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

VINICIUS DE MOURA SIQUEIRA

USO DE REGRESSÃO LINEAR PARA GENERALIZAR A OBTENÇÃO DO COEFICIENTE DE DECAIMENTO DE TURBINAS EM PLANTAS DE PROPULSÃO NAVAL

Linhares
2021

VINICIUS DE MOURA SIQUEIRA

USO DE REGRESSÃO LINEAR PARA GENERALIZAR A OBTENÇÃO DO COEFICIENTE DE DECAIMENTO DE TURBINAS EM PLANTAS DE PROPULSÃO NAVAL

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 uma regressão linear para prever valores do coeficiente de decaimento de turbinas em plantas de propulsão naval. Como fonte de dados para essa implementação, usou-se uma base, disponibilizada no UCI Machine Learning Repository, que continha a leitura de 16 sensores (como entrada) e 2 resultados do coeficiente de decaímento obtido em cada situação (como saída).

Palavras-chave: Regressão Linear. Coeficiente de decaimento. Plantas de Propulsão Naval.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Considerando que a regressão linear pode ser usada para resumir a relação de dados de entrada com as saídas, faz-se úítil o uso dessa técnica para generalizar a relação de informações obtidas por sensores com os resultados que esses valores, juntos, implicam em um objeto real.

1.1 CONJUNTO DE DADOS UTILIZADO

Uma base de dados disponibilizada pela UCI Machine Learning Repository é entitulada como "Condition Based Maintenance of Naval Propulsion Plants"ou, "Manutenção baseada na condição em plantas de propulsores navais". Esse conjunto de dados foi construído em um simulador sofisticado de turbinas a gás, montado em um navio destacado por ter uma propulsão do tipo "COmbined Diesel eLetric And Gas (CODLAG)".

A base de dados conta com 11934 instâncias variáveis e possui 16 parâmetros de entrada para 2 variáveis de saída. O conjunto foi disponibilizado para uso público em 2014 e já possui mais de 75000 acessos.

A planta utilizada para realizar as simulações recebeu constantes melhorias, e teve a veracidade incrementada com diversas comparações a plantas de propulsão reais similares ao que estava sendo proposto no ambiente de simulação. Assim, os dados disponibilizados estão de acordo com o que poderia acontecer em uma implementação com um navio real.

As possibilidades do estado de degradação podem ser descritas por uma combinação de três variaveis: velocidade do navio, coeficiente de degradação do compressor e coeficiente de degradação da turbina. A variação da velocidade do navio pode variar entre 3 nós até 27 nós.

Para as variáveis de saída, os coeficientes de decaimento foram analisados em dois domínios diferentes, sendo [1; 0.95] para o coeficiente de decaimento do compressor e [1; 0.975] para o coeficiente de decaimento da turbina.

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

- Lever position (lp)
- Ship speed (v)
- Gas Turbine (GT) shaft torque

- GT rate of revolutions (GTn)
- Gas Generator rate of revolutions (GGn)
- Starboard Propeller Torque (Ts)
- Port Propeller Torque (Tp)
- Hight Pressure (HP) Turbine exit temperature (T48)
- GT Compressor inlet air temperature
- GT Compressor outlet air temperature
- HP Turbine exit pressure (P48)
- GT Compressor inlet air pressure (P1)
- GT Compressor outlet air pressure (P2)
- GT exhaust gas pressure (Pexh)
- Turbine Injection Control (TIC)
- Fuel flow (mf)

Para os dois dados de saída, tem-se:

- GT Compressor decay state coefficient
- GT Turbine decay state coefficient

Os dados foram fornecidos em um documento de texto nomeado como "data.txt"que contém os 18 dados espaçados igualmente.

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  import matplotlib.pyplot as plt #plot de gráficos
3
4  import pandas as pd # manipulacao de arquivos (csv, txt)
5
6  from sklearn.model_selection import train_test_split
7  from sklearn.metrics import r2_score, mean_squared_error
8
9  from sklearn.preprocessing import MinMaxScaler
10 from sklearn.linear_model import LinearRegression
```

O pacote *numpy* foi importado para possibilitar operações numéricas no programa. Também foi utilizado o *matplotlib.pyplot*, que possibilita a exibição de gráficos da forma como é feito originalmente no $MATLAB$, facilitando a manipulação. O pacote *pandas* foi utilizado para permitir a manipulação de arquivos com a extensão "*.txt*", 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 pacote *r2_score* calculará a medida R^2, que permite a comparação do desempenho da rede com o mais básico método de seleção: a escolha aleatória, com 50% de chance para cada saída. Na mesma linha, há também a importação do *mean_squared_error* que possibilita o cálculo do erro médio quadrático, medida que indica a diferença entre o valor estimado e do parâmetro ao quadrado.

O modelo $MinMaxScaler$ permite o processamento diferenciado dos intervalos de existência dos dados, para que a visualização fique coerente no gráfico, sem deixar os pontos muito distantes e desconexos.

Por fim, há a importação do modelo de regressão linear. Ele carrega consigo as características de uma regressão linear propriamente dita, além de todos os métodos de ajuste para que ela se enquadre nos parâmetros necessários.

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

```
titulos_tabela = np.array([
    "lp", #Lever Position
    "v", #Ship Speed
    "GTT", #Gas Turbine Shaft Torque
    "GTn", #Gas Turbine Rate of Revolutions
    "GGn", #Gas Generator Rate of Revolutions
    "Ts", #Starboard Propeller Torque
    "Tp", #Port Propeller Torque
    "T48", #HP Turbine Exit Temperature
    "T1", #GT Compressor Inlet air Pressure
    "T2", #GT Compressor Outlet air Temperature
    "P48", #HP Turbine Exhaust Gas Pressure
    "P1", #GT Compressor Inlet Air Pressure
    "P2", #GT Compressor Outlet Air Pressure
    "Pexh", #Gas Turbine Exhaust Gas Pressure
    "TIC", #Turbine Injecton Control
    "mf", #Fuel Flow
    "C1", #GT Compressor Decay State Coefficient
    "C2"]) #GT Turbine Decay State Coefficient

dataframe = pd.read_csv(
    "data.txt",
    sep="   ",
    header=None,
    names=titulos_tabela)
```

O primeiro passo foi nomear as 18 colunas que serão importadas do arquivo que armazena as informações e, em seguida, utilizar uma função da biblioteca *pandas* para realizar a leitura desses dados em um dataframe adequado que permitirá a manipulação desses dados. O parâmetro *sep* se refere ao critério de separação das informaçoes dentro do documento, que nesse caso, são 3 espaços entre as colunas.

	lp	v	GTT	GTn	GGn	Ts	Tp	T48	T1	T2	P48	P1	P2	Pexh	TIC	mf	C1	C2
0	1.138	3.0	289.964	1349.489	6677.380	7.584	7.584	464.006	288.0	550.563	1.096	0.998	5.947	1.019	7.137	0.082	0.95	0.975
1	2.088	6.0	6960.180	1376.166	6828.469	28.204	28.204	635.401	288.0	581.658	1.331	0.998	7.282	1.019	10.655	0.287	0.95	0.975
2	3.144	9.0	8379.229	1386.757	7111.811	60.358	60.358	606.002	288.0	587.587	1.389	0.998	7.574	1.020	13.086	0.259	0.95	0.975
3	4.161	12.0	14724.395	1547.465	7792.630	113.774	113.774	661.471	288.0	613.851	1.658	0.998	9.007	1.022	18.109	0.358	0.95	0.975
4	5.140	15.0	21636.432	1924.313	8494.777	175.306	175.306	731.494	288.0	645.642	2.078	0.998	11.197	1.026	26.373	0.522	0.95	0.975
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
11929	5.140	15.0	21624.934	1924.342	8470.013	175.239	175.239	681.658	288.0	628.950	2.087	0.998	10.990	1.027	23.803	0.471	1.00	1.000
11930	6.175	18.0	29763.213	2306.745	8800.352	245.954	245.954	747.405	288.0	658.853	2.512	0.998	13.109	1.031	32.671	0.647	1.00	1.000
11931	7.148	21.0	39003.867	2678.052	9120.889	332.389	332.389	796.457	288.0	680.393	2.982	0.998	15.420	1.036	42.104	0.834	1.00	1.000
11932	8.206	24.0	50992.579	3087.434	9300.274	438.024	438.024	892.945	288.0	722.029	3.594	0.998	18.293	1.043	58.064	1.149	1.00	1.000
11933	9.300	27.0	72775.130	3560.400	9742.950	644.880	644.880	1038.411	288.0	767.595	4.531	0.998	22.464	1.052	86.067	1.704	1.00	1.000

11934 rows × 18 columns

Figura 1 – Exibição do formato do dataframe com os dados importados

Na Figura 1 é possível visualizar o formato dos dados identificados pelas abreviações introduzidas no início deste trabalho. As duas saídas, coeficiente de decaimento do compressor e da turbina foram identificadas como $C1$ e $C2$, respectivamente.

Em seguida, os dados são extraídos do dataframe para que possam ser tratados individualmente (entradas e saídas). No trabalho em questão, as entradas foram tratadas como *dados* e as saídas como *resultados*.

```
dados = dataframe.iloc[:, :16].values
#Obtencao dos dados do compressor na saída C1
resultados_C1 = dataframe.iloc[:, 16].values

#Obtencao dos dados da turbina na saída C2
resultados_C2 = dataframe.iloc[:, 17].values
```

A variável de entrada foi povoada com todos os 11934 registros, com os dados das 16 primeiras colunas, identificadas na Figura 1 como o intervalo de variáveis entre lp e nf. Já a variável de saída recebeu todos os 11934 registros, assim como a variável de entrada, porem contém apenas os dois últimos dados da tabela, $C1$ e $C2$.

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

```
#Divisao dos dados de treino e teste para C1
dados_train_C1, dados_test_C1, resultados_train_C1, resultados_test_C1 =
    train_test_split(dados, resultados_C1, test_size=0.33)

#Divisao dos dados de treino e teste para C2
dados_train_C2, dados_test_C2, resultados_train_C2, resultados_test_C2 =
    train_test_split(dados, resultados_C2, 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.

Como os dados estão em unidades diferentes, é esperado que as medidas estejam muito distantes uma das outras (quando a unidade é desconsiderada). Para que facilite a visualização e o entendimento do resultado criado, e necessário normalizar os dados em um intervalo que abrangerá desde os valores mínimos até os valores máximos de cada variável.

```
#Tratamento da escala para C1
scalerDados_C1 = MinMaxScaler()
```

```
3  scalerDados_C1.fit(dados_train_C1)
4  dadosTrain_norm_C1 = scalerDados_C1.transform(dados_train_C1)
5  dadosTest_norm_C1 = scalerDados_C1.transform(dados_test_C1)
6
7  scalerResultados_C1 = MinMaxScaler()
8  scalerResultados_C1.fit(resultados_train_C1.reshape(-1, 1))
9  resultadosTrain_norm_C1 = scalerResultados_C1.transform(resultados_train_C1
       .reshape(-1, 1))
10 resultadosTest_norm_C1 = scalerResultados_C1.transform(resultados_test_C1.
       reshape(-1, 1))
11
12 #Tratamento da escala para C1
13 scalerDados_C2 = MinMaxScaler()
14 scalerDados_C2.fit(dados_train_C2)
15 dadosTrain_norm_C2 = scalerDados_C2.transform(dados_train_C2)
16 dadosTest_norm_C2 = scalerDados_C2.transform(dados_test_C2)
17
18 scalerResultados_C2 = MinMaxScaler()
19 scalerResultados_C2.fit(resultados_train_C2.reshape(-1, 1))
20 resultadosTrain_norm_C2 = scalerResultados_C2.transform(resultados_train_C2
       .reshape(-1, 1))
21 resultadosTest_norm_C2 = scalerResultados_C2.transform(resultados_test_C2.
       reshape(-1, 1))
```

Para isso, uma nova escala é criada com o tipo $MinMaxScaler()$, que armazena os limites superior e inferior do conjunto de dados a que se refere (uma para os dados de entrada, e outra para os dados de saída).

Em seguida, as novas variáveis são criadas para armazenar todos esses valores normalizados (dentro de um mesmo intervalo) para a entrada e saída. Elas serão utilizadas para gerar um modelo de regressão linear e, em seguida, será feito um processo para retornar os valores para a escala normal.

Para criar o modelo de regressão linear que será adaptado aos dados do conjunto utilizado, utilizamos o modelo importado anteriormente, o $LinearRegression$.

```
1  #Regressao linear para C1
2  modelo_C1 = LinearRegression()
3  modelo_C1.fit(dadosTrain_norm_C1, resultadosTrain_norm_C1)
4
5  #Regressao linear para C2
6  modelo_C2 = LinearRegression()
7  modelo_C2.fit(dadosTrain_norm_C2, resultadosTrain_norm_C2)
```

Em seguida, utilizam-se os dados normalizados para criar a regressão linear.

Uma nova variável deve ser criada para armazenar os dados previstos pela regressão linear (ainda normalizados) e, em seguida, utilizar uma função de transformação inversa que retornará os valores das variáveis normalizadas para o intervalo inicial (armazenado com o modelo $MinMaxScaler()$).

```
#Predição dos dados para C1
resultadosPred_norm_C1 = modelo_C1.predict(dadosTest_norm_C1)

resultados_pred_C1 = scalerResultados_C1.inverse_transform(
    resultadosPred_norm_C1)

#Predição dos dados para C2
resultadosPred_norm_C2 = modelo_C2.predict(dadosTest_norm_C2)

resultados_pred_C2 = scalerResultados_C2.inverse_transform(
    resultadosPred_norm_C2)
```

Para mensurar a diferença média com relação aos dados já existentes, usa-se a função *mean_squared_error*, que calculará o erro médio quadrático entre os resultados (saídas) dos dados de teste e os que foram previstos pela regressão linear.

```
#Calculo do Erro Médio Quadrático para C1
erro_C1 = mean_squared_error(resultados_test_C1, resultados_pred_C1)
print("O erro médio quadrático de C1 é: %.7f" % np.sqrt(erro_C1))
# erro médio quadrático gerado  para C1 nessa instância de testes:
    0.0058707

#Calculo do Erro Médio Quadrático para C2
erro_C2 = mean_squared_error(resultados_test_C2, resultados_pred_C2)
print("O erro médio quadrático de C2 é: %.7f" % np.sqrt(erro_C2))
# erro médio quadrático gerado para C2 nessa instância de testes: 0.0022257
```

Outro método utilizado para mensurar o desempenho da regressão linear é o cálculo da medida R^2.

```
#Calculo da medida R2 para C1
medidaR2_C1 = r2_score(resultados_test_C1, resultados_pred_C1)
print("A medida R2 para C1 é: %.5f" % medidaR2_C1)
# medida R2 gerada para C1 nessa instância de testes: 0.84228

#Calculo da medida R2 para C2
medidaR2_C2 = r2_score(resultados_test_C2, resultados_pred_C2)
print("A medida R2 para C2 é: %.5f" % medidaR2_C2)
# medida R2 gerada para C1 nessa instância de testes: 0.91222
```

Ela é responsável por relacionar a escolha aleatória (50% de chance para a saída) com a acurácia da regressão linear.

Para visualizar a previsão das duas variáveis de saída ($C1$ e $C2$), plotam-se as variáveis correspondentes aos resultados previstos com os resultados do conjunto de testes (usa-se o conjunto de testes pois a regressão linear foi criada com base nos dados disponíveis no conjunto dos dados de treino).

```
# Plot do gráfico comparando as informaç es  do resultado 1 (#GT Compressor
    Decay State Coefficient )

plt.figure(figsize=(10, 10))

plt.scatter(resultados_pred_C1, resultados_test_C1, color = "#2E6AB8")
plt.scatter(resultados_pred_C2, resultados_test_C2, color = "#2EFAF8")

plt.plot(resultados_pred_C1, resultados_pred_C1, color="#1B3E6B")
plt.grid()
```

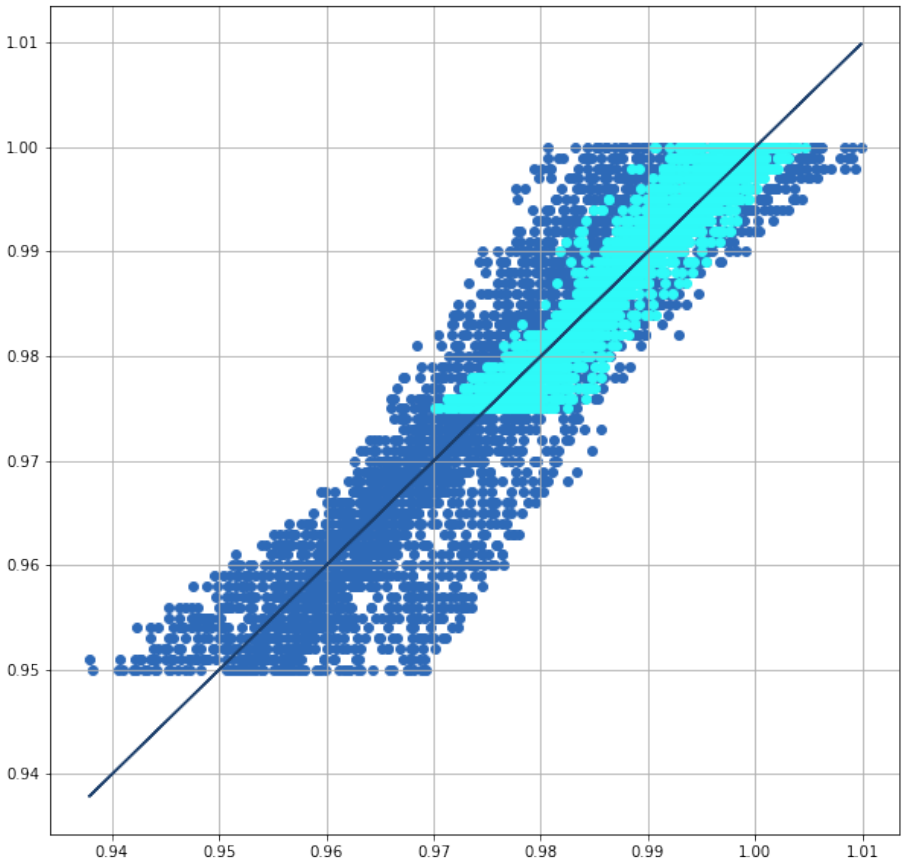

Figura 2 – Gráfico da saída geral

A reta na diagonal da Figura 2 representa uma função em que $y = x$, e na relação em que o eixo vertical corresponde aos dados previstos e o eixo horizontal aos dados do conjunto de teste, dados espalhados seguindo exatamente uma função $y = x$ representaria 100% de acerto. Os dados em azul escuro correspondem à saída $C1$, e os em azul claro representam a saída $C2$.

Os dados de $C2$ estão corretamente dispostos em um intervalo menor, pois os registros para o coeficiente de decaimento da turbina foram medidos em um domínio menor.

3 RESULTADOS E DISCUSSÕES

Para analisar o desempenho do modelo criado (com regressão linear), deve-se obter os dados sobre o erro médio quadrático e medida R^2.

3.1 ANÁLISE DA PREVISÃO PARA O COEFICIENTE DE DECAIMENTO DO COMPRESSOR

Para a primeira saída ($C1$), o erro médio quadrático foi de 0.0058707 e a medida R^2 foi de 0.84228. Considerando que o erro médio quadrático pode ser interpretado como a diferença entre o valor previsto e o parâmetro já existente, deve-se considerar que a escala (domínio) de existência para as saídas é considerado preciso até a faixa de 10^{-2}. Assim, o erro médio quadrático obtido com a regressão linear pode ser considerado bom, pois indica que as diferenças entre os valores são, em média, pouco impactantes para o resultado (numericamente) por indicarem variações na faixa de 10^{-3}.

Sabendo que a medida R^2 deve ser maior que 0.5 (para ser considerada melhor que um palpite aleatório) e menor que 1.0 (limite da medida), os 84% obtidos pela regressão podem ser considerados como satisfatórios.

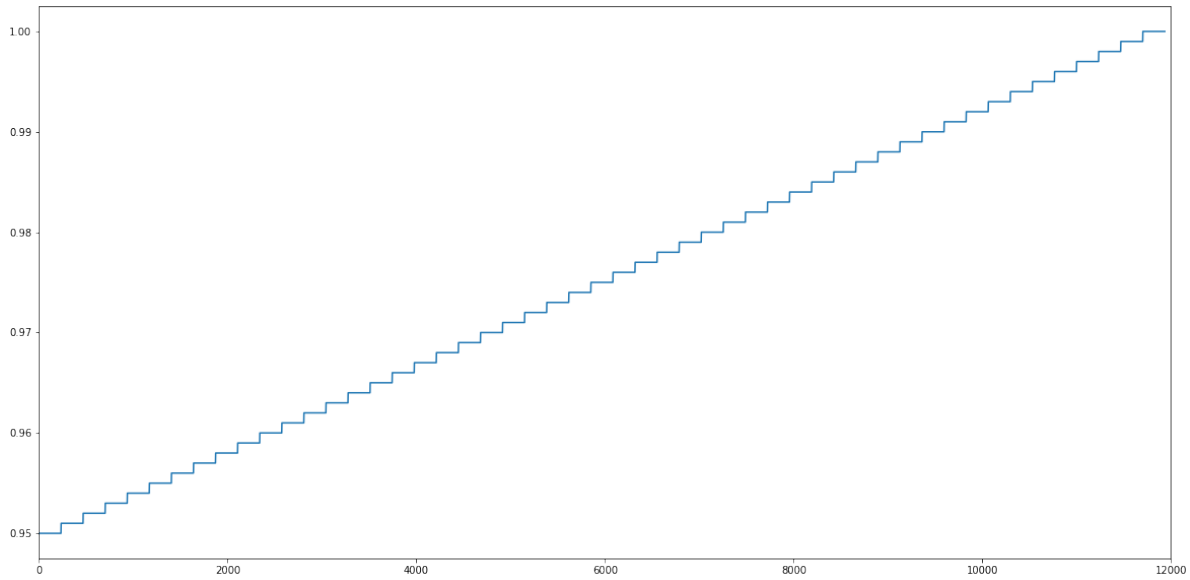

Figura 3 – Gráfico com os valores de C1 obtidos no conjunto de dados

Analisando a Figura 3 é possível perceber que ela representa uma excelente aproximação de reta. Se o ruído for desconsiderado, a taxa de crescimento seria muito bem interpretada como constante. Também graças à isso, os valores do erro médio quadrático e medida R^2 são menos satisfatórios que os que podem ser verificados a partir de uma análise do segundo caso (para $C2$). Isso acontece pois o valor da média já pode ser considerado como válido por se tratar de uma reta (e a regressão linear também formularia uma reta).

3.2 ANÁLISE DA PREVISÃO PARA O COEFICIENTE DE DECAIMENTO DA TURBINA

Para a segunda saída ($C2$), o erro médio quadrático foi de 0.0058707 e a medida R^2 foi de 0.84228.

Seguindo a mesma justificativa dada para o para a dimensão do em $C1$, o ero médio quadrático obtido com a regressão linear pode ser considerado bom. Para a medida R^2

Sabendo que a medida R^2 deve ser maior que 0.5 (para ser considerada melhor que um palpite aleatório) e menor que 1.0 (limite da medida), os 84% obtidos pela regressão podem ser considerados como satisfatórios.

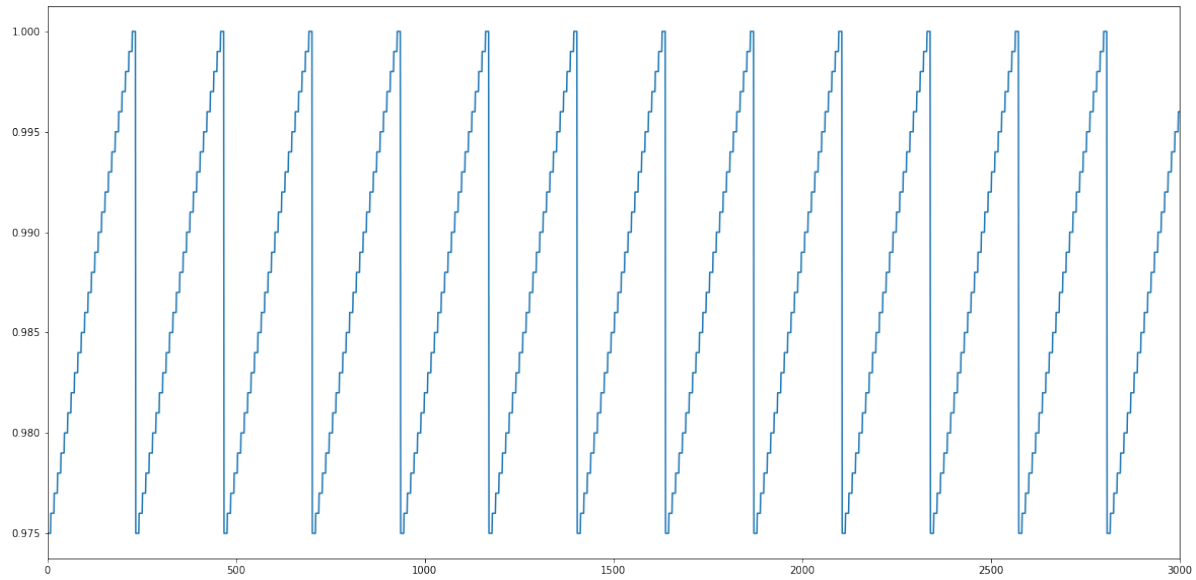

Figura 4 – Gráfico com os valores de C2 obtidos no conjunto de dados

Analisando a Figura 4 é possível perceber que ela representa uma função periódica. A medida R^2 foi ainda melhor para esse segundo caso. Isso acontece pois, diferente do formato da função de saída para $C1$, em $C2$ a média não apresenta um erro baixo. Para esse caso, a média representaria (aproximadamente) uma reta que passa no meio do gráfico. Como a medida R^2 é uma comparação com essa média, a regressão criada foi ainda melhor que esse valor médio. Isso é válido pois um valor médio não é suficiente para aproximar todos os valores e ainda há de se considerar a inclinação na subida para cada período de repetição da função de saída $C2$.

Já sobre o erro médio quadrático, ele ainda pode ser considerado satisfatório, mesmo sendo maior do que o erro encontrado para a primeira variável ($C1$). Essa diferença entre o erro encontrado para as duas saídas deve-se ao formato da função de saída.

Como uma regressão linear formula uma reta e a função do coeficiente de decaimento do motor também pode ser aproximada de uma reta (quando desconsidera-se o ruído),

é compreensível que a diferença entre os valores previstos sejam menores do que em um caso em que a comparação seja feita entre uma função periódica (função do coeficiente de decaimento da turbina) e uma reta (regressão linear).

4 CONCLUSÃO

Baseando-se nos resultados e valores evidenciados pelos gráficos plotados, é possível afirmar que a regressão linar criada pode ser considerada satisfatória. Mesmo com uma pequena variação entre as regressões para as saídas $C1$ e $C2$, as duas apresentaram valores com pouco desvio, realizando a avaliação com o cálculo do erro médio quadrático e da medida R^2.

APÊNDICES

APÊNDICE A – CÓDIGO FONTE

O código fonte também está disponível em um repositório online (*GitHub*).

```
# Importaçao das bibliotecas

import numpy as np #operaç es  matemátcas
import matplotlib.pyplot as plt #plot de gráficos

import pandas as pd # manipulaçao de arquivos (csv, txt)

from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_squared_error

from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LinearRegression

titulos_tabela = np.array([
    "lp", #Lever Position
    "v", #Ship Speed
    "GTT", #Gas Turbine Shaft Torque
    "GTn", #Gas Turbine Rate of Revolutions
    "GGn", #Gas Generator Rate of Revolutions
    "Ts", #Starboard Propeller Torque
    "Tp", #Port Propeller Torque
    "T48", #HP Turbine Exit Temperature
    "T1", #GT Compressor Inlet air Pressure
    "T2", #GT Compressor Outlet air Temperature
    "P48", #HP Turbine Exhaust Gas Pressure
    "P1", #GT Compressor Inlet Air Pressure
    "P2", #GT Compressor Outlet Air Pressure
    "Pexh", #Gas Turbine Exhaust Gas Pressure
    "TIC", #Turbine Injecton Control
    "mf", #Fuel Flow
    "C1", #GT Compressor Decay State Coefficient
    "C2"]) #GT Turbine Decay State Coefficient

dataframe = pd.read_csv(
    "data.txt",
    sep="   ",
    header=None,
    names=titulos_tabela)

    dados = dataframe.iloc[:, :16].values
#Obtençao dos dados do compressor na saída C1
resultados_C1 = dataframe.iloc[:, 16].values

```

```
44 #Obtençao dos dados da turbina na saída C2
45 resultados_C2 = dataframe.iloc[:, 17].values
46
47 dados_train_C1, dados_test_C1, resultados_train_C1, resultados_test_C1 =
       train_test_split(dados, resultados_C1, test_size=0.33)
48 dados_train_C2, dados_test_C2, resultados_train_C2, resultados_test_C2 =
       train_test_split(dados, resultados_C2, test_size=0.33)
49
50 scalerDados_C1 = MinMaxScaler()
51 scalerDados_C1.fit(dados_train_C1)
52 dadosTrain_norm_C1 = scalerDados_C1.transform(dados_train_C1)
53 dadosTest_norm_C1 = scalerDados_C1.transform(dados_test_C1)
54
55 scalerResultados_C1 = MinMaxScaler()
56 scalerResultados_C1.fit(resultados_train_C1.reshape(-1, 1))
57 resultadosTrain_norm_C1 = scalerResultados_C1.transform(resultados_train_C1
       .reshape(-1, 1))
58 resultadosTest_norm_C1 = scalerResultados_C1.transform(resultados_test_C1.
       reshape(-1, 1))
59
60 scalerDados_C2 = MinMaxScaler()
61 scalerDados_C2.fit(dados_train_C2)
62 dadosTrain_norm_C2 = scalerDados_C2.transform(dados_train_C2)
63 dadosTest_norm_C2 = scalerDados_C2.transform(dados_test_C2)
64
65 scalerResultados_C2 = MinMaxScaler()
66 scalerResultados_C2.fit(resultados_train_C2.reshape(-1, 1))
67 resultadosTrain_norm_C2 = scalerResultados_C2.transform(resultados_train_C2
       .reshape(-1, 1))
68 resultadosTest_norm_C2 = scalerResultados_C2.transform(resultados_test_C2.
       reshape(-1, 1))
69
70 modelo_C1 = LinearRegression()
71 modelo_C1.fit(dadosTrain_norm_C1, resultadosTrain_norm_C1)
72
73 modelo_C2 = LinearRegression()
74 modelo_C2.fit(dadosTrain_norm_C2, resultadosTrain_norm_C2)
75
76 resultadosPred_norm_C1 = modelo_C1.predict(dadosTest_norm_C1)
77
78 resultadosPred_norm_C2 = modelo_C2.predict(dadosTest_norm_C2)
79
80 resultadosPred_norm_C1 = modelo_C1.predict(dadosTest_norm_C1)
81
82 resultados_pred_C1 = scalerResultados_C1.inverse_transform(
       resultadosPred_norm_C1)
83
84 resultadosPred_norm_C2 = modelo_C2.predict(dadosTest_norm_C2)
```

```
85
86  resultados_pred_C2 = scalerResultados_C2.inverse_transform(
        resultadosPred_norm_C2)
87
88  erro_C1 = mean_squared_error(resultados_test_C1, resultados_pred_C1)
89  print("O erro médio quadrático de C1 é: %.7f" % np.sqrt(erro_C1))
90  # erro médio quadrático gerado  para C1 nessa instância de testes:
        0.0058707
91
92  erro_C2 = mean_squared_error(resultados_test_C2, resultados_pred_C2)
93  print("O erro médio quadrático de C2 é: %.7f" % np.sqrt(erro_C2))
94  # erro médio quadrático gerado para C2 nessa instância de testes: 0.0022257
95
96  medidaR2_C1 = r2_score(resultados_test_C1, resultados_pred_C1)
97  print("A medida R2 para C1 é: %.5f" % medidaR2_C1)
98  # medida R2 gerada para C1 nessa instância de testes: 0.84228
99
100 medidaR2_C2 = r2_score(resultados_test_C2, resultados_pred_C2)
101 print("A medida R2 para C2 é: %.5f" % medidaR2_C2)
102 # medida R2 gerada para C1 nessa instância de testes: 0.91222
103
104 # Plot do gráfico comparando as informaçoes do resultado 1 (#GT Compressor
        Decay State Coefficient )
105
106 plt.figure(figsize=(10, 10))
107
108 plt.scatter(resultados_pred_C1, resultados_test_C1, color = "#2E6AB8")
109 plt.scatter(resultados_pred_C2, resultados_test_C2, color = "#2EFAF8")
110
111 plt.plot(resultados_pred_C1, resultados_pred_C1, color="#1B3E6B")
112 plt.grid()
113
114 dataframe['C1'].plot(figsize = (20, 10), xlim=[0, 12000])
115
116 dataframe['C2'].plot(figsize = (20, 10), xlim=[0, 3000])
```