UNIVERSIDADE FEDERAL DO CEARÁ - CAMPUS SOBRAL

CURSO ENGENHARIA DA COMPUTAÇÃO

DISCIPLINA: INTELIGÊNCIA COMPUTACIONAL

PROFESSOR: JARBAS JOACI DE MESQUITA SÁ JUNIOR

RELATÓRIO

Andressa Gomes Moreira – 402305

Sobral – CE
2020

1. Questão 01

As soluções referentes à questão 01 encontram-se no arquivo questao01.py. Dessa forma, para o desenvolvimento utilizou-se a linguagem Python, versão 3.7.3 e a IDE PyCharm 2018.3.1. A priori, importou-se as bibliotecas necessárias no decorrer do desenvolvimento: Numpy e Matplotlib. A base de dados "aerogerador.dat" foi carregada e dividida em duas colunas, na qual a primeira estaria relacionada à variável de entrada x (Velocidade do vento) e a segunda seria a variável dependente y (Potência gerada). Desse modo, sabe-se que a relação entre xi e y é caracterizada por um modelo matemático chamado equação de regressão. Primeiramente, tratou-se do modelo em que há apenas uma variável de saída e uma de entrada, chamada Regressão Simples.

Na regressão linear simples a relação entre a variável independente e dependente é definido por uma reta. Assim sendo, y pode ser descrita pelo modelo: $y = \beta0 + \beta1 * x$, na qual o intercepto (β0) e inclinação (β1) são constantes desconhecidas. Para definir as estimativas de β0 e β1:

```
# Para determinar os parâmetros desconhecidos B0 e B1 iremos precisar:
n = len(x)  # Quantidade de elementos da amostra => n = len(y)
sum_xy = 0  # Somatório xi*yi
sum_x = 0   # Somatório xi
sum_y = 0   # Somatório yi
sum_x2 = 0  # Somatório (xi**2)

# Realizando os somatórios:
for i in range(n):
    sum_xy += float(x[i])*float(y[i])
    sum_x += float(x[i])
    sum_y += float(y[i])
    sum_x2 += float(x[i])**2

# B1 e B0 são iguais a:
B1 = (sum_xy - ((1/n) * sum_y * sum_x)) / (sum_x2 - ((1/n) * (sum_x**2)))
B0 = ((1/n) * sum_y) - (B1 * ((1/n) * sum_x))
```

Figura 01 - Estimativas de β0 e β1. Fonte: Elaborada pelo autor.

Como resultado, achou-se do os valores para o intercepto β0 e para a inclinação β1, visto na figura 02.

```
B0 = -217.69027909515972 (Intercepto)
B1 = 56.44385544805463 (Inclinação)
```

Figura 02 - Resultado de β0 e β1. Fonte: Elaborada pelo autor.

Logo, sabendo os valores dos parâmetros β0 e β1 tornou-se possível determinar os valores para a reta de regressão descrita pelo modelo y = β0 + β1* x:

```
y_reta_aux = list()  # Exibe a lista com os valores da reta de regressão
linhas1 = list()     # Variável auxiliar para receber as linhas da matriz.
colunas1 = list()    # Variável auxiliar para receber as colunas da matriz.

# Portanto, a reta de regressão será: y_reta = B0 + B1 * x
for i in range(n):
    y_reta = B0 + (B1 * float(x[i]))
    y_reta_aux.append(y_reta)

    linhas1 = (float(x[i]), float(y_reta))
    colunas1.append(linhas1)
```

Figura 03 – Cálculo Regressão Linear. Fonte: Elaborada pelo autor.

Ademais, é necessário determinar os modelos de regressão polinomial de graus 2, 3, 4, 5 com parâmetros estimados pelo método dos mínimos quadrados. Para isso criou-se a matriz X correspondente a cada grau do polinômio e determinou a estimativa de quadrados mínimos de β da seguinte forma:

$$\hat{\beta} = (X^T X)^{-1} X^T y \quad (1)$$

Portanto, para cada matriz determinou-se a transposta e a inversa correspondente realizou-se operações de multiplicação para assim determinar a matriz dos coeficientes.

```
# Estimativa de quadrados mínimos de β:
def min_quadrados(grau):
    # Determina qual o grau do polinômio
    if grau == 2:
        X = matriz2
    elif grau == 3:
        X = matriz3
    elif grau == 4:
        X = matriz4
    elif grau == 5:
        X = matriz5

    # Transposta, multiplicação, inversa => (XT * X)^-1:
    XT = X.T                              # TRANSPOSTA
    multip = np.dot(XT, X)                # Multiplicação XT * X
    inversa = np.linalg.inv(multip)       # Inversa (XT * X)^-1

    # Logo,  estimativa de quadrados mínimos de β é igual a:
    multip1 = np.dot(inversa, XT)
    beta = np.dot(multip1, y_2)

    return beta
```

Figura 04 – Método dos mínimos quadrados

Em seguida, o modelo de regressão ajustado (preditor) foi calculado da seguinte forma:

$$\hat{y} = X\hat{\beta} \tag{2}$$

Por fim, determinou-se o coeficiente de determinação e o de determinação ajustada, usado para definir a adequação de um modelo de regressão, para cada grau do polinômio.

```
# Função para determinar os coeficientes de determinação
def coef_determinacao(y_aux, grau, SQe=0, Syy=0):
    p = grau + 1
    for i in range(n):
        SQe += ((float(y[i])) - (float(y_aux[i]))) ** 2
        Syy += ((float(y[i])) - media) ** 2


    # Resultado do coeficiente de determinação.
    R2 = 1 - (SQe / Syy)

    # Resultado do coeficiente de determinação ajustado.
    R2_aj = 1 - ((SQe / (n - p)) / (Syy / (n - 1)))

    return R2, R2_aj
```

Figura 05 – Coeficientes de Determinação

Portanto, plotou-se os gráficos para cada grau do polinômio e comparou-se os resultados, utilizando os parâmetros estimados pelo método dos mínimos quadrados e avaliou-se a qualidade de cada modelo pelos coeficiente de determinação e coeficientes de determinação ajustado. Dessa forma, as figuras seguintes exibem o gráfico para cada modelo, como também o resultado dos coeficientes.

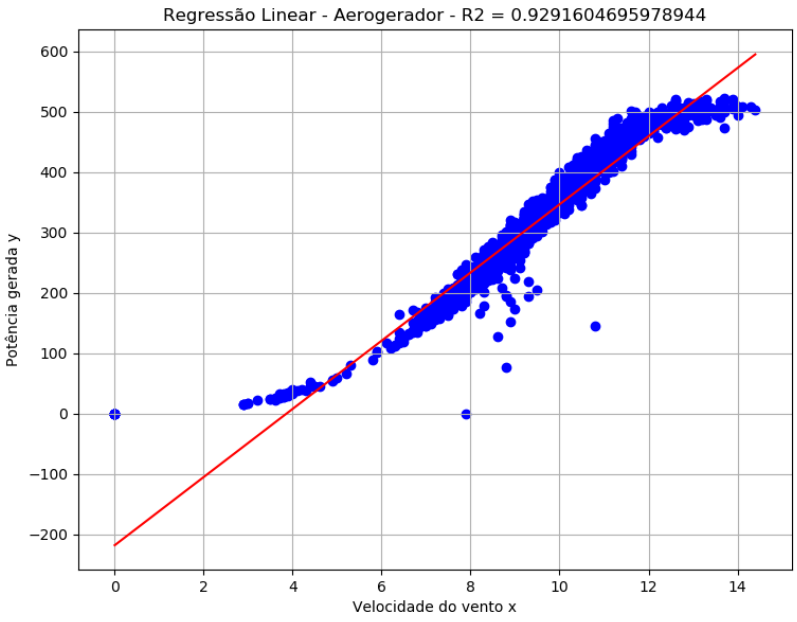

Figura 06 – Regressão Linear

É possível observar pela imagem e pelo valor do coeficiente de determinação que o modelo de regressão linear não é o mais adequado para esse problema, por tal motivo teve-se que aplicar o modelo polinomial de ordem 2, 3, 4 e 5, por meio da regressão múltipla.

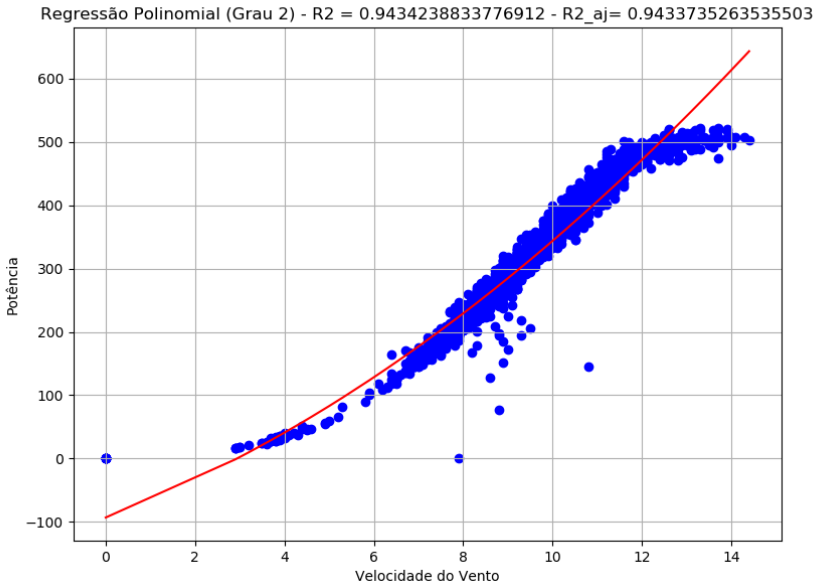

Figura 07 – Regressão Polinomial (Grau 2)

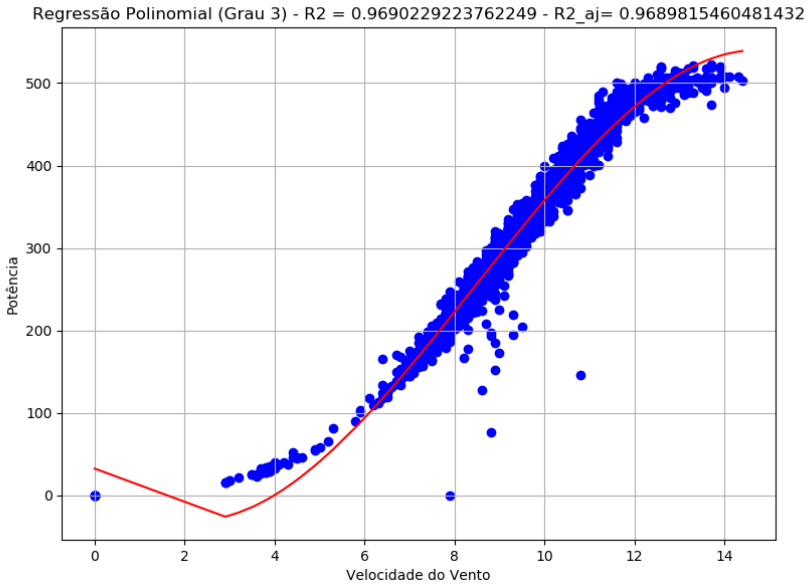

Figura 08 – Regressão Polinomial (Grau 3)

É possível observar pelas imagens e pelos coeficientes de determinação que a medida em que ocorre o aumento do grau da regressão polinomial o modelo se adequa aos dados, porém a partir do polinômio de grau 4 essa melhoria torna-se mais discreta.

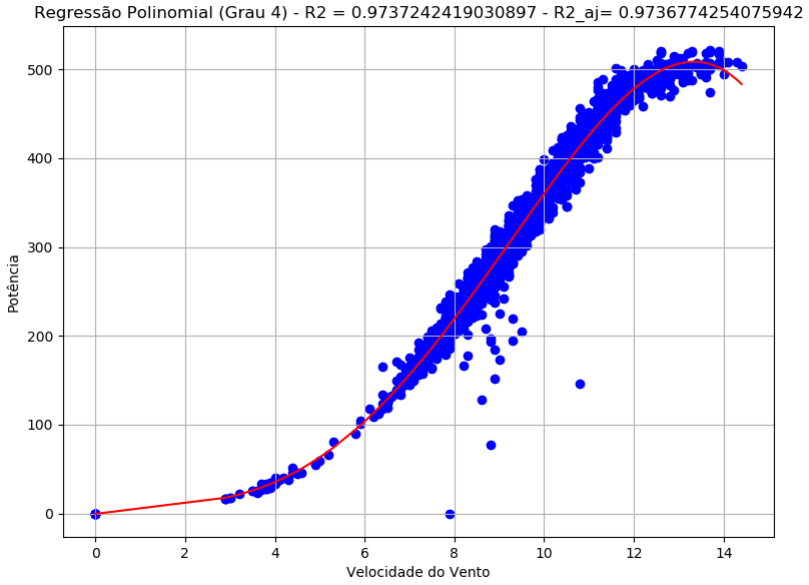

Figura 09 – Regressão Polinomial (Grau 4)

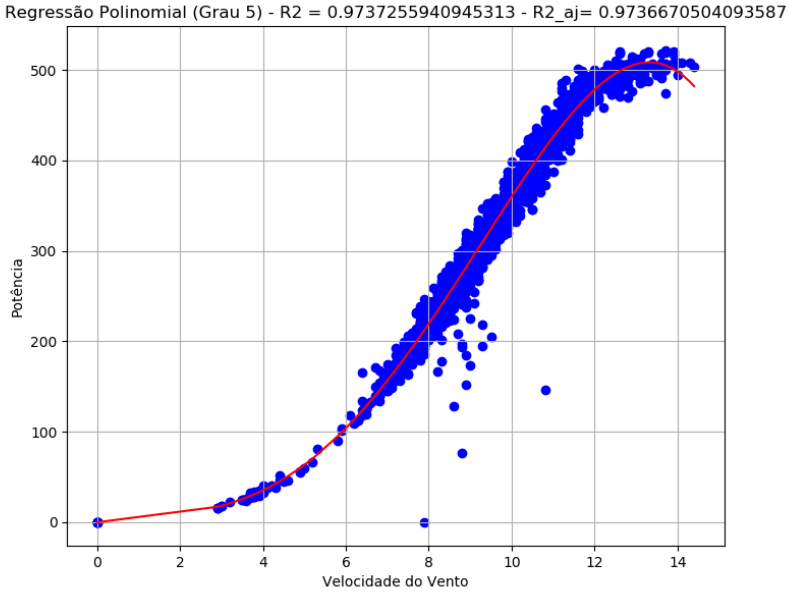

Figura 09 – Regressão Polinomial (Grau 5)

A primeira tabela exibe os resultados para a regressão linear (referentes aos valores vistos no gráfico 01).

Regressão Linear	$\widehat{\boldsymbol{\beta}}_{\boldsymbol{0}}$	$\widehat{\boldsymbol{\beta}}_{\boldsymbol{1}}$	$\boldsymbol{R^2}$
	-217.69027909515972	56.44385544805463	0.9291604695978944

Tabela 01 – Regressão Linear

A tabela a seguir faz a comparação entre os valores obtidos, exibindo os resultados dos coeficientes de determinação e coeficientes de determinação.

Regressão Polinomial (Grau)	R^2	R^2_{aj}
2	0.9434238833776912	0.9433735263535503
3	0.9690229223762249	0.9689815460481432
4	0.9737242419030897	0.9736774254075942
5	0.9737255940945313	0.9736670504093587

Tabela 02 – Coeficientes de determinação

E por fim, a próxima tabela exibe os valores para as estimativas de quadrados mínimos de β.

Regressão Polinomial (Grau)	**β**
2	[[-92.98003005][26.72314144][1.6931193]]
3	[[32.62351025][-58.76042398][15.05191299][-0.59240797]]
4	[[-0.39132613][10.37288673][-5.00359968][1.43389503][-0.06766974]]
5	[[-1.79826238e-01][8.16387611e+00][-3.93045527e+00][1.24622593e+00] [-5.37024696e-02][-3.75305437e-04]]

Tabela 03 – Estimativas de quadrados mínimos de β.

2. Questão 02

As soluções referentes à questão 02 encontram-se no arquivo questao02.py. A priori, organizou-se a base de dados, na qual os valores das variáveis de entrada (x1 e x2) foram armazenadas em matriz X e os valores da variável dependente (y) armazenados em uma matriz Y.

```
# Organizando a base de dados:
x = [[1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
     [122, 114, 86, 134, 146, 107, 68, 117, 71, 98],
     [139, 126, 90, 144, 163, 136, 61, 62, 41, 120]]

y = [0.115, 0.120, 0.105, 0.090, 0.100, 0.120, 0.105, 0.080, 0.100, 0.115]

# Ajustando a base de dados na forma matricial
for i in range(n):
    linhas_X = (1, float(x[1][i]), float(x[2][i]))
    colunas_X.append(linhas_X)

    linhas_y = [y[i]]
    colunas_y.append(linhas_y)

X = np.array(colunas_X)  # Matriz com valores das variáveis regressoras (x1 e x2)
Y = np.array(colunas_y)  # Matriz com valores da variável dependete (y)
```

Figura 10 – Organização da base de dados

Para determinar o modelo de regressão múltipla iniciou-se aplicando o método dos mínimos quadrados e determinando o modelo de regressão ajustado, como visto nas equações [1] e [2].

```
# Função que irá estipular a estimativa de quadrados mínimos β:
def min_quadrados():
    # Transposta, multiplicação, inversa => (XT * X)^-1:
    XT = X.T                              # TRANSPOSTA
    multip = np.dot(XT, X)                # Multiplicação (XT * X)
    inversa = np.linalg.inv(multip)       # Inversa (XT * X)^-1

    multip1 = np.dot(inversa, XT)         # Multiplicação ((XT * X)^-1 * XT
    beta = np.dot(multip1, Y)             # Resultado: β = (XT * X)^-1 * XT * y

    return beta                           # Retorna a estimativa de quadrados
                                          # mínimos β


# Resultado da Regressão
beta = min_quadrados()                  # Estimativa de quadrados mínimos de β
y_beta = np.dot(X, beta)                # Modelo de regressão ajustado
```

Figura 10 – Método dos mínimos quadrados e modelo de regressão ajustado

Em seguida, para determinar a adequação do modelo de regressão determinou-se os valores referentes ao coeficiente de determinação e o coeficiente de determinação ajustado.

```
# Variáveis que serão utilizadas no cálculo do coeficiente de determinação
SQe = 0             # Soma de quadrados dos resíduos
Syy = 0
media = sum(Y/n)    # Variável auxiliar para o cálculo de Syy

# Função para determinar os coeficientes de determinação
def coef_determinacao(y_aux, k, SQe=0, Syy=0):
    p = k + 1
    for i in range(n):
        SQe += ((float(Y[i])) - (float(y_aux[i]))) ** 2
        Syy += ((float(Y[i])) - media) ** 2

    # Resultado do coeficiente de determinação,
    R2 = 1 - (SQe / Syy)

    # Resultado do coeficiente de determinação ajustado.
    R2_aj = 1 - ((SQe / (n - p)) / (Syy / (n - 1)))

    return R2, R2_aj

R2, R2_aj = coef_determinacao(y_beta, 2)
```

Figura 11 – Coeficientes de determinação

Portanto, a tabela 04 exibe os valores para o método dos mínimos quadrados β, para o coeficiente de determinação e para o coeficiente de determinação ajustado. Desse modo, avaliando a qualidade do modelo pela métrica R2 é possível observar que a adequação do modelo não é ideal, pois os valores encontram-se distantes de 1.

β	R^2	R^2_{aj}
[[0.12737897][-0.00065924][0.00044084]]	[0.72388235]	[0.64499159]

Tabela 04 – Resultados.

Por fim, plotou-se o gráfico a partir dos valores determinados.

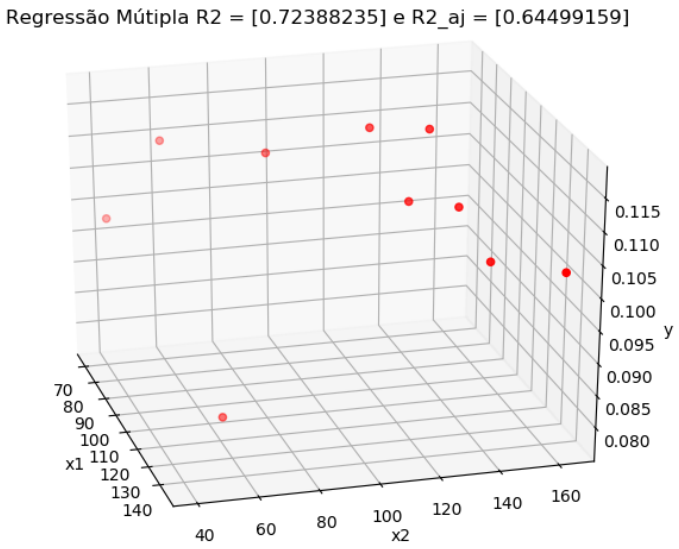

Figura 12 – Plotagem do gráfico