




# Mercado De Leilões

Grupo 29:

- Carlos Freitas - 201504749
  - Luis Martins - 201503344
  - João Conde - 201503256
- 

# Descrição do problema



No cenário em questão, um ou mais compradores possuem uma lista de produtos que pretendem comprar. Simultaneamente, vários vendedores tentam leiloar os seus produtos.

Cada vendedor fixa um preço mínimo para cada produto que vende, sendo este o valor inicial do produto em leilão (English auction), segundo uma distribuição normal. O mesmo produto, oferecido por diferentes vendedores, pode ter preços diferentes.

Cada comprador estará disposto a gastar um valor máximo por cada produto, valor este também obtido através da mesma distribuição normal. No entanto este valor pode reduzir caso o vendedor tenha má reputação. A reputação pode variar entre 0 e 1.

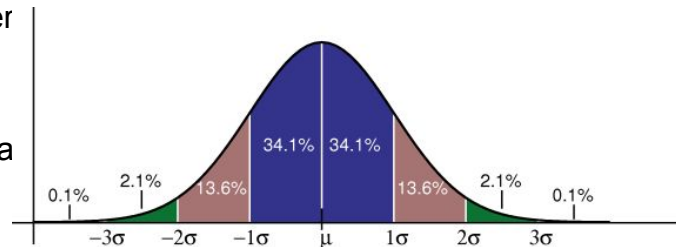
Um comprador que ganhe 2 ou mais leilões do mesmo produto pode escolher qual prefere, conforme o rating dos vendedores que licitaram os produtos ganhos

# Descrição do problema

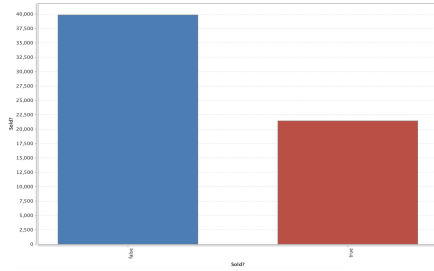
Por forma a poder prever se um produto leiloado será ou não vendido (problema de classificação) é utilizado um dataset de 61000 linhas de dados de execução, sendo que a variável dependente é se o produto é ou não vendido.

As variáveis independentes são:

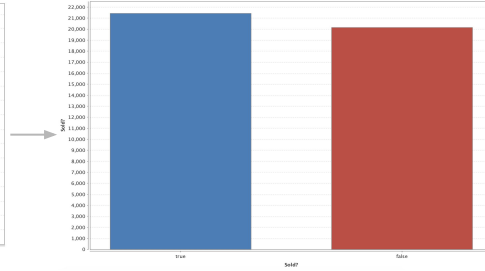
- Desvio percentual entre o **shipment delay (1)** do seller que está a vender o produto e a média do shipment delay de todos os sellers que vendem este mesmo produto
  - Desvio percentual entre a média da distribuição normal(  $\mu$  ) e o valor inicial do leilão
  - Desvio percentual entre a média do valor máximo que os compradores interessados estão dispostos a dar e o valor inicial do leilão
  - Número total de leilões do mesmo produto (para que se possa ter em consideração o número de competidores)
  - Número de interessados no produto (para que se possa ter em consideração a dimensão do público alvo)
- (1) - Valor máximo de dias que um vendedor pode atrasar a entrega da compra.



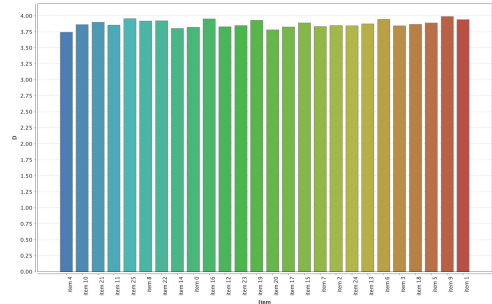
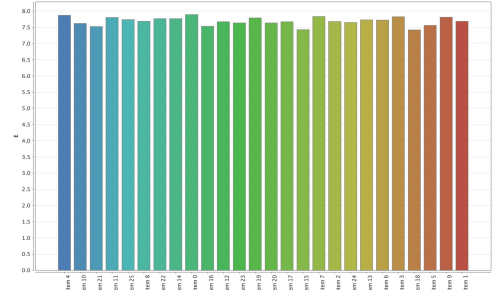
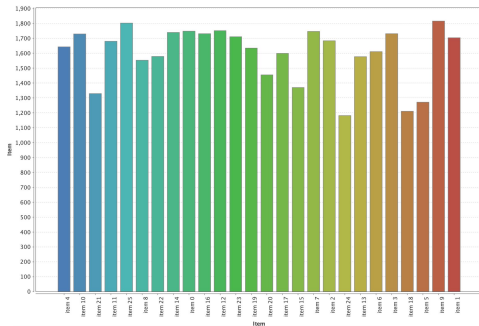
# Estatísticas sobre os dados recolhidos



Dataset original



Dataset depois de sampling



- A** - distância percentual entre o valor inicial do leilão e o valor médio do leilão
- B** - distância percentual entre o valor máximo que os buyers estão dispostos a gastar e o valor inicial do leilão
- C** - distância percentual entre o shipment delay do vendedor e o shipment delay dos seus concorrentes diretos
- D** - número de concorrentes diretos
- E** - número de buyers interessados

Attribu...	A	B	C	D	E
A	1	-0.041	-0.000	-0.001	0.005
B	-0.041	1	0.007	0.005	-0.003
C	-0.000	0.007	1	-0.002	0.001
D	-0.001	0.005	-0.002	1	0.005
E	0.005	-0.003	0.001	0.005	1

# Análise dos dados com o rapidminer

Para as experiências foram geradas cerca de 61000 linhas de dados de execução do programa às quais aplicámos variados modelos, tais como:

1. Decision Tree
2. Random Forest
3. K-NN
4. Deep Learning
5. Neural Net

Para cada modelo foram analisadas as seguintes variáveis:

- **Matriz confusão** - tabela que mostra os resultados das previsões do modelo aplicado
- **Accuracy (teste e treino) (%)** - precisão do modelo quando aplicado os dados de treino e os dados de teste
- **Classification\_error (%)** - erro do modelo (100% - accuracy(%))
- **Validation runtime (ms)** - tempo que o modelo demora a correr. Este ponto é importante pois caso existam muitos dados este pode ser um fator determinante.

De notar que para análise dos dados, e uma vez que é utilizada split validation (treino e teste) foi aplicado stratified sampling de forma a que os subsets aleatórios dos dados tenha uma distribuição das classes a analisar igual os subset inicial. Por exemplo, neste caso de classificação binomial, este tipo de sampling constrói sets que contém aproximadamente as mesmas proporções das duas classes de labels, para treino e para teste

# Comparação dos diferentes modelos aplicados

	<b>Decision Tree</b>	<b>Random Forest</b>	<b>K-NN</b>	<b>Deep Learning</b>	<b>Neural Net</b>	<b>Gradient Boosted Trees</b>
<b>Accuracy training (%)</b>	83.00	83.79	86.34	81.31	81.65	84.82
<b>Accuracy testing (%)</b>	82.94	83.26	80.42	80.77	81.15	84.14
<b>Validation runtime (ms)</b>	362	33546	70463	19203	333832	11296

De notar que o melhor modelo em termos de precisão é o Gradient Boosted Trees, que apesar de demorar mais do que o modelo decision tree tem mais 2% de precisão o que é relevante.

# Gradient Boosted Trees

Uma vez que o modelo que obteve melhor resultado verificou-se então qual seria o erro caso se escolhe-se sempre a escolha mais popular.

Nº true:  $5416 + 1014 = 6430$

Nº false:  $5081 + 965 = 6046$

Baseline error:  $6046 / (6430 + 6046) = 48.5\%$

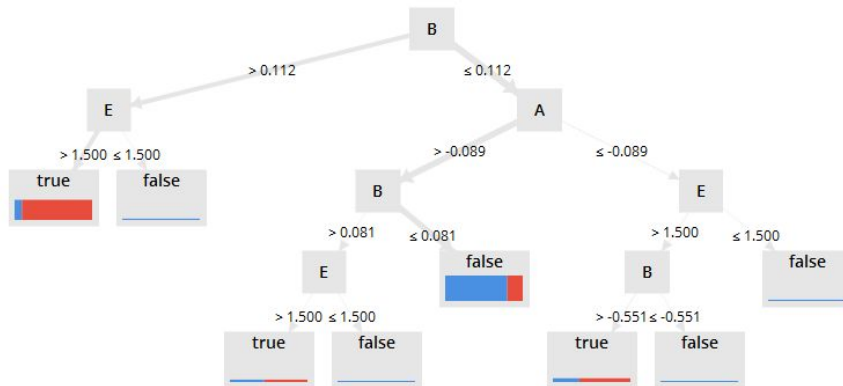
classification_error: 15.86%			
	true false	true true	class precision
pred. false	5081	1014	83.36%
pred. true	965	5416	84.88%
class recall	84.04%	84.23%	

Uma vez que os dados são bastante equilibrados e o *baseline error* é muito mais alto do que o erro de classificação conclui-se que este modelo é muito mais vantajoso que simplesmente considerar sempre a classe mais comum.

# Decision Tree

classification\_error: 16.78%

	true false	true true	class precision
pred. false	5274	1322	79.96%
pred. true	772	5108	86.87%
class recall	87.23%	79.44%	





# Decision Tree

Como pode ser observado, o algoritmo da decision tree consegue prever se um produto é vendido ou não com uma precisão de 82% (com um erro de classificação de 18%).

Ao analisar a árvore de decisão criada (com um depth máximo de 5 ) podemos concluir que a venda de um produto depende consideravelmente do número de indivíduos interessados no mesmo e do valor que os compradores estão dispostos a dar pelo produto. Quanto maior o número de compradores interessados e maior o valor que estes estão dispostos a dar pelo produto comparativamente ao seu valor inicial, maior a probabilidade de o produto ser vendido.

# Teste com variação de atributos

Uma vez que existe correlação considerável entre os atributos B e C, foi também testado um caso em que se retirou o atributo C verificando o impacto desse atributo no resultado final.

	Decision Tree	Random Forest	K-NN	Deep Learning	Neural Net	Gradient Boosted Trees
Accuracy testing (%)	82.94	83.26	82.60	80.40	81.40	81.40

Ao observar os resultados obtidos, é possível verificar que em geral são piores, com exceção do K-NN. A variação destes resultados não são significativos por isso podemos concluir que o atributo C (distância percentual entre o shipment delay do vendedor e o shipment delay dos seus concorrentes diretos) é pouco útil para a experiência realizada.

# Conclusões

Ao observar os resultados obtidos de cada modelo implementado, pode-se concluir que todos os modelos apresentam resultados bastante similares. Verifica-se que com os dados gerados foi obtido em qualquer modelo uma precisão de pelo menos 80% , ou seja em 80% dos dados testados o modelo concluiu corretamente se o produto foi ou não vendido.


Após as análises realizadas concluiu-se que o melhor modelo para esta classificação é Gradient Boosted Tree, uma vez que tem uma accuracy de 84.14%.

Ao testar o modelo, sem a avaliação do atributo C, também é possível concluir que foram obtidos piores resultados. Por exemplo, a precisão do Gradient Boosted Tree passou de 84.14% para 81.4%. Esta variação de 3% revela que esta variável apesar de ter alguma correlação com outra variável é útil para a experiência realizada.



# Parte 2

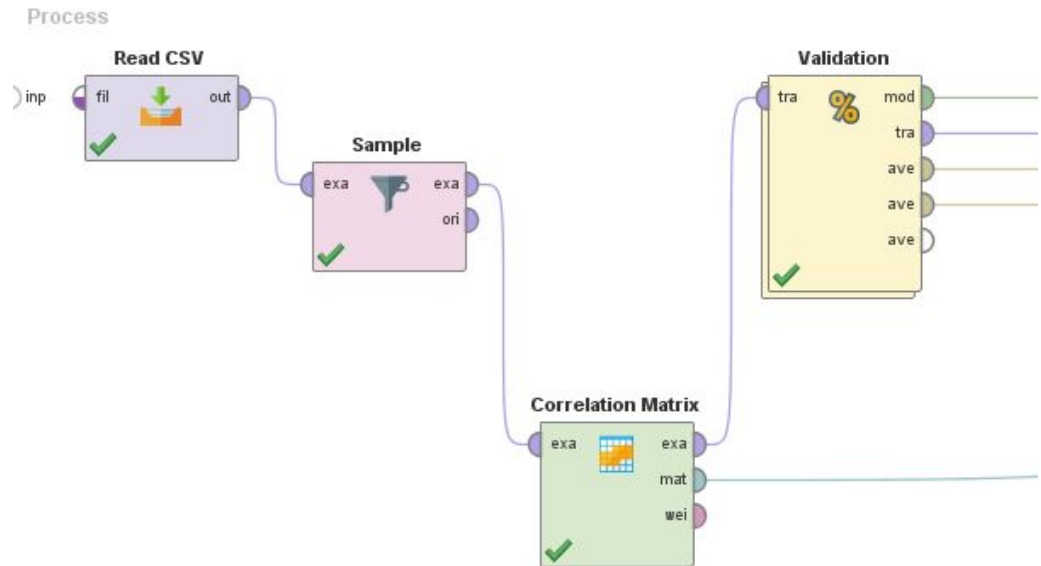
Informação adicional



# Processos rapidminer - Classificação

O processo utilizado na classificação de se um produto seria vendido ou não compreende os seguintes passos:

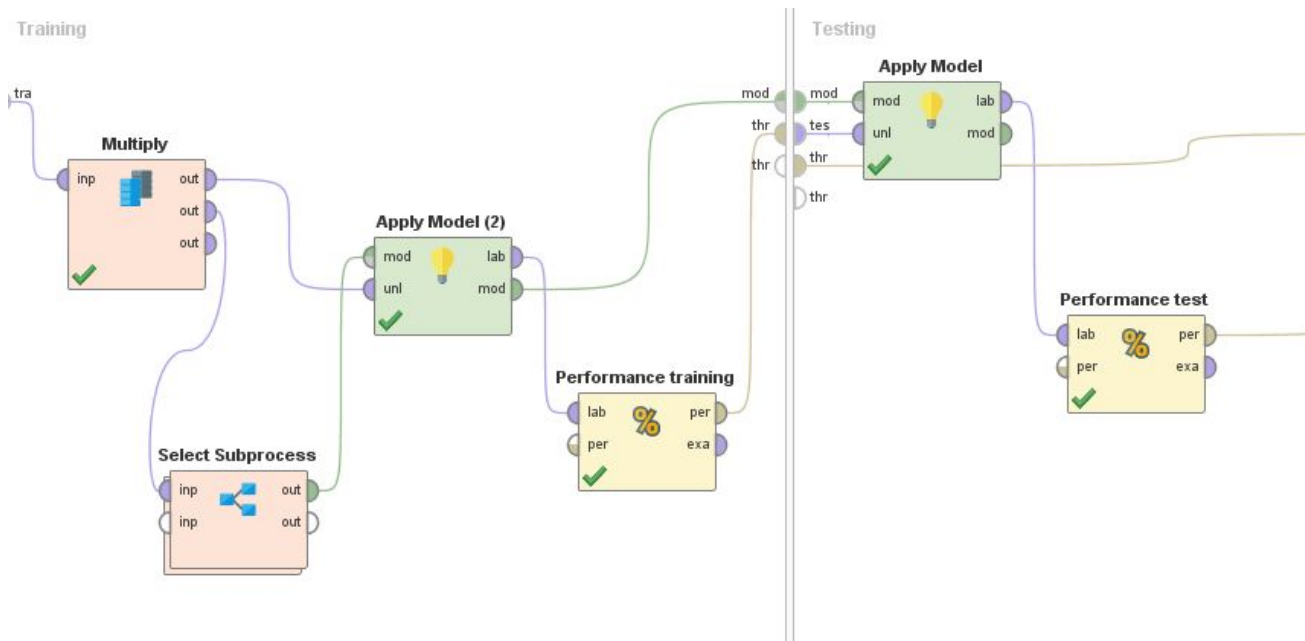
1. leitura dos dados (ficheiro .csv)
2. sampling dos mesmos, garantindo uma proporção equilibrada de dados de ambas as classes
3. construção da matriz de correlação
4. validação e aplicação do modelo



# Processos rapidminer - Classificação

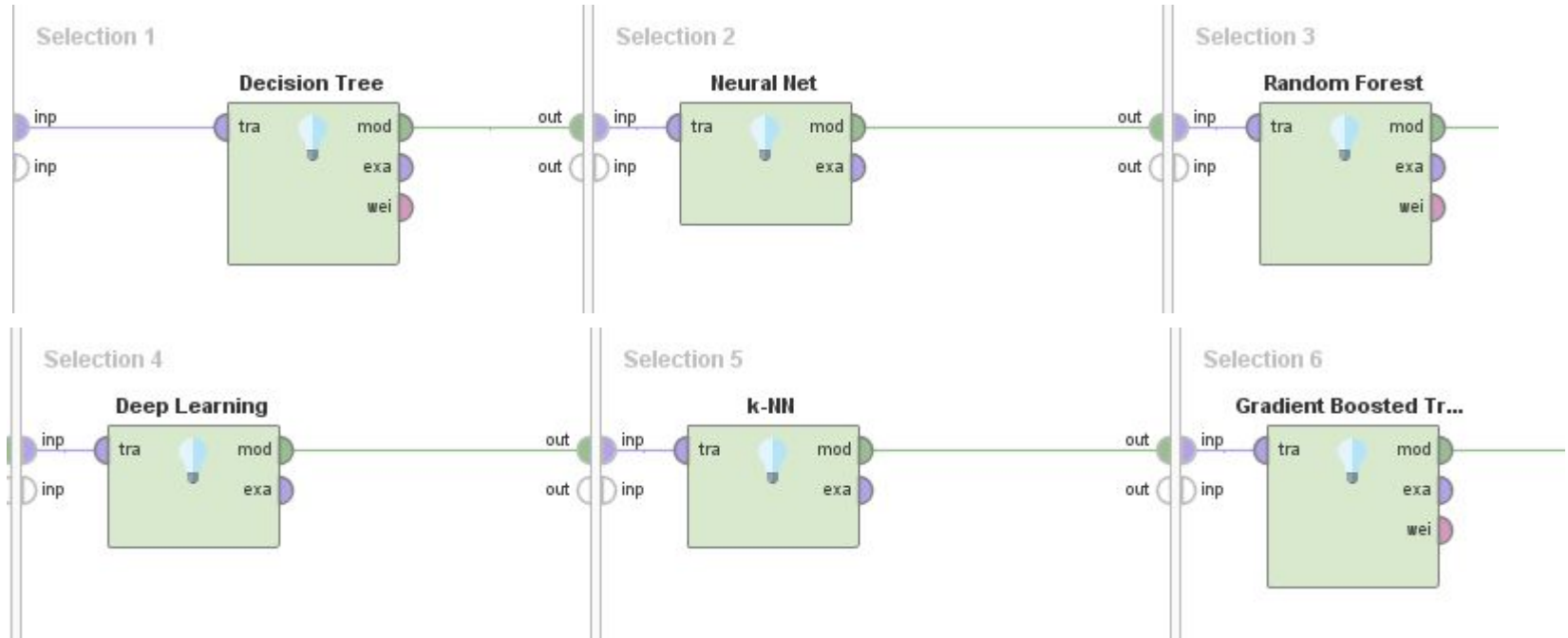
Na etapa de validação, aplicamos 70% dos dados para treino e 30% para teste.

Para tal aplicamos o modelo desejado e posteriormente medimos a sua performance.

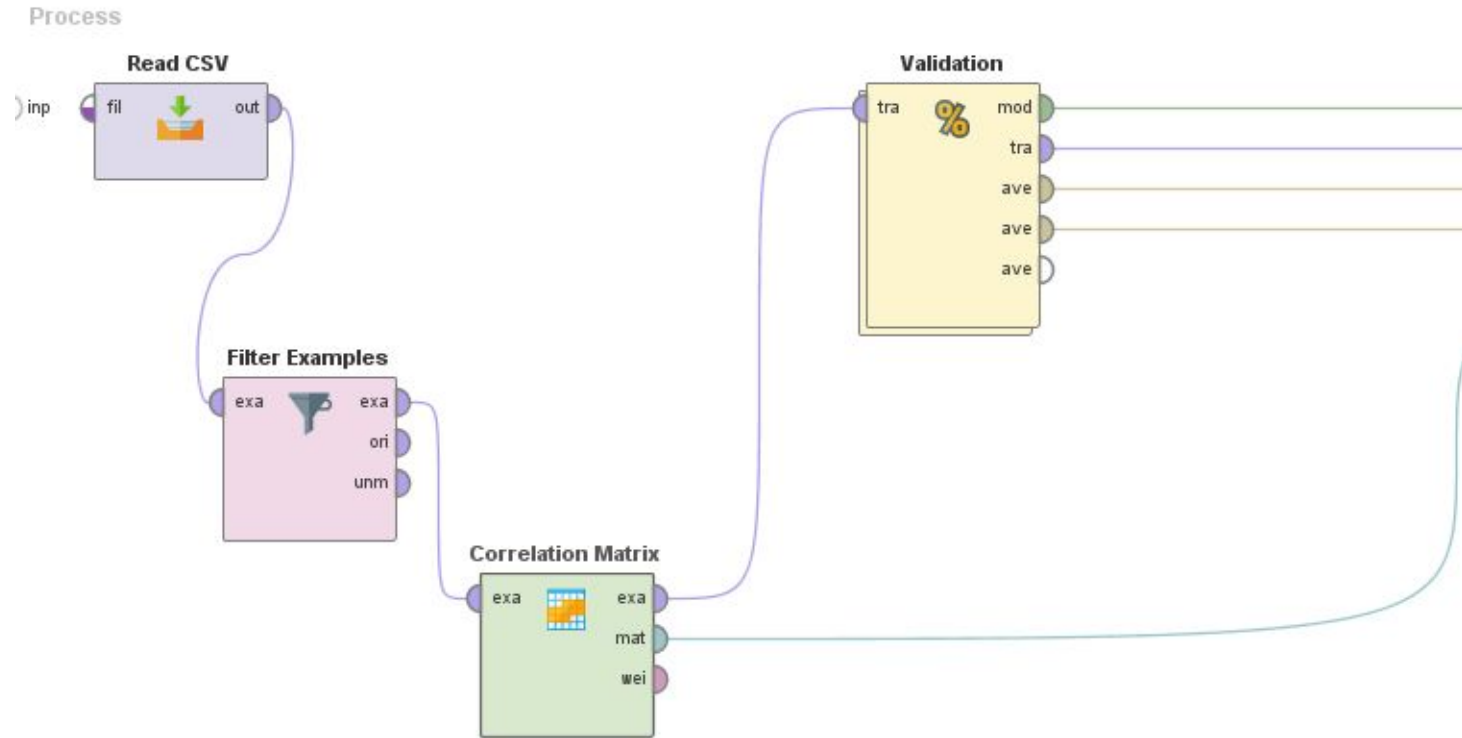


# Processos rapidminer - Classificação

Procuramos variar o modelo aplicado, considerando os já anteriormente mencionados.



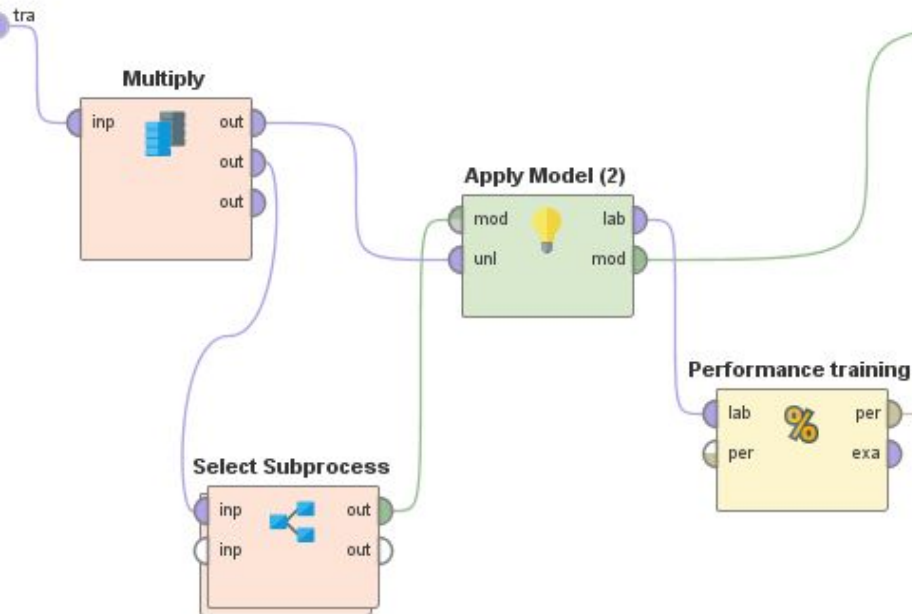
# Processos rapidminer - Regressão



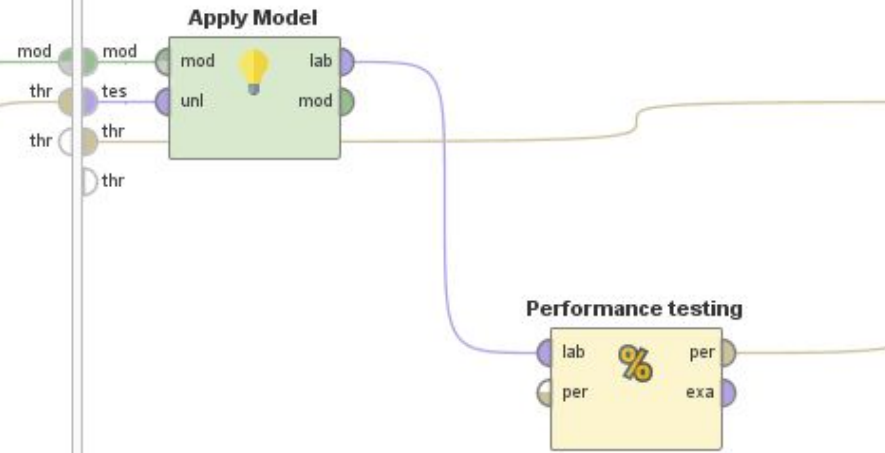


# Processos rapidminer - Regressão

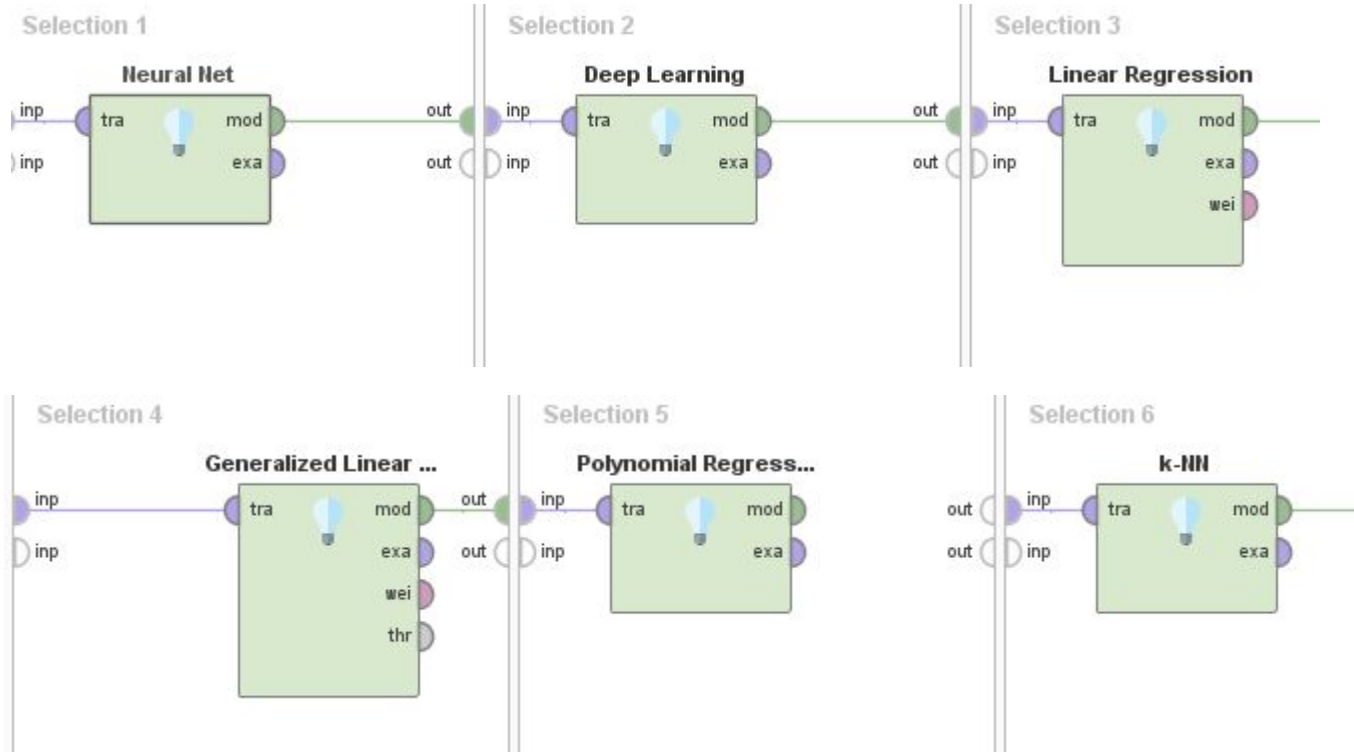
Training



Testing



# Processos rapidminer - Regressão



# Resultados de outras experiências interessantes

Seria também útil tentar prever qual seria o valor de venda de um produto (filtrando por nome de produto). Foram então aplicados alguns modelos, utilizando com variáveis independentes:

- Valor inicial do produto
- Desvio percentual entre o shipment delay do seller que está a vender o produto e a média do shipment delay de todos os sellers que vendem este mesmo produto
- Número total de leilões do mesmo produto (para que se possa ter em consideração o número de competidores)
- Número de interessados no produto (para que se possa ter em consideração a dimensão do público alvo)

Os modelos utilizados foram:

- Neural Net
- Deep Learning
- Linear Regression
- Generalized Linear Model
- Polynomial Regression
- K-NN

# Resultados de outras experiências interessantes

Por se tratar de um problema de regressão foram analisadas as seguintes variáveis:

- **Root Mean Squared Error (RMSE)** - mede a diferença entre a previsão e o valor efetivamente verificado
- **Validation runtime** - tempo de execução do modelo tendo em conta treino teste
- **Root Relative Squared Error (RRSE)** - compara o resultado obtido pelo nosso modelo relativamente a uma previsão mais fraca onde se considera apenas a média dos resultados. A análise deste parâmetro deve ser feita tendo em consideração os seguintes ranges de valores:
  - 0 : modelo perfeito
  - ]0,1[ : modelo útil
  - 1 : modelo igual ao trivial (valor médio)
  - > 1 : modelo pior que o trivial

# Resultados de outras experiências interessantes

	Neural Net	Deep Learning	Linear Regression	Generalized Linear Regression	Polynomial Regression	K-NN
Root mean squared error	2.651 +/- 0.00	2.864 +/- 0.00	2.857 +/- 0.00	2.857 +/- 0.00	57.064 +/- 0.00	2.978 +/- 0.00
Root relative squared error	0.716	0.774	0.772	0.772	15.415	0.804
Validation Runtime (ms)	15401	1838	415	246	509	491

Todos os modelos acima testados são úteis, pois o root relative squared error é inferior a 1 com exceção do *Polynomial Regression* que oferece resultados muito piores do que o modelo trivial. De notar ainda que o modelo que obteve melhores resultados foi o *Neural Net* uma vez que o erro é menor logo tem maior precisão, sendo que no entanto em termos de execução demora bastante mais do que modelos como a Linear Regression o que pode colocar alguma resistência à utilização deste modelo.

# Resultados de outras experiências interessantes

Foram também realizados testes utilizando como atributo a variável que identifica se o produto é ou não vendido, obtendo-se os seguintes resultados:

	Deep Learning	Generalized Linear Regression	K-NN
Root mean squared error	0.522 +/- 0.00	0.588 +/- 0.00	1.521 +/- 0.00
Root relative squared error	0.141	0.159	0.411
Validation Runtime (ms)	2081	239	646

Desta segunda experiência consegue-se concluir que utilizando o atributo “sold” a previsão do valor de venda é bastante melhor principalmente se for usado o modelo deep learning, pelo que esta variável tem bastante relevância para a previsão do valor final.