```
# Instala a biblioteca
!pip install numpy
!pip install matplotlib
!pip install tensorflow
import tensorflow as tf
print(tf.__version__)
```

```
Requirement already satisfied: numpy in /usr/local/lib/python3.12/dist-packages (2.0.2)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-packages (3.10.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1.3.3)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (4.59
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1.4.9
Requirement already satisfied: numpy>=1.23 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (2.0.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (25.0)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (3.2.3)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (2
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.7->matpl
Requirement already satisfied: tensorflow in /usr/local/lib/python3.12/dist-packages (2.19.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (1.6.3
Requirement already satisfied: flatbuffers>=24.3.25 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (25
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.12/dist-packages (from
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (0.2
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (18.1.1
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (3.4.0
Requirement already satisfied: packaging in /usr/local/lib/python3.12/dist-packages (from tensorflow) (25.0)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<6.0.0dev,>=3.20.3 in /
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (2.3
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-packages (from tensorflow) (75.2.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (1.17.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (3.1.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.12/dist-packages (from tensorflow)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (1.17.3)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (1.7
Requirement already satisfied: tensorboard~=2.19.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (2.1
Requirement already satisfied: keras>=3.5.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (3.10.0)
Requirement already satisfied: numpy<2.2.0,>=1.26.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (2
Requirement already satisfied: h5py>=3.11.0 in /usr/local/lib/python3.12/dist-packages (from tensorflow) (3.14.0)
Requirement already satisfied: ml-dtypes<1.0.0,>=0.5.1 in /usr/local/lib/python3.12/dist-packages (from tensorflow)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.12/dist-packages (from astunparse>=1.6.0
Requirement already satisfied: rich in /usr/local/lib/python3.12/dist-packages (from keras>=3.5.0->tensorflow) (13.9
Requirement already satisfied: namex in /usr/local/lib/python3.12/dist-packages (from keras>=3.5.0->tensorflow) (0.1
Requirement already satisfied: optree in /usr/local/lib/python3.12/dist-packages (from keras>=3.5.0->tensorflow) (0
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests<3
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2.21.0->te
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2.21
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests<3,>=2.21
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.12/dist-packages (from tensorboard~=2.19.0-
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.12/dist-packages (fro
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from tensorboard~=2.19.0-
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.12/dist-packages (from werkzeug>=1.0.1->t
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.12/dist-packages (from rich->keras>=3
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.12/dist-packages (from rich->keras>
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.12/dist-packages (from markdown-it-py>=2.2.0->ri
2.19.0
```

```
# Importando pacotes necessários
import numpy as np
import pandas as pd
from sklearn.model_selection import StratifiedKFold
from sklearn.preprocessing import StandardScaler
from keras.models import Sequential
from keras.layers import Dense
import warnings
warnings.filterwarnings('ignore')
```

```
# Carrega conjunto de dados
dados = pd.read_csv('/content/Treino.txt')
dados.head()   # Ver dados Carregados
```

	Diâmetro do furo (mm)	Diâmetro Externo (mm)	Largura B (mm)	Carga Dinâmica (kN)	Carga Estática (kN)	Velocidade de Referência (r/min)	Velocidade Limite (r/min)	Vedação (m)	Anel Externo (Unid)	Tipo 1 a 5
0	66.1598	175.048	8862.47	0.766	0.8093	89217	33909	0.800532	1	4
1	49.4644	169.377	2730.17	0.515	0.9082	90129	24723	0.400155	9	2
2	81.8460	981.657	9544.16	0.934	0.4951	33906	68493	0.600479	5	3
3	59.5039	684.376	5072.05	0.484	0.8883	22805	53880	1.000808	3	5
4	47.2217	478.536	1949.27	0.262	0.7622	7735	13208	0.200695	7	1

```
# Divide em Entrada e Resposta
X = dados.iloc[:,0:9]    #  Entrada
scaler = StandardScaler()  # Normaliza dos Dados
X = scaler.fit_transform(X)
X    # Ver as entradas Normalizadas
```

```
array([[ 0.59540181, -1.16554868,  1.35608754, ..., -0.62883194,
         0.66927164, -1.30277672],
       [ 0.01162623, -1.18606998, -0.71401454, ..., -0.95527487,
        -0.75195475,  1.44533501],
       [ 1.14388944,  1.75327761,  1.58620803, ...,  0.60017987,
        -0.04086199,  0.07127915],
       ...,
       [-0.00438482, -1.52933401,  1.12194992, ...,  1.26671195,
         0.670825  ,  0.07127915],
       [-0.54488267, -0.38733516, -1.30317147, ...,  0.75593857,
        -0.04002887, -1.30277672],
       [-0.33828107,  0.31614985, -1.3334957 , ...,  1.68263707,
        -0.74967227, -0.95926275]])
```

```
Y1 = dados.iloc[:,9] # Resposta
# Transforma os dados
Y2 = []
for i in range(len(Y1)):
    linha = []
    for j in range(5):
      if (j+1) == Y1[i]:
        linha += [1]
      else:
        linha += [0]
    Y2.append(linha)

Y = pd.DataFrame(data=Y2,columns=['Tipo1', 'Tipo2', 'Tipo3', 'Tipo4', 'Tipo5'])

Y.head() # Ver a Resposta de Treinamento
```

	Tipo1	Tipo2	Tipo3	Tipo4	Tipo5
0	0	0	0	1	0
1	0	1	0	0	0
2	0	0	1	0	0
3	0	0	0	0	1
4	1	0	0	0	0

```
# Define o Modelo
modelo = Sequential()
modelo.add(Dense(9, input_dim=9, activation='relu'))
modelo.add(Dense(5, activation='sigmoid'))
# Compila o modelo
modelo.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
# Treina o Modelo
resultado = modelo.fit(X, Y, batch_size = 400, epochs = 2000, verbose=0)
```

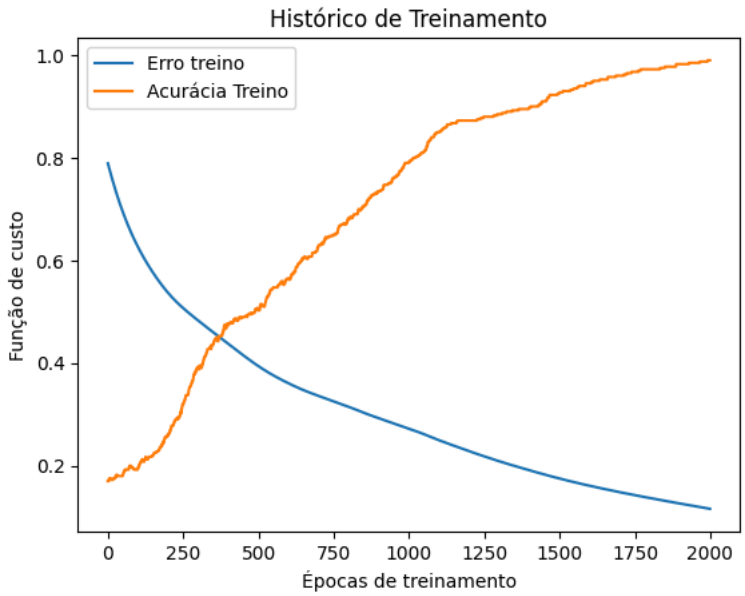
```
modelo.summary()  # Mostra a rede
```

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 9)	90
dense_1 (Dense)	(None, 5)	50

Total params: 422 (1.65 KB)
Trainable params: 140 (560.00 B)
Non-trainable params: 0 (0.00 B)
Optimizer params: 282 (1.11 KB)

```
# Mostra Resultado
import matplotlib.pyplot as plt
plt.plot(resultado.history['loss'])
plt.plot(resultado.history['accuracy'])
plt.title('Histórico de Treinamento')
plt.ylabel('Função de custo')
plt.xlabel('Épocas de treinamento')
plt.legend(['Erro treino', 'Acurácia Treino'])
plt.show()
```




```
# Carrega conjunto de teste
testes = pd.read_csv('/content/Teste.txt')
testes.head()  # Ver testes
```

	Diâmetro do furo (mm)	Diâmetro Externo (mm)	Largura B (mm)	Carga Dinâmica (kN)	Carga Estática (kN)	Velocidade de Referência (r/min)	Velocidade Limite (r/min)	Vedação (m)	Anel Externo (Unid)
0	16.9380	581.346	7797.45	0.897	0.2970	82749	29031	0.200444	2
1	18.0830	301.272	9500.78	0.074	0.5473	94154	95180	0.400566	1
2	45.2607	496.625	1300.09	0.436	0.5219	83996	6532	0.600666	2
3	90.9008	131.941	5343.58	0.568	0.0034	83362	79481	0.800469	9

```
Xtestes = testes.iloc[:,0:9] # Entrada dos Testes
Xtestes  # Ver os testes
```

```
Xtestes = scaler.fit_transform(Xtestes)  # Normaliza dos Testes
Xtestes
```

```
array([[-1.43752933e+00,  8.72886916e-01,  9.38322212e-01,
         1.05050514e+00, -3.08351685e-01,  6.07744086e-01,
        -9.32643721e-01, -1.41417218e+00, -4.76731295e-01],
       [-1.39775020e+00, -3.37884597e-01,  1.52124341e+00,
        -1.66143253e+00,  6.81140965e-01,  9.34796050e-01,
         1.27265753e+00, -7.06712579e-01, -8.34279766e-01],
       [-4.53553590e-01,  5.06634521e-01, -1.28523332e+00,
        -4.68575378e-01,  5.80729006e-01,  6.43503296e-01,
        -1.68272419e+00,  6.68894247e-04, -4.76731295e-01],
       [ 1.13205622e+00, -1.06990953e+00,  9.85478689e-02,
        -3.36108921e-02, -1.46901905e+00,  6.25322591e-01,
         7.49278123e-01,  7.06996897e-01,  2.02610800e+00],
       [-4.46487150e-01,  3.02037938e-01,  1.00566514e+00,
        -1.44065510e+00,  1.88845641e+00, -9.60843591e-01,
         2.91608687e-01,  1.41524342e+00, -1.19182824e+00],
       [ 1.04638345e+00,  2.38857984e+00,  9.95381296e-01,
         6.55082877e-01,  4.58574300e-02,  1.00840783e+00,
         1.37740676e+00, -1.41416829e+00, -8.34279766e-01],
       [ 1.21812157e+00, -1.09367761e+00, -1.73852537e-01,
        -3.99376482e-01,  5.52661096e-01, -4.67498392e-01,
         1.59001901e-02, -7.05614920e-01,  9.53462589e-01],
       [ 4.07065169e-01,  5.15907445e-01,  1.09577626e+00,
         1.08082084e-01, -1.10690348e+00, -1.55185990e+00,
         1.30609594e+00, -6.94606446e-04, -4.76731295e-01],
       [-9.34081944e-01, -1.21747223e+00,  2.33931866e-01,
         1.02743884e+00, -8.02900348e-01,  4.73482157e-01,
        -6.03780493e-02,  7.05660264e-01,  5.95914118e-01],
       [ 6.44492694e-01, -2.27443857e-01, -1.16014686e+00,
        -6.86057621e-01,  1.54650230e+00,  6.59992107e-01,
        -1.86338493e+00,  1.41335672e+00, -1.19182824e-01],
       [ 6.46611932e-01, -9.00476567e-01, -5.54594363e-01,
         5.56227312e-01,  1.14248252e+00, -7.28222325e-01,
        -3.32252631e-01, -1.41555265e+00,  2.02610800e+00],
       [ 6.25989322e-01,  3.46790063e-01,  8.18909830e-01,
        -1.78994476e+00, -6.13145465e-01, -1.52771454e+00,
         5.50247927e-01, -7.07256282e-01, -1.19182824e+00],
       [-2.52271219e-01,  1.24700292e+00, -7.45435835e-01,
         1.15265589e+00, -6.10378207e-01, -1.70089007e+00,
        -7.29179519e-01,  1.56045472e-03, -1.19182824e-01],
       [ 1.02666412e+00, -1.39625593e-01, -1.60437817e+00,
         1.03402921e+00, -1.30693676e+00,  9.95991043e-01,
        -6.91140414e-01,  7.05404674e-01,  5.95914118e-01],
       [-1.82571105e+00, -1.19334966e+00, -1.18413680e+00,
         8.95631418e-01, -2.20194729e-01,  9.87789652e-01,
         7.28508306e-01,  1.41528018e+00, -4.76731295e-01]])
```

```
# Testa a rede
Y_predito = modelo.predict(Xtestes)
print("Valores Preditos:",Y_predito)

Y_predito1 = np.array([])
Y_Resposta1 = np.array([])
for i in Y_predito:
    Y_parte = ([1 if max(i)==y else 0 for y in i])
    print(Y_parte)
    Y_predito1 = np.concatenate((Y_predito1,Y_parte))
    for j in range(len(Y_parte)):
      if Y_parte[j]==1:
        Resp = ([j+1])

    Y_Resposta1 = np.concatenate((Y_Resposta1,Resp))

Y_Resposta1 # Ver os Resuladados
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 186ms/step
Valores Preditos: [[8.5716987e-01 3.8377279e-01 5.3628753e-03 2.3952913e-05 1.9275503e-05]
 [1.8250515e-01 3.3777463e-01 1.4627956e-01 1.5264659e-03 5.5801953e-05]
 [2.8486464e-03 2.0752673e-01 6.6349560e-01 1.8194123e-01 1.8499963e-03]
 [1.3079279e-04 4.8719463e-03 6.8003297e-02 7.7797502e-01 5.1782642e-02]
 [7.6210267e-06 4.9541974e-05 8.9179853e-04 6.8690583e-02 9.7901458e-01]
 [8.4410751e-01 2.1502097e-01 6.2153498e-03 1.6784419e-05 2.2724193e-05]
 [1.6954276e-01 3.5645449e-01 1.7215568e-01 2.1360286e-03 8.9354347e-05]
 [5.0605885e-03 1.4475778e-01 7.4116665e-01 1.4925916e-01 1.5701770e-03]
 [1.4906772e-04 3.4753026e-03 6.1239365e-02 7.8768319e-01 4.9100753e-02]
 [7.2827752e-06 6.2464896e-05 6.4921781e-04 6.9794387e-02 9.6241188e-01]
 [8.8108122e-01 2.4954480e-01 7.0965108e-03 2.5777774e-05 2.5981928e-05]
 [1.7536804e-01 3.1722480e-01 1.6989958e-01 1.4824140e-03 4.1310010e-05]
 [3.8311353e-03 2.1022794e-01 6.9097584e-01 1.9022933e-01 2.8460175e-03]
 [1.0657411e-04 3.7277923e-03 6.9467016e-02 7.8952801e-01 4.1274585e-02]
```

```
 [7.9188494e-06 3.5937650e-05 7.7804708e-04 7.5822234e-02 9.5910305e-01]]
[1, 0, 0, 0, 0]
[0, 1, 0, 0, 0]
[0, 0, 1, 0, 0]
[0, 0, 0, 1, 0]
[0, 0, 0, 0, 1]
[1, 0, 0, 0, 0]
[0, 1, 0, 0, 0]
[0, 0, 1, 0, 0]
[0, 0, 0, 1, 0]
[0, 0, 0, 0, 1]
[1, 0, 0, 0, 0]
[0, 1, 0, 0, 0]
[0, 0, 1, 0, 0]
[0, 0, 0, 1, 0]
[0, 0, 0, 0, 1]
array([1., 2., 3., 4., 5., 1., 2., 3., 4., 5., 1., 2., 3., 4., 5.])
```

```
Y_Resposta = pd.DataFrame(data=Y_Resposta1, dtype=np.int8, columns=['Tipo 1 a 5'])
Y_Resposta  # Ver a Resposta
```

	Tipo 1 a 5
0	1
1	2
2	3
3	4
4	5
5	1
6	2
7	3
8	4
9	5
10	1
11	2
12	3
13	4
14	5

```
# Mostra Pesos
for layerNum, layer in enumerate(modelo.layers):
    weights = layer.get_weights()[0]
    biases = layer.get_weights()[1]

    for toNeuronNum, bias in enumerate(biases):
        print(f'{layerNum}B -> L{layerNum+1}N{toNeuronNum}: {bias}')

    for fromNeuronNum, wgt in enumerate(weights):
        for toNeuronNum, wgt2 in enumerate(wgt):
            print(f'L{layerNum}N{fromNeuronNum} \
                  -> L{layerNum+1}N{toNeuronNum} = {wgt2}')
```

```
0B -> L1N0: 0.31181657314300537
0B -> L1N1: 1.9725333452224731
0B -> L1N2: -0.7538341879844666
0B -> L1N3: 0.6177404522895813
0B -> L1N4: 0.49979162216186523
0B -> L1N5: -1.2521060705184937
0B -> L1N6: 1.8149276971817017
0B -> L1N7: 0.5405175089836121
```

```
0B -> L1N8: 0.09129773080348969
L0N0                    -> L1N0 = 0.025492941960692406
L0N0                    -> L1N1 = -0.04147998243570328
L0N0                    -> L1N2 = 3.0102131859166548e-05
L0N0                    -> L1N3 = -0.0073944395408034325
L0N0                    -> L1N4 = -0.11845748126506805
L0N0                    -> L1N5 = 0.0001683716691331938
L0N0                    -> L1N6 = 0.08019407838582993
L0N0                    -> L1N7 = 0.01465736236423254
L0N0                    -> L1N8 = 3.3604410418774933e-06
L0N1                    -> L1N0 = -0.03713059052824974
L0N1                    -> L1N1 = -0.39480826258659363
L0N1                    -> L1N2 = -0.00024858303368091583
L0N1                    -> L1N3 = -0.07263470441102982
L0N1                    -> L1N4 = -0.17024755477905273
L0N1                    -> L1N5 = -0.0007736980332992971
L0N1                    -> L1N6 = 0.3293956518173218
L0N1                    -> L1N7 = 0.146325945854187
L0N1                    -> L1N8 = -0.00021044365712441504
L0N2                    -> L1N0 = 0.007434753235429525
L0N2                    -> L1N1 = -0.11093560606241226
L0N2                    -> L1N2 = 0.00026865926338359714
L0N2                    -> L1N3 = -0.04003934562206268
L0N2                    -> L1N4 = -0.09404151886701584
L0N2                    -> L1N5 = 0.0009160063927993178
L0N2                    -> L1N6 = 0.03273417428135872
L0N2                    -> L1N7 = 0.06410622596740723
L0N2                    -> L1N8 = 0.00043589595588855445
L0N3                    -> L1N0 = 0.03133640065789223
L0N3                    -> L1N1 = -0.001516046584583819
L0N3                    -> L1N2 = 0.000647893117275089
L0N3                    -> L1N3 = 0.08685853332281113
L0N3                    -> L1N4 = -0.0479290708899498
L0N3                    -> L1N5 = 0.001658910303376615
L0N3                    -> L1N6 = -0.06448067724704742
L0N3                    -> L1N7 = -0.23911626636981964
L0N3                    -> L1N8 = -0.0012824386358261108
L0N4                    -> L1N0 = 0.023490821942687035
L0N4                    -> L1N1 = -0.1936616152524948
L0N4                    -> L1N2 = -0.00013734947424381971
L0N4                    -> L1N3 = 0.02829572930932045
L0N4                    -> L1N4 = -0.17281796038150787
L0N4                    -> L1N5 = -0.0016392999095842242
L0N4                    -> L1N6 = 0.13362661004066467
L0N4                    -> L1N7 = -0.0921023041009903
L0N4                    -> L1N8 = -0.00031391967786476016
L0N5                    -> L1N0 = 0.004149658139795065
L0N5                    -> L1N1 = 0.06827863305807114
L0N5                    -> L1N2 = -0.0007958905771374702
L0N5                    -> L1N3 = 0.031320031732320786
```