```
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
import pandas as pd
import matplotlib.pyplot as plt

temp = pd.read_csv('AMP_EsforcoCorrecDefeitos.csv')
x = temp['EST_HORAS_CORRECAO'].to_numpy()
print(x)
y = temp['REAL_HORAS_CORRECAO'].to_numpy()
print(y)
x = x.reshape((-1,1));
y = y.reshape((-1,1));

modelo1 = LinearRegression().fit(x, y)
print(f"b0: {modelo1.intercept_}")
print(f"b1: {modelo1.coef_}")

# Calcular R2
R2_m1 = modelo1.score(x,y)
print('R2_m1: {:.4f}'.format(R2_m1))

if (R2_m1 >= 0.92):
    print('Ajuste do modelo: excelente')
elif (R2_m1 >= 0.90):
    print('Ajuste do modelo: muito bom')
elif (R2_m1 >= 0.88):
    print('Ajuste do modelo: bom')
elif (R2_m1 >= 0.84):
    print('Ajuste do modelo: adequado')
else:
    print('Ajuste do modelo: não adequado')

# Realizar previsões
x_novo = np.array([25, 39, 46, 51, 60, 73]).reshape((-1, 1))
print(x_novo)
y_novo = modelo1.predict(x_novo)
print(y_novo)


# Plotar diagrama de dispersão e reta de regressão
fig, ax = plt.subplots()
ax.scatter(x, y);
x_ = np.arange(x.min(), x.max(), (x.max()-x.min())/1000).reshape((-1,
1))
y_ = modelo1.predict(x_)
ax.plot(x_,y_,color='red');

# Preparar a matriz x_
```

```
x_ = PolynomialFeatures(degree=2, include_bias=True).fit_transform(x)
print(x_)

modelo2 = LinearRegression(fit_intercept=False).fit(x_, y)
print(f"Coeficientes: {modelo2.coef_}")

# Calcular R2
R2_m2 = modelo2.score(x_, y)
print('R2_m2: {:.4f}'.format(R2_m2))

if (R2_m2 >= 0.92):
    print('Ajuste do modelo: excelente')
elif (R2_m2 >= 0.90):
    print('Ajuste do modelo: muito bom')
elif (R2_m2 >= 0.88):
    print('Ajuste do modelo: bom')
elif (R2_m2 >= 0.84):
    print('Ajuste do modelo: adequado')
else:
    print('Ajuste do modelo: não adequado')

# Realizar previsões
x_novo = np.array([25, 39, 46, 51, 60, 73]).reshape((-1, 1))
x_novo_ = PolynomialFeatures(degree=2,
include_bias=True).fit_transform(x_novo)
print(x_novo)
y_novo = modelo2.predict(x_novo_)
print(y_novo)

# Plotar diagrama de dispersão e reta de regressão
fig, ax = plt.subplots()
ax.scatter(x, y);
x_plot = np.arange(x.min(), x.max(), (x.max()-
x.min())/1000).reshape((-1, 1))
x_plot_ = PolynomialFeatures(degree=2,
include_bias=True).fit_transform(x_plot)
y_plot = modelo2.predict(x_plot_)
ax.plot(x_plot,y_plot,color='red');

# Preparar a matriz x_
x_ = PolynomialFeatures(degree=3, include_bias=True).fit_transform(x)
print(x_)

modelo3 = LinearRegression(fit_intercept=False).fit(x_, y)
print(f"Coeficientes: {modelo3.coef_}")

# Calcular R2
R2_m3 = modelo3.score(x_, y)
print('R2_m3: {:.4f}'.format(R2_m3))
```

```
if (R2_m3 >= 0.92):
    print('Ajuste do modelo: excelente')
elif (R2_m3 >= 0.90):
    print('Ajuste do modelo: muito bom')
elif (R2_m3 >= 0.88):
    print('Ajuste do modelo: bom')
elif (R2_m3 >= 0.84):
    print('Ajuste do modelo: adequado')
else:
    print('Ajuste do modelo: não adequado')

# Realizar previsões
x_novo = np.array([25, 39, 46, 51, 60, 73]).reshape((-1, 1))
x_novo_ = PolynomialFeatures(degree=3,
include_bias=True).fit_transform(x_novo)
print(x_novo)
y_novo = modelo3.predict(x_novo_)
print(y_novo)

# Plotar diagrama de dispersão e reta de regressão
fig, ax = plt.subplots()
ax.scatter(x, y);
x_plot = np.arange(x.min(), x.max(), (x.max()-
x.min())/1000).reshape((-1, 1))
x_plot_ = PolynomialFeatures(degree=3,
include_bias=True).fit_transform(x_plot)
y_plot = modelo3.predict(x_plot_)
ax.plot(x_plot,y_plot,color='red');

print("==-=-=-=-=-=-=-=-FORMATIVA==-=-=-==-=-==-=-=-=")
# Imprimir R2 para modelo 1
print(f'MODELO 1: {R2_m1}\n')

# Imprimir R2 para modelo 2
print(f'MODELO 2: {R2_m2}\n')

# Imprimir R2 para modelo 3
print(f'MODELO 3: {R2_m3}\n')

# Cria o vetor x_novo
x_novo = np.array([32, 46, 52, 57, 67, 73]).reshape((-1, 1))

# Usar Polinomial features para completar a matriz x-novo
x_novo = PolynomialFeatures(degree=3,
include_bias=True).fit_transform(x_novo)

# Mostrar x_novo
print('matriz x_no
```

```
print(x_novo)

# Usar método predict do melhor modelo para calcular as previsões
y_novo = modelo3.predict(x_novo)
# Mostrar y_novo
print(y_novo)
```

```
[ 3  7 10 11 13 18 23 26 33 42 48 56 58 60 66 70]
[ 4 12 18 23 27 32 35 30 26 32 36 44 60 72 86 97]
b0: [4.98276833]
b1: [[1.01888917]]
R2_m1: 0.8065
Ajuste do modelo: não adequado
[[25]
 [39]
 [46]
 [51]
 [60]
 [73]]
[[30.4549975 ]
 [44.71944583]
 [51.85167   ]
 [56.94611584]
 [66.11611834]
 [79.36167751]]
[[1.000e+00 3.000e+00 9.000e+00]
 [1.000e+00 7.000e+00 4.900e+01]
 [1.000e+00 1.000e+01 1.000e+02]
 [1.000e+00 1.100e+01 1.210e+02]
 [1.000e+00 1.300e+01 1.690e+02]
 [1.000e+00 1.800e+01 3.240e+02]
 [1.000e+00 2.300e+01 5.290e+02]
 [1.000e+00 2.600e+01 6.760e+02]
 [1.000e+00 3.300e+01 1.089e+03]
 [1.000e+00 4.200e+01 1.764e+03]
 [1.000e+00 4.800e+01 2.304e+03]
 [1.000e+00 5.600e+01 3.136e+03]
 [1.000e+00 5.800e+01 3.364e+03]
 [1.000e+00 6.000e+01 3.600e+03]
 [1.000e+00 6.600e+01 4.356e+03]
 [1.000e+00 7.000e+01 4.900e+03]]
Coeficientes: [[ 1.99456888e+01 -3.30550615e-01  1.86745381e-02]]
R2_m2: 0.8743
Ajuste do modelo: adequado
[[25]
 [39]
 [46]
 [51]
 [60]
 [73]]
```

```
[[23.35350973]
 [35.45818724]
 [44.25568309]
 [51.66008098]
 [67.34098898]
 [95.33210731]]
[[1.00000e+00 3.00000e+00 9.00000e+00 2.70000e+01]
 [1.00000e+00 7.00000e+00 4.90000e+01 3.43000e+02]
 [1.00000e+00 1.00000e+01 1.00000e+02 1.00000e+03]
 [1.00000e+00 1.10000e+01 1.21000e+02 1.33100e+03]
 [1.00000e+00 1.30000e+01 1.69000e+02 2.19700e+03]
 [1.00000e+00 1.80000e+01 3.24000e+02 5.83200e+03]
 [1.00000e+00 2.30000e+01 5.29000e+02 1.21670e+04]
 [1.00000e+00 2.60000e+01 6.76000e+02 1.75760e+04]
 [1.00000e+00 3.30000e+01 1.08900e+03 3.59370e+04]
 [1.00000e+00 4.20000e+01 1.76400e+03 7.40880e+04]
 [1.00000e+00 4.80000e+01 2.30400e+03 1.10592e+05]
 [1.00000e+00 5.60000e+01 3.13600e+03 1.75616e+05]
 [1.00000e+00 5.80000e+01 3.36400e+03 1.95112e+05]
 [1.00000e+00 6.00000e+01 3.60000e+03 2.16000e+05]
 [1.00000e+00 6.60000e+01 4.35600e+03 2.87496e+05]
 [1.00000e+00 7.00000e+01 4.90000e+03 3.43000e+05]]
Coeficientes: [[-4.53930055e+00  3.42826733e+00 -1.09374629e-01
1.17633013e-03]]
R2_m3: 0.9697
Ajuste do modelo: excelente
[[25]
 [39]
 [46]
 [51]
 [60]
 [73]]
[[ 31.18839797]
 [ 32.58304185]
 [ 36.22355153]
 [ 41.86029172]
 [ 61.49538341]
 [120.47923537]]
==-=-=-=-=-=-=-=-FORMATIVA==-=-=-==-=-==-=-=-=
MODELO 1: 0.8065164133431957

MODELO 2: 0.8742969188970658

MODELO 3: 0.9696890509047211

[[1.00000e+00 3.20000e+01 1.02400e+03 3.27680e+04]
 [1.00000e+00 4.60000e+01 2.11600e+03 9.73360e+04]
 [1.00000e+00 5.20000e+01 2.70400e+03 1.40608e+05]
 [1.00000e+00 5.70000e+01 3.24900e+03 1.85193e+05]
 [1.00000e+00 6.70000e+01 4.48900e+03 3.00763e+05]
```

```
 [1.00000e+00 7.30000e+01 5.32900e+03 3.89017e+05]]
[[ 31.7116198 ]
 [ 36.22355153]
 [ 43.38303111]
 [ 53.36187386]
 [ 87.96848041]
 [120.47923537]]
```