Relatório: Classificação de motores usando Machine Learning

Thiago Brandenburg

Dezembro 2022

1 Experimento

O experimento consistiu em realizar uma analise exploratória de dados sobre uma base de dados de experimentos realizados sobre um motor universal, em seguida aplicar uma rede neural artificial para classificar os motores em 3 categorias: sem defeito, armadura com segmentos em curto circuito e armadura com segmento quebrado, por fim se realiza uma analise dos resultados obtidos. Foram disponibilizados 500 experimentos, cada qual com 1000 medições em sequência de corrente e outras 1000 em sequência para a tensão.

2 Inspeção Visual

Para realizar uma inspeção visual dos dados, foi escolhido experimentos aleatórios para cada estado de motor, exibidos seus gráficos e seus boxplots para definir o comportamento dos dados, a figura 1 representa 3 destas amostras, sendo a linha vermelha representa a média dos valores.

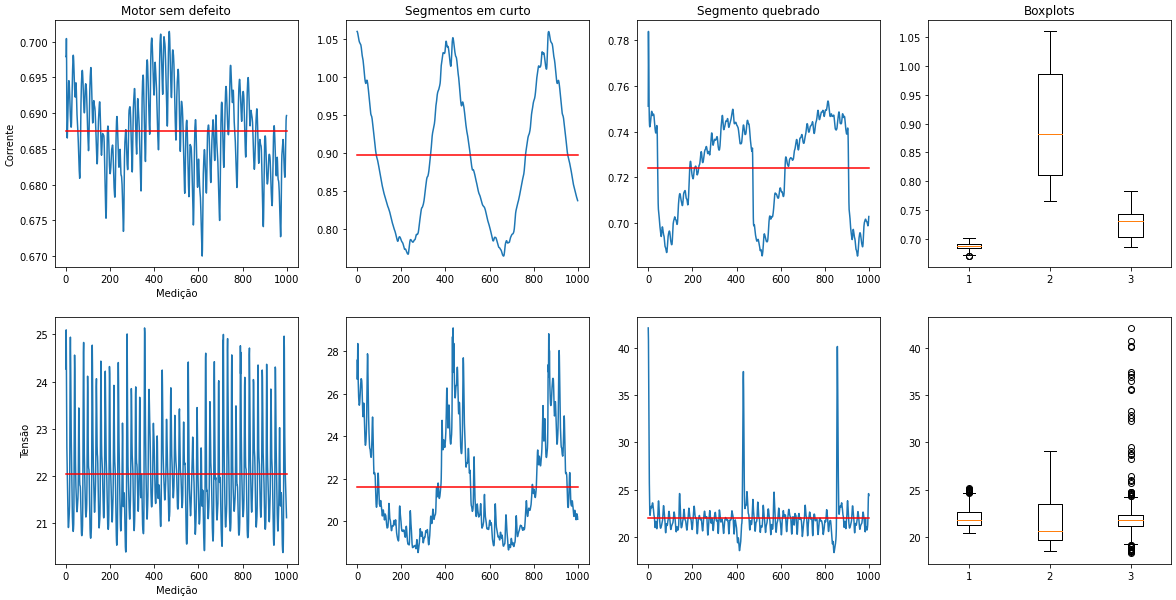

Figura 1: Comportamento dos dados

A primeira observação é que a corrente e a tensão em todos os experimentos oscilam dentro de intervalos bem definidos, apresentando como diferença a frequência da oscilação, os valores de desvio padrão e a modo de oscilação. Para as correntes, o motor sem defeito possui uma oscilação continua, com menor desvio de todos, oscilando em um intervalo por volta de 0.05, já o motor com segmentos em curto apresentam um desvio maior, motores com segmento quebrado apresentam um desvio curto, mas a oscilação tem picos e quedas abruptas bem características.

A tensão no motor sem defeito oscila em uma frequência bem alta e um desvio curto comparado aos outros, o desvio para o motor com segmentos em curtos também é baixo, mas apresenta uma oscilação bem definida que acompanha sua respectiva corrente, já o motor com segmento quebrado tem picos bem nítidos nos momentos de queda da corrente.

No geral, a inspeção visual é bem promissora, pois é possível perceber visualmente as diferenças entre os estados dos motores, portanto é muito provável a obtenção de bons resultados na classificação pela rede neural artificial. No que se diz a tratamento de dados, levando em conta o fato que os dados estão contidos em intervalos, foi utilizado a média, o teste Z e os percentis para tratar os dados.

3 Imputação e Tratamento de dados

Imputou-se os valores faltantes utilizando a média dos valores no inicio da análise. Os outliers foram identificados utilizando o teste Z, visto que há uma quantidade significativa de dados.

Os outliers identificados não foram removidos, mas sim tratados, para um outlier acima do intervalo de dados, ele foi realocado em um ponto entre o percentil 99 e 98, para um outlier no abaixo do intervalo de dados, foram realocados entre o percentil 1 e 2,. Esta escolha se justifica pois os percentis 1 e 99 limitam bem o intervalo de dados, e supõe-se que um outlier superior provem de uma região onde dados se acumulam na região superior, e a mesma lógica para os outliers inferiores. A figura 2 a seguir

mostra os dados incluindo os percentis 1 e 99, onde percebe-se que estes contem com aceitável precisão o intervalo onde os dados estão presentes. Vale ressaltar que somente os outliers são realocados para dentro dos percentis, existem dados fora dos percentis 1 e 99 que não são outliers pelo teste Z e portanto não serão realocados, afim de preservar o comportamento dos dados o máximo possível.

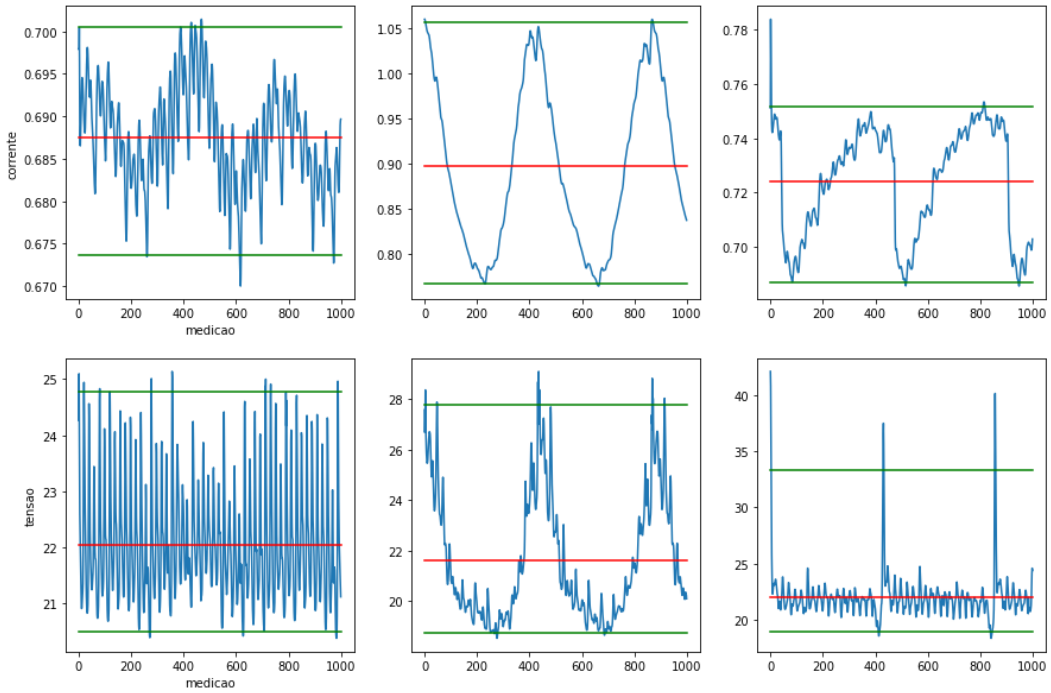

Figura 2: Testes com percentis 1 e 99

4 Escalonamento dos dados

Os dados de tensão tem valores de aproximadamente 20 à 40, visto que a rede neural utilizada trabalha com valores de 0 à 1 é necessário o escalonamento. Os dados de corrente estão presentes na região em torno de 0.6 à 1, como há possibilidade de valores acima de 1 que não são outliers também é necessário o escalonamento. Os escalonamentos foram realizados separadamente para corrente e para tensão, em ambos foi utilizado o escalonamento MinMax. A imagem 3 mostra um valor final de tensão, com dados tratados e normalizados.

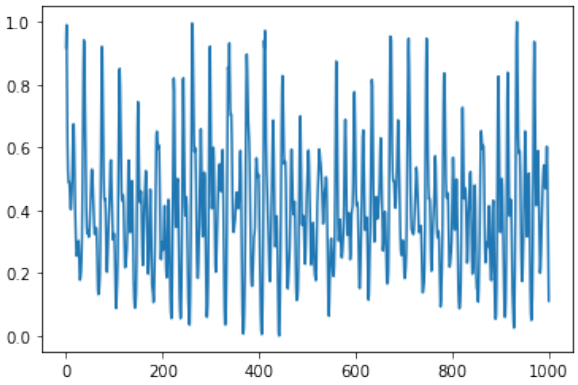

Figura 3: Tensão tratada e normalizada

5 Separação dos dados

Para separar os dados deve se ter em mente que a amostra de dados é heterogênea, pois há 300 testes de motores bons, 100 testes de motores com segmentos em curto e 100 testes com motores com segmento quebrado. Escolheu-se separar os dados de forma que estas proporções fossem mantidas nas amostras de treino, avaliação e teste, portanto a base foi separada em três partes por estado do motor, cada base de estados foi separada em treino e teste por hold-out de 70-30, sendo 10% da base de treino foi separada para avaliação, por fim os dados de cada estado são concatenados à sua respectiva base e tem sua ordem randomizada, compondo as bases finais de treino, validação e teste.

A forma final dos dados é de 3 arrays com formatos (315,2,1000), (35,2,1000) e (150,2,1000), representando respectivamente treino, validação e teste. Para cada array, p primeiro valor representa o número de testes, o segundo as colunas de corrente e tensão e o último a quantidade de medições por teste. Também há outros 3 arrays com as labels de cada respectiva base.

6 Rede Neural Aplicada

A rede neural aplicada utilizando as biblioteca Keras e Tensorflow em Python, o modelo constitui de uma rede Perceptron com uma camada de tratamento e duas camadas de neurônios, a camada inicial realoca os dados em um único array de 2000 posições para cada experimento, primeira camada tem 1024 neurônios tem função de ativação ReLU, a segunda camada é a de saída e possui 16 neurônios com função de ativação Sigmóide. O otimizador escolhido foi Adam, disponibilizado pelo Keras, a perda foi calculada utilizando a entropia cruzada esparsa, recomendada quando o eixo Y (neste caso, o estado do motor) é um numero inteiro. A métrica de avaliação foi a acurácia. A implementação da rede consta no código a seguir.

Listing 1: Rede Neural Artificial Implementada

```
import numpy as np
import tensorflow as tf
from tensorflow import keras

#Os xdados representam as bases de treino, validacao e teste
#Os xlabels representam o estado do motor para cada elemento da base

epocas = 10

model = keras.Sequential([
    keras.layers.Flatten(input_shape=(2,1000)),
    keras.layers.Dense(1024,activation = "relu"),
    keras.layers.Dense(16,activation = "sigmoid")
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

teste = model.fit(xdados_treino,
                  xlabel_treino,
                  validation_data=(xdados_validacao, xlabel_validacao),
                  epochs=epocas)
print('TESTE')
model.evaluate(xdados_teste, xlabel_teste)
```

7 Resultados

A figura 4 apresenta os valores do erro para cada época para , onde é possível perceber que os valores do erro estabilizam próximo a zero a partir 5 iteração.

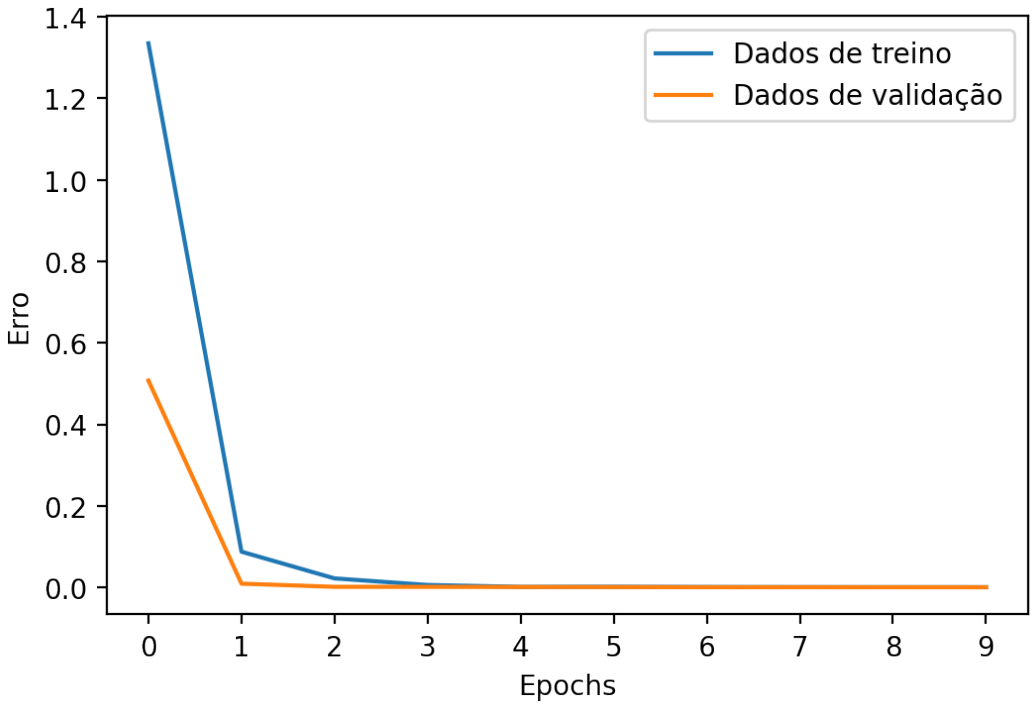

Figura 4: Erro para cada iteração/época

O gráfico demonstra um comportamento adequado, pois há convergência dos erros de treino e validação, não se caracterizando um overfitting ou underfitting perceptível. Portanto, pode-se afirmar que o modelo de classificação conseguiu resultados satisfatórios.