS
REALIZADAS**

01

**ESTATÍSTICAS SOBRE
DADOS RECOLHIDOS**

02

ANÁLISE DE DADOS

03

ÍNDICE

04

CONCLUSÕES

05

**INFORMAÇÃO
ADICIONAL**



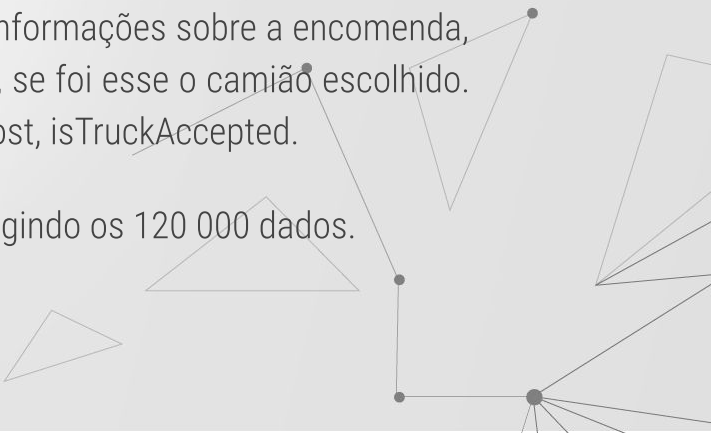
01

EXPERIÊNCIAS REALIZADAS

Foi desenvolvido um gerador de situações através de um ficheiro txt, sendo que cada linha deste ficheiro contém o número de camiões grandes, número de camiões pequenos e número de encomendas desejados no seguinte formato *nrBigTrucks-nrSmallTrucks-nrPackages*. Para cada situação foi gerado aleatoriamente para cada camião o seu custo por km e velocidade e para cada encomenda um x, y e a sua prioridade.

Foi apenas produzido um ficheiro csv, que em cada linha tinha informações sobre a encomenda, o camião disponível para fazer a entrega, o custo de entrega e por fim, se foi esse o camião escolhido. Cada linha tem o seguinte formato *priority, x, y, costPerKm, capacity, cost, isTruckAccepted*.

Para ambas as experiências foram gerados dados de treino, atingindo os 120 000 dados.





01

EXPERIÊNCIAS REALIZADAS

Experiência 1

Objetivo:

Responder ao problema 1 (**classificação**)

Variáveis Independentes:

priority inteiro | prioridade de cada encomenda;

x inteiro | coordenada x de cada encomenda;

y inteiro | coordenada y de cada encomenda;

costPerKm inteiro | em centavos, custo por km de cada caminhão;

speed inteiro | velocidade de cada caminhão;

capacity inteiro | capacidade de cada caminhão.

Variável Dependente:

isTruckAccepted booleano | se o caminhão é aceite ou não para levar uma encomenda.

Experiência 2

Objetivo:

Responder ao problema 2 (**regressão**)

Variáveis Independentes:

priority inteiro | prioridade de cada encomenda;

x inteiro | coordenada x de cada encomenda;

y inteiro | coordenada y de cada encomenda;

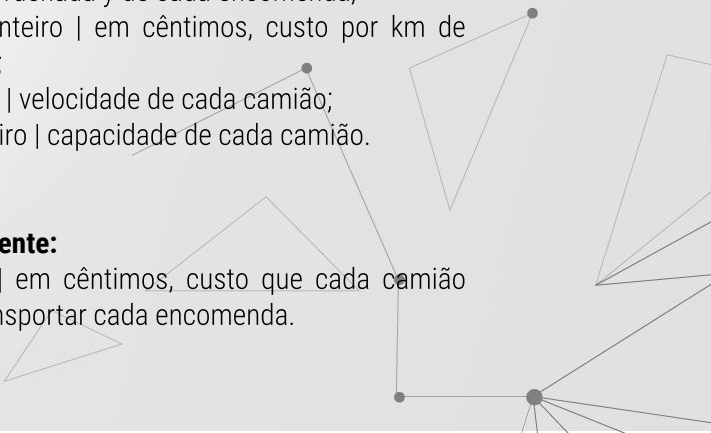
costPerKm inteiro | em centavos, custo por km de cada caminhão;

speed inteiro | velocidade de cada caminhão;

capacity inteiro | capacidade de cada caminhão.








Variável Dependente:

cost inteiro | em centavos, custo que cada caminhão leva para transportar cada encomenda.



02

ESTATÍSTICAS SOBRE DADOS RECOLHIDOS (classificação)

 Label isTruckAccepted	Polynomial	0	Least true (4128)	Most false (115887)	Values false (115887), true (4128)
 priority	Integer	0	Min 1	Max 10	Average 5.533
 x	Integer	0	Min 1	Max 30	Average 15.747
 y	Integer	0	Min 1	Max 30	Average 15.451
 costPerKm	Integer	0	Min 4	Max 59	Average 28.098
 speed	Integer	0	Min 5	Max 18	Average 10.045
 capacity	Integer	0	Min 1	Max 2	Average 1.550

02

ESTATÍSTICAS SOBRE DADOS RECOLHIDOS (regressão)

✓ Label cost	Integer	0	Min 11	Max 2503	Average 810.768
✓ priority	Integer	0	Min 1	Max 10	Average 5.533
✓ x	Integer	0	Min 1	Max 30	Average 15.747
✓ y	Integer	0	Min 1	Max 30	Average 15.451
✓ cost_per_km	Integer	0	Min 4	Max 59	Average 28.098
✓ speed	Integer	0	Min 5	Max 18	Average 10.045
✓ capacity	Integer	0	Min 1	Max 2	Average 1.550



03

ANÁLISE DE DADOS COM RAPIDMINER (classificação)

Decision Tree

KNN

Random Forest

```
PerformanceVector:
accuracy: 96.50%
ConfusionMatrix:
True:   true   false
true:   0       0
false: 1261    34743
classification_error: 3.50%
ConfusionMatrix:
True:   true   false
true:   0       0
false: 1261    34743
absolute_error: 0.067 +/- 0.171
relative_error: 6.68% +/- 17.13%
root_mean_squared_error: 0.184 +/- 0.000
root_relative_squared_error: 5.249
squared_error: 0.034 +/- 0.171
```

```
PerformanceVector:
accuracy: 96.62%
ConfusionMatrix:
True:   true   false
true:   3       29
false: 1187    34785
classification_error: 3.38%
ConfusionMatrix:
True:   true   false
true:   3       29
false: 1187    34785
absolute_error: 0.066 +/- 0.185
relative_error: 6.62% +/- 18.55%
root_mean_squared_error: 0.197 +/- 0.000
root_relative_squared_error: 5.958
squared_error: 0.039 +/- 0.168
```

```
PerformanceVector:
accuracy: 96.39%
ConfusionMatrix:
True:   true   false
true:   0       0
false: 1299    34705
classification_error: 3.61%
ConfusionMatrix:
True:   true   false
true:   0       0
false: 1299    34705
absolute_error: 0.067 +/- 0.173
relative_error: 6.66% +/- 17.30%
root_mean_squared_error: 0.185 +/- 0.000
root_relative_squared_error: 5.138
squared_error: 0.034 +/- 0.170
```




03

ANÁLISE DE DADOS COM RAPIDMINER (classificação)

Naive Bayes

```
PerformanceVector:
accuracy: 96.69%
ConfusionMatrix:
True:   true   false
true:   0      0
false: 1190    34814
classification_error: 3.31%
ConfusionMatrix:
True:   true   false
true:   0      0
false: 1190    34814
absolute_error: 0.065 +/- 0.166
relative_error: 6.47% +/- 16.60%
root_mean_squared_error: 0.178 +/- 0.000
root_relative_squared_error: 5.389
squared_error: 0.032 +/- 0.164
```

Deep Learning

```
PerformanceVector:
accuracy: 96.49%
ConfusionMatrix:
True:   true   false
true:   0      6
false: 1257    34741
classification_error: 3.51%
ConfusionMatrix:
True:   true   false
true:   0      6
false: 1257    34741
absolute_error: 0.053 +/- 0.176
relative_error: 5.29% +/- 17.57%
root_mean_squared_error: 0.183 +/- 0.000
root_relative_squared_error: 5.255
squared_error: 0.034 +/- 0.174
```

SVM

```
PerformanceVector:
accuracy: 96.53%
ConfusionMatrix:
True:   true   false
true:   0      0
false: 1250    34754
classification_error: 3.47%
ConfusionMatrix:
True:   true   false
true:   0      0
false: 1250    34754
absolute_error: 0.285 +/- 0.085
relative_error: 28.50% +/- 8.46%
root_mean_squared_error: 0.297 +/- 0.000
root_relative_squared_error: 8.563
squared_error: 0.088 +/- 0.085
```



03

ANÁLISE DE DADOS COM RAPIDMINER (regressão)

Linear Regression

PerformanceVector:

root_mean_squared_error: 157.055 +/- 0.000
absolute_error: 116.757 +/- 105.043
relative_error: 25.90% +/- 57.05%
root_relative_squared_error: 0.351
squared_error: 24666.265 +/- 45195.765
correlation: 0.936
squared_correlation: 0.876

KNN

PerformanceVector:

root_mean_squared_error: 69.900 +/- 0.000
absolute_error: 31.998 +/- 62.146
relative_error: 5.18% +/- 9.82%
root_relative_squared_error: 0.156
squared_error: 4885.940 +/- 25675.236
correlation: 0.988
squared_correlation: 0.976

Deep Learning

PerformanceVector:

root_mean_squared_error: 8.713 +/- 0.000
absolute_error: 6.580 +/- 5.711
relative_error: 1.29% +/- 4.30%
root_relative_squared_error: 0.019
squared_error: 75.911 +/- 157.200
correlation: 1.000
squared_correlation: 1.000

Neural Net

PerformanceVector:

root_mean_squared_error: 20.693 +/- 0.000
absolute_error: 15.186 +/- 14.056
relative_error: 3.03% +/- 5.40%
root_relative_squared_error: 0.046
squared_error: 428.191 +/- 920.913
correlation: 0.999
squared_correlation: 0.998

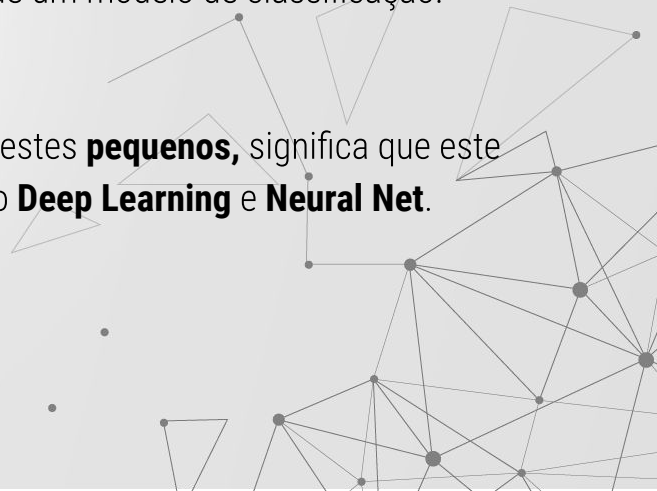


04

ANÁLISE DE DADOS COM RAPIDMINER

No modelo de classificação, os resultados não foram os esperados pois os dados recolhidos não têm em consideração se os camiões estão ocupados ou não e por isso, por vezes, apesar de um camião ser melhor que outro (o custo de entrega da encomenda é menor), não fica encarregue de entregar a encomenda porque estava ocupado a fazer outra entrega. Por outro lado, os níveis de **accuracy** são bastante altos, **variando entre 96,39% e 96,69%** o que significa que estes os dados recolhidos são bastante relevantes para a criação de um modelo de classificação. Podemos assim concluir que os melhores modelos são o **KNN** e o **Naive Bayes**.

No modelo de regressão, todos os **erros** são todos aproximados e sendo estes **pequenos**, significa que este modelo é bastante útil. Podemos assim concluir que os melhores modelos são o **Deep Learning** e **Neural Net**.



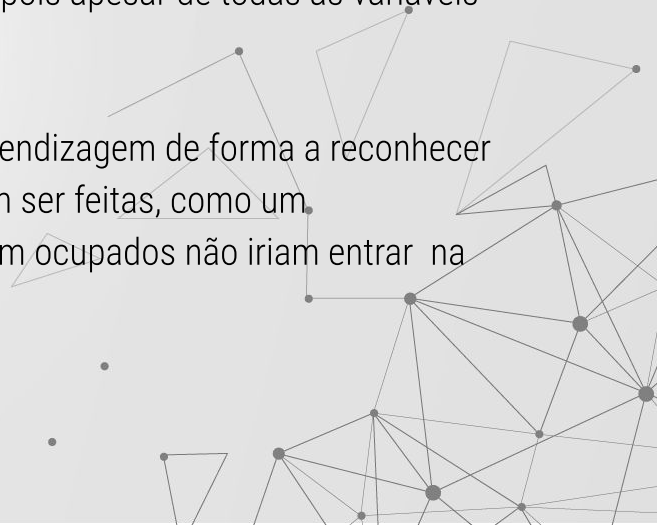
04

CONCLUSÕES

No final de todas estas experiências pode-se concluir que o número de dados gerado foi suficiente fazer uma análise que fosse fiável e que os erros fossem os menores possíveis.

Relativamente ao trabalho feito na segunda entrega, apenas foi alterado o launcher para gerar agentes com as suas variáveis em modo aleatório, o que pode alterar um pouco as experiências pois apesar de todas as variáveis estarem dentro de um intervalo, existem valores muito díspares entre elas.

Com este trabalho foi possível aplicar alguns métodos conhecidos de aprendizagem de forma a reconhecer padrões que um humano dificilmente conseguiria ver. Algumas melhorias podem ser feitas, como um pré-processamento do conjunto de dados inicial em que os camiões que estariam ocupados não iriam entrar na corrida de melhor camião a fazer a entrega da encomenda.





05

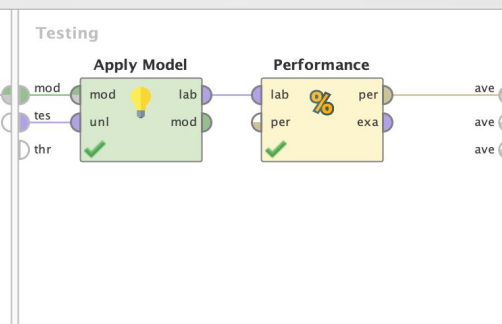
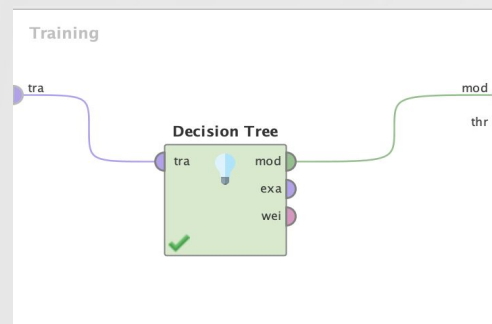
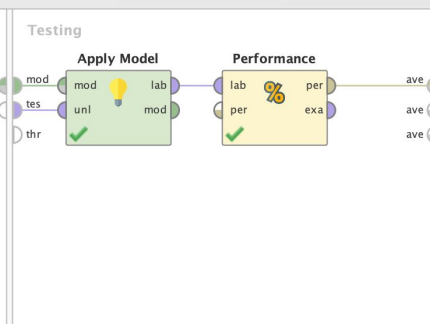
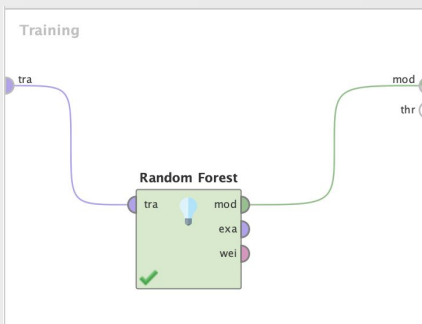
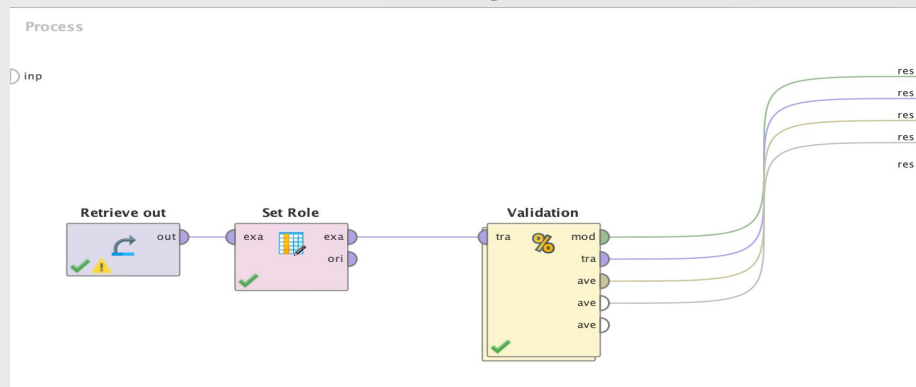
PROCESSOS RAPIDMINER

Os dados recolhidos permitiram-nos realizar diferentes processos no RapidMiner. Todos os processos possuem a mesma arquitetura, o único aspeto que os distingue é o algoritmo utilizado.

- Numa primeira fase, é definido qual o parâmetro a ser analisado com o operador **Set Role**, seguindo-se a separação dos dados pelo operador **Split Validation**: 70% para testar o modelo criado, 30% para o algoritmo. O algoritmo desenvolve um modelo, que é testado pelo operador **Apply Model**. Para classificação foram usados os algoritmos KNN, Decision Tree, Random Forest, Naive Bayes, Deep Learning e SVM. Para regressão, Linear Regression, KNN, Deep Learning e Neural Net. Por fim, é utilizado o operador **Performance**, o qual, através de várias medidas de erro, consegue transmitir o quão bem o modelo gerado consegue responder a cada problema.

05

PROCESSOS RAPIDMINER (classificação)



05

PROCESSOS RAPIDMINER (regressão)

