INSTITUTO FEDERAL DO ESPÍRITO SANTO
BACHARELADO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

VINICIUS DE MOURA SIQUEIRA

USO DA REGRESSÃO LOGÍSTICA PARA RESOLVER UM PROBLEMA DE CLASSIFICAÇÃO SIMPLES

Linhares
2021

VINICIUS DE MOURA SIQUEIRA

USO DA REGRESSÃO LOGÍSTICA PARA RESOLVER UM PROBLEMA DE CLASSIFICAÇÃO SIMPLES

Trabalho apresentado na disciplina de Redes Neurais Artificiais no Ifes Campus Linhares

Orientador: Lucas de Assis Soares

Linhares
2021

RESUMO

O presente trabalho tem como objetivo analisar o desempenho de um modelo de regressão logística para prever o estado de funcionamento de uma máquina com base em dados de entrada. Os dados de entrada são leitura de sensores de temperatura, rotação e frequência. Como fonte de dados para essa implementação, usou-se uma base disponibilizada no Kaggle, uma plataforma de aprendizado e prática de Data Science. Essa fonte de dados era composta por 10000 registros com 10 colunas (dados de entrada ou saída) cada.

Palavras-chave: Classificação. Funcionamento de uma máquina. Kaggle.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Considerando que a classificação pode ser usada para prever a saída de uma fonte de dados tomando como base uma série de dados de entrada, faz-se útil o uso dessa técnica para generalizar a relação de informações obtidas por sensores com os resultados que esses valores, juntos, implicam no funcionamento de um objeto real (maquinário industrial).

1.1 CONJUNTO DE DADOS UTILIZADO

Uma base de dados disponibilizada pelo Kaggle é entitulada como "Machine Predictive Maintenance Classification" ou, "Classificação da Manutenção Preditiva de Máquinas". O autor da base de dados a descreve como uma fonte de dados para prever a falha de uma máquina (resultado binário) e o tipo de falha (multiclasse). Para testar a capacidade de classificação da rede treinada, optou-se por desenvolver os treinamentos em um ambiente com uma única saída, que nesse caso foi a saída binária que continha o estado de funcionamento da máquina de acordo com uma determinada associação de dados de entrada.

A base de dados conta com 10000 instâncias variáveis e possui 10 parâmetros de entrada para 2 variáveis de saída. O conjunto foi disponibilizado para uso público dia 05 de Novembro de 2021 e já possui mais de 6000 acessos e 500 dowloads.

O post realizado no Kaggle fornece os créditos para uma fonte de dados disponibilizada na UCI Machine Learning Repository. Por ser a fonte principal, ele já conta com mais de 40000 acessos e está disponível desde 2020.

O autor da fonte de dados justifica o fato de dados reais sobre manutenções preditivas serem geralmente difíceis de se obter e ainda mais difíceis de serem publicados (em virtude da confidencialidade dos dados de cada indústria). Por esse motivo ele desenvolveu uma fonte sintética de dados que reflete a realidade encontrada em industrias.

Uma lista completa dos parâmetros de entrada (da forma como foi fornecido no conjunto de dados) pode ser visualizada abaixo:

- Unique ID (UDI)
- Product ID (product id)
- Type (Product Type)
- Air temperature [K] (air temp in K)

- Process temperature [K] (process temp in K)
- Rotational speed [rpm] (Rotational speed)
- Torque [Nm] (torque)
- Tool wear [min] (tool wear in mins)

Para os dois dados de saída, tem-se:

- Target (if failed or not)
- Failure Type (type of failure)

Os dados foram fornecidos em um documento de texto nomeado como "predictive_maintenance.csv" que contém os 10 dados separados por ",".

2 DESENVOLVIMENTO

Para desenvolver o trabalho, usou-se o suporte da máquina virtual disponibilizada no Google Colaboratory, acessada diretamente pelo Google Drive.

```
1 import numpy as np #operacoes matemátcas
2
3 import pandas as pd # manipulacao de arquivos (csv, txt)
4
5 from sklearn.model_selection import train_test_split # divisao dos dados em
      treino e teste
6
7 from sklearn.linear_model import LogisticRegression # modelo de
    classificacao
```

O pacote *numpy* foi importado para possibilitar operações numéricas no programa. O pacote *pandas* foi utilizado para permitir a manipulação de arquivos com a extensão "*.csv*", que contém os dados de interesse para criar a regressão linear no contexto descrito.

Na linha 6 do trecho acima, há a importação do *train_test_split* que dividirá o conjunto de dados disponíveis em partes de treino e teste (para verificar que a rede não está apenas decorando os valores, mas sim, generalizando da maneira correta).

Em seguida, o modelo de classificação, o *LogisticRegression*, é importado. Esse será o tipo do modelo que será treinado para classificar os dados (máquina apresentou defeito ou não).

O arquivo ".csv" foi carregado para uma conta do Google Drive para facilitar a manipulação do mesmo.

```
1 # Carregamento do arquivo .csv com os dados do Google Drive
2
3 from google.colab import drive
4 drive.mount('/content/drive')
```

Em seguida, deve-se incluir os dados disponíveis no conjunto para uma variável que será utilizada internamente.

```
1 # Leitura do arquivo .csv com os dados para uma variável local
2
3 dataset = pd.read_csv('/content/drive/MyDrive/Arquivos/Aplicac o es - 
    Redes Neurais/Trabalhos/T2: Classificaç o /predictive_maintenance.csv')
```

Para obter uma noção da visão geral do tipo dos dados e da proporção que cada um deles pode tomar na distribuição dos 10000 dados, usa-se a função *describe()*.

	UDI	Air temperature [K]	Process temperature [K]	Rotational speed [rpm]	Torque [Nm]	Tool wear [min]	Target
count	10000.00000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000
mean	5000.50000	300.004930	310.005560	1538.776100	39.986910	107.951000	0.033900
std	2886.89568	2.000259	1.483734	179.284096	9.968934	63.654147	0.180981
min	1.00000	295.300000	305.700000	1168.000000	3.800000	0.000000	0.000000
25%	2500.75000	298.300000	308.800000	1423.000000	33.200000	53.000000	0.000000
50%	5000.50000	300.100000	310.100000	1503.000000	40.100000	108.000000	0.000000
75%	7500.25000	301.500000	311.100000	1612.000000	46.800000	162.000000	0.000000
max	10000.00000	304.500000	313.800000	2886.000000	76.600000	253.000000	1.000000

Figura 1 – Exibição da descrição do dataset com os dados importados

Como as colunas já estão nomeadas em um formato legível e que possibilita a interpretação natural, não foi necessário renomear cada variável.

	UDI	Product ID	Type	Air temperature [K]	Process temperature [K]	Rotational speed [rpm]	Torque [Nm]	Tool wear [min]	Target	Failure Type
0	1	M14860	M	298.1	308.6	1551	42.8	0	0	No Failure
1	2	L47181	L	298.2	308.7	1408	46.3	3	0	No Failure
2	3	L47182	L	298.1	308.5	1498	49.4	5	0	No Failure

Figura 2 – Exibição do formato do dataframe com os dados importados

Na Figura 2 é possível visualizar o formato dos dados identificados como entradas. A saída utilizada é o penúltimo valor exibido na coluna, titulado como "Target".

Em seguida, os dados são extraídos do dataframe para que possam ser tratados individualmente (entradas e saídas).

```
1 # Carrega uma variável local com os dados de entrada
2 dados = dataset.iloc[:, 3:8].values
3
4 # Carrega uma variável local com a opç o  binária de saída (0 ou 1 para
      quebra, 1 quando positivo)
5 resultado = dataset.iloc[:, 8].values
```

No trabalho em questão, as entradas foram tratadas como *dados* e as saídas como *resultados*.

Para alimentar a rede com dados considerados úteis para o processo de qualificação, escolheu-se o intervalo da terceira à sétima variável, que se referem ao estado de funcionamento da máquina ou ao ambiente em que ele está inserido.

A variável de entrada foi povoada com todos os 10000 registros, com os dados das 5 colunas definidas no intervalo entre "Process temperature [K]" e "Tool wear [min]", identificadas

na Figura 2. Já a variável de saída recebeu todos os 10000 registros, assim como a variável de entrada, porém contém apenas o penúltimo dado da tabela, "Target".

Em seguida é necessário dividir os dados de treino e teste para possibilitar a validação da capacidade de generalização da classificação criada.

```
1 #Divisao dos dados de treino e teste
2 dados_train, dados_test, resultado_train, resultado_test = train_test_split
      (dados, resultado, test_size=0.33)
```

Para definir a nova divisão dos dados (treino e teste), deve-se especificar o parâmetro *test_size* com a proporção para os dados de teste. Nesse caso, 33% dos dados serão separados para teste.

Para criar o modelo de classificação que será adaptado aos dados do conjunto utilizado, utilizamos o modelo importado anteriormente, o *LogisticRegression()*.

```
1 # Criacao do modelo sequencial para formular a rede
2
3 modelo = LogisticRegression()
4 modelo.fit(dados_train, resultado_train)
```

Com o método *fit* do modelo criado, ele é adaptado para se "encaixar" nos dados disponíveis para treinamento (para que consiga performar de maneira satisfatória na classificação nos dados de teste).

Para verificar a qualidade do modelo treinado, verifica-se a pontuação dele, avaliando a capacidade de "prever" o estado da máquina (com algum defeito ou não) com base nos dados fornecidos pelas variáveis de entrada (leitura de sensores).

```
1 acuracia = modelo.score(dados_test, resultado_test)
2
3 # acuracia obtida na instancia: 0.9681818
```

Criando uma variável para armazenar os valores previstos pelo modelo para cada variável de entrada, obtém-se a saída final do treinamento.

```
1 resultado_pred = modelo.predict(dados_test)
```

Verificando o nível de certeza do modelo quanto à previsão do estado de funcionamento (0 ou 1 para a máquina sem defeito ou com defeito, respectivamente), pode-se perceber se os

resultados obtidos são consistentes ou poderiam variar com a mínima variação na leitura de algum sensor.

```
probs = modelo.predict_proba(dados_test)

# o vetor probs possui o formato [possibilidade de 0, posibilidade de 1], e a saída na instancia atual foi:

# array([[0.97703206, 0.02296794],
#        [0.9962657 , 0.0037343 ],
#        [0.99638468, 0.00361532],
#        ...,
#        [0.97479471, 0.02520529],
#        [0.9974022 , 0.0025978 ],
#        [0.94657652, 0.05342348]])
```

3 RESULTADOS E DISCUSSÕES

Para analisar o desempenho do modelo criado (com treinamento da regressão logística), deve-se avaliar os dados sobre a acurácia no modelo de testes..

Com uma acurácia de 0.968 no conjunto de testes, pode-se considerar que ela obteve um desempenho bom, visto que o limite para esse indicador é de uma unidade inteira.

Verificou-se, por meio da atribuição na variável *probs* que o modelo possui uma boa consistência na classificação, não ficando "indeciso"na maioria das classificações.

4 CONCLUSÃO

Baseando-se nos resultados e valores evidenciados pelas variáveis de avaliação, é possível afirmar que a classificação criada pode ser considerada satisfatória. Uma acurácia de 100% é quase sempre ilusória nesse tipo de aplicação (podendo até mesmo indicar um *overfitting*, então, quanto mais se aproximar desse valor (sem atingi-lo), melhor. Também é importante verificar que mesmo treinando na base de treinamentos, ela ainda manteve um bom desempenho na base de testes, o que mostra que a acurácia é, de fato, um bom indicador para esse caso.