Análise de um *dataset* através de *Deep Learning*

André Pereira (pg38923)
Carlos Lemos (pg38410)
João Barreira (a73831)
Rafael Costa (a61799)

Maio 2019

Universidade do Minho
Escola de Engenharia

Trabalho Prático 3
Sistemas Inteligentes – Sistemas Autónomos
MEI / MIEI – Universidade do Minho

1 Introdução

O presente trabalho prático, proposto pela Unidade Curricular de Classig, tem como principal objetivo a exploração de uma arquitetura de redes neuronais recorrentes chamada *Long short-term memory* (*LSTM*), pertencente ao ramo de *deep learning*. Este tipo de estrutura tem sido cada vez mais utilizado em problemas de classificação e de previsão, assim como reconhecimento de escrita e de voz, operando sobretudo, sobre dados que referem a uma determinada sequência temporal. Como caso de estudo, propôs-se a construção de uma *LSTM* que consiga prever as vendas de um determinado produto dietético a partir de um *dataset* de pequenas dimensões.

No relatório deste trabalho prático descreve-se a análise efetuada ao *dataset* recebido e seu respetivo pré-processamento. Posteriormente, são detalhados os inúmeros testes e configurações efetuadas à *LSTM* desenvolvida, tendo-se avaliado a eficácia da mesma a partir da métrica *loss* dos dados de teste face aos dados de treino, identificando-se situações de *overfitting* e *underfitting* do modelo produzido.

2 Análise do *dataset*

O *dataset* sobre o qual incide a realização deste trabalho corresponde a um conjunto de dados referentes às vendas e custos de publicidade referentes à comercialização de um produto dietético de controlo de peso, ao longo de 3 anos (36 meses) consecutivos.

Por forma a podermos compreender melhor os valores do *dataset* com os quais iríamos trabalhar, começámos por gerar uma tabela com informação sobre os mesmos para um ficheiro (*dataset_description.txt*), bem como os respetivos gráficos de distribuição (*dist-advertising.png* e *dist-sales.png*), como indicam as **Figuras 1**, **2** e **3**.

	Advertising	Sales
count	36	36
mean	24.25	28.53
std	6.18	18.78
min	12	1
25%	20.3	15.75
50%	24.25	23
75%	28.6	41
max	36.5	65

Figura 1: Descrição das *features* do *dataset*

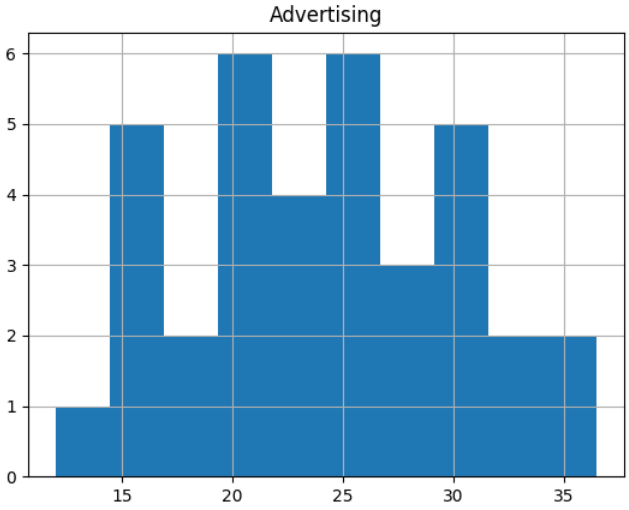

Figura 2: Distribuição dos valores da coluna *Advertising*

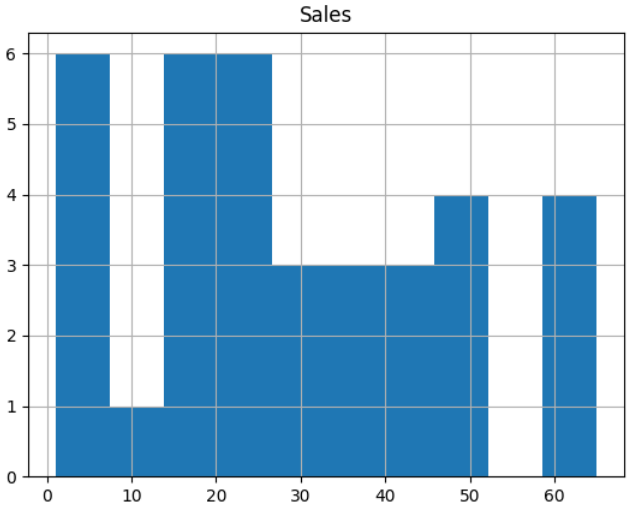

Figura 3: Distribuição dos valores da coluna *Sales*

3 Pré-processamento

Como foi dito na secção anterior, o *dataset* continha apenas 3 atributos e 36 linhas de dados. O primeiro desses atributos correspondia ao mês e ao ano referentes a uma venda que continha o seguinte formato: "ano-mês". Devido a esse facto, fez-se um *split* pelo caráter "-" e retirou-se a informação relativa ao ano desse atributo, restando apenas informação relativa ao nome do mês. Através deste resultado atribuiu-se um valor numérico entre 1 a 12 a cada mês, ou seja, efetuou-se a técnica de *label encoding*.

Face ao facto de o *dataset* ser bastante pequeno e com poucos atributos, tentou-se enriquecer este *dataset* através de *data augmentation*. Para isso criou-se um atributo derivado relativo às estações do ano a que um mês pertence, ou seja, *Spring*, *Summer*, *Autumn* e *Winter*. Considerou-se que este novo atributo deu qualidade ao *dataset*, já que a venda de produtos dietéticos varia de acordo com as diferentes estações do ano. A outra técnica de *data augmentation* usada consistiu em usar *one hot encoding* aos atributos referentes ao mês e à estação do ano. Esta técnica foi conseguida com recurso à função *get_dummies* em *Python*.

Finalmente, avaliou-se a possibilidade de se utilizar normalização ou estandardização aos atributos *Advertising* e *Sales*. No entanto, verificou-se que estes atributos apresentavam valores na mesma escala e que não continham *outliers*, pelo que se considerou não ser eficaz o uso de tais técnicas.

4 Testes

De seguida, seguiu-se a fase de testes cujo objetivo seria obter um modelo otimizado para o problema em questão, com os parâmetros mais adequados.

Assim sendo, começámos por testar as camadas a utilizar no modelo e o número de neurónios, seguido de testes de *dropout*, funções de ativação, *initializers*, *optimizers*, tamanho da janela deslizante e *batch size*.

Para isso – e tendo por base o exercício realizado na aula prática –, resolvemos escolher um número inicial de *epochs* que achámos adequado (500) e o *batch size* por defeito (32). Em relação à janela, optámos por escolher o valor 3 por forma a que a previsão de um valor para um determinado mês tivesse em consideração os resultados do trimestre anterior. Relativamente a este parâmetro não quisemos optar por um valor demasiado elevado pois perderíamos obrigatoriamente uma grande quantidade de informação no processo da passagem da janela deslizante pelo *dataset* que, por si, já é pequeno. Isto porque para uma janela de tamanho N, os primeiros N valores do *dataset* não iriam ser utilizados como *label* durante este processo. Também face à pequena dimensão do *dataset*, foram utilizados os dados de teste para efeitos de validação.

Como resultado deste processo, apresentamos nas subsecções seguintes um resumo dos valores obtidos em cada teste (e restantes parâmetros) sob a forma de várias tabelas. Os valores dos resultados correspondem ao valor do *RMSE* (*Root Mean Square Error*), pelo que como não foi feita a normalização dos dados da coluna *Sales*, é imediatamente percetível a dimensão do erro no

contexto da escala dos valores reais presentes no *dataset* (que variam entre 1 e 65).

Todos os gráficos gerados durante e após a execução do código correspondente encontram-se no final deste relatório em anexo.

4.1 Camadas e número de neurónios

Por forma a testar qual o número de camadas e respetivos neurónios que melhor se adequariam face ao problema em questão, começámos por utilizar um modelo bastante simples composto por quatro camadas: duas *LSTM* com 8 e 4 neurónios e duas *Dense* compostas por 4 e 1 neurónios.

Assim, a **Figura 4** apresenta a tabela dos resultados desta etapa:

Train	Test	Epochs	Batch size	Janela	Camadas do modelo				
35	22	500	32	3	LSTM(8)	LSTM(4)	Dense(4)	Dense(1)	
21	8	500	32	3	LSTM(16)	LSTM(8)	Dense(8)	Dense(1)	
18	10	500	32	3	LSTM(32)	LSTM(16)	Dense(16)	Dense(1)	
11.51	17.38	500	32	3	LSTM(64)	LSTM(32)	Dense(16)	Dense(1)	
19	12	500	32	3	LSTM(64)	LSTM(32)	Dense(8)	Dense(1)	
4	25	500	32	3	LSTM(64)	LSTM(32)	Dense(32)	Dense(1)	
19	11	500	32	3	LSTM(64)	LSTM(32)	LSTM(16)	Dense(16)	Dense(1)
20.28	10.69	500	32	3	LSTM(64)	LSTM(32)	LSTM(8)	Dense(16)	Dense(1)
13.38	13.3	415	32	3	LSTM(64)	LSTM(32)	Dense(16)	Dense(1)	

Figura 4: Resultados dos testes das camadas e número de neurónios

Nos primeiros três testes (**Figuras 15**, **16** e **17**, em anexo), o modelo encontrava-se numa situação clara de *underfitting* em que lhe faltava a complexidade necessária para captar a totalidade da informação correspondente ao problema em estudo. Esta situação pode ser identificada pelo facto de o erro ser demasiado grande tanto na fase de treino como na de validação. Neste ponto poderíamos ter aumentado substancialmente o número de *epochs* por forma a tentar perceber se com um maior "tempo" de treino, o modelo conseguiria estudar melhor o *dataset*, o que corresponderia a menores valores para o *RMSE* do treino e validação e consequente "abatimento" de ambas as curvas. No entanto, optámos por manter o número de *epochs*, adicionando complexidade ao modelo.

Assim sendo, com o aumento da complexidade do modelo no quarto teste, foi possível obter valores bastante melhores para o *RMSE* de treino e validação. No entanto, como é possível observar pelo gráfico presente na **Figura 18**, ao fim de X *epochs* o modelo começava a entrar numa situação de *overfitting*, em que o erro do treino é bastante menor do que o da validação – o modelo tinha sido treinado demais. Assim sendo, optou-se por reduzir o "tempo" de treino através da diminuição de *epochs* para 415, procurando alcançar o instante imediatamente anterior ao início da observação da situação de *overfitting*. Com isto conseguiu-se obter os melhores valores de *RMSE* para esta fase, presentes na última linha da tabela (nono teste) e nas **Figuras 23** e **24** em anexo. No gráfico da fase de testes presente nesta última figura, podemos ver que a reta

das previsões começa a acompanhar ligeiramente o modelo, principalmente a última descida e os mínimos locais existentes.

Com o aumento da complexidade nos quinto, sexto, sétimo e oitavo testes (**Figuras 19**, **20**, **21** e **22**, em anexo) não foram obtidos valores tão satisfatórios como no teste número 4, pelo que os seus modelos não foram utilizados daí em diante.

4.2 *Dropout*

Como foi dito na subsecção anterior, conseguiu-se sair da situação de *underfitting* através da adicionação de complexidade ao modelo, mas entrou-se agora numa situação de *overfitting*, colmatada no nono teste dessa subsecção com a limitação do número de *epochs*.

No entanto, foi também testada a alternativa de adicionar *dropout* por forma a tentar reduzir a disparidade entre os resultados de treino e validação, cuja tabela se encontra na **Figura 5**.

Train	Test	Epochs	Batch size	Janela	Camadas do modelo					
11.26	17.99	500	32	3	LSTM(64)	Dropout(0.2)	LSTM(32)	Dense(16)	Dense(1)	
12.3	15.08	400	32	3	LSTM(64)	Dropout(0.2)	LSTM(32)	Dense(16)	Dense(1)	
11.43	18.19	500	32	3	LSTM(64)	Dropout(0.2)	LSTM(32)	Dropout(0.2)	Dense(16)	Dense(1)
11.54	16.81	450	32	3	LSTM(64)	Dropout(0.2)	LSTM(32)	Dropout(0.2)	Dense(16)	Dense(1)

Figura 5: Resultados dos testes do *dropout*

No entanto, os resultados obtidos presentes na tabela acima e nas **Figuras 25**, **16**, **27** e **28** (em anexo), não foram satisfatórios, pelo que se manteve o modelo presente no nono teste da subsecção anterior (sem *dropouts*).

4.3 *Activation functions*

De seguida, tentou-se melhorar os resultados mais favoráveis obtidos até agora (correspondentes ao modelo no nono teste na primeira subseção de testes), através da otimização da utilização de diferentes funções de ativação das últimas duas camadas (*Dense*) do modelo.

Face às inúmeras opções disponíveis para este parâmetro, cingimos os nossos testes à comparação entre as funções de ativação *LeakyReLU* e *PReLU* e o valor por defeito utilizado nos testes anteriores (*None* – nenhuma).

Os resultados destes testes está, então, presente na **Figura 6**:

Train	Test	Epochs	Batch size	Janela	Camadas do modelo			
11.26	17.76	500	32	3	LSTM(64)	LSTM(32)	Dense(16, LeakyReLU)	Dense(1)
13.36	13.27	345	32	3	LSTM(64)	LSTM(32)	Dense(16, LeakyReLU)	Dense(1)
13.28	13.09	350	32	3	LSTM(64)	LSTM(32)	Dense(16, LeakyReLU)	Dense(1, LeakyReLU)
13.17	13.19	347	32	3	LSTM(64)	LSTM(32)	Dense(16, LeakyReLU)	Dense(1, LeakyReLU)
11.85	15.97	347	32	3	LSTM(64)	LSTM(32)	Dense(16, PReLU)	Dense(1, PReLU)
13.33	13.44	325	32	3	LSTM(64)	LSTM(32)	Dense(16, PReLU)	Dense(1, PReLU)

Figura 6: Resultados dos testes das *activation functions*

Nos primeiros testes (**Figuras 29**, **30**, **32** e **33**, em anexo) começámos por utilizar *LeakyReLU* como função de ativação da penúltima camada e depois das duas últimas camadas, limitando o número de *epochs* nos teste seguintes correspondentes.

Os melhores resultados do *RMSE* foram obtidos para a utilização de *LeakyReLU* nas duas últimas camadas, cujo teste se encontra presente nas **Figuras 33** e **34**. Na primeira figura, pode-se comprovar a diminuição dos erros de teste e validação, sendo que na segunda se pode ainda observar um acompanhamento ligeiramente melhor das previsões face aos valores reais do *dataset*, principalmente na fase final de descida e respetivos mínimos locais.

Como os testes relativos à função de ativação *PReLU* (**Figuras 35** e **36**, em anexo) foram ligeiramente piores do que os de *LeakyReLU*, o modelo de referência passou a ser o correspondente ao quarto teste desta subsecção.

4.4 *Initializers*

De seguida, tentou-se melhorar os resultados mais favoráveis obtidos até agora (correspondentes ao modelo do quarto teste da subsecção anterior), através da otimização da utilização de diferentes *initializers* das últimas duas camadas (*Dense*) do modelo.

Face às inúmeras opções disponíveis para este parâmetro, cingimos os nossos testes à comparação entre os *initializers RandomNormal* e *GlorotUniform*.

Os resultados destes testes está, então, presente na **Figura 7**:

Train	Test	Epochs	Batch size	Janela	Camadas do modelo			
12.24	14.91	347	32	3	LSTM(64)	LSTM(32)	Dense(16, LeakyReLU, RandomNormal)	Dense(1, LeakyReLU, RandomNormal)
13.22	13.19	328	32	3	LSTM(64)	LSTM(32)	Dense(16, LeakyReLU, RandomNormal)	Dense(1, LeakyReLU, RandomNormal)
9.16	16.45	347	32	3	LSTM(64)	LSTM(32)	Dense(16, LeakyReLU, GlorotUniform)	Dense(1, LeakyReLU, GlorotUniform)
12.63	12.93	329	32	3	LSTM(64)	LSTM(32)	Dense(16, LeakyReLU, GlorotUniform)	Dense(1, LeakyReLU, GlorotUniform)

Figura 7: Resultados dos testes dos *initializers*

No primeiro teste (**Figura 37**, em anexo), começámos por utilizar o *initializer RandomNormal*, não obtendo uma melhoria de resultados mesmo com a limitação das *epochs* no teste seguinte (**Figura 38**, em anexo).

Posteriormente, testámos a utilização do *initilizer GlorotUniform* (**Figura 39**, em anexo), tendo obtido melhores resultados após a limitação das *epochs* (**Figuras 40** e **41**, em anexo). O modelo correspondente ao quarto teste, passou, então, a ser o modelo de referência.

4.5 *Optimizers*

De seguida, tentou-se melhorar os resultados mais favoráveis obtidos até agora (correspondentes ao modelo do último teste da subsecção anterior), através da otimização da utilização de diferentes *optimizers* para a compilação do modelo (função do *Keras model.compile()*).

Face às inúmeras opções disponíveis para este parâmetro, cingimos os nossos testes à comparação entre os *initializers Adadelta* e *Rmsprop*.

Os resultados destes testes está, então, presente na **Figura 8**:

Train	Test	Epochs	Camadas do modelo			
0.71	11.84	329	LSTM(64)	LSTM(32)	Dense(16, LeakyReLU, GlorotUniform)	Dense(1, LeakyReLU, GlorotUniform) + Adadelta
12.15	14.82	87	LSTM(64)	LSTM(32)	Dense(16, LeakyReLU, GlorotUniform)	Dense(1, LeakyReLU, GlorotUniform) + Adadelta
10.46	11.04	115	LSTM(64) + Dropout(0.2)	LSTM(32) + Dropout(0.2)	Dense(16, LeakyReLU, GlorotUniform) + Dropout(0.2)	Dense(1, LeakyReLU, GlorotUniform) + Adadelta
4.11	21.59	329	LSTM(64)	LSTM(32)	Dense(16, LeakyReLU, GlorotUniform)	Dense(1, LeakyReLU, GlorotUniform) + Rmsprop
12.33	14.21	215	LSTM(64)	LSTM(32)	Dense(16, LeakyReLU, GlorotUniform)	Dense(1, LeakyReLU, GlorotUniform) + Rmsprop
13.22	14.81	215	LSTM(64) + Dropout(0.2)	LSTM(32) + Dropout(0.2)	Dense(16, LeakyReLU, GlorotUniform) + Dropout(0.2)	Dense(1, LeakyReLU, GlorotUniform) + Rmsprop

Figura 8: Resultados dos testes dos *optimizers* (omitidos batch size (32) e janela (3) para poupar espaço horizontal na tabela)

No primeiro teste (**Figura 42**, em anexo), utilizou-se o *optimizer Adadelta*, tendo, de seguida, sido realizadas mais duas séries de testes em que se limitou o número de *epochs* e se adicionou três *dropouts* (**Figuras 43** e **44**, em anexo), por forma a limitar o *overfitting* que era visível no teste anterior.

De seguida, foi feito o mesmo procedimento mas para o *optimizer Rmsprop* (**Figuras 46**, **47** e **48**, em anexo).

Os melhores resultados foram obtidos no terceiro teste do *optimizer Adadelta* (**Figura 44** e **45**, em anexo), pelo que o modelo correspondente passou a ser o modelo de referência daí em diante.

4.6 Tamanho da janela deslizante

De seguida, tentou-se melhorar os resultados mais favoráveis obtidos até agora (correspondentes ao modelo do terceiro teste da subsecção anterior), através da utilização de um tamanho diferente da janela deslizante.

Inicialmente, como já foi dito, optámos por escolher o valor 3 por forma a que a previsão de um valor para um determinado mês tivesse em consideração os resultados do trimestre anterior. Não quisemos optar por um valor demasiado elevado pois perderíamos obrigatoriamente uma grande quantidade de informação no processo da passagem da janela deslizante pelo *dataset* que, por

si, já é pequeno. Isto porque para uma janela de tamanho N, os primeiros N valores do *dataset* não iriam ser utilizados como *label* durante este processo.

Assim sendo, limitámos os nossos testes aos valores de janela entre 1 e 6, cujos resultados se apresentam na **Figura 9**.

Train	Test	Epochs	Batch size	Janela
16.61	16.78	150	32	6
11.44	12.33	178	32	5
11.02	12.14	195	32	4
12.39	13.79	78	32	2
12.16	14.19	62	32	1

Figura 9: Resultados dos testes do tamanho da janela deslizante

No entanto, não foram registadas quaisquer melhorias dos valores do *RMSE* para o treino e teste – como se pode verificar pelas **Figuras 49**, **50**, **51**, **52** e **53**, em anexo –, pelo que se manteve o modelo de referência anterior.

4.7 *Batch size*

Finalmente, tentou-se melhorar os resultados mais favoráveis obtidos até agora (correspondentes ao modelo do terceiro teste da subsecção dos *optimizers*), através da utilização de valores diferentes para o *batch size*.

A **Figura 10** apresenta, então, os resultados deste conjunto de testes:

Train	Test	Epochs	Batch size	Janela
8.48	8.76	115	16	3
10.47	14.8	115	64	3
10.05	12.22	115	128	3
10.79	11.74	115	254	3
11.43	11.58	115	512	3

Figura 10: Resultados dos testes do *batch size*

Com o aumento do valor do *batch size*, não foram obtidas melhorias dos valores do *RMSE*, como se pode verificar pelas figuras correspondentes aos testes 2 a 5 desta subsecção (**Figuras 58**, **59**, **60** e **61**, em anexo).

No entanto, para os testes referentes a um *batch size* menor (16), foi registada uma notória melhoria, como se pode verificar pelas **Figuras 54** e **55**, em anexo.

5 Melhor modelo encontrado

Após a exaustiva fase de testes anterior, chegámos, então, ao melhor modelo encontrado para tratar o problema em questão, cuja esquematização se encontra na **Figura 11**.

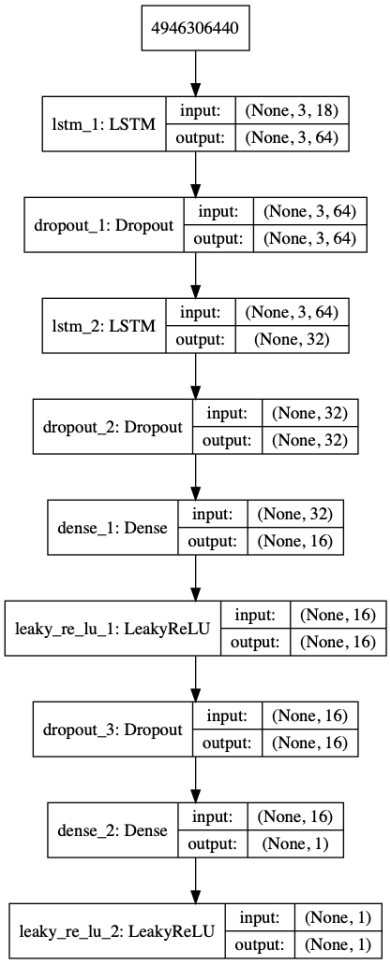

Figura 11: Esquematização do melhor modelo encontrado

Este modelo possui quatro camadas: duas *LSTM* com 64 e 32 neurónios e duas *Dense* com 16 e 1 neurónios. Foram utilizados três *dropouts* entre estas camadas e, no caso das duas últimas foi utilizado a função de ativação *LeakyReLU* e o *initializer GlorotUniform*. Para a compilação do modelo foi utilizado o *optimizer Adadelta*. Foi ainda utilizado um tamanho de janela 3, 16 de *batch size* e 115 *epochs*.

Após a execução dos testes referentes a este modelo foram obtidos os seguintes gráficos de treino + validação e teste, presentes nas **Figuras 12** e **13**, respetivamente.

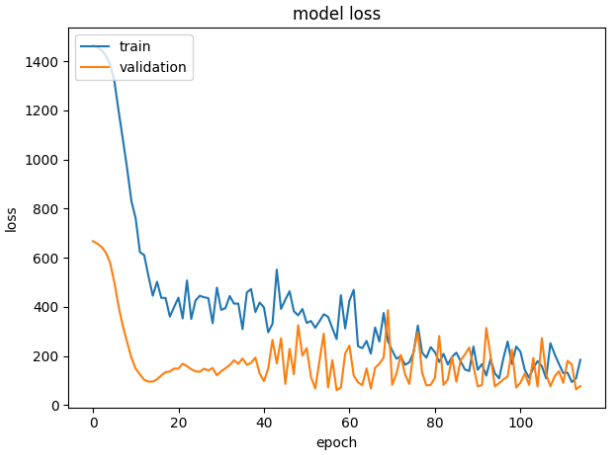

Figura 12: Resultado do treino + validação do melhor modelo encontrado

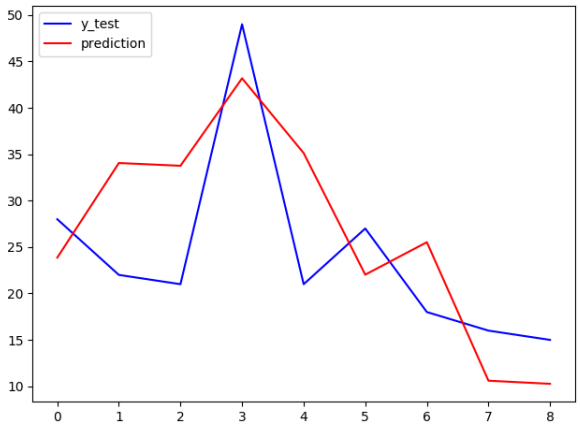

Figura 13: Resultado do teste do melhor modelo encontrado

Como podemos verificar pela figura anterior, este modelo consegue captar muito melhor a variação dos valores reais do *dataset*, nomeadamente no primeiro mínimo local, no seguinte máximo absoluto e depois na descida final e respetivos máximo e mínimos locais.

A **Figura 14** possui ainda o output final do nosso programa com as previsões correspondente ao teste presente das duas figuras anteriores, em que

se pode verificar o *RMSE* do treino + validação de 8.478 e 8.765 e as previsões da fase de teste.

```
Train Score: 71.879 MSE (8.478 RMSE)
Test Score: 76.823 MSE (8.765 RMSE)
['loss', 'acc']
Value: 28.000000 ---> Prediction: 23.860889 Diff: 4.139111
Value: 22.000000 ---> Prediction: 34.055500 Diff: 12.055500
Value: 21.000000 ---> Prediction: 33.756969 Diff: 12.756969
Value: 49.000000 ---> Prediction: 43.181381 Diff: 5.818619
Value: 21.000000 ---> Prediction: 35.125240 Diff: 14.125240
Value: 27.000000 ---> Prediction: 22.027266 Diff: 4.972734
Value: 18.000000 ---> Prediction: 25.523239 Diff: 7.523239
Value: 16.000000 ---> Prediction: 10.604321 Diff: 5.395679
Value: 15.000000 ---> Prediction: 10.269248 Diff: 4.730752
```

Figura 14: *Output* do programa para o melhor modelo encontrado

6 Conclusão e Trabalho Futuro

Dado por concluído este trabalho, o grupo considera que fez um bom trabalho referente à tentativa de otimização partindo de um modelo inicial, tendo obtido resultados que considera satisfatórios face ao contexto do problema.

Em retrospetiva, consideramos, no entanto, que a reduzida dimensão do *dataset* relativamente ao número de registos presentes acabou por ser um entrave à obtenção de resultados ainda melhores.

Como trabalho futuro, sugere-se que sejam estudados mais parâmetros que possam otimizar o modelo obtido, bem como efetuar um maior número de testes mesmo para os parâmetros já abordados neste relatório.

A Anexos

A.1 Testes – Camadas e número de neurónios

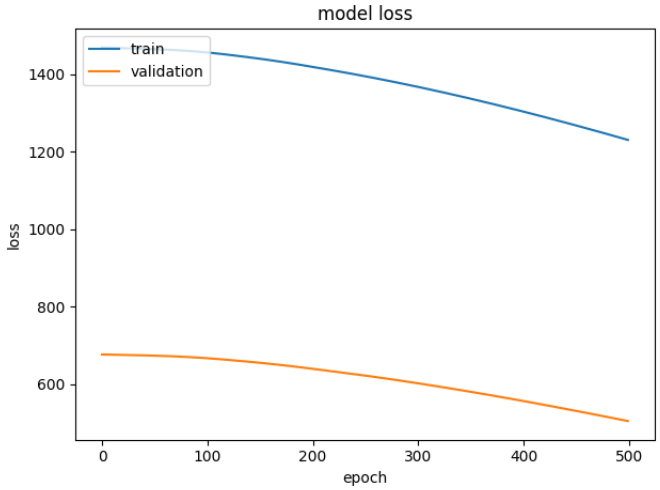

Figura 15: Primeiro teste (camadas e número de neurónios)

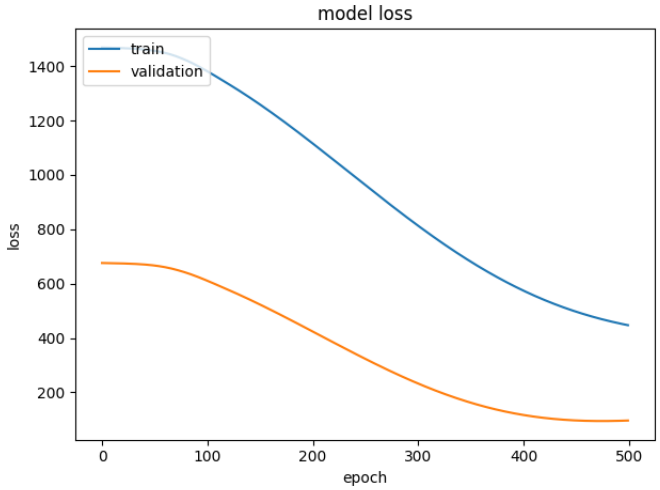

Figura 16: Segundo teste (camadas e número de neurónios)

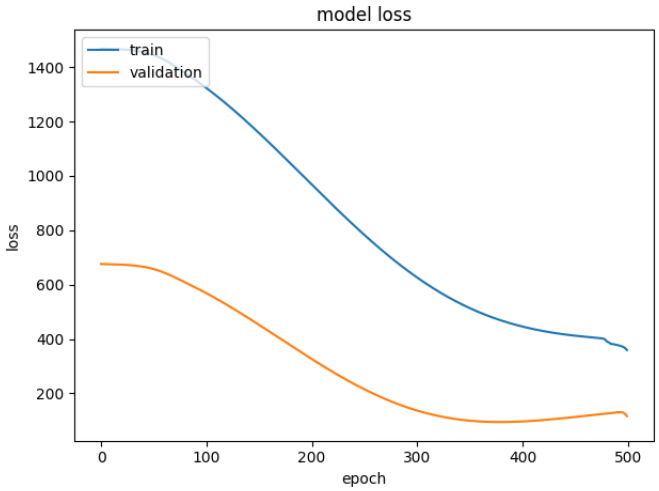

Figura 17: Terceiro teste (camadas e número de neurónios)

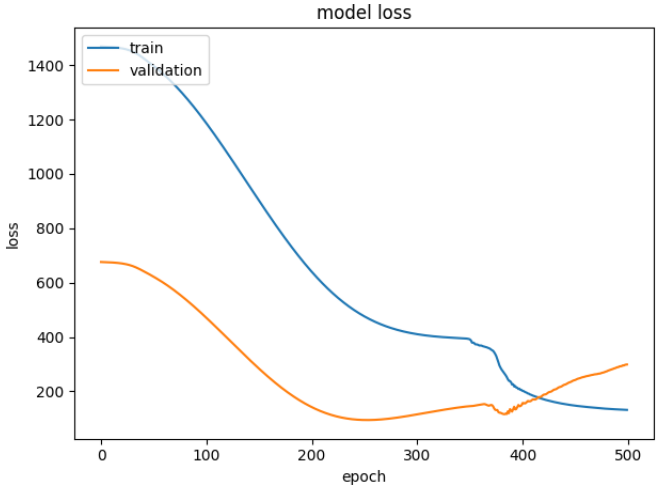

Figura 18: Quarto teste (camadas e número de neurónios)

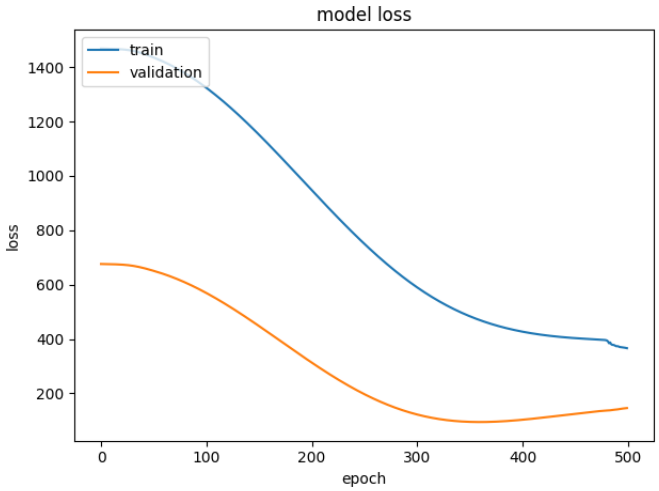

Figura 19: Quinto teste (camadas e número de neurónios)

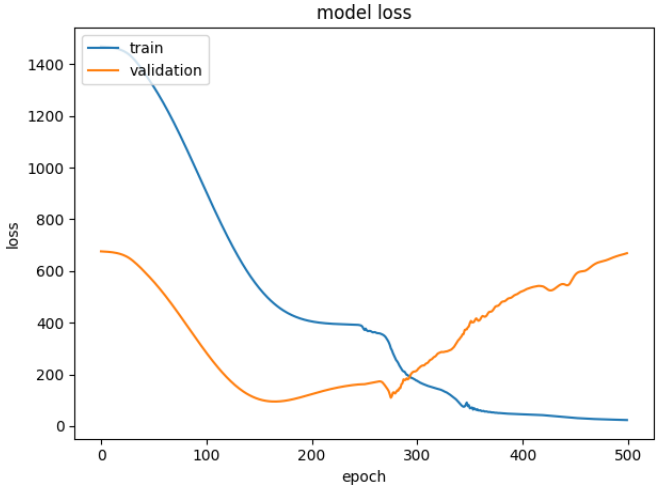

Figura 20: Sexto teste (camadas e número de neurónios)

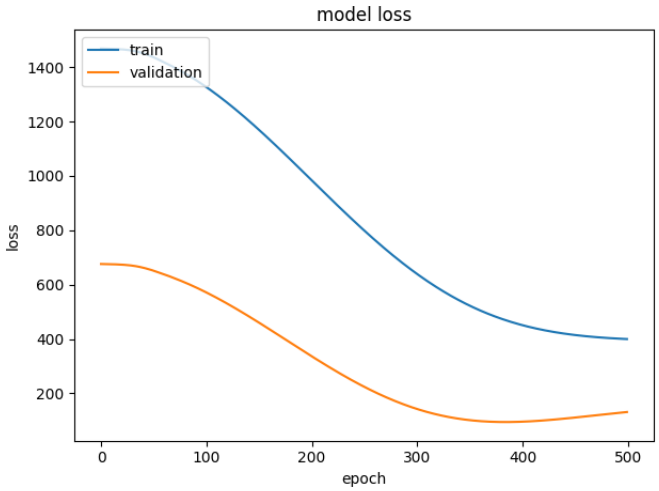

Figura 21: Sétimo teste (camadas e número de neurónios)

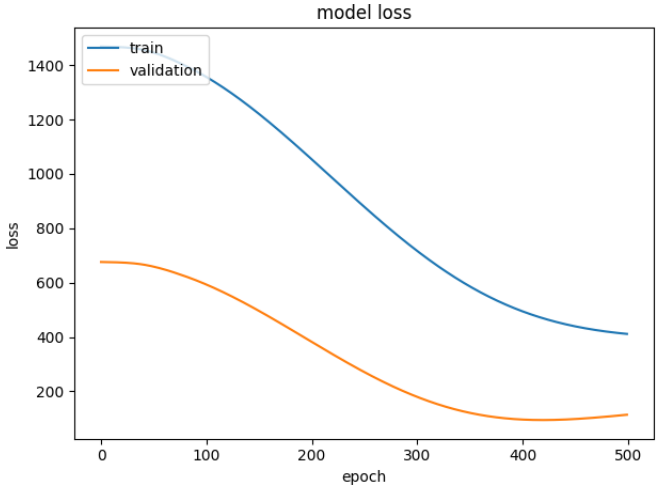

Figura 22: Oitavo teste (camadas e número de neurónios)

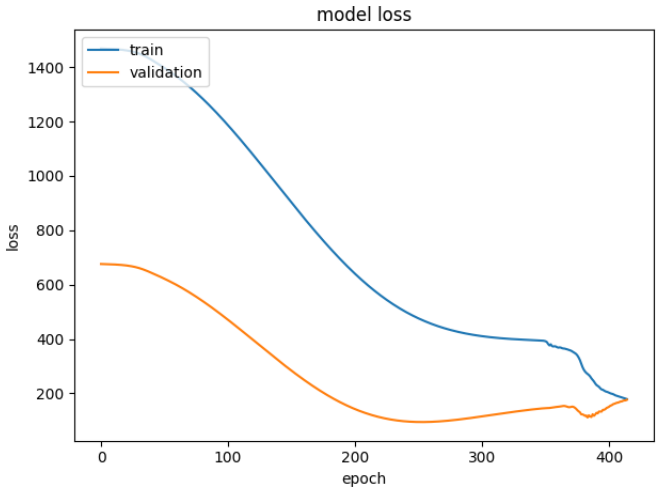

Figura 23: Nono teste (camadas e número de neurónios)

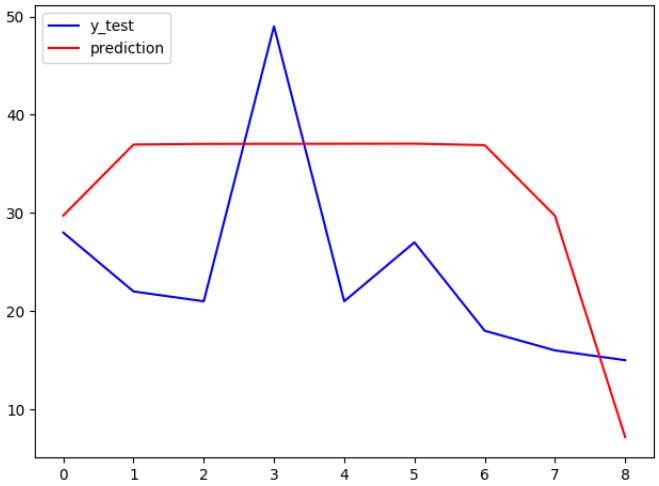

Figura 24: Nono teste (camadas e número de neurónios)

A.2 Testes – *Dropout*

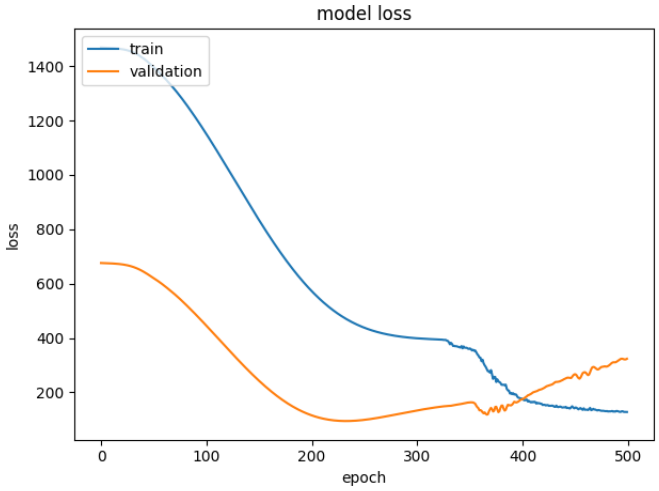

Figura 25: Primeiro teste (*dropout*)

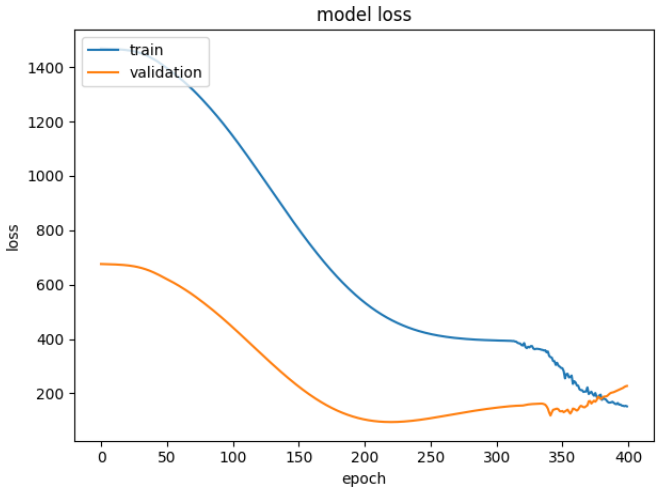

Figura 26: Primeiro teste (*dropout*)

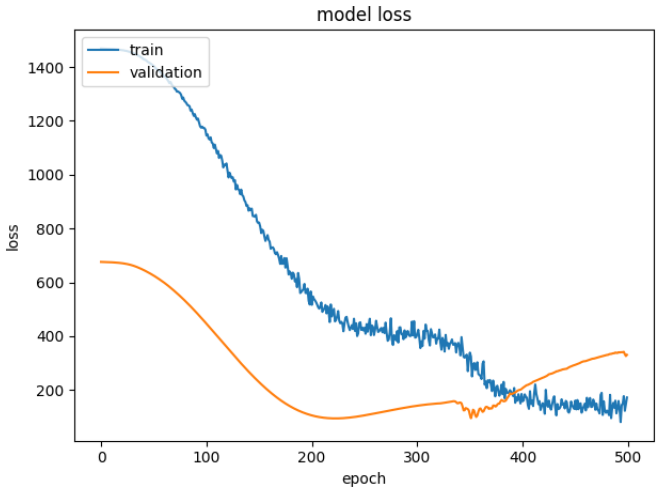

Figura 27: Segundo teste (*dropout*)

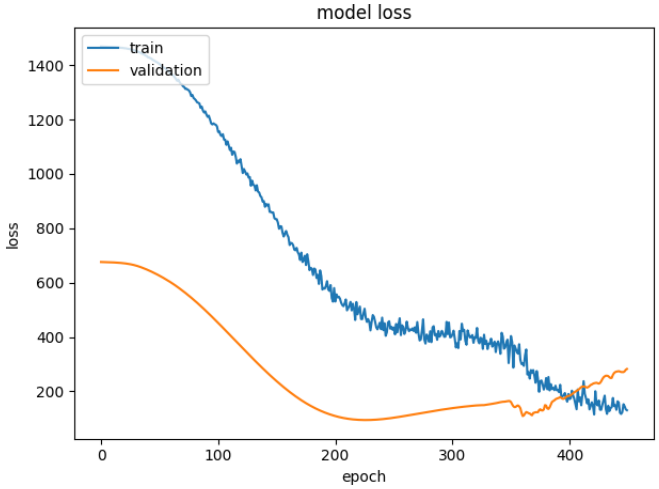

Figura 28: Segundo teste (*dropout*)

A.3 Testes – *Activation functions*

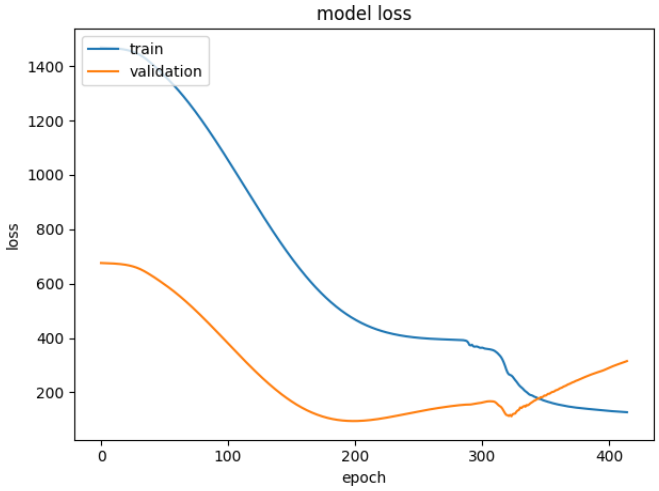

Figura 29: Primeiro teste (*activation functions*)

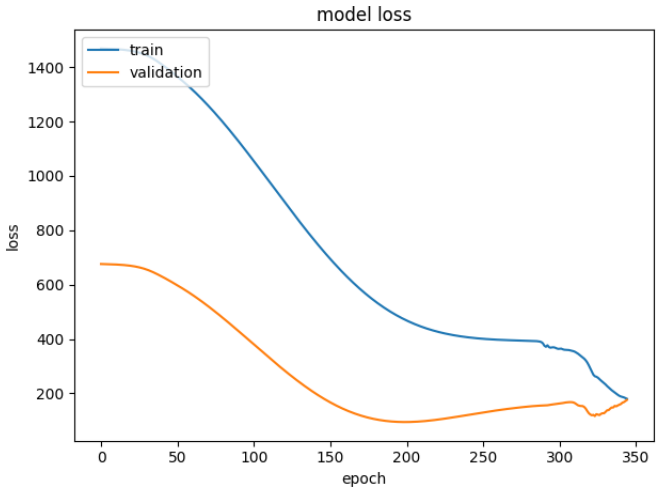

Figura 30: Primeiro teste (*activation functions*)

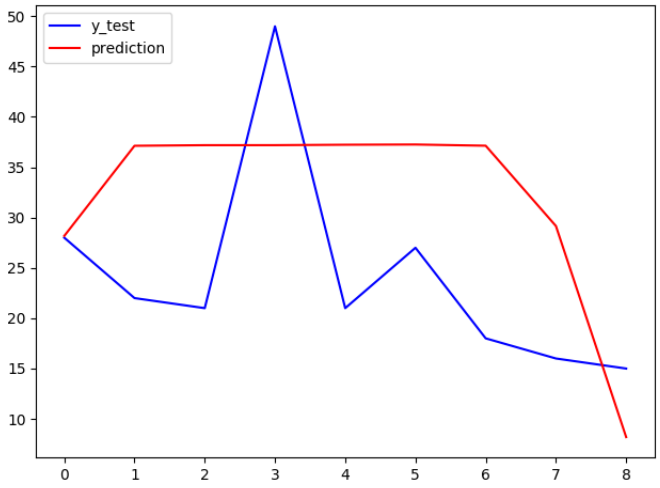

Figura 31: Primeiro teste (*activation functions*)

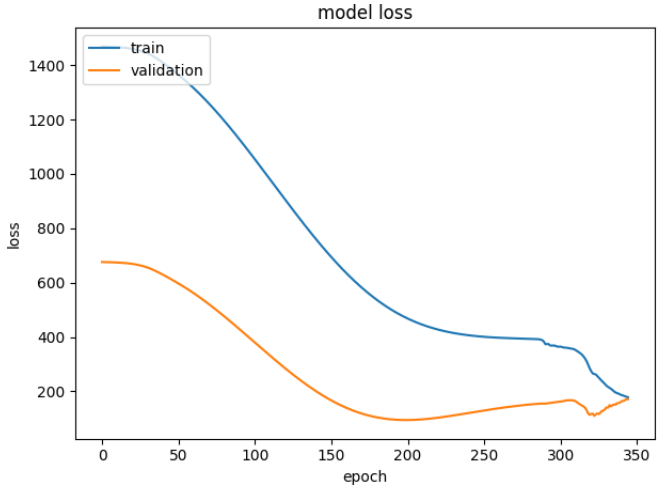

Figura 32: Segundo teste (*activation functions*)

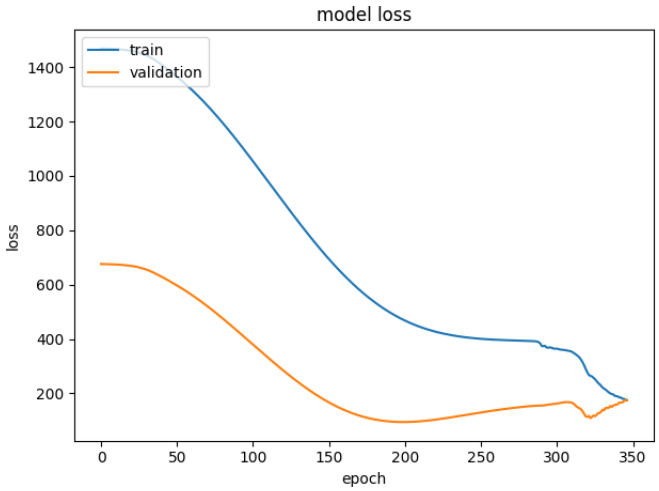

Figura 33: Segundo teste (*activation functions*)

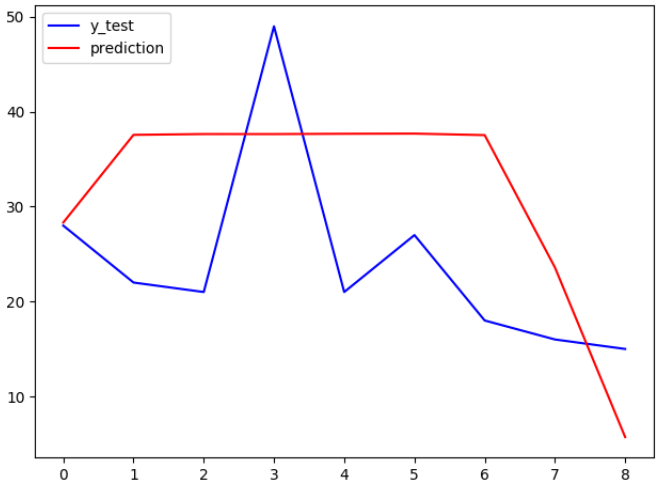

Figura 34: Segundo teste (*activation functions*)

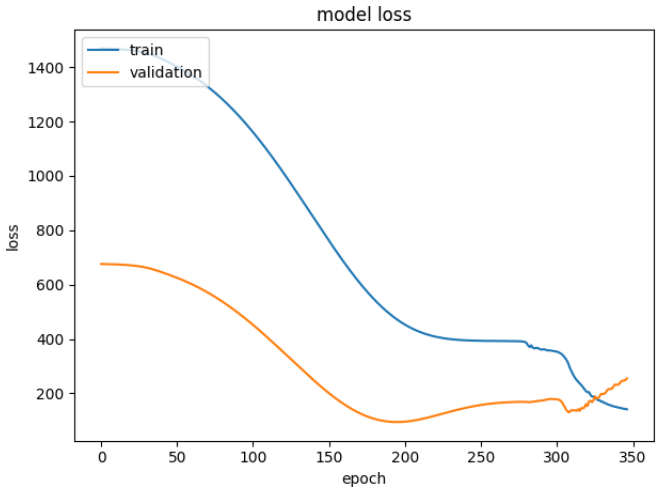

Figura 35: Terceiro teste (*activation functions*)

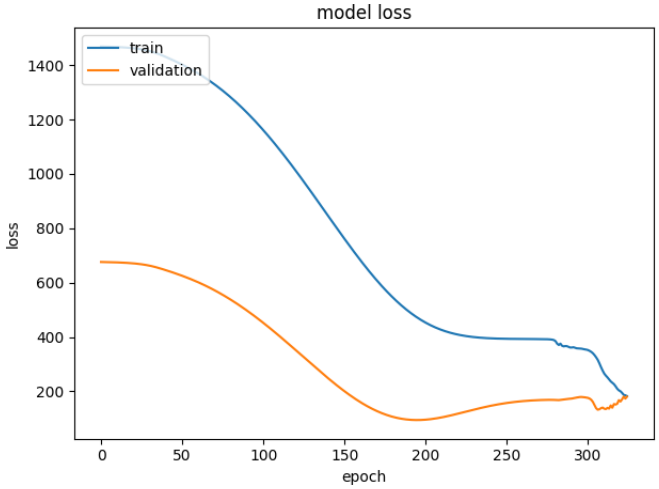

Figura 36: Terceiro teste (*activation functions*)

A.4 Testes – *Initializers*

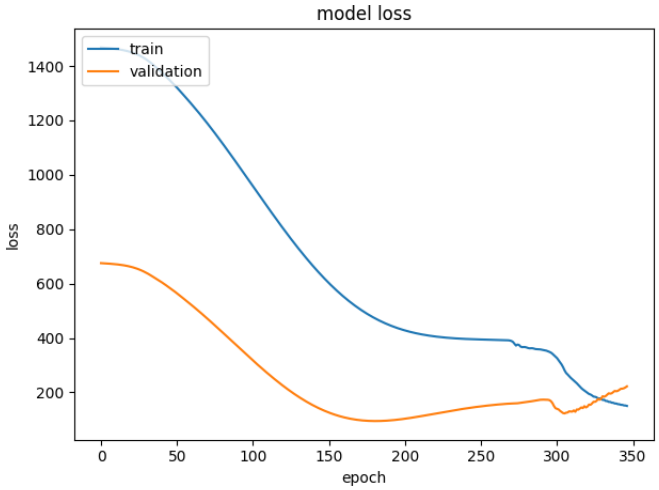

Figura 37: Primeiro teste (*initializers*)

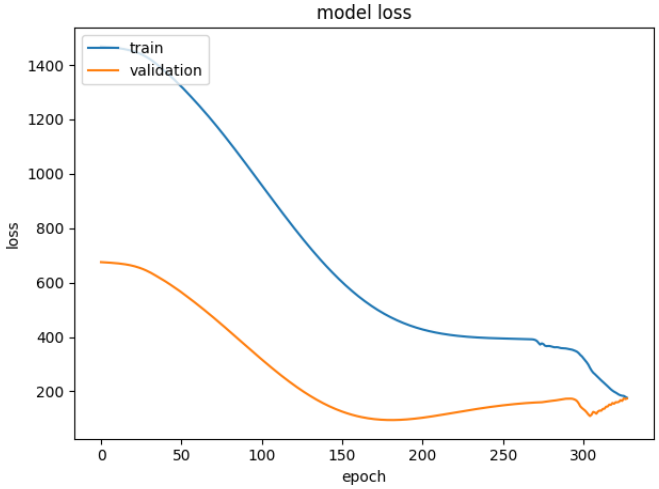

Figura 38: Primeiro teste (*initializers*)

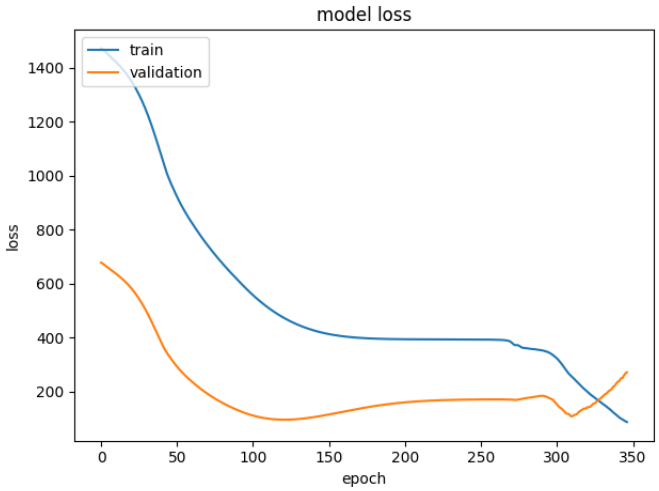

Figura 39: Segundo teste (*initializers*)

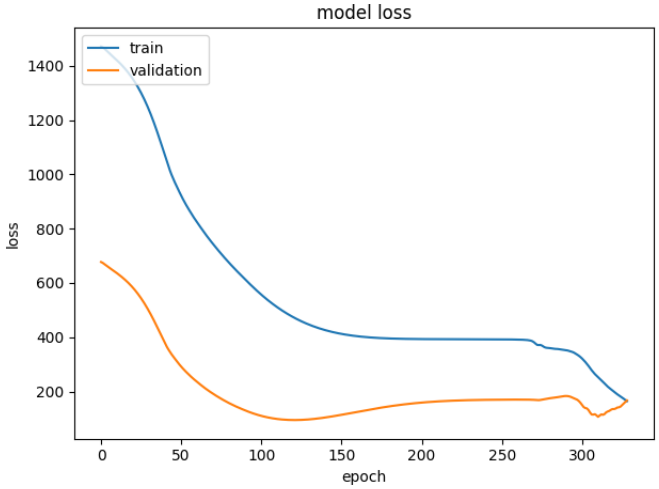

Figura 40: Segundo teste (*initializers*)

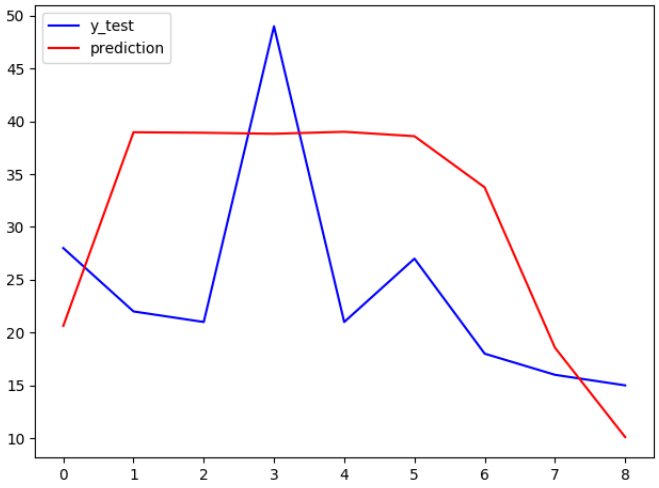

Figura 41: Segundo teste (*initializers*)

A.5 Testes – *Optimizers*

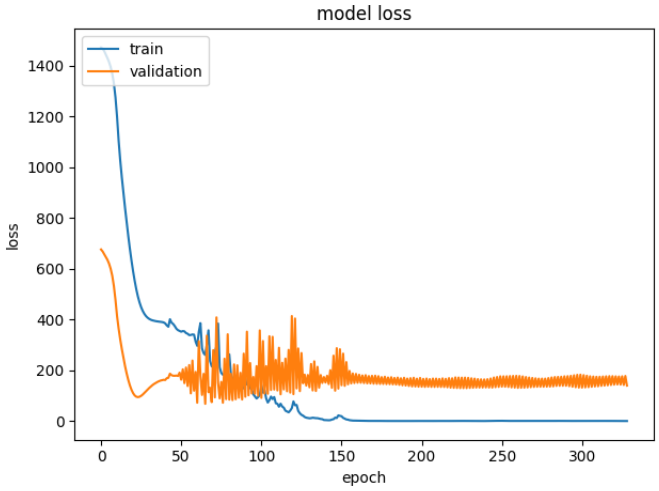

Figura 42: Primeiro teste (*optimizers*)

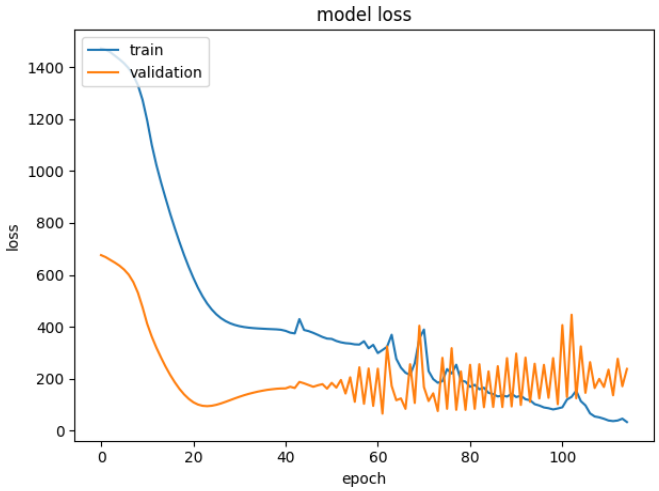

Figura 43: Primeiro teste (*optimizers*)

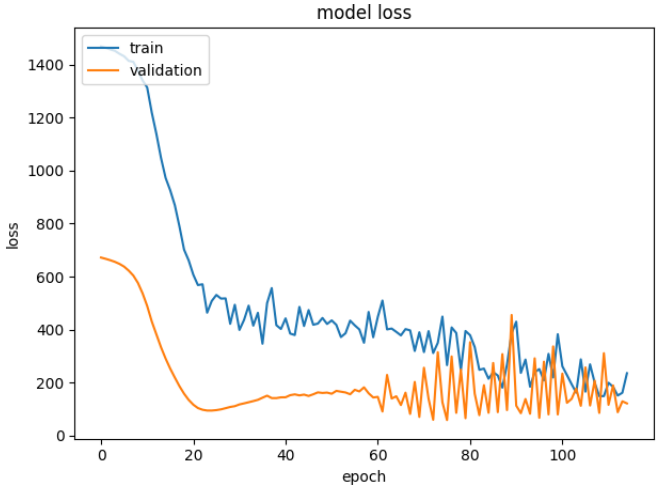

Figura 44: Primeiro teste + dropout (*optimizers*)

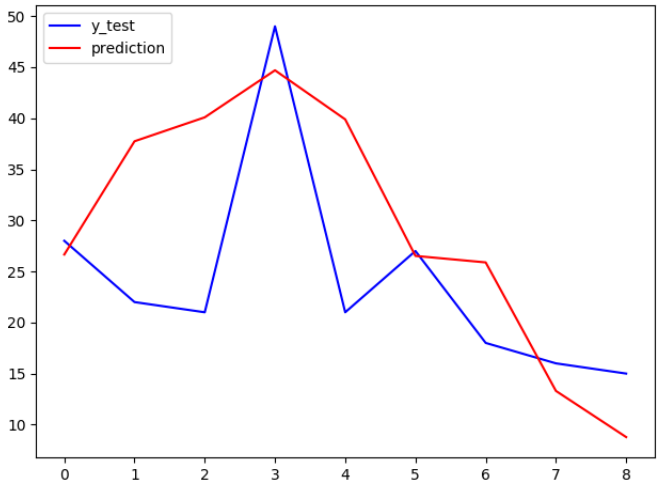

Figura 45: Primeiro teste + dropout (*optimizers*)

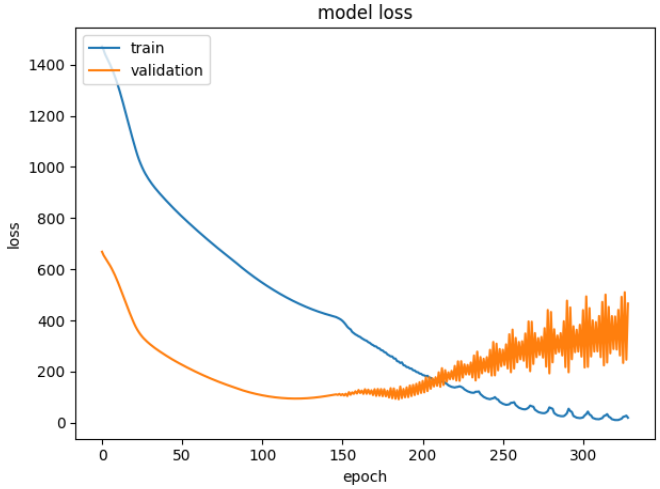

Figura 46: Segundo teste (*optimizers*)

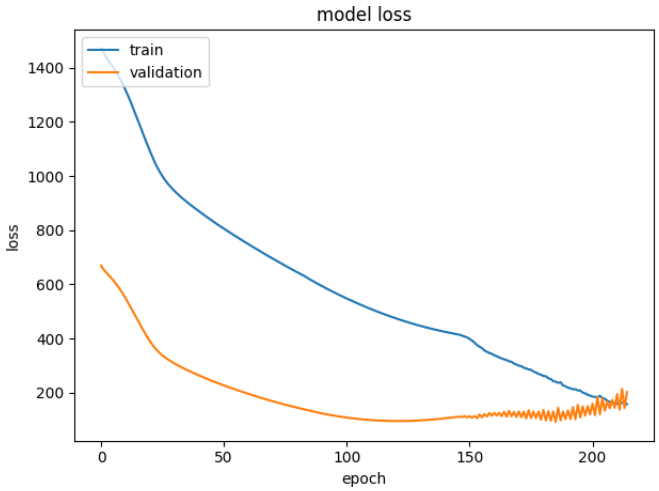

Figura 47: Segundo teste (*optimizers*)

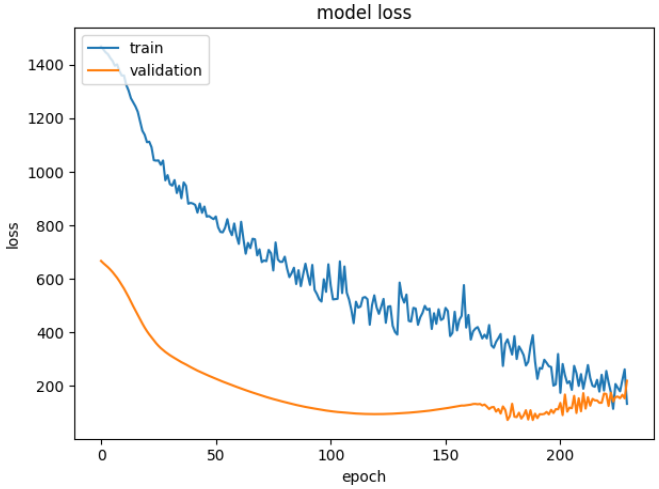

Figura 48: Segundo teste + dropout (*optimizers*)

A.6 Testes – Tamanho da janela deslizante

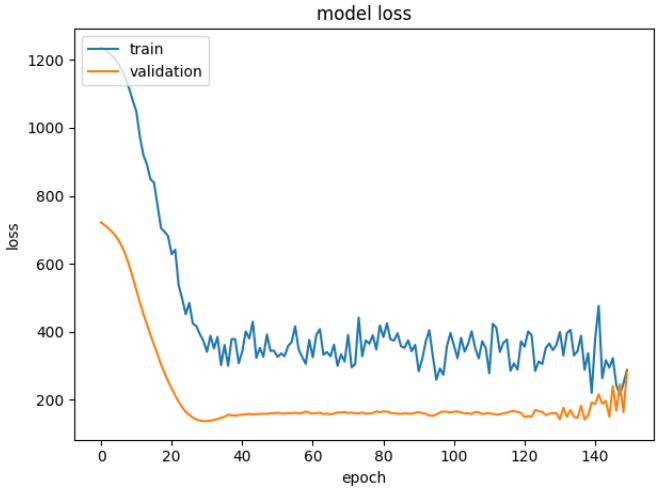

Figura 49: Primeiro teste (tamanho da janela deslizante)

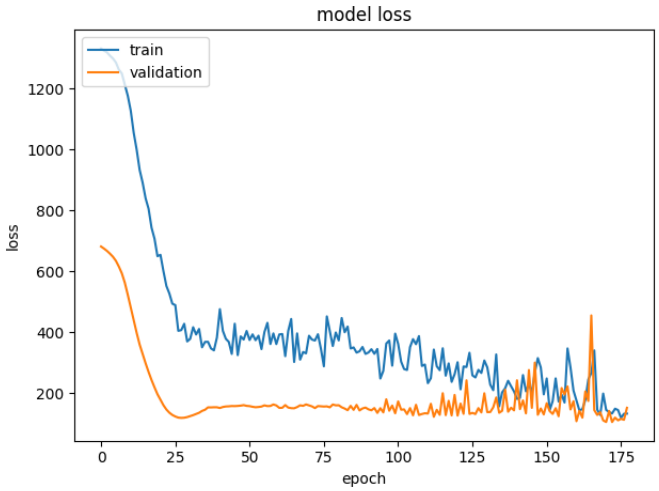

Figura 50: Segundo teste (tamanho da janela deslizante)

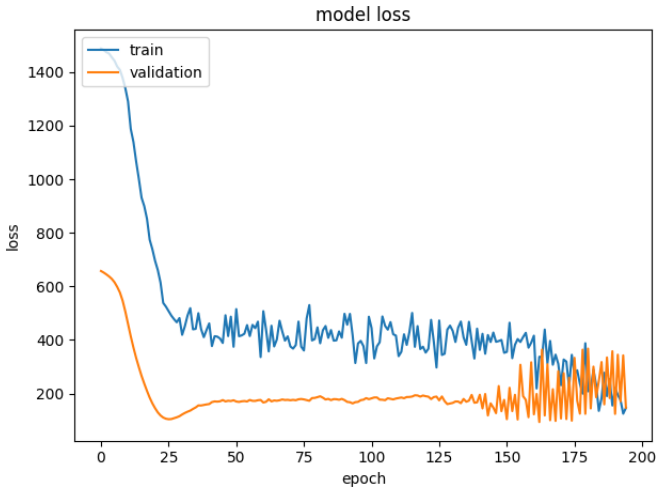

Figura 51: Terceiro teste (tamanho da janela deslizante)

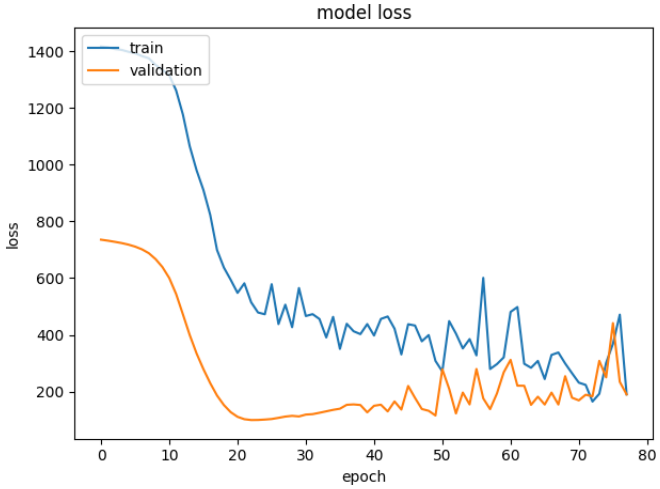

Figura 52: Quarto teste (tamanho da janela deslizante)

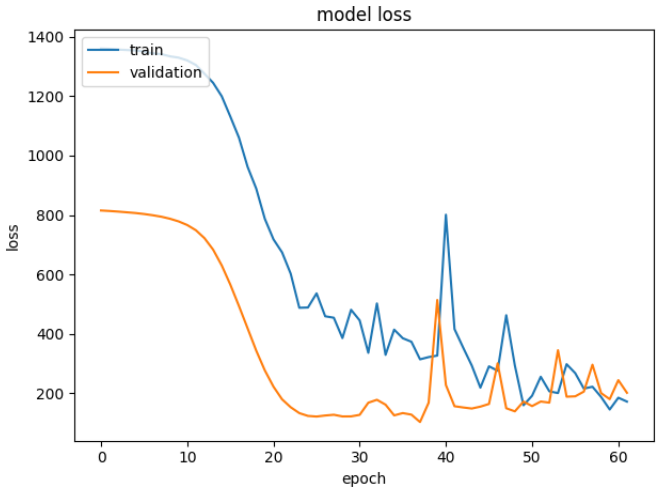

Figura 53: Quinto teste (tamanho da janela deslizante)

A.7 Testes – *Batch size*

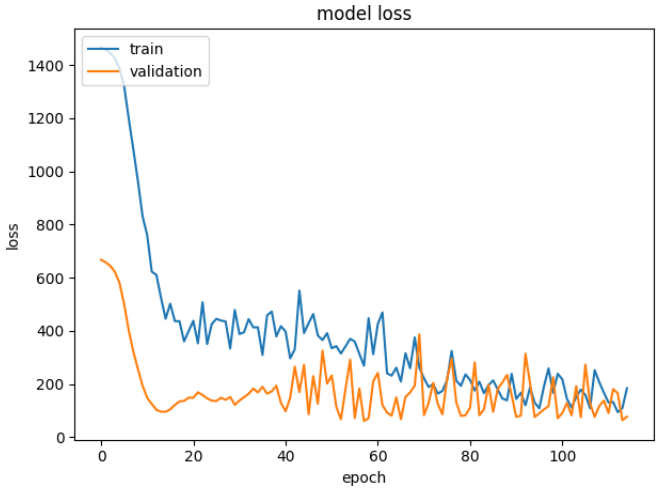

Figura 54: Primeiro teste (*batch size*)

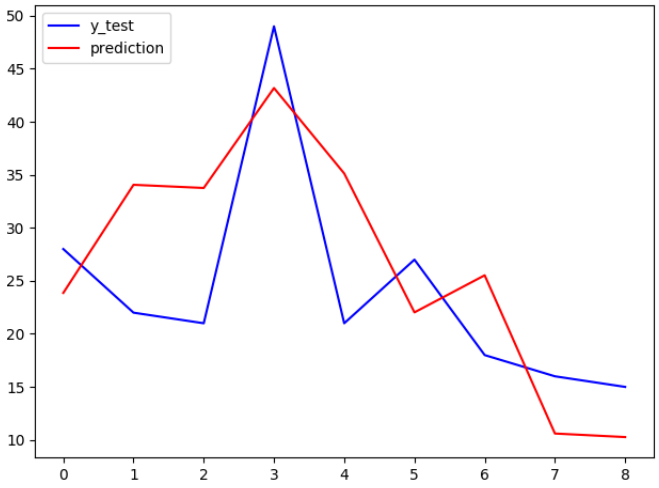

Figura 55: Primeiro teste (*batch size*)

```
Train Score: 71.879 MSE (8.478 RMSE)
Test Score: 76.823 MSE (8.765 RMSE)
['loss', 'acc']
Value: 28.000000 ---> Prediction: 23.860889 Diff: 4.139111
Value: 22.000000 ---> Prediction: 34.055500 Diff: 12.055500
Value: 21.000000 ---> Prediction: 33.756969 Diff: 12.756969
Value: 49.000000 ---> Prediction: 43.181381 Diff: 5.818619
Value: 21.000000 ---> Prediction: 35.125240 Diff: 14.125240
Value: 27.000000 ---> Prediction: 22.027266 Diff: 4.972734
Value: 18.000000 ---> Prediction: 25.523239 Diff: 7.523239
Value: 16.000000 ---> Prediction: 10.604321 Diff: 5.395679
Value: 15.000000 ---> Prediction: 10.269248 Diff: 4.730752
```

Figura 56: Primeiro teste resultados (*batch size*)

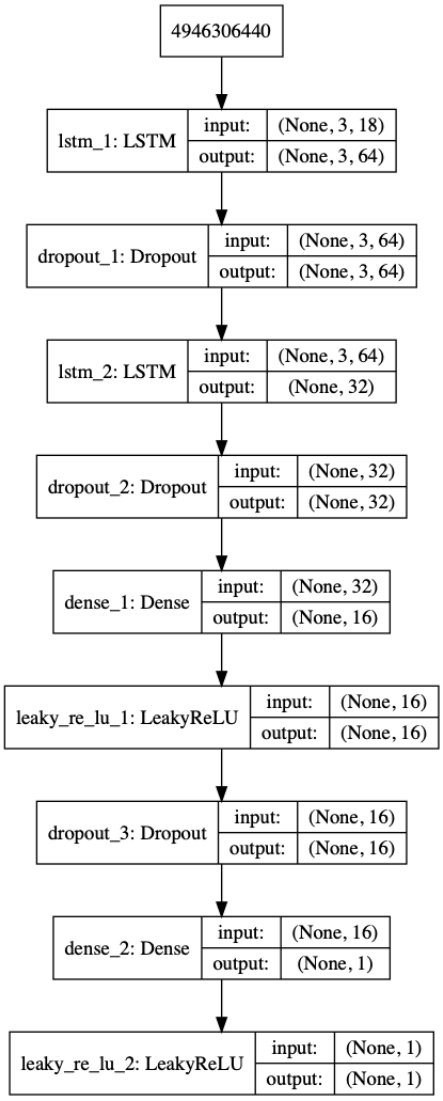

Figura 57: Primeiro teste modelo (*batch size*)

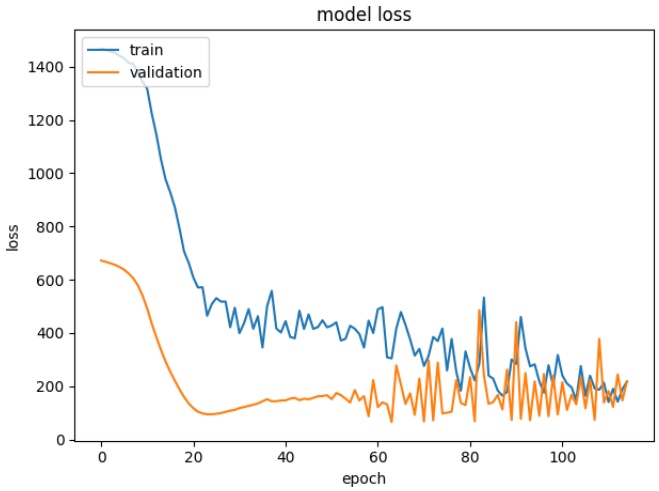

Figura 58: Segundo teste (*batch size*)

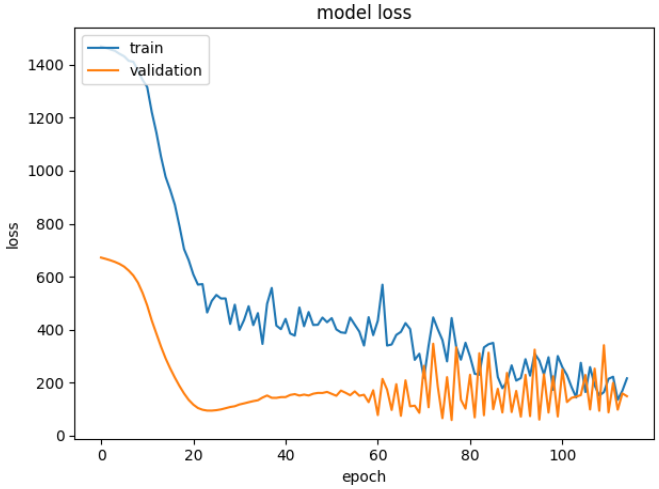

Figura 59: Terceiro teste (*batch size*)

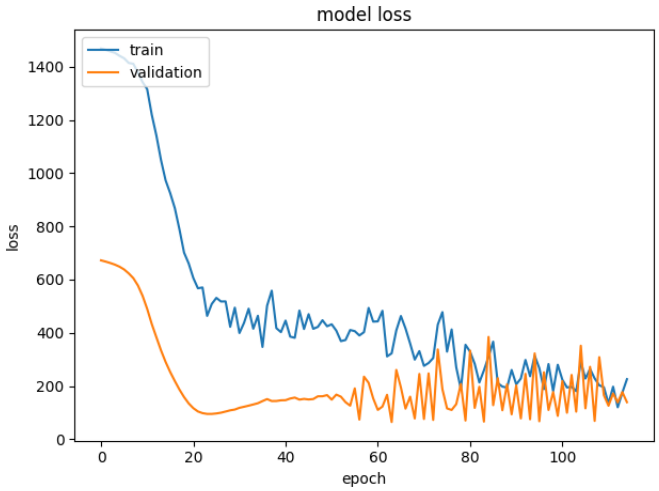

Figura 60: Quarto teste (*batch size*)

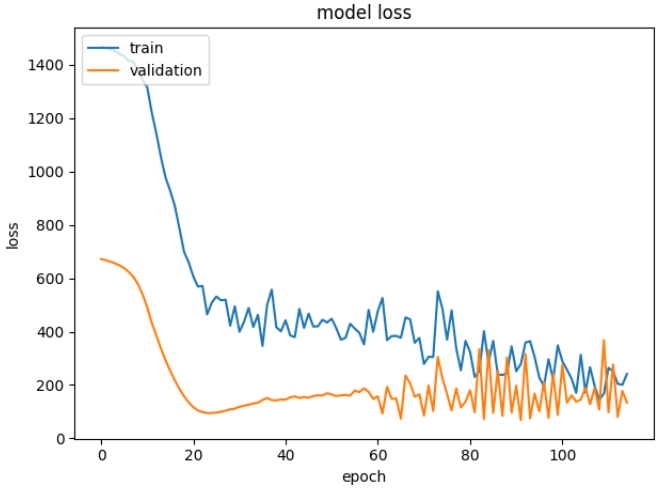

Figura 61: Quinto teste (*batch size*))