UNIFESP

UNIVERSIDADE FEDERAL DE SÃO PAULO

1933

ANÁLISE DE DEMANDA VIA INTELIGÊNCIA ARTIFICIAL NO RESTAURANTE UNIVERSITÁRIO DO INSTITUTO DE CIÊNCIA E TECNOLOGIA DA UNIFESP

Douglas Diniz Landim, ddlandim@unifesp.br
RA 76681
Ciência da Computação.
Disciplina de Inteligência Artificial
Prof. Dr. Fabio Augusto Faria

Motivação

428.620 refeições subsidiadas no banco de dados do sistema antigo, no período de 2011 à 2016.

111.454 refeições no período de 2017 a 01/08/2018 que fecham o modelo de contrato antigo.

Valor pago em cada refeição pela UNIFESP: R$9,14. Total investido: R$4.936.276,36

Valor pago pelo aluno: R$2,50 pelo aluno. Total investido R$1.350.185,00.

Movimentação do restaurante: R$6.286.461,36

Já no 2o semestre de 2018, iniciando em 01/08/2018 à 31/10/2018, o valor total subsidiado pela universidade é de R$179.019,52, R$82.270,00 pelos alunos e o faturamento bruto acumulado pelo restaurante é de R$261.289,52

Referências:

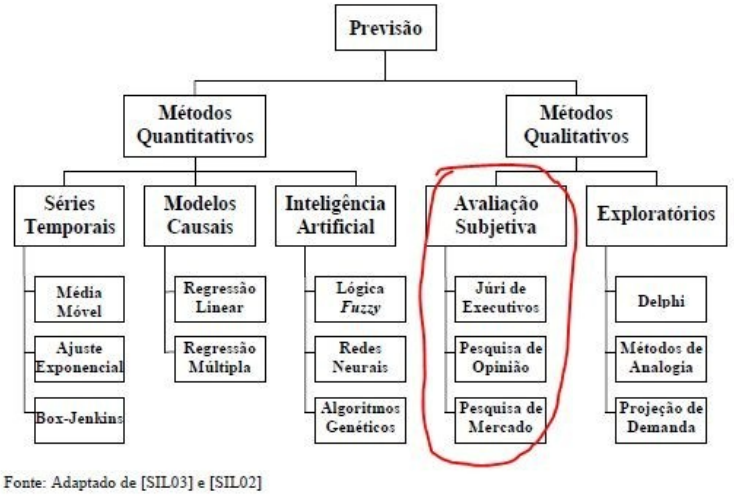
Total de refeições: ti.sjc@unifesp.br (Francismar / Fiscal de Contrato do R.U)

Valores por refeição: Ederson Barroso, ederbarroso@gmail.com , gerente nutrimenta.

Desperdício

ATUALMENTE O RESTAURANTE NÃO TEM NENHUM MODELO PREDITIVO.

- MÉTODOS SUBJETIVOS
- ANÁLISE DOS DIAS ANTERIORES



Fonte: Adaptado de [SIL03] e [SIL02]

Retirado de Junior, 2007. Análise de previsão de demanda baseado em séries temporais em uma empresa do setor de perfumes e cosméticos.

Trabalhos anteriores

DATA	VENDAS	TEMPERATURA
10/08/16	316	18
17/08/16	303	28
24/08/16	291	23
07/09/16	310	21
14/09/16	381	27.5
21/09/16	291	20
28/09/16	291	24

TABELA 1

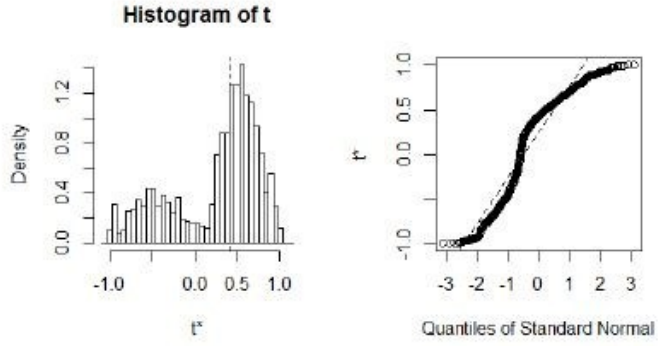
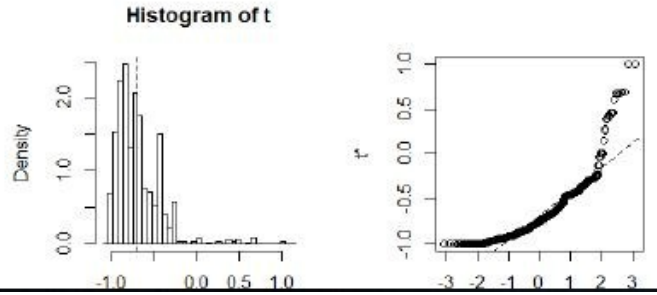


GRÁFICO 2

GRÁFICO 2: Cálculo bootstrap de 1000 reamostragens t da TABELA 1 em função da densidade.
Intervalo de confiança obtido por Bca: 95% (-0.8666, 0.9290), outras estatísticas obtidas pela biblioteca: Original: 0.4040055, Bias: -0.148679, Erro padrão: 0.51477

DATA	VENDAS	TEMPERATURA
28/09/16	291	27
05/10/16	284	21
19/10/16	78	35
26/10/16	277	30
09/11/16	274	31



Principais referências de heurísticas de previsão de demanda.

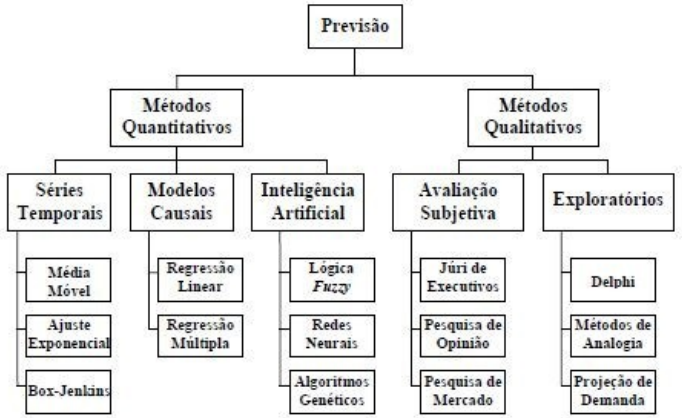
ALBINO MILESKI JUNIOR

ANÁLISE DE MÉTODOS DE PREVISÃO DE DEMANDA BASEADOS EM SÉRIES TEMPORAIS EM UMA EMPRESA DO SETOR DE PERFUMES E COSMÉTICOS

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Produção e Sistemas da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Engenharia de Produção e Sistemas.

CURITIBA
2007

Figura 2.3: Esquema do trabalho.



Fonte: Adaptado de [SIL03] e [SIL02]

Figura 3.4: Métodos para previsão da demanda.

Principais referências de modelos estatísticos.

Modelos de Regressão

Clarice Garcia Borges Demétrio
Departamento de Ciências Exatas, ESALQ, USP
Caixa Postal 9
13418-900 Piracicaba, SP
Email: Clarice@carpa.ciagri.usp.br
Fax: 019 34294346

Sílvio Sandoval Zocchi
Departamento de Ciências Exatas, ESALQ, USP
Caixa Postal 9
13418-900 Piracicaba, SP
Email: sszocchi@carpa.ciagri.usp.br
Fax: 019 34294346

29 de março de 2011

y	x_1	x_2	$\cdots$	x_p
y_1	x_{11}	x_{12}	$\cdots$	x_{1p}
y_2	x_{21}	x_{22}	$\cdots$	x_{2p}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
y_n	x_{n1}	x_{n2}	$\cdots$	x_{np}

Tabela 2.2.1: Representação dos dados.

$$\widehat{\beta} = (X'X)^{-1}X'Y.$$

Notemos que os estimadores de mínimos quadrados dos parâmetros do "Modelo 2.2" podem ser facilmente encontrados considerando a notação matricial dos dados, que é de fácil manipulação. Desta forma, considerando a entrada de dados apresentada na Tabela 2.2.1, o modelo de Regressão Linear Múltipla pode ser escrito como

$$Y = X\beta + \varepsilon,$$

com

$$Y = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix}, \quad X = \begin{bmatrix} 1 & x_{11} & x_{12} & \ldots & x_{1p} \\ 1 & x_{21} & x_{22} & \ldots & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \ldots & x_{np} \end{bmatrix}, \quad \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix} \text{ e } \varepsilon = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix},$$

em que

- Y é um vetor $n \times 1$ cujos componentes corresponde às n respostas;
- X é uma matriz de dimensão $n \times (p+1)$ denominada matriz do modelo;
- ε é um vetor de dimensão $n \times 1$ cujos componentes são os erros e
- β é um vetor $(p+1) \times 1$ cujos elementos são os coeficientes de regressão.

O método de mínimos quadrados tem como objetivo encontrar o vetor $\widehat{\beta}$ que minimiza

$$L = \sum_{i=1}^{n} \varepsilon_i^2 = \varepsilon'\varepsilon = (Y - X\beta)'(Y - X\beta) =$$

$$= Y'Y - Y'X\beta - \beta'X'Y + \beta'X'X\beta = Y'Y - 2\beta'X'Y + \beta'X'X\beta,$$

sendo que $Y'X\beta = \beta'X'Y$ pois o produto resulta em um escalar. A notação X' representa o transposto da matriz X enquanto que Y' e β' representam os transpostos dos vetores Y e β, respectivamente. Usando a técnica de derivação (em termos matriciais) obtemos

$$\frac{\partial L}{\partial \beta} = -2X'Y + 2X'X\beta.$$

Principais referências de modelos de inteligência artificial em R.U.

IV CONGRESSO NACIONAL DE EXCELÊNCIA EM GESTÃO
31 de Julho a 02 de Agosto de 2008

MODELOS PARA PREVISÃO DE DEMANDA NO RESTAURANTE UNIVERSITÁRIO UTILIZANDO TÉCNICAS DE REDES NEURAIS

Liliane Lopes Cordeiro (DMA - UFV)
lililopescordeiro@yahoo.com.br
Heverton Augusto Pereira (Unicamp)
hevertonaugusto@yahoo.com.br

Resumo

Um dos grandes problemas enfrentados hoje no mundo é a elevação dos preços dos alimentos. Isto tem causado preocupações para a população em geral e também para as empresas como restaurantes que sofrem diretamente os reflexos da variação no preço dos alimentos. Atualmente o Restaurante Universitário (R.U.) da Universidade Federal de Viçosa não possui um sistema que ajude na gestão de compras dos alimentos. O objetivo deste trabalho é utilizar a técnica de Redes Neurais Artificiais do tipo MLP (Perceptron Multiplas Camadas) para fazer a predição do número de usuários que irão fazer suas refeições no R.U. em uma, duas e três semanas para a administração poder determinar a política de compras de alimentos. As redes desenvolvidas utilizam o dia da semana e os cinco dias anteriores ao que se deseja prever. Para validar os modelos propostos foram separados conjuntos de dados para realização de comparações e analises da eficácia da nova forma de gestão das compras.

Abstract

Nowadays, one of the major problems in the world is the rising of food prices. The problem concerns general population and also businesses such as restaurants that suffer directly the consequences of changes in the food prices. Currently tthe Universtty Restaurant (R.U.), at Federal University of Viçosa does not have a system that helps in the management to buy food. This work uses the technique of Artificial Neural Networks MLP type (Multiple Layers Perceptron) to predict the number of users who will have their meals in the R.U. in one, two and three weeks to support the administration's decision of food storing. The developed networks have as input variables: day of the week and the previous five days until the day that want to predict. Different data were used to validate the models through comparisons and analysis of the new management food buying benefits.

Palavras-chaves: Redes Neurais Artificiais, Previsão, Gestão

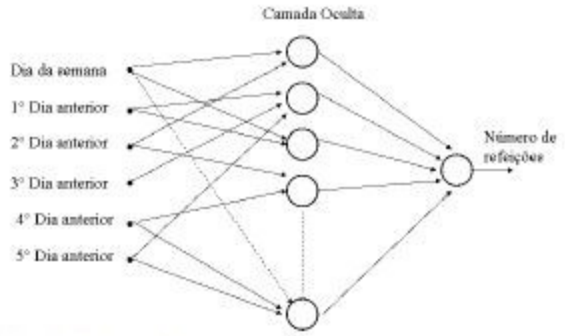


Figura 4 – Entradas e saída da rede proposta

Media do erro (Refeições)		Total (5 dias)			
Treinamento	Validação	Previsto	Esperado	Erro (Refeições)	% erro
-4,3	-44,9	17124	16900	224	1,32

Media do erro (Refeições)		Total (10 dias)			
Treinamento	Validação	Previsto	Esperado	Erro (Refeições)	% erro
0,92	-107,2	36534	35462	1072	3,0

Media do erro (Refeições)		Total (15 dias)			
Treinamento	Validação	Previsto	Esperado	Erro (Refeições)	% erro
1,7	46,8	51785	52488	703	1,34

Principais referências de modelos de inteligência artificial em R.U.

ORIGINAL | *ORIGINAL*

Utilização de redes neurais artificiais para a determinação do número de refeições diárias de um restaurante universitário

Use of artificial neural networks to determine the daily number of meals served by a university cafeteria

José Celso ROCHA[1]
Felipe Delestro MATOS[1]
Fernando FREI[1]

RESUMO

Objetivo
Construir uma rede neural artificial para auxiliar os gestores de restaurantes universitários na previsão de refeições diárias.

Métodos
O estudo foi desenvolvido a partir do levantamento de oito variáveis que influenciam o número de refeições diárias servidas no restaurante universitário. Utiliza-se o algoritmo de treinamento Backpropagation. Os resultados por meio da rede são comparados com os da série estudada e com resultados da estimação por média aritmética simples.

Resultados
A rede proposta acompanha as inúmeras alterações que ocorrem no número de refeições diárias do restaurante universitário. Em 73% dos dias analisados, o método das redes neurais artificiais apresenta uma taxa de acerto maior do que o método da média aritmética simples.

Conclusão
A rede neural artificial mostrou-se mais adequada para a previsão do número de refeições do que a metodologia de média simples ou quando a decisão do número de refeições é feita de forma subjetiva, sem critérios científicos.

Termos de indexação: Desperdícios de alimentos. Redes neurais artificiais. Serviços de alimentação.

[1] Universidade Estadual Paulista Júlio de Mesquita Filho, Faculdade de Ciências e Letras, Curso de Engenharia Biotecnológica, Departamento de Ciências Biológicas. Av. Dom Antônio, 2100, 19806-900, Assis, SP, Brasil. Correspondência para/*Correspondence to*: J.C. ROCHA. E-mail: <jcelso@assis.unesp.br>.

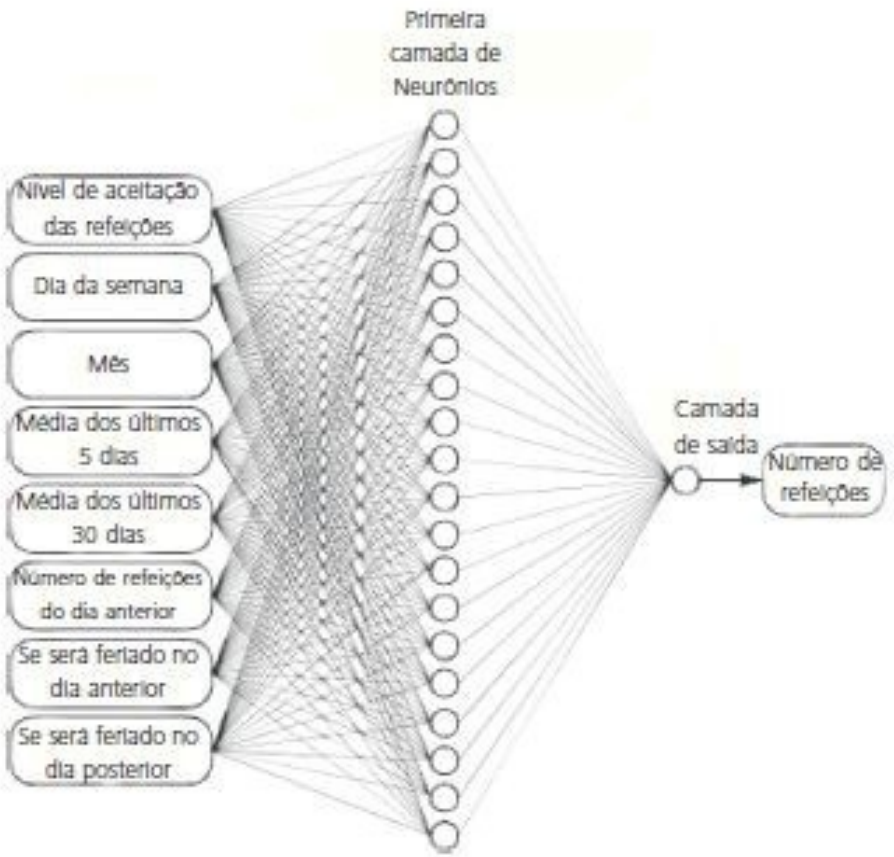


Figura 1. Arquitetura da rede neural artificial.

Principais referências de modelos de inteligência artificial em R.U.

Utilização de redes neurais artificiais para a determinação do número de refeições diárias de um restaurante universitário

Use of artificial neural networks to determine the daily number of meals served by a university cafeteria

José Celso ROCHA[1]
Felipe Delestro MATOS[1]
Fernando FREI[1]

RESUMO

Objetivo
Construir uma rede neural artificial para auxiliar os gestores de restaurantes universitários na previsão de refeições diárias.

Métodos
O estudo foi desenvolvido a partir do levantamento de oito variáveis que influenciam o número de refeições diárias servidas no restaurante universitário. Utiliza-se o algoritmo de treinamento Backpropagation. Os resultados por meio da rede são comparados com os da série estudada e com resultados da estimação por média aritmética simples.

Resultados
A rede proposta acompanha as inúmeras alterações que ocorrem no número de refeições diárias do restaurante universitário. Em 73% dos dias analisados, o método das redes neurais artificiais apresenta uma taxa de acerto maior do que o método da média aritmética simples.

Conclusão
A rede neural artificial mostrou-se mais adequada para a previsão do número de refeições do que a metodologia de média simples ou quando a decisão do número de refeições é feita de forma subjetiva, sem critérios científicos.

Termos de indexação: Desperdícios de alimentos. Redes neurais artificiais. Serviços de alimentação.

[1] Universidade Estadual Paulista Júlio de Mesquita Filho, Faculdade de Ciências e Letras, Curso de Engenharia Biotecnológica, Departamento de Ciências Biológicas. Av. Dom Antônio, 2100, 19806-900, Assis, SP, Brasil. Correspondência para/Correspondence to: J.C. ROCHA. E-mail: <jcelso@assis.unesp.br>.

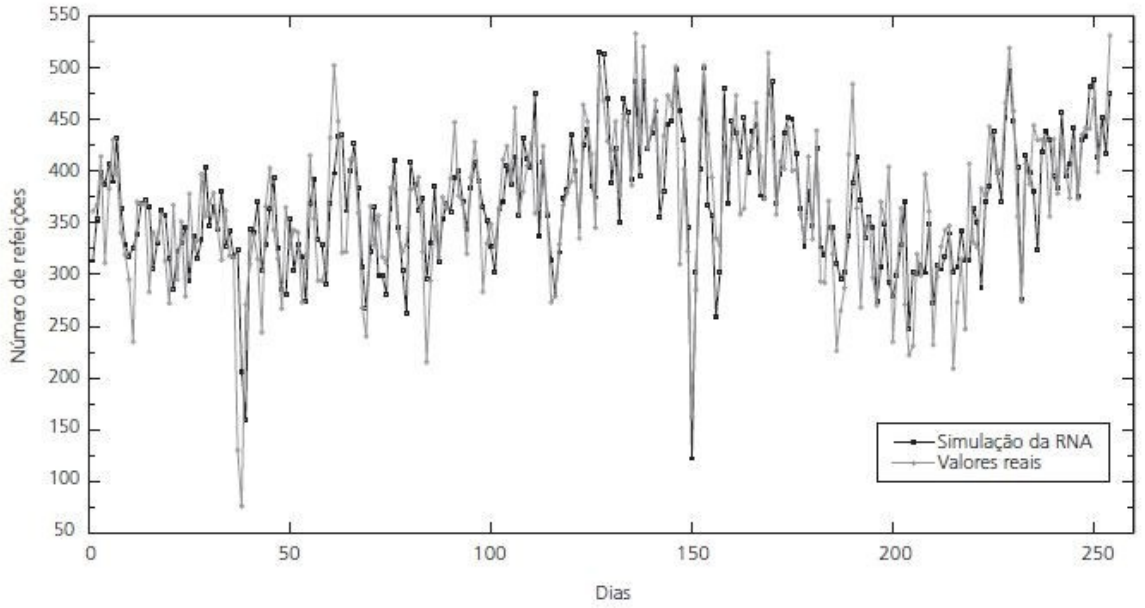
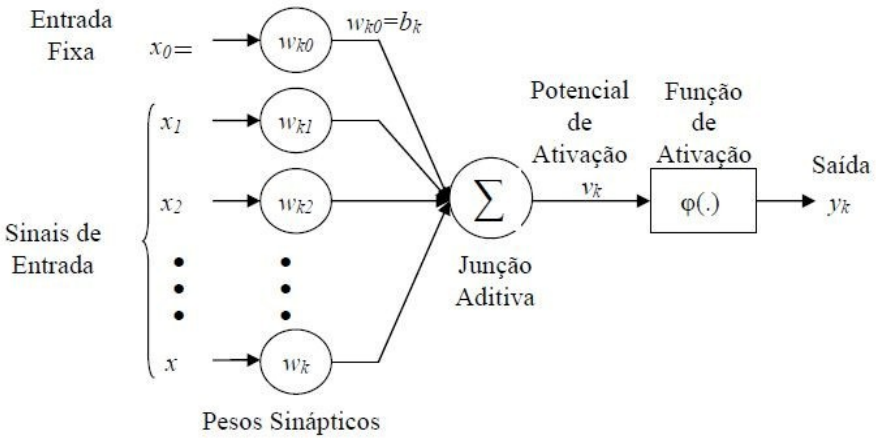
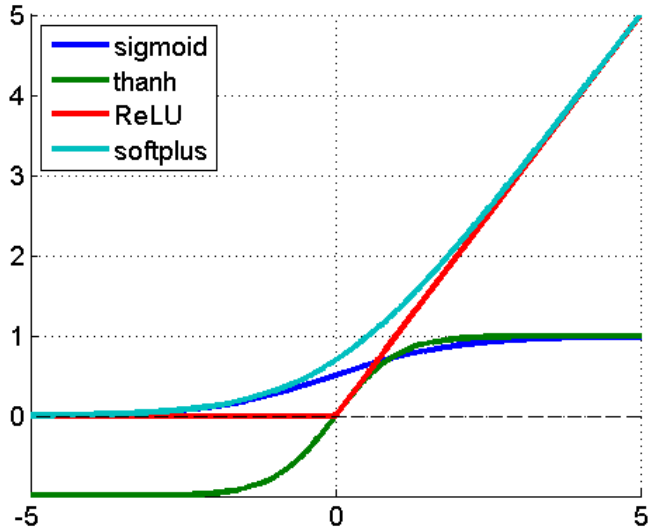


Figura 3. Desempenho da rede neural artificial em comparação ao número de refeições reais.

Para o estudo em pauta, o erro geral obtido pela metodologia da RNA foi de 9,5%.

Técnica do modelo : Perceptron com treino Backpropagation



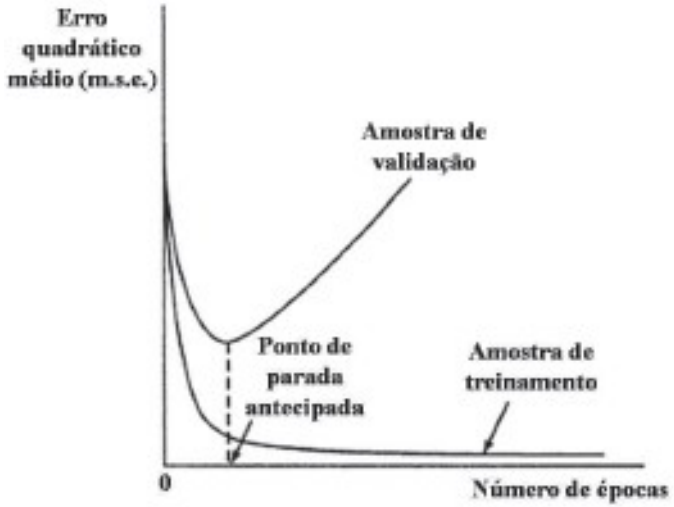


$$w \leftarrow w - \eta \frac{\partial E}{\partial w}$$

o valor do peso na iteração atual será o valor do peso na iteração anterior, corrigido de valor proporcional ao gradiente.

Perceptron com treino Backpropagation.

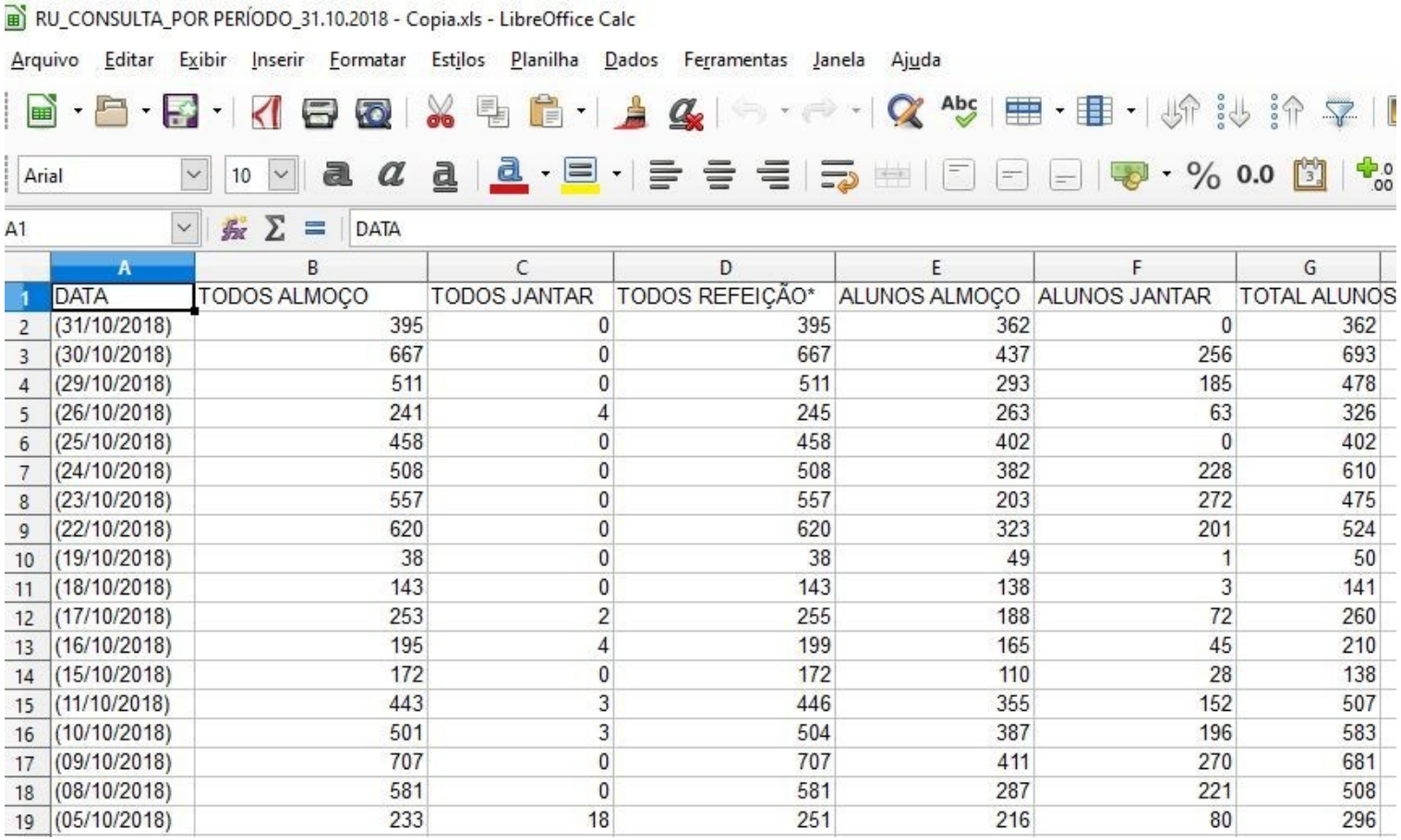
Condição de parada da atualização dos pesos:



Early Stopping

```
>>> from keras.callbacks import EarlyStopping
>>> early_stopping_monitor = EarlyStopping(patience=2)
>>> model3.fit(x_train4,
               y_train4,
               batch_size=32,
               epochs=15,
               validation_data=(x_test4,y_test4),
               callbacks=[early_stopping_monitor])
```

Dados extraídos, T.I.


RU_CONSULTA_POR PERÍODO_31.10.2018 - Copia.xls - LibreOffice Calc

Arquivo Editar Exibir Inserir Formatar Estilos Planilha Dados Ferramentas Janela Ajuda

Arial 10

A1 DATA

	A	B	C	D	E	F	G
1	DATA	TODOS ALMOÇO	TODOS JANTAR	TODOS REFEIÇÃO*	ALUNOS ALMOÇO	ALUNOS JANTAR	TOTAL ALUNOS
2	(31/10/2018)	395	0	395	362	0	362
3	(30/10/2018)	667	0	667	437	256	693
4	(29/10/2018)	511	0	511	293	185	478
5	(26/10/2018)	241	4	245	263	63	326
6	(25/10/2018)	458	0	458	402	0	402
7	(24/10/2018)	508	0	508	382	228	610
8	(23/10/2018)	557	0	557	203	272	475
9	(22/10/2018)	620	0	620	323	201	524
10	(19/10/2018)	38	0	38	49	1	50
11	(18/10/2018)	143	0	143	138	3	141
12	(17/10/2018)	253	2	255	188	72	260
13	(16/10/2018)	195	4	199	165	45	210
14	(15/10/2018)	172	0	172	110	28	138
15	(11/10/2018)	443	3	446	355	152	507
16	(10/10/2018)	501	3	504	387	196	583
17	(09/10/2018)	707	0	707	411	270	681
18	(08/10/2018)	581	0	581	287	221	508
19	(05/10/2018)	233	18	251	216	80	296

Dados extraídos, BMDEP.

Não seguro | www.inmet.gov.br/projetos/rede/pesquisa/form_mapas_c_diario.php

unifesp | utilitarios | pessoal | profissional | Importado do Firefox | monografy-ann-dem | ScrumHalf - Painel | Segredos Unifesp | FINANÇAS REP POR

BDMEP - Série Histórica - Dados Diários

Período - Data inicio (dd/mm/aaaa) : 01/01/2017 fim : 31/10/2018

Região : Todas (OU) Estado : Todos

Selecionar Variáveis :

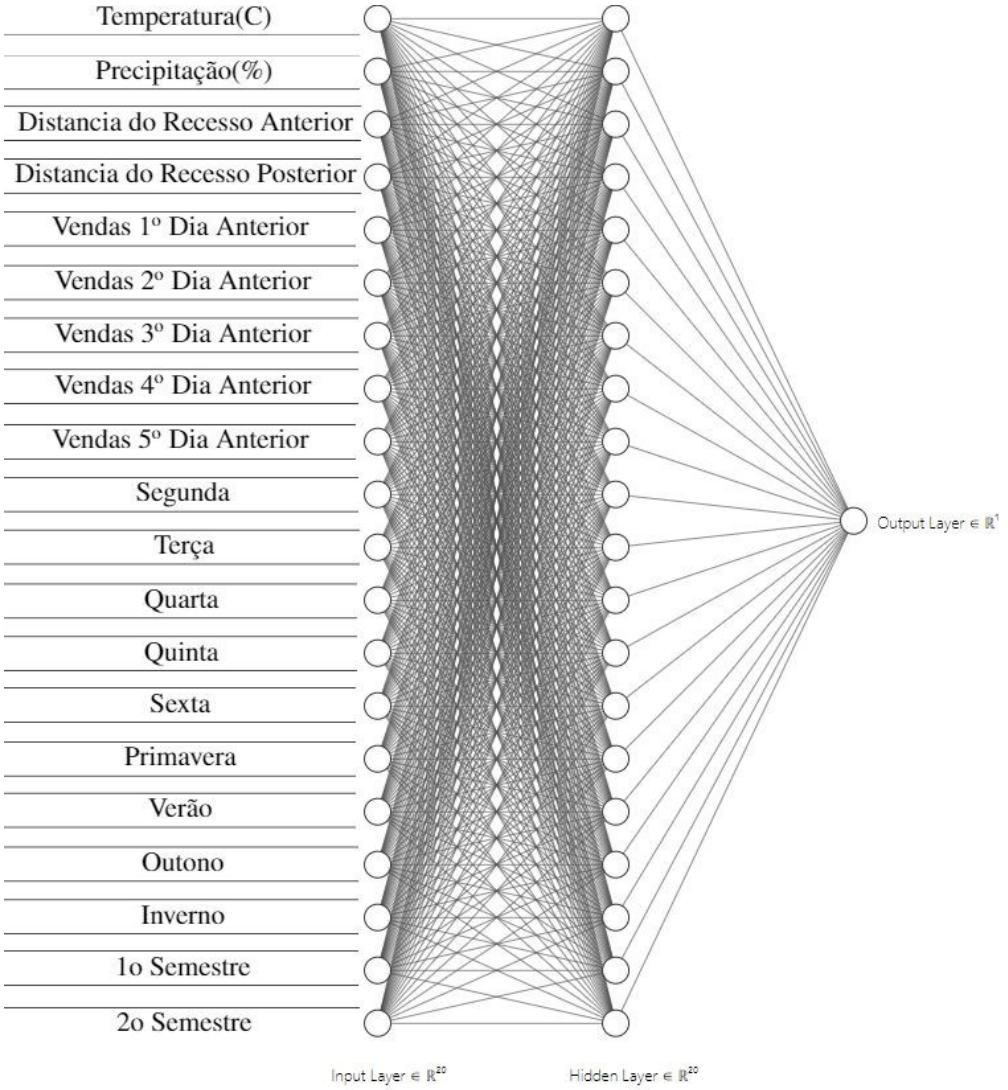
- [x] Precipitação(mm)
- [x] Temp Máxima(°C)
- [x] Temp Mínima(°C)
- [x] Insolação(horas)
- [x] Evaporação do Piche(mm)
- [x] Temperatura Compensada Média(°C)
- [x] Umidade Relativa Média(%)
- [x] Velocidade Vento Média(mps)

Pesquisa

J1 | Umidade Relativa Media

	A	B	C	D	E	F	G	H	I	J	K
1	Estacao	Data	Hora	Precipitacao	TempMaxima	TempMinima	Insolacao	Evaporacao Piche	Temp Comp Media	Umidade Relativa Media	Velocidade do Vento Media
1076	83781	22/06/2018	0		24.6		5.1	5.3	19.12	59.75	1.733333
1077	83781	22/06/2018	1200	0		15					
1078	83781	23/06/2018	0		26.4		6.8	4	20.28	60.25	1.866667
1079	83781	23/06/2018	1200	0		14.6					
1080	83781	24/06/2018	0		28.5		7.4	4.4	20.72	67	2.1
1081	83781	24/06/2018	1200	0		17.5					
1082	83781	25/06/2018	0		27.2		8.4	3.1	21.14	51.25	1.9
1083	83781	25/06/2018	1200	0		15.9					
1084	83781	26/06/2018	0		26.5		2.5	8	21.46	45	1.633333
1085	83781	26/06/2018	1200	0		16.6					
1086	83781	27/06/2018	0		23.4		7.3	6	18.62	70.75	1.9
1087	83781	27/06/2018	1200	0		15.5					
1088	83781	28/06/2018	0	26			5.2	3	19.58	70	1.6
1089	83781	28/06/2018	1200	0		15.2					
1090	83781	29/06/2018	0		26.4		8.5	3.5	20.06	57.5	1.966667
1091	83781	29/06/2018	1200	0		14.7					
1092	83781	30/06/2018	0		26.8		6.6	4.7	20.18	72	0.666667
1093	83781	30/06/2018	1200	0		15.1					
1094	83781	01/07/2018	0		27.4		7.9		20.56	61.5	0.51444
1095	83781	01/07/2018	1200	0		15					
1096	83781	02/07/2018	0		27.3		7.9		20.9	50.5	1.37184
1097	83781	02/07/2018	1200	0		15					
1098	83781	03/07/2018	0		24.7		5.9		18.72	78	2.5722
1099	83781	03/07/2018	1200	0		16.2					
1100	83781	04/07/2018	0		23.7		5.7		17.96	83.75	2.5722
1101	83781	04/07/2018	1200	0		15.6					

1º Modelo proposto.



O sinal de saída do perceptron entende-se então por: $y = \delta(\sum_{i=1}^{n} XiWi + b)$

- Xi - sinais de entrada do neurônio;
- Wi - pesos sinápticos do neurônio;
- b - bias ou limiar de ativação;
- $\delta(.)$ - função de ativação;
- y - sinal de saída do neurônio.

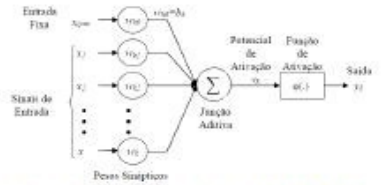
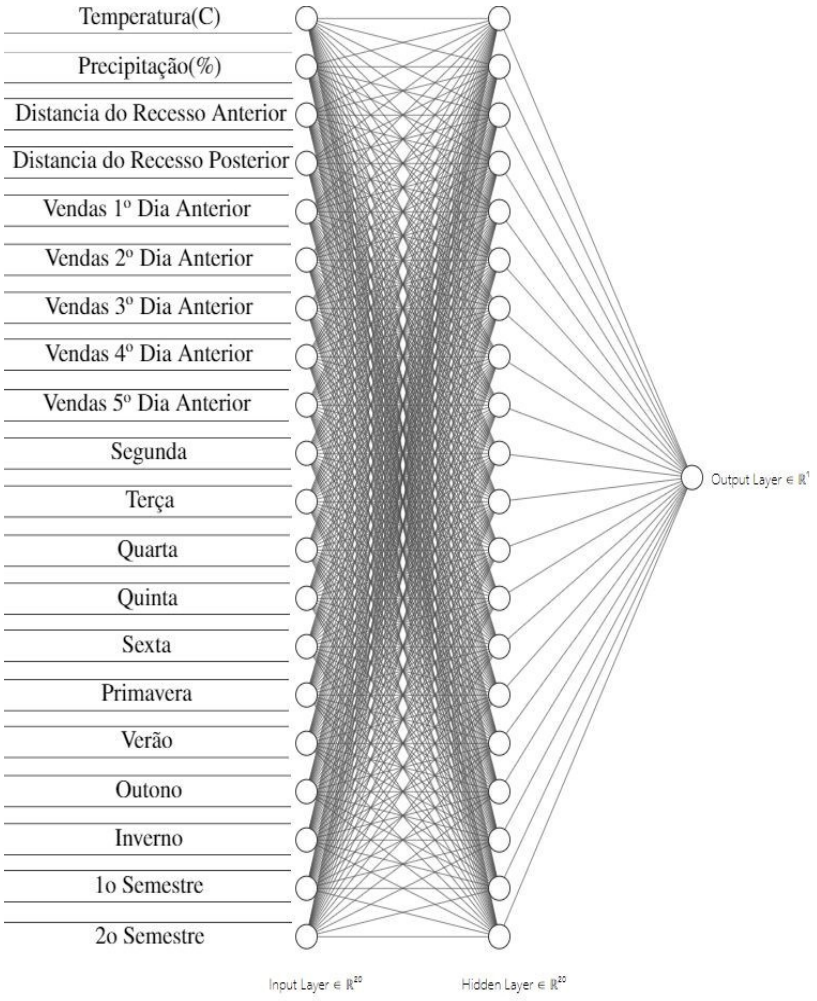
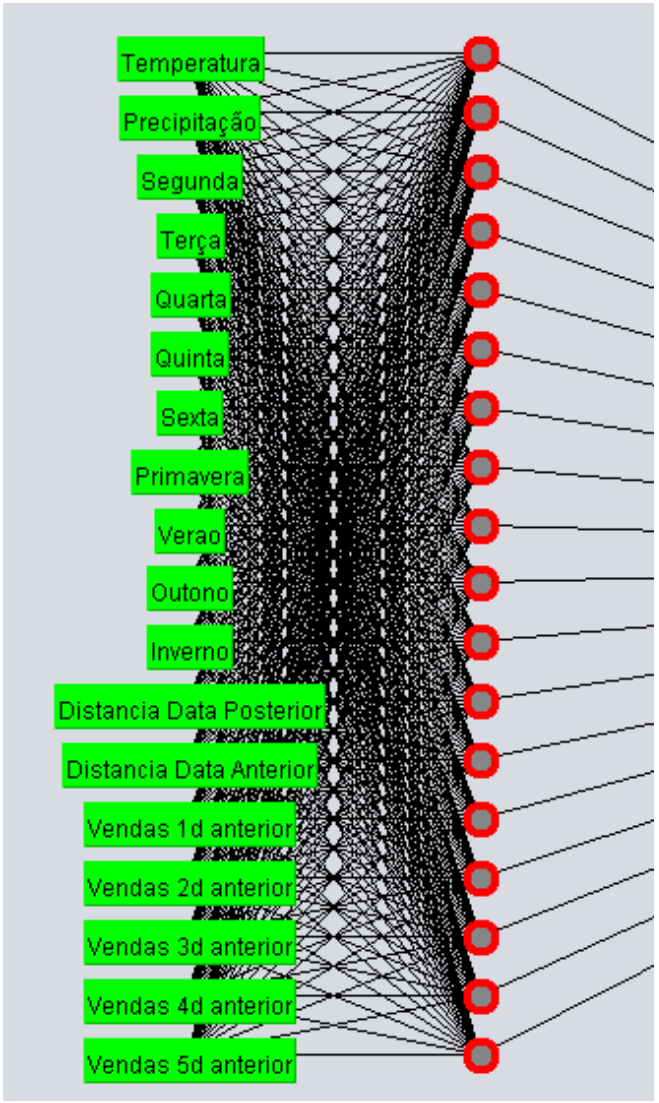


Figura 7 – Neurônio Artificial Perceptron, retirado de (JUNIOR, 2007) ()

Adaptação do modelo, exclusão e troca de entradas com conteúdo ambíguo.

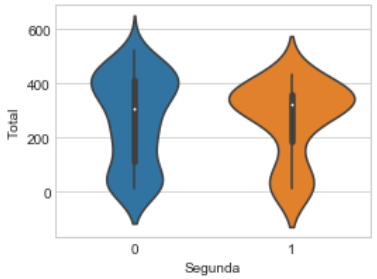


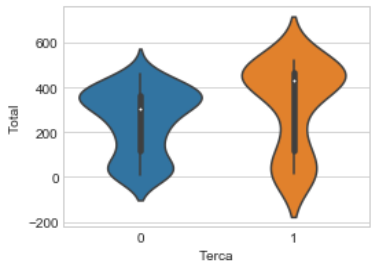


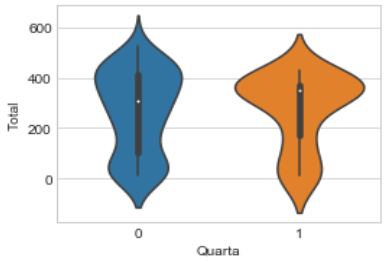
```
In [7]: original_data.describe()
```

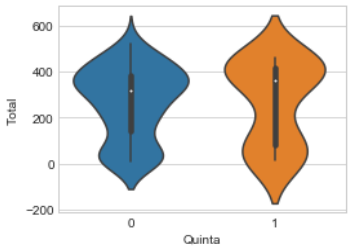
Out[7]:

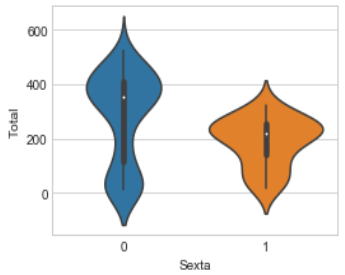
	Temperatura	Precipitacao	Segunda	Terca	Quarta	Quinta	Sexta	Primavera	Verao	Outono	Inverno
count	148.000000	148.000000	148.000000	148.000000	148.000000	148.000000	148.000000	148.000000	148.0	148.000000	148.000000
mean	25.097297	73.442568	0.216216	0.216216	0.209459	0.189189	0.168919	0.391892	0.0	0.297297	0.310811
std	4.246006	12.661813	0.413061	0.413061	0.408305	0.392989	0.375953	0.489830	0.0	0.458621	0.464397
min	16.000000	35.500000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000
25%	22.275000	68.875000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000
50%	25.100000	74.500000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000
75%	28.200000	80.812500	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.0	1.000000	1.000000
max	34.700000	96.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	0.0	1.000000	1.000000











In [14]:
```
X_train = original_data.iloc[:,1:19]
```

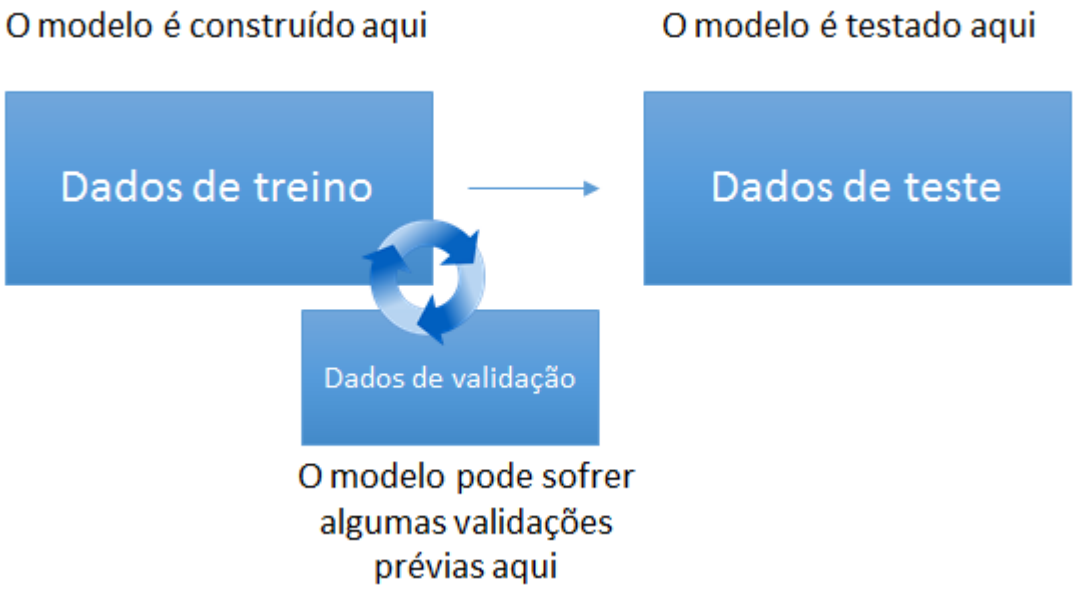
In [15]:
```
X_train.values
```

Out[15]:
```
array([[ 29.9 ,  77.25,   0.  , ..., 101.  , 178.  , 215.  ],
       [ 29.2 ,  74.25,   1.  , ..., 178.  , 215.  , 263.  ],
       [ 31.9 ,  67.5 ,   0.  , ..., 215.  , 263.  , 145.  ],
       ...,
       [ 24.1 ,  88.25,   1.  , ...,   0.  ,   0.  ,   0.  ],
       [ 20.3 ,  78.75,   0.  , ...,   0.  ,   0.  ,   0.  ],
       [ 25.2 ,  83.5 ,   0.  , ...,   0.  ,   0.  ,   0.  ]])
```

In [24]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler().fit(X_train)
standardized_X = scaler.transform(X_train)
standardized_X_test = scaler.transform(X_test)
```

In [25]:
```
standardized_X
```

Out[25]:
```
array([[ 1.13495156,  0.30172306, -0.52522573, ..., -1.02827871,
        -0.53945104, -0.29949702],
       [ 0.96953093,  0.06398563,  1.90394328, ..., -0.54679855,
        -0.30940401, -0.00345878],
       [ 1.60758192, -0.47092356, -0.52522573, ..., -0.31543795,
        -0.01096462, -0.73121946],
       ...,
       [-0.23567649,  1.17342694,  1.90394328, ..., -1.65983061,
        -1.64616377, -1.62550165],
       [-1.13367418,  0.42059177, -0.52522573, ..., -1.65983061,
        -1.64616377, -1.62550165],
       [ 0.02427021,  0.79700935, -0.52522573, ..., -1.65983061,
        -1.64616377, -1.62550165]])
```



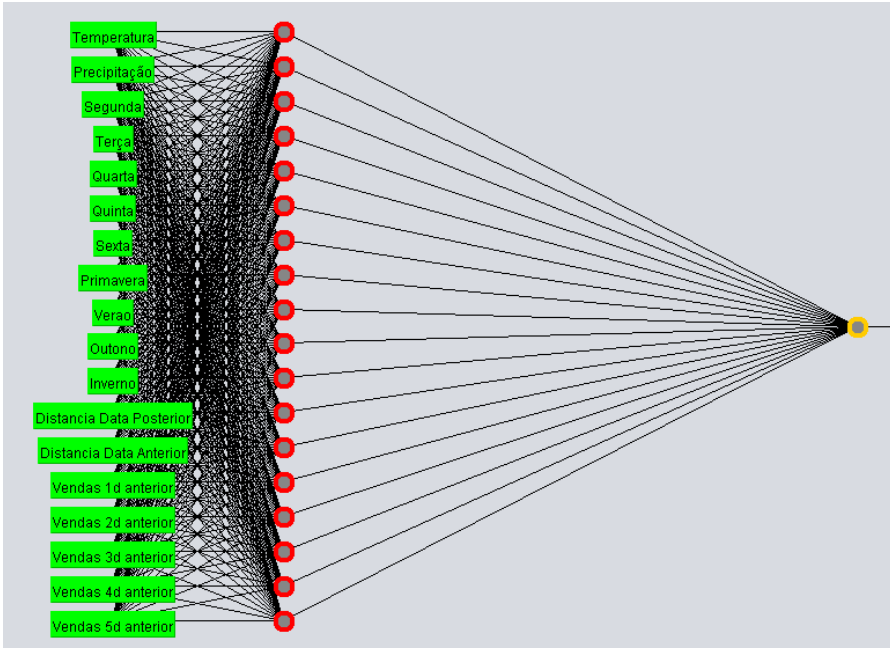
TREINO 2017: 12/04/2017 à 19/12/2017 – 148 REGISTROS

VALIDAÇÃO (2017) código abaixo

```
>>> from sklearn.model_selection import train_test_split
>>> standardized_X, standardized_X_validation, y_train , y_validation =
train_test_split(x,y,test_size=0.33,random_state=42)
```

TESTE 2018: 26/02/2018 à 31/10/2018 – 150 REGISTROS

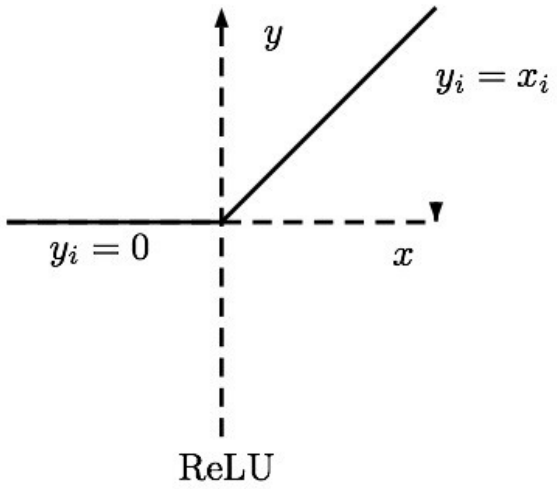
Treino com o primeiro modelo proposto:




```
from keras import Sequential
from keras.layers import Dense
model = Sequential()
model.add(Dense(standardized_X.s
hape[1],activation='relu',input_dim=
standardized_X.shape[1]))
model.add(Dense(1))
model.compile(optimizer='rmsprop',
loss='mse',
metrics=['mae'])
```

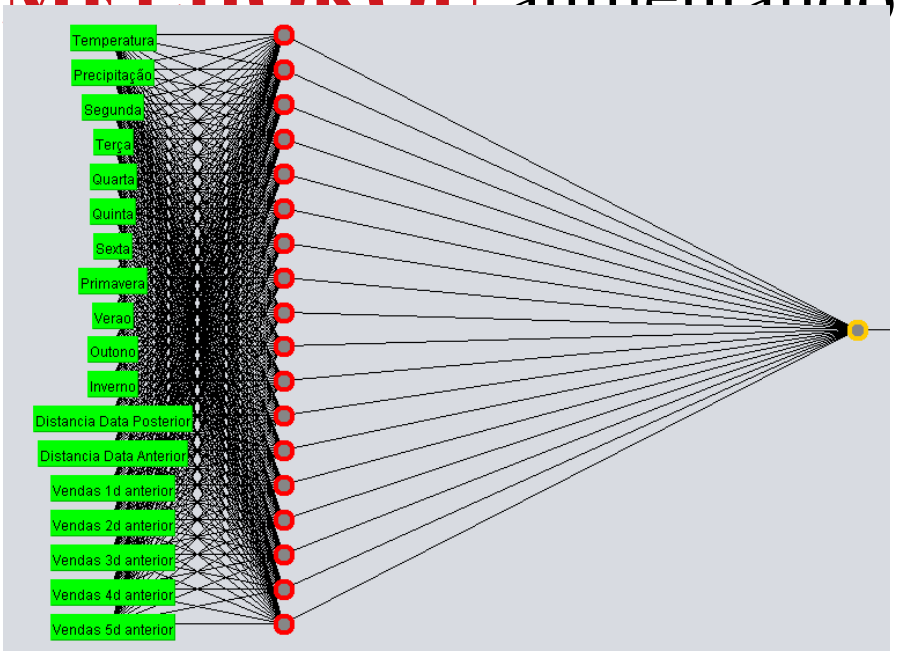
batch_size=36, **epochs=500**,

148/148
[==============================]
- 0s 81us/step -
loss: 14931.1859 -
mean_absolute_error: 73.5863 -
val_loss: 24922.2952 -
val_mean_absolute_error: **118.9186**



Treino com o primeiro modelo proposto:

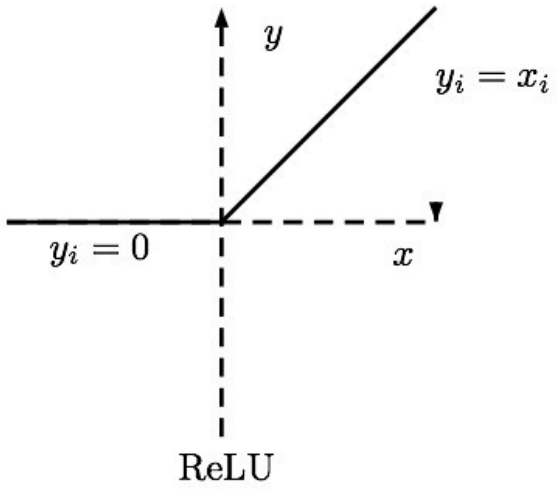
MELHOROU aumentando épocas de 500 pra 1000




```
from keras import Sequential
from keras.layers import Dense
model = Sequential()
model.add(Dense(standardized_X.s
hape[1],activation='relu',input_dim=
standardized_X.shape[1]))
model.add(Dense(1))
model.compile(optimizer='rmsprop',
loss='mse',
metrics=['mae'])
```

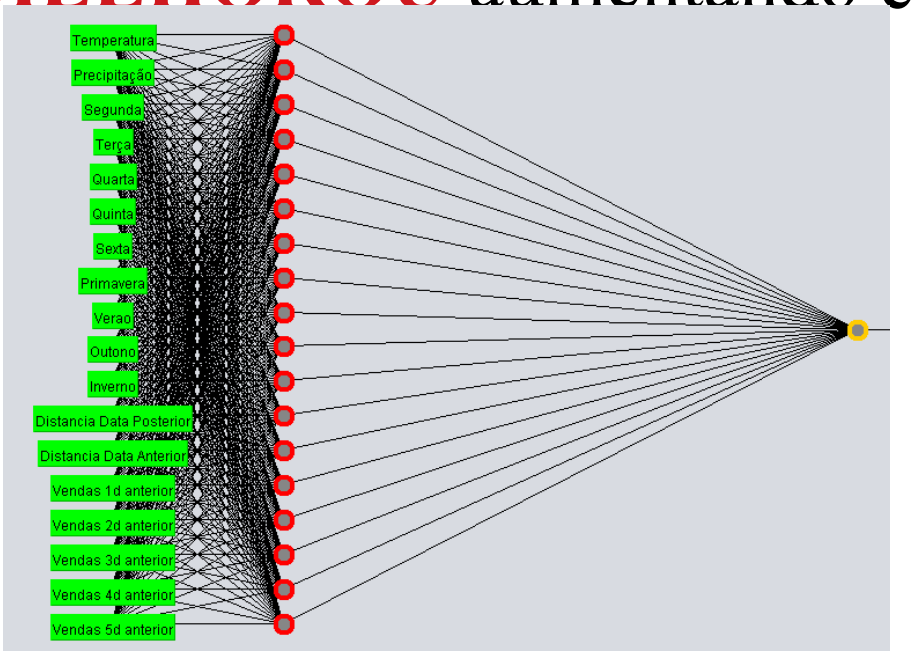
batch_size=36, **epochs=1000,**

148/148
[==============================]
- 0s 81us/step -
loss: 7397.2767 -
mean_absolute_error: 60.5002 -
val_loss: 11375.0160 -
val_mean_absolute_error: **72.5417**



Treino com o primeiro modelo proposto:

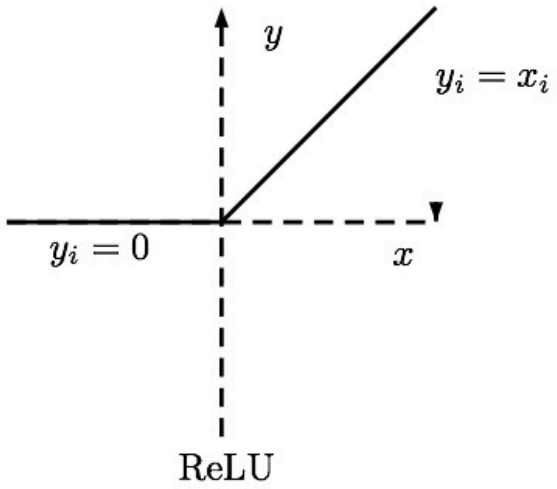
MELHOROU aumentando épocas de 1000 pra 250



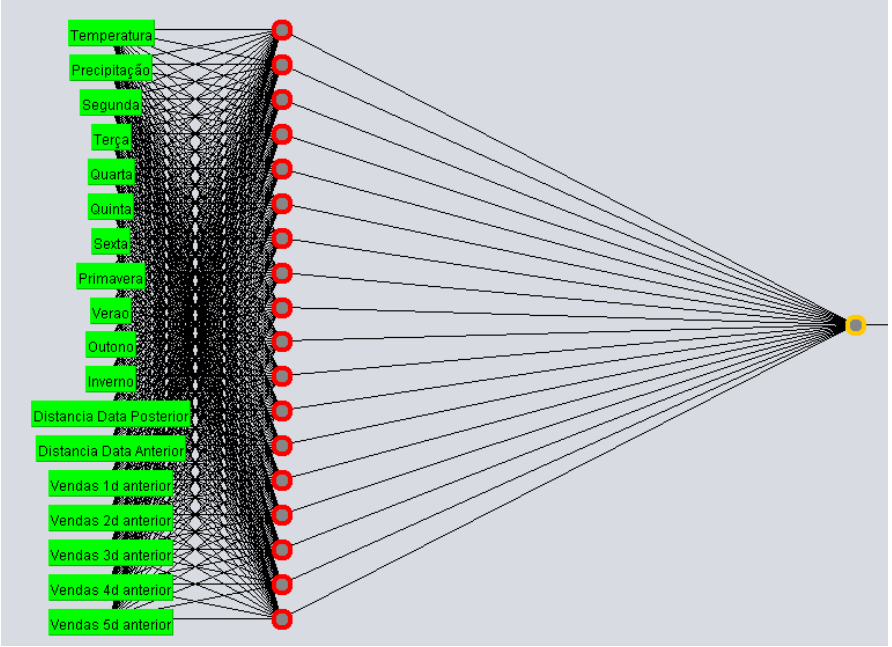

```
from keras import Sequential
from keras.layers import Dense
model = Sequential()
model.add(Dense(standardized_X.s
hape[1],activation='relu',input_dim=
standardized_X.shape[1]))
model.add(Dense(1))
model.compile(optimizer='rmsprop',
loss='mse',
metrics=['mae'])
```

batch_size=36, **epochs=250**,

148/148
[==============================]
- 0s 81us/step -
loss: 6943.7007 -
mean_absolute_error: 59.2704 -
val_loss: 10794.9711 -
val_mean_absolute_error: **69.8469**



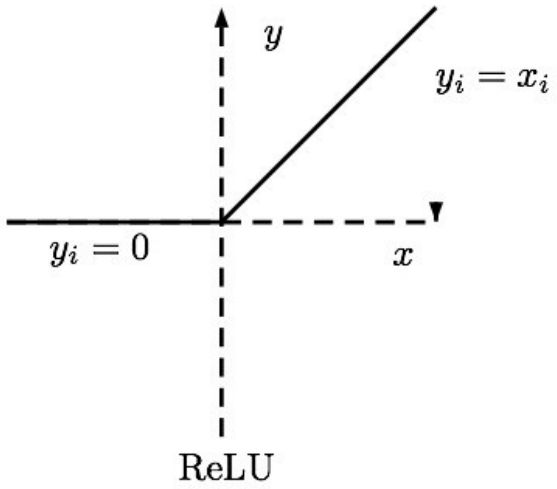
Treino com o primeiro modelo proposto: **MELHOROU** aumentando épocas de 250 pra 125



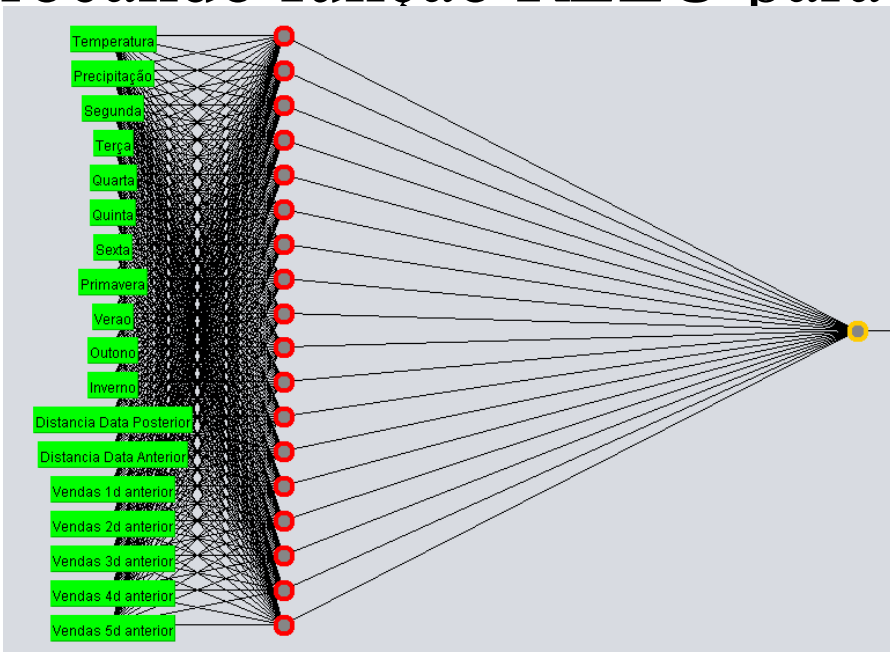

```
from keras import Sequential
from keras.layers import Dense
model = Sequential()
model.add(Dense(standardized_X.s
hape[1],activation='relu',input_dim=
standardized_X.shape[1]))
model.add(Dense(1))
model.compile(optimizer='rmsprop',
loss='mse',
metrics=['mae'])
```

batch_size=36, **epochs=125,**

148/148
[==============================]
0s 81us/step -
loss: 6454.3501 -
mean_absolute_error: 57.6243 -
val_loss: 10239.3892 -
val_mean_absolute_error: **67.4350**



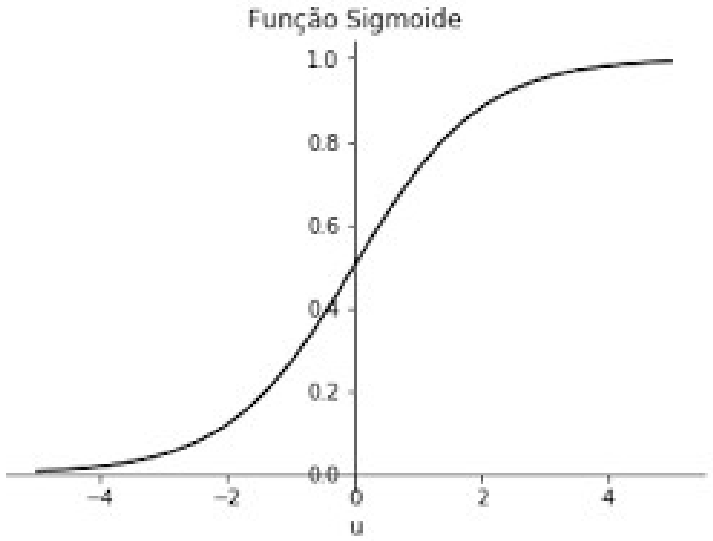
Treino com o primeiro modelo proposto: **PIOROU** trocando função RELU para SIGMOIDE



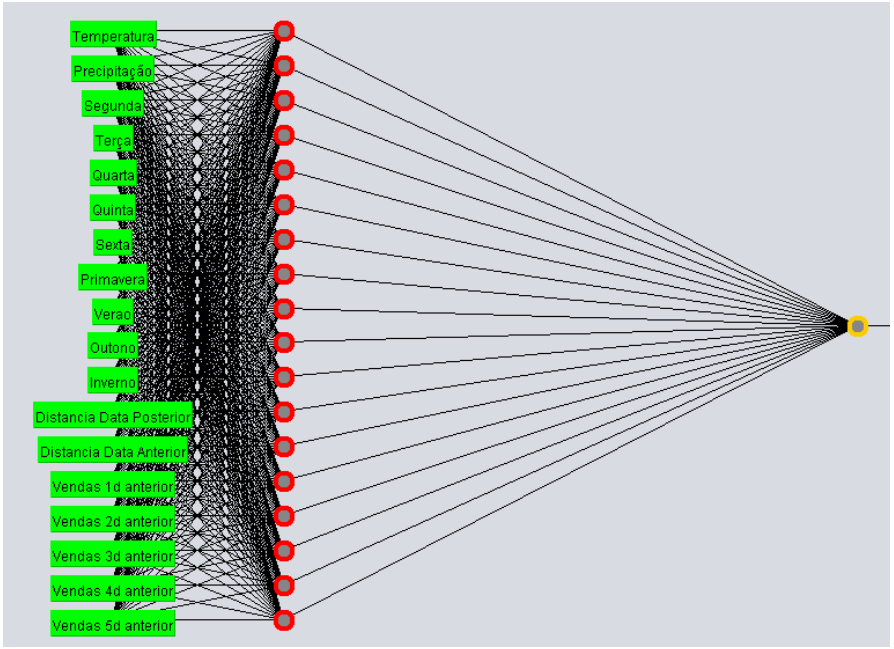

```
model = Sequential()
model.add(Dense(standardized_X.s
hape[1],activation='sigmoid',input_
dim=standardized_X.shape[1]))
model.add(Dense(1))
model.compile(optimizer='rmsprop',
loss='mse',
metrics=['mae'])
```

batch_size=36, **epochs=125,**

148/148
[==============================]
0s 108us/step
- loss: 90516.9781 -
mean_absolute_error: 258.0064 -
val_loss: 151921.6487 -
val_mean_absolute_error: **368.2700**



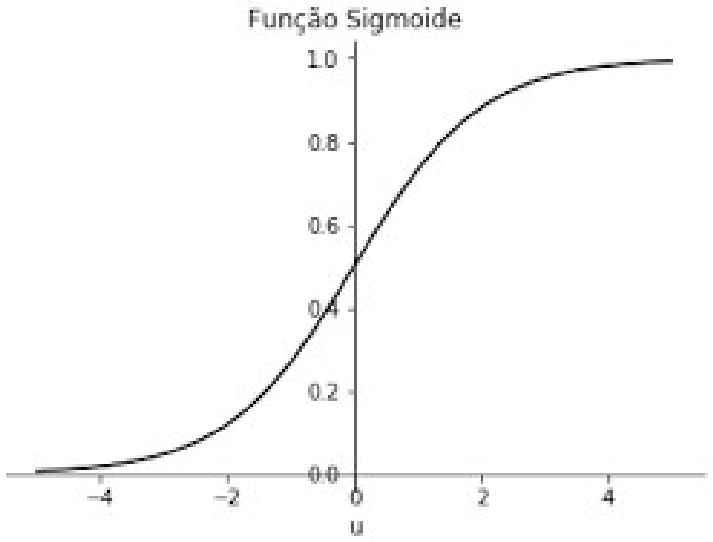
Treino com o primeiro modelo proposto: **MELHOROU** aumentando número de épocas



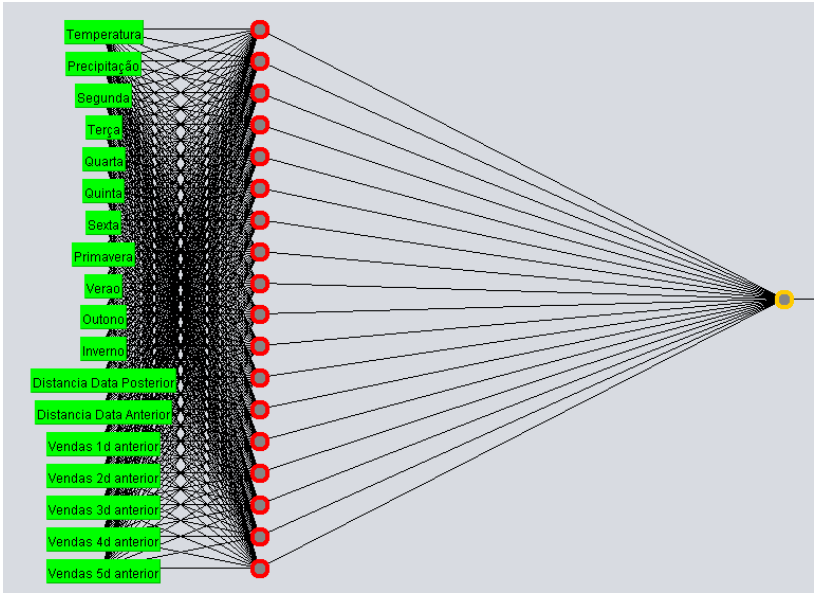

```
model = Sequential()
model.add(Dense(standardized_X.s
hape[1],activation='sigmoid',input_
dim=standardized_X.shape[1]))
model.add(Dense(1))
model.compile(optimizer='rmsprop',
loss='mse',
metrics=['mae'])
```

batch_size=36, **epochs=5000,**

148/148
[==============================]
0s 54us/step -
loss: 6390.4238 -
mean_absolute_error: 58.5402 -
val_loss: 14054.7375 -
val_mean_absolute_error: **99.0611**



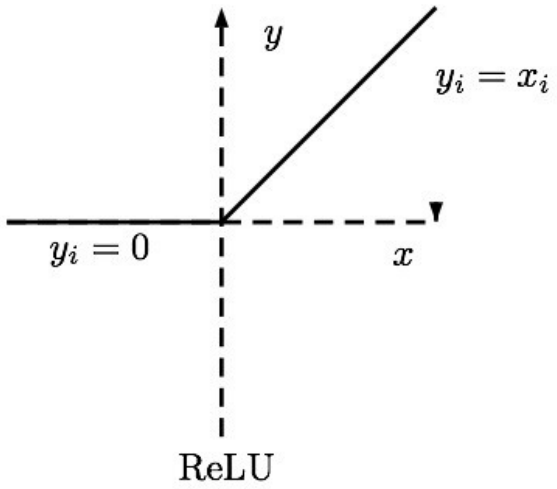
Treino com o primeiro modelo proposto: **MELHOROU** voltando pra RELU e usando otimizador ADAM.



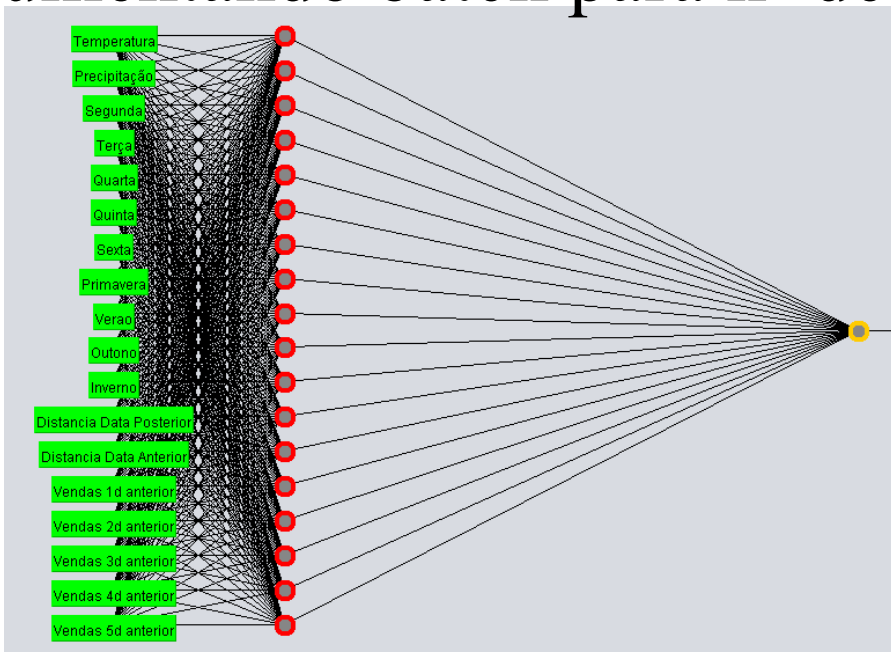

```
from keras import Sequential
from keras.layers import Dense
model = Sequential()
model.add(Dense(standardized_X.s
hape[1],activation='relu',input_dim=
standardized_X.shape[1]))
model.add(Dense(1))
model.compile(optimizer='adam',
loss='mse',
metrics=['mae'])
```

batch_size=36, **epochs=5000,**

148/148
[==============================]
0s 54us/step -
loss: 6019.6716 -
mean_absolute_error: 52.0574 -
val_loss: 10161.9818 -
val_mean_absolute_error: **65.4172**



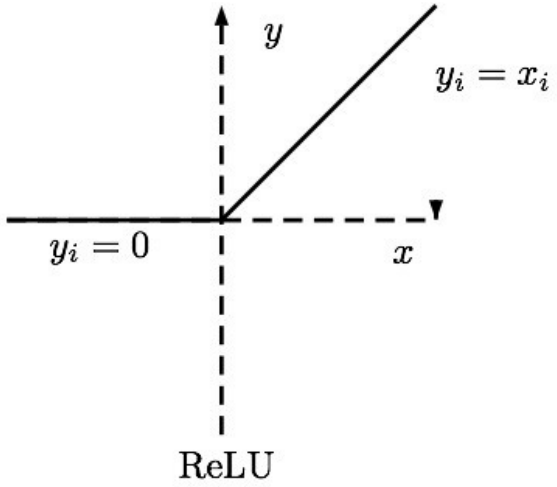
Treino com o primeiro modelo proposto: MELHOROU aumentando batch para nº de registros




```
from keras import Sequential
from keras.layers import Dense
model = Sequential()
model.add(Dense(standardized_X.s
hape[1],activation='relu',input_dim=
standardized_X.shape[1]))
model.add(Dense(1))
model.compile(optimizer='adam',
loss='mse',
metrics=['mae'])
```

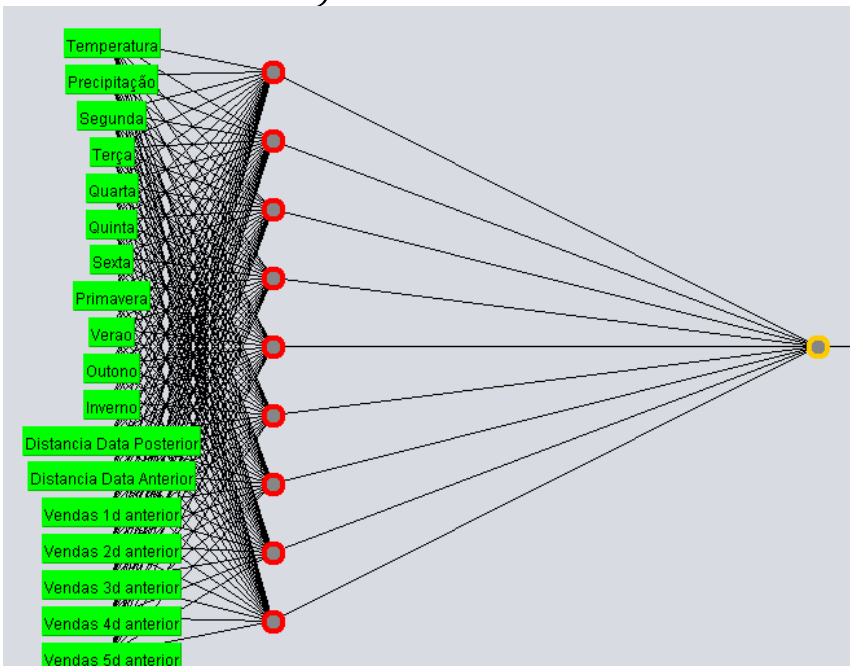
batch_size=148, epochs=5000,

148/148
[==============================]
0s 27us/step -
loss: 3658.2805 -
mean_absolute_error: 42.7853 -
val_loss: 8063.6133 -
val_mean_absolute_error: **60.9182**



Treino com o segundo modelo proposto: 27/38

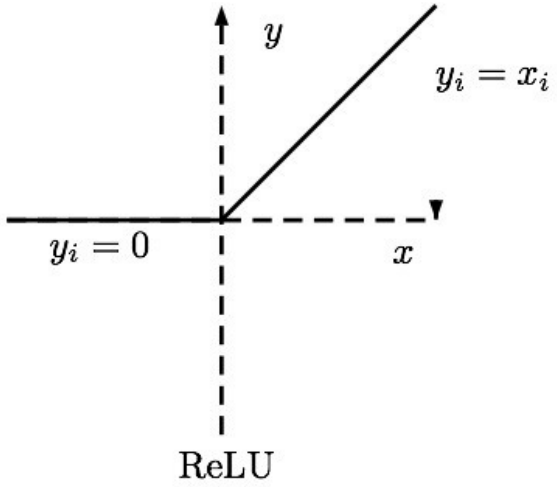
PIOROU, DIMINUINDO Nº DE NEURÔNIOS



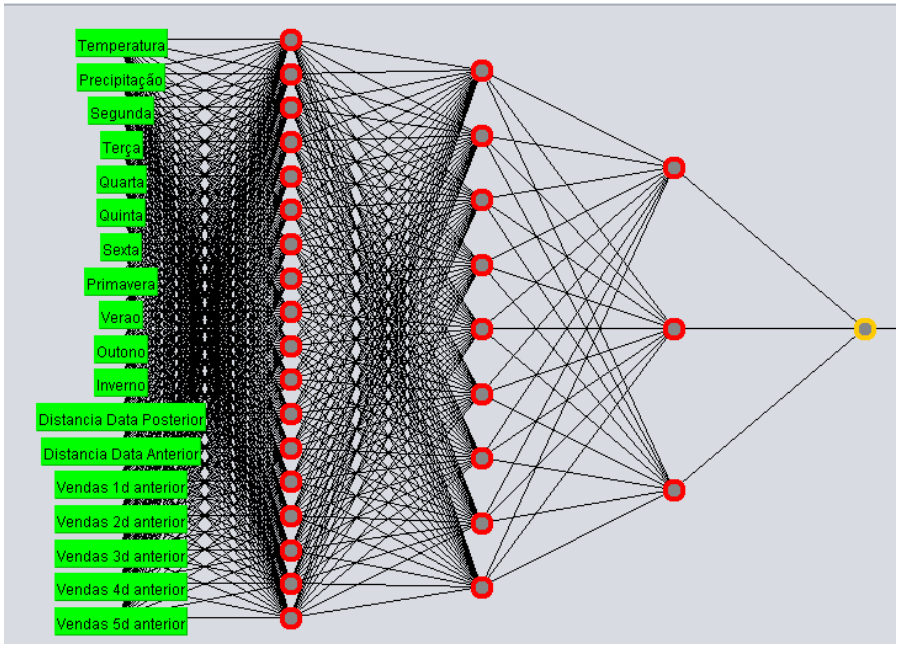

```
model1 = Sequential()
model1.add(Dense(9,activation='relu'
,input_dim=standardized_X.shape[1])
)
model1.add(Dense(1))
model1.compile(optimizer='adam',
loss='mse',
metrics=['mae'])
```

batch_size=148, epochs=5000,

148/148
[==============================]
0s 54us/step -
loss: 10035.1689 -
mean_absolute_error: 60.3270 -
val_loss: 17238.3887 -
val_mean_absolute_error: **95.9984**



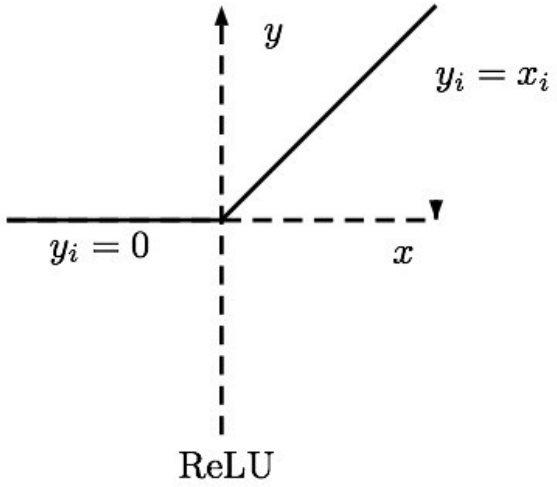
Treino com o terceiro modelo proposto: **MELHOROU** aumentando neurônios / camadas



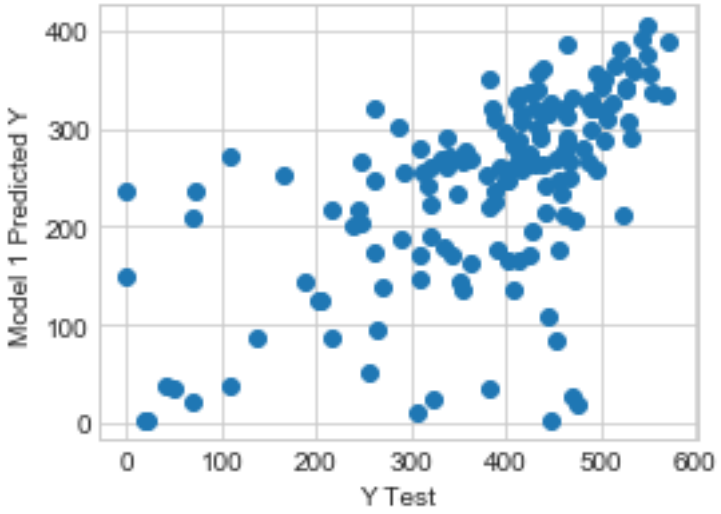

```
model3 = Sequential()
model3.add(Dense(standardized_X.s
hape[1],activation='relu',input_dim=st
andardized_X.shape[1]))
model3.add(Dense(9,activation='relu',
input_dim=standardized_X.shape[1]))
model3.add(Dense(3,activation='relu',
input_dim=9))
model3.add(Dense(1))
model3.compile(optimizer='adam',
loss='mse', metrics=['mae'])
```

batch_size=148, epochs=5000,

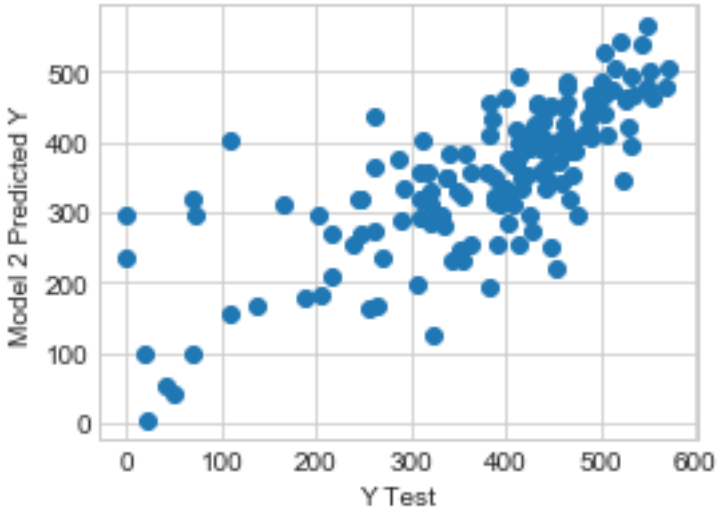
148/148
[==============================]
0s 20us/step -
loss: 3721.4502 -
mean_absolute_error: 43.0977 -
val_loss: 7648.9710 -
val_mean_absolute_error: **59.6000**



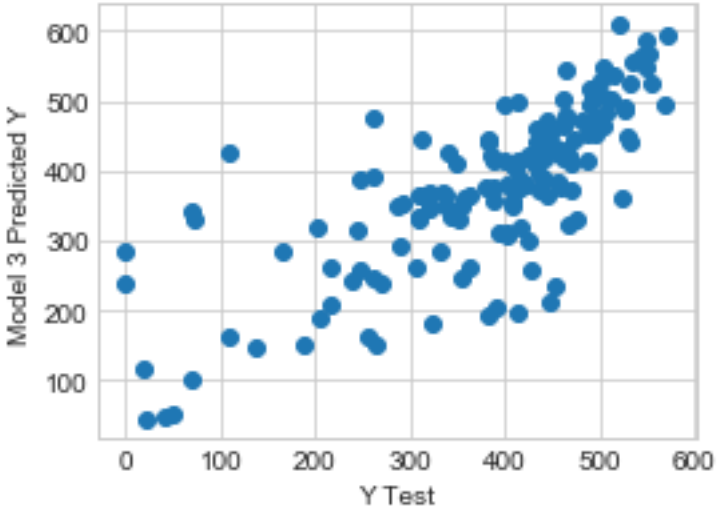
MODELO 1: 18,1



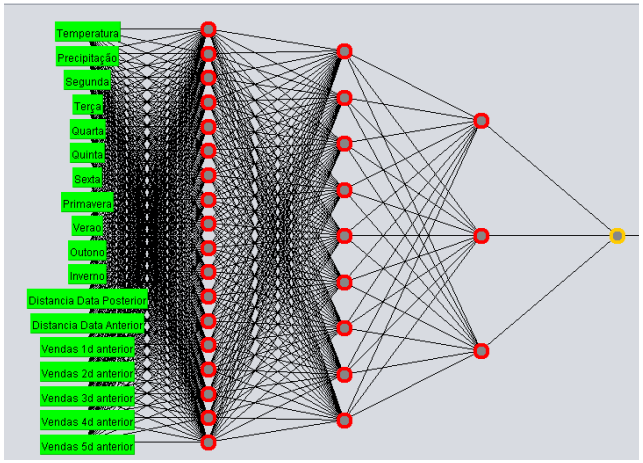
MODELO 2: 9,1



MODELO 3: 18,9,3,1

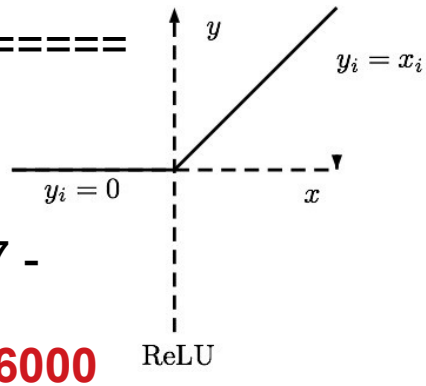


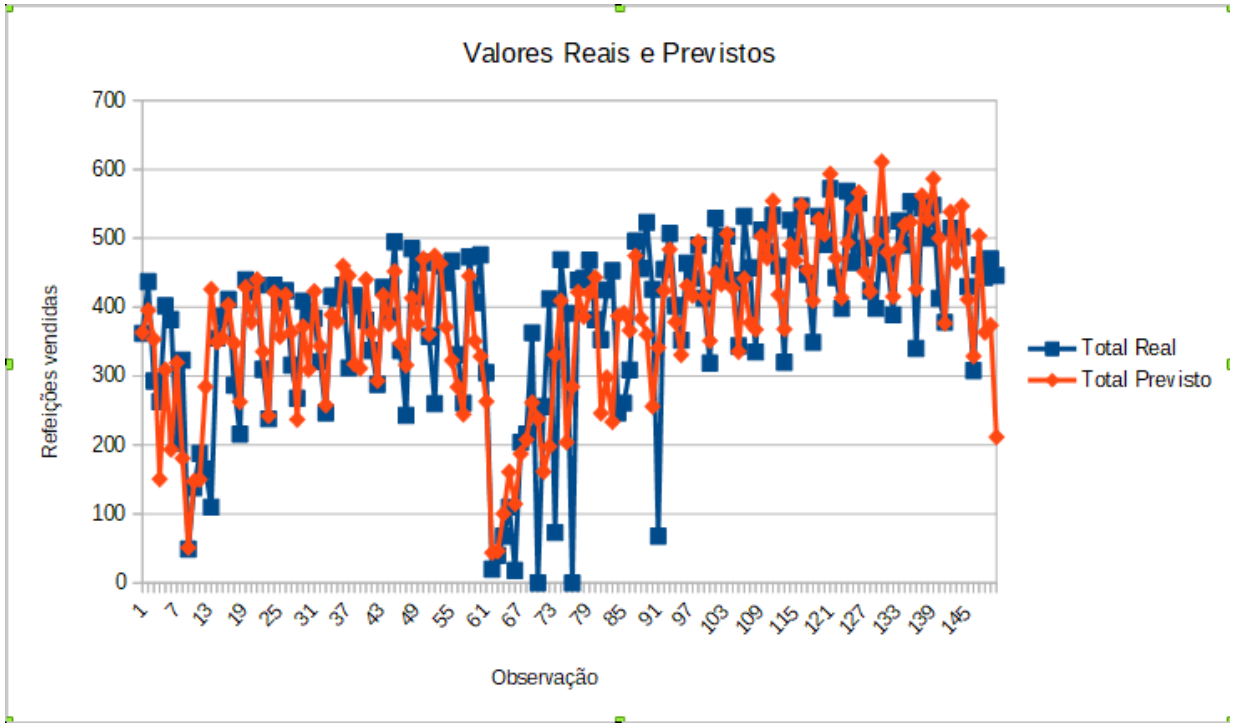
ESCOLHA DO 3º MODELO

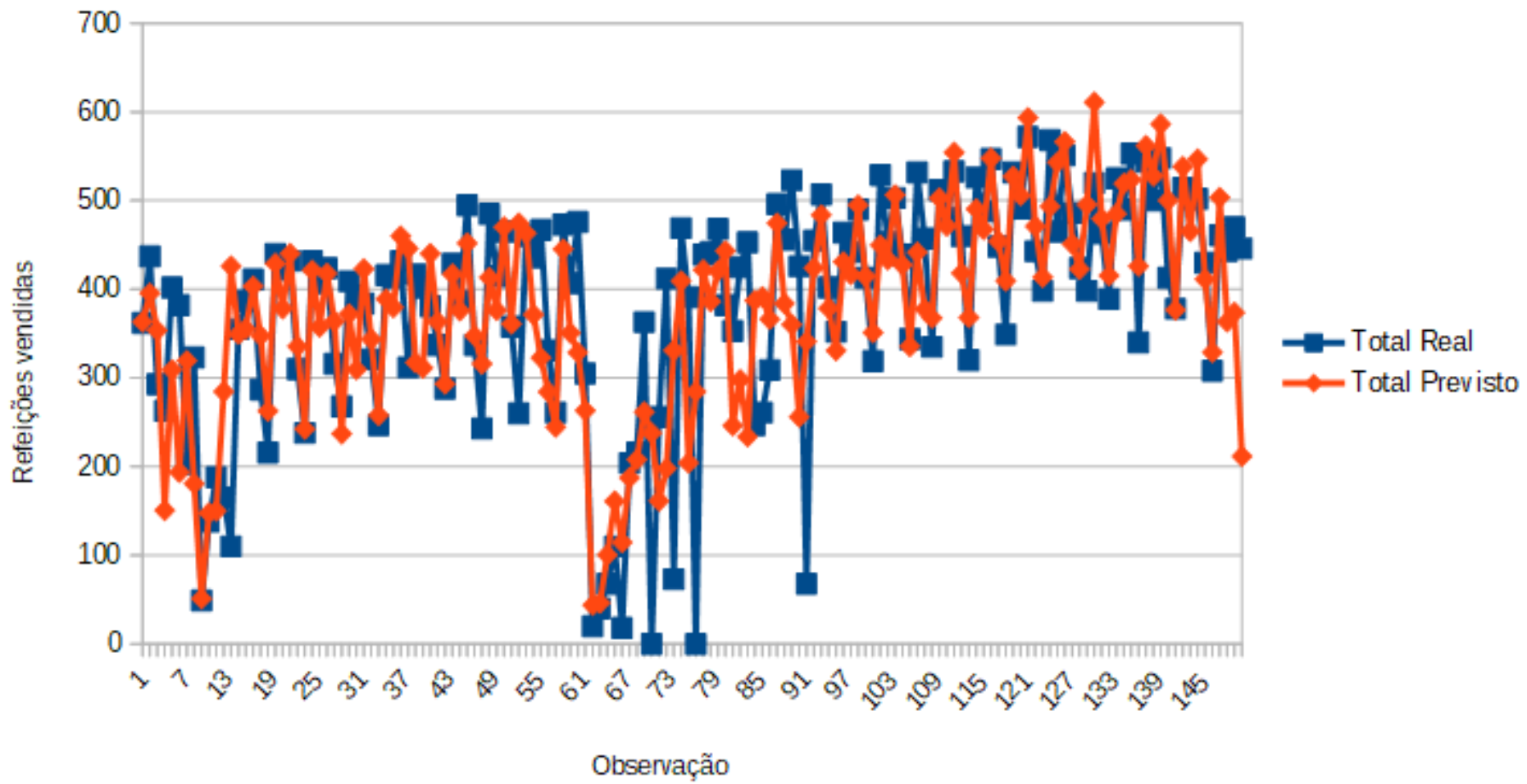


148/148
[==============================
===]
0s 20us/step -
loss: 3721.4502 -
mean_absolute_error: 43.0977 -
val_loss: 7648.9710 -
val_mean_absolute_error: **59.6000**

batch_size=148, epochs=5000,






Valores Reais e Previstos
Refeições vendidas
Observação
Total Real
Total Previsto
0
100
200
300
400
500
600
700
1
7
13
19
25
31
37
43
49
55
61
67
73
79
85
91
97
103
109
115
121
127
133
139
145

DIFICULDADES:

GRANDE DIFICULDADE EM MONTAR A BASE DE DADOS:
O desafio está em procurar as pessoas e fontes certas!
- Vendas: Gerente do R.U (que tentei obter os dados por meses) vs T.I
- Climáticos: (climatempo sem exportação) vs (INPE – BNDE)
- Cruzar as informações da forma certa (Montei na mão no excell)

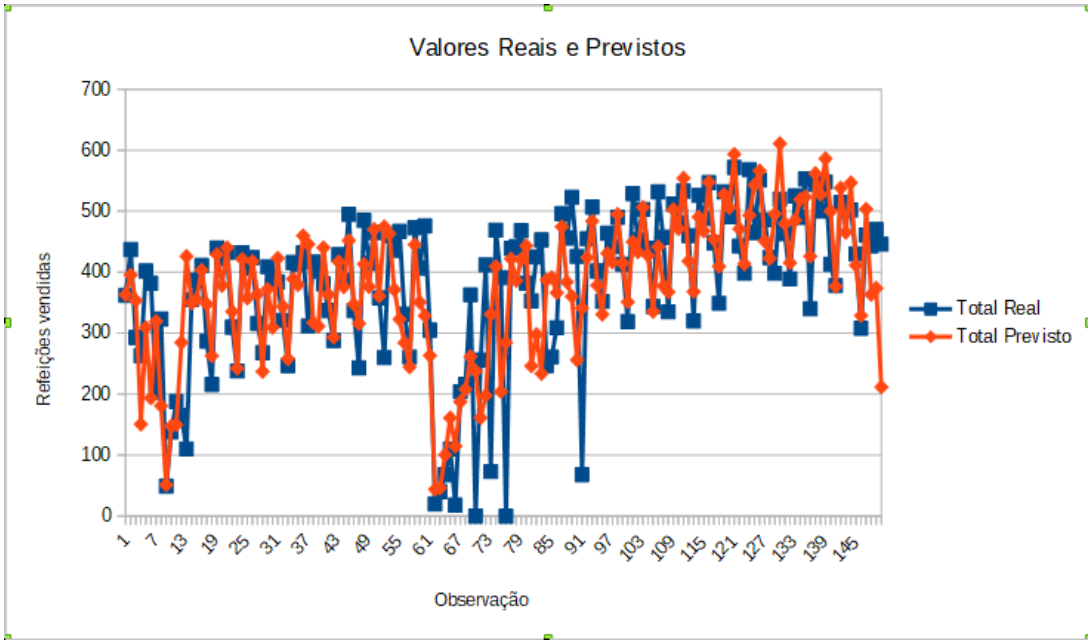
NORMALIZAÇÃO DOS DADOS:
DEU PAU:

```
from sklearn.preprocessing import  MinMaxScaler
sc= MinMaxScaler(feature_range=(0,1))
X = X.reshape(-1,1)
X = sc.fit_transform(X)
y = y.reshape(-1,1)
```

VS
DEU CERTO

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler().fit(X_train)
standardized_X = scaler.transform(X_train)
standardized_X_test = scaler.transform(X_test)
```

CONCLUSÕES:



- O modelo deve ser treinado por mais dados (1 ano só ainda é pouco). Existe uma base para ser tratada ainda e explorada que foi exportada em dump.sql da talim, do sistema antigo 2008-2016.

- 59,6% de erro ainda é o mesmo erro que análise exploratória atual

- Otimizador ADAM se saiu melhor que RMSPROP

- Função RELU se saiu melhor que sigmóide

- AUMENTO da profundidade e neurônios da rede foi melhor

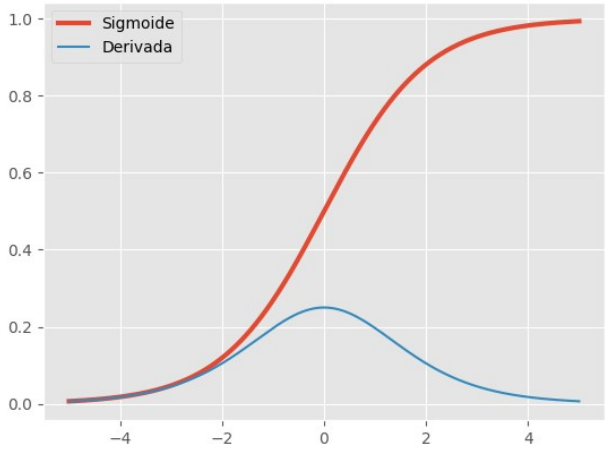
- Modelo final acompanhou tendencia linear com os dados de teste. INCLUSIVE TROCA DE SEMESTRE.

CONCLUSÕES - ATIVAÇÃO:

http://www.deeplearningbook.org/contents/mlp.html

Ativação Sigmoide (Logística)

Até pouco tempo atrás, a função sigmoides era a mais utilizada em RNAs, por serem biologicamente mais plausível. Como neurônios biológicos funcionam de foma binária (ativando vs não ativando), a função sigmoide é uma boa forma de modelar esse comportamento, já que assume valores apenas entre 0 (não ativação) e 1 (ativação). No entanto, se olharmos sua derivada, podemos ver que ela satura para valores acima de 5 e abaixo de -5. Com essas derivadas tendendo a zero, a propagação do gradiente desvanece nessas regiões, causando dificuldades no treinamento.



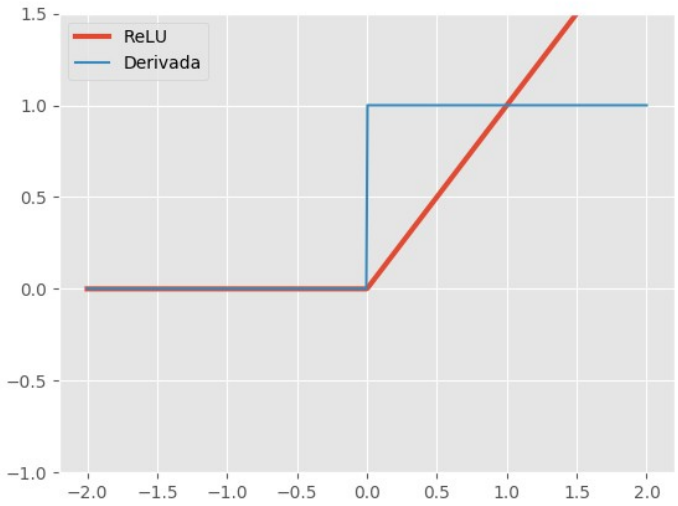
Mais ainda, repare que a derivada da função sigmoide é sempre <1. Isso é problemático, fazendo com que desvaneça o produto dado pela regra da cadeia na propagação dos gradientes. **Assim, não é mais recomendado utilizar a função logística como não linearidade de ativação nas redes neurais artificiais.**

CONCLUSÕES - ATIVAÇÃO:

http://www.deeplearningbook.org/contents/mlp.html

Ativação ReLU

Redes com a função ReLU são fáceis de otimizar, já que a ReLU é extremamente parecida com a função identidade. A única diferença é que a ReLU produz zero em metade do seu domínio. Como consequência, as derivadas se mantêm grandes enquanto a unidade estiver ativa.



Teoricamente, a derivada não está definida em 0, mas podemos implementá-la como sendo 0 ou 1 sem maiores preocupações. Note que as derivadas não são apenas grandes, mas também estáveis, sendo 1, quando x>0 e 0 quando x<0. Note também que a segunda derivada é zero em todo o domínio. A ativação ReLU é muito mais eficiente do que as funções sigmoidais vistas acima e é uma das descobertas que contribuiu de forma significante para a recente popularidade de Deep Learning. **Essa não linearidade é um ótimo exemplo de como a simplicidade pode ser extremamente poderosa.**

CONCLUSÕES - OTIMIZADOR:

https://blog.paperspace.com/intro-to-optimization-momentum-rmsprop-adam/

RMSProp

RMSprop ou Root Mean Square Propogation tem uma história interessante. Foi inventado pelo lendário Geoffrey Hinton, enquanto sugeria uma ideia aleatória durante uma aula do Coursera.

O RMSProp também tenta atenuar as oscilações, mas de um modo diferente do momentum. O suporte do RMS também elimina a necessidade de ajustar a taxa de aprendizado e o faz automaticamente. Mais ainda, o RMSProp escolhe uma taxa de aprendizado diferente para cada parâmetro.

No RMS prop, cada atualização é feita de acordo com as equações descritas abaixo. Esta atualização é feita separadamente para cada parâmetro.

$$\nu_t = \rho\nu_{t-1} + (1-\rho) * g_t^2$$

$$\Delta\omega_t = -\frac{\eta}{\sqrt{\nu_t + \epsilon}} * g_t$$

$$\omega_{t+1} = \omega_t + \Delta\omega_t$$

CONCLUSÕES - OTIMIZADOR:

https://blog.paperspace.com/intro-to-optimization-momentum-rmsprop-adam/

ADAM

Enquanto o momentum acelera nossa busca na direção dos mínimos, o RMSProp impede nossa busca na direção das oscilações.

Os algoritmos Adam ou Adaptive Moment Optimization combinam as heurísticas de Momentum e RMSProp. Aqui estão as equações de atualização.

$$For\ each\ Parameter\ w^j$$

$$(j\ subscript\ dropped\ for\ clarity)$$

$$\nu_t = \beta_1 * \nu_{t-1} - (1 - \beta_1) * g_t$$

$$s_t = \beta_2 * s_{t-1} - (1 - \beta_2) * g_t^2$$

$$\Delta\omega_t = -\eta \frac{\nu_t}{\sqrt{s_t + \epsilon}} * g_t$$

$$\omega_{t+1} = \omega_t + \Delta\omega_t$$

OBRIGADO!

https://github.com/ddlandim/monografy-ann-demand-prediction