Solução Lista 04

Nome: Julia Xavier
E-mail: julia.xavier@aluno.ufabc.edu.br
Nome: Leonardo Bernardes Lério da Silva
E-mail: leonardo.lerio@aluno.ufabc.edu.br

01 de novembro, 2024

Exercício 01

```
import pandas as pd
import numpy as np
import re
from sklearn.linear_model import LassoCV
from sklearn.linear_model import RidgeCV
import matplotlib.pyplot as plt
from tqdm import tqdm
```

```
file_url = "https://drive.google.com/uc?export=download&id=1jiWcGsl_tbqK5F0ryUTq48kcDTKWTTuk"
df = pd.read_csv(file_url)
df.head()
```

```
##    Unnamed: 0               Name  Age  ...    RW   RWB    ST
## 0           0  Cristiano Ronaldo   32  ...  91.0  66.0  92.0
## 1           1           L. Messi   30  ...  91.0  62.0  88.0
## 2           2             Neymar   25  ...  89.0  64.0  84.0
## 3           3          L. Suárez   30  ...  87.0  68.0  88.0
## 4           4           M. Neuer   31  ...   NaN   NaN   NaN
##
## [5 rows x 75 columns]
```

```
selected_columns = ['Age', 'Overall', 'Potential', 'Wage', 'Special', 'Acceleration', 'Aggression', 'A
                    'Balance', 'Ball control', 'Composure', 'Crossing', 'Curve', 'Dribbling', 'Finishin
                    'Positioning', 'Stamina', 'Interceptions', 'Strength', 'Vision', 'Volleys', 'Jumpin
                    'Penalties', 'Shot power', 'Sprint speed', 'Heading accuracy', 'Long passing', 'Sho

df = df[selected_columns].copy()
df.head()
```

```
##    Age  Overall  Potential  ...  Heading accuracy  Long passing  Short passing
## 0   32       94         94  ...                88            77             83
## 1   30       93         93  ...                71            87             88
## 2   25       92         94  ...                62            75             81
## 3   30       92         92  ...                77            64             83
```

```
## 4    31         92          92  ...                 25              59              55
##
## [5 rows x 28 columns]
```

```python
df["Wage"] = df["Wage"].apply(lambda x: re.sub(r"\D", "", str(x)))
```

```python
def trata_modificadores(valor):
  if not isinstance(valor, str):
    valor = str(valor)

  if "-" in valor:
    valor = valor.split("-")
    valor = int(valor[0]) - int(valor[1])
  elif "+" in valor:
    valor = valor.split("+")
    valor =  int(valor[0])  +  int(valor[1])

  if not isinstance(valor, int):
    valor = int(valor)
  return valor
```

```python
for column in tqdm(selected_columns):
  df[column] = df.apply(lambda x: trata_modificadores(x[column]), axis=1)
```

```
##   0%|          | 0/28 [00:00<?, ?it/s]  7%|7         | 2/28 [00:00<00:01, 14.70it/s] 14%|#4        |
```

```python
df = df.astype(int)
df = df.dropna()
```

```python
X = df.drop("Wage", axis=1)
y = df["Wage"]
```

```python
lasso = LassoCV(cv=10)
lasso.fit(X, y)
```

```
## LassoCV(cv=10)
```

```python
melhor_lambda = lasso.alpha_
melhor_lambda
```

```
## 2.307721132735995
```

```python
lasso_final = LassoCV(cv=10, alphas=[melhor_lambda])
lasso_final.fit(X, y)
```

```
## LassoCV(alphas=[2.307721132735995], cv=10)
```

```
coef = pd.Series(lasso_final.coef_, index=X.columns)
coef
```

```
## Age                  -0.000000
## Overall               1.375303
## Potential             0.764636
## Special              -0.001090
## Acceleration         -0.000000
## Aggression           -0.015404
## Agility              -0.021620
## Balance              -0.000000
## Ball control         -0.000000
## Composure             0.043667
## Crossing             -0.000000
## Curve                -0.000000
## Dribbling            -0.000000
## Finishing             0.000000
## Positioning           0.000000
## Stamina              -0.011703
## Interceptions        -0.000000
## Strength             -0.000000
## Vision                0.022117
## Volleys               0.067780
## Jumping               0.016544
## Penalties             0.020043
## Shot power           -0.010764
## Sprint speed         -0.009136
## Heading accuracy     -0.000000
## Long passing         -0.000000
## Short passing        -0.000000
## dtype: float64
```

```
sel_var = coef[coef != 0].index.tolist()
sel_var
```

```
## ['Overall', 'Potential', 'Special', 'Aggression', 'Agility', 'Composure', 'Stamina', 'Vision', 'Voll
```

```
df.info()
```

```
## <class 'pandas.core.frame.DataFrame'>
## RangeIndex: 17981 entries, 0 to 17980
## Data columns (total 28 columns):
##  #   Column            Non-Null Count  Dtype
## ---  ------            --------------  -----
##  0   Age               17981 non-null  int32
##  1   Overall           17981 non-null  int32
##  2   Potential         17981 non-null  int32
##  3   Wage              17981 non-null  int32
##  4   Special           17981 non-null  int32
##  5   Acceleration      17981 non-null  int32
##  6   Aggression        17981 non-null  int32
##  7   Agility           17981 non-null  int32
```

```
## 8   Balance          17981 non-null  int32
## 9   Ball control     17981 non-null  int32
## 10  Composure        17981 non-null  int32
## 11  Crossing         17981 non-null  int32
## 12  Curve            17981 non-null  int32
## 13  Dribbling        17981 non-null  int32
## 14  Finishing        17981 non-null  int32
## 15  Positioning      17981 non-null  int32
## 16  Stamina          17981 non-null  int32
## 17  Interceptions    17981 non-null  int32
## 18  Strength         17981 non-null  int32
## 19  Vision           17981 non-null  int32
## 20  Volleys          17981 non-null  int32
## 21  Jumping          17981 non-null  int32
## 22  Penalties        17981 non-null  int32
## 23  Shot power       17981 non-null  int32
## 24  Sprint speed     17981 non-null  int32
## 25  Heading accuracy 17981 non-null  int32
## 26  Long passing     17981 non-null  int32
## 27  Short passing    17981 non-null  int32
## dtypes: int32(28)
## memory usage: 1.9 MB
```

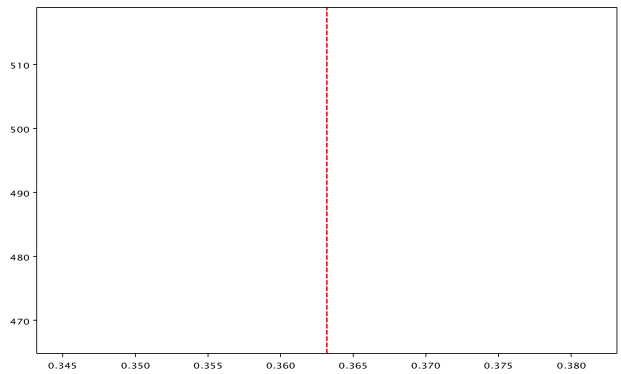
```
mse_values = []
for alpha in lasso_final.alphas_:
    lasso = LassoCV(cv=10, alphas=[alpha])
    lasso.fit(X, y)
    mse_values.append(np.mean(lasso.mse_path_, axis=0))
```

```
## LassoCV(alphas=[2.307721132735995], cv=10)
```

```
mse_values
```

```
## [491.8692567659583]
```

```
plt.figure(figsize=(10, 6))
plt.plot(np.log10(lasso_final.alphas_), mse_values, ':')
plt.axvline(np.log10(melhor_lambda), linestyle='--', color='r',
            label='Melhor Lambda: {:.4f}'.format(np.log10(melhor_lambda)))
plt.show()
```



Exercício 02

```
df_ridge = df.copy()
df_ridge.head()
```

```
##    Age  Overall  Potential  ...  Heading accuracy  Long passing  Short passing
## 0   32       94         94  ...                88            77             83
## 1   30       93         93  ...                71            87             88
## 2   25       92         94  ...                62            75             81
## 3   30       92         92  ...                77            64             83
## 4   31       92         92  ...                25            59             55
##
## [5 rows x 28 columns]
```

```
X = df.drop("Wage", axis=1)
y = df["Wage"]
```

```
ridge = RidgeCV(cv=10)
ridge.fit(X, y)
```

```
## RidgeCV(cv=10)
```

```
melhor_alpha = ridge.alpha_
melhor_alpha
```

```
## 10.0
```

```
ridge_final = RidgeCV(cv=10, alphas=[melhor_alpha])
ridge_final.fit(X, y)
```

```
## RidgeCV(alphas=[10.0], cv=10)
```

```
coef = pd.Series(ridge_final.coef_, index=X.columns)
coef
```

```
## Age                  -0.339056
## Overall               1.698431
## Potential             0.591346
## Special              -0.010173
## Acceleration         -0.015879
## Aggression           -0.013098
## Agility              -0.047591
## Balance               0.024752
## Ball control         -0.109887
## Composure             0.087602
## Crossing              0.024616
## Curve                -0.002092
## Dribbling             0.042050
## Finishing             0.011187
## Positioning           0.051796
## Stamina              -0.019561
## Interceptions         0.043924
## Strength             -0.017182
## Vision                0.048548
## Volleys               0.104411
## Jumping               0.056939
## Penalties             0.065561
## Shot power           -0.047438
## Sprint speed         -0.019061
## Heading accuracy      0.009730
## Long passing          0.041981
## Short passing        -0.043940
## dtype: float64
```

```
sel_vars = coef[coef != 0].index.tolist()
sel_vars
```

```
## ['Age', 'Overall', 'Potential', 'Special', 'Acceleration', 'Aggression', 'Agility', 'Balance', 'Ball
```

```
alphas = np.logspace(-4, 2, num=100)
coef_path = []
for alpha in alphas:
    ridge = RidgeCV(cv=10, alphas=[alpha])
    ridge.fit(X, y)
    coef_path.append(ridge.coef_)
```

```
## RidgeCV(alphas=[0.0001], cv=10)
## RidgeCV(alphas=[0.00011497569953977356], cv=10)
## RidgeCV(alphas=[0.00013219411484660288], cv=10)
## RidgeCV(alphas=[0.0001519911082952933], cv=10)
## RidgeCV(alphas=[0.0001747528400007683], cv=10)
## RidgeCV(alphas=[0.00020092330025650479], cv=10)
## RidgeCV(alphas=[0.00023101297000831605], cv=10)
## RidgeCV(alphas=[0.00026560877829466864], cv=10)
```

```
## RidgeCV(alphas=[0.0003053855508833416], cv=10)
## RidgeCV(alphas=[0.0003511191734215131], cv=10)
## RidgeCV(alphas=[0.0004037017258596554], cv=10)
## RidgeCV(alphas=[0.0004641588833612782], cv=10)
## RidgeCV(alphas=[0.0005336699231206312], cv=10)
## RidgeCV(alphas=[0.0006135907273413176], cv=10)
## RidgeCV(alphas=[0.0007054802310718645], cv=10)
## RidgeCV(alphas=[0.0008111308307896872], cv=10)
## RidgeCV(alphas=[0.0009326033468832199], cv=10)
## RidgeCV(alphas=[0.0010722672220103231], cv=10)
## RidgeCV(alphas=[0.0012328467394420659], cv=10)
## RidgeCV(alphas=[0.0014174741629268048], cv=10)
## RidgeCV(alphas=[0.0016297508346206436], cv=10)
## RidgeCV(alphas=[0.001873817422860383], cv=10)
## RidgeCV(alphas=[0.0021544346900318843], cv=10)
## RidgeCV(alphas=[0.0024770763559917113], cv=10)
## RidgeCV(alphas=[0.002848035868435802], cv=10)
## RidgeCV(alphas=[0.0032745491628777285], cv=10)
## RidgeCV(alphas=[0.0037649358067924675], cv=10)
## RidgeCV(alphas=[0.004328761281083062], cv=10)
## RidgeCV(alphas=[0.004977023564332114], cv=10)
## RidgeCV(alphas=[0.00572236765935022], cv=10)
## RidgeCV(alphas=[0.006579332246575682], cv=10)
## RidgeCV(alphas=[0.007564633275546291], cv=10)
## RidgeCV(alphas=[0.008697490026177835], cv=10)
## RidgeCV(alphas=[0.01], cv=10)
## RidgeCV(alphas=[0.011497569953977356], cv=10)
## RidgeCV(alphas=[0.013219411484660288], cv=10)
## RidgeCV(alphas=[0.01519911082952933], cv=10)
## RidgeCV(alphas=[0.01747528400007685], cv=10)
## RidgeCV(alphas=[0.02009233002565048], cv=10)
## RidgeCV(alphas=[0.023101297000831605], cv=10)
## RidgeCV(alphas=[0.026560877829466867], cv=10)
## RidgeCV(alphas=[0.030538555088334154], cv=10)
## RidgeCV(alphas=[0.03511191734215131], cv=10)
## RidgeCV(alphas=[0.040370172585965536], cv=10)
## RidgeCV(alphas=[0.04641588833612782], cv=10)
## RidgeCV(alphas=[0.05336699231206313], cv=10)
## RidgeCV(alphas=[0.061359072734131756], cv=10)
## RidgeCV(alphas=[0.07054802310718646], cv=10)
## RidgeCV(alphas=[0.08111308307896872], cv=10)
## RidgeCV(alphas=[0.093260334688322], cv=10)
## RidgeCV(alphas=[0.10722672220103231], cv=10)
## RidgeCV(alphas=[0.12328467394420659], cv=10)
## RidgeCV(alphas=[0.14174741629268048], cv=10)
## RidgeCV(alphas=[0.16297508346206452], cv=10)
## RidgeCV(alphas=[0.1873817422860385], cv=10)
## RidgeCV(alphas=[0.21544346900318845], cv=10)
## RidgeCV(alphas=[0.24770763559917114], cv=10)
## RidgeCV(alphas=[0.2848035868435802], cv=10)
## RidgeCV(alphas=[0.32745491628777285], cv=10)
## RidgeCV(alphas=[0.37649358067924676], cv=10)
## RidgeCV(alphas=[0.43287612810830617], cv=10)
## RidgeCV(alphas=[0.49770235643321137], cv=10)
```

```
## RidgeCV(alphas=[0.572236765935022], cv=10)
## RidgeCV(alphas=[0.6579332246575682], cv=10)
## RidgeCV(alphas=[0.7564633275546291], cv=10)
## RidgeCV(alphas=[0.8697490026177834], cv=10)
## RidgeCV(alphas=[1.0], cv=10)
## RidgeCV(alphas=[1.1497569953977356], cv=10)
## RidgeCV(alphas=[1.3219411484660286], cv=10)
## RidgeCV(alphas=[1.5199110829529332], cv=10)
## RidgeCV(alphas=[1.747528400007683], cv=10)
## RidgeCV(alphas=[2.009233002565046], cv=10)
## RidgeCV(alphas=[2.310129700083158], cv=10)
## RidgeCV(alphas=[2.6560877829466896], cv=10)
## RidgeCV(alphas=[3.0538555088334185], cv=10)
## RidgeCV(alphas=[3.5111917342151346], cv=10)
## RidgeCV(alphas=[4.037017258596558], cv=10)
## RidgeCV(alphas=[4.641588833612782], cv=10)
## RidgeCV(alphas=[5.336699231206313], cv=10)
## RidgeCV(alphas=[6.135907273413176], cv=10)
## RidgeCV(alphas=[7.054802310718645], cv=10)
## RidgeCV(alphas=[8.111308307896872], cv=10)
## RidgeCV(alphas=[9.326033468832199], cv=10)
## RidgeCV(alphas=[10.722672220103231], cv=10)
## RidgeCV(alphas=[12.32846739442066], cv=10)
## RidgeCV(alphas=[14.174741629268048], cv=10)
## RidgeCV(alphas=[16.297508346206435], cv=10)
## RidgeCV(alphas=[18.73817422860383], cv=10)
## RidgeCV(alphas=[21.544346900318867], cv=10)
## RidgeCV(alphas=[24.77076355991714], cv=10)
## RidgeCV(alphas=[28.48035868435805], cv=10)
## RidgeCV(alphas=[32.745491628777316], cv=10)
## RidgeCV(alphas=[37.649358067924716], cv=10)
## RidgeCV(alphas=[43.287612810830616], cv=10)
## RidgeCV(alphas=[49.770235643321136], cv=10)
## RidgeCV(alphas=[57.223676593502205], cv=10)
## RidgeCV(alphas=[65.79332246575683], cv=10)
## RidgeCV(alphas=[75.64633275546291], cv=10)
## RidgeCV(alphas=[86.97490026177834], cv=10)
## RidgeCV(alphas=[100.0], cv=10)
```

```
alphas
```

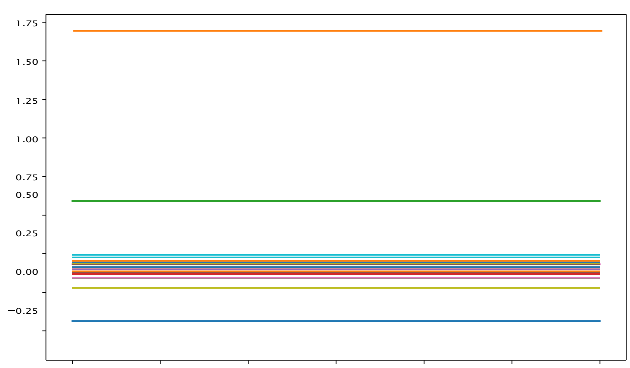
```
## array([1.00000000e-04, 1.14975700e-04, 1.32194115e-04, 1.51991108e-04,
##        1.74752840e-04, 2.00923300e-04, 2.31012970e-04, 2.65608778e-04,
##        3.05385551e-04, 3.51119173e-04, 4.03701726e-04, 4.64158883e-04,
##        5.33669923e-04, 6.13590727e-04, 7.05480231e-04, 8.11130831e-04,
##        9.32603347e-04, 1.07226722e-03, 1.23284674e-03, 1.41747416e-03,
##        1.62975083e-03, 1.87381742e-03, 2.15443469e-03, 2.47707636e-03,
##        2.84803587e-03, 3.27454916e-03, 3.76493581e-03, 4.32876128e-03,
##        4.97702356e-03, 5.72236766e-03, 6.57933225e-03, 7.56463328e-03,
##        8.69749003e-03, 1.00000000e-02, 1.14975700e-02, 1.32194115e-02,
##        1.51991108e-02, 1.74752840e-02, 2.00923300e-02, 2.31012970e-02,
##        2.65608778e-02, 3.05385551e-02, 3.51119173e-02, 4.03701726e-02,
##   4.64158883e-02, 5.33669923e-02, 6.13590727e-02, 7.05480231e-02,
```

```
##        8.11130831e-02, 9.32603347e-02, 1.07226722e-01, 1.23284674e-01,
##        1.41747416e-01, 1.62975083e-01, 1.87381742e-01, 2.15443469e-01,
##        2.47707636e-01, 2.84803587e-01, 3.27454916e-01, 3.76493581e-01,
##        4.32876128e-01, 4.97702356e-01, 5.72236766e-01, 6.57933225e-01,
##        7.56463328e-01, 8.69749003e-01, 1.00000000e+00, 1.14975700e+00,
##        1.32194115e+00, 1.51991108e+00, 1.74752840e+00, 2.00923300e+00,
##        2.31012970e+00, 2.65608778e+00, 3.05385551e+00, 3.51119173e+00,
##        4.03701726e+00, 4.64158883e+00, 5.33669923e+00, 6.13590727e+00,
##        7.05480231e+00, 8.11130831e+00, 9.32603347e+00, 1.07226722e+01,
##        1.23284674e+01, 1.41747416e+01, 1.62975083e+01, 1.87381742e+01,
##        2.15443469e+01, 2.47707636e+01, 2.84803587e+01, 3.27454916e+01,
##        3.76493581e+01, 4.32876128e+01, 4.97702356e+01, 5.72236766e+01,
##        6.57933225e+01, 7.56463328e+01, 8.69749003e+01, 1.00000000e+02])
```

```
coef_path = np.array(coef_path)
coef_path
```

```
## array([[-0.33918812,  1.69861634,  0.59123188, ...,  0.00973075,
##          0.04199479, -0.04394393],
##        [-0.33918812,  1.69861634,  0.59123188, ...,  0.00973075,
##          0.04199479, -0.04394393],
##        [-0.33918812,  1.69861634,  0.59123188, ...,  0.00973075,
##          0.04199479, -0.04394393],
##        ...,
##        [-0.33818763,  1.69721632,  0.59209349, ...,  0.00972462,
##          0.04189227, -0.04391386],
##        [-0.33803821,  1.69700708,  0.59222216, ...,  0.00972371,
##          0.04187694, -0.04390937],
##        [-0.33786653,  1.69676664,  0.59236998, ...,  0.00972265,
##          0.04185931, -0.0439042 ]])
```

```
plt.clf()
plt.figure(figsize=(10, 6))
plt.plot(np.log10(alphas), coef_path)
plt.show()
```



Exercício 03

```
import numpy as np
import statsmodels.api as sm

data = sm.datasets.get_rdataset("mtcars").data
data
```

```
##                       mpg  cyl   disp   hp  drat  ...   qsec  vs  am  gear  carb
## Mazda RX4            21.0    6  160.0  110  3.90  ...  16.46   0   1     4     4
## Mazda RX4 Wag        21.0    6  160.0  110  3.90  ...  17.02   0   1     4     4
## Datsun 710           22.8    4  108.0   93  3.85  ...  18.61   1   1     4     1
## Hornet 4 Drive       21.4    6  258.0  110  3.08  ...  19.44   1   0     3     1
## Hornet Sportabout    18.7    8  360.0  175  3.15  ...  17.02   0   0     3     2
## Valiant              18.1    6  225.0  105  2.76  ...  20.22   1   0     3     1
## Duster 360           14.3    8  360.0  245  3.21  ...  15.84   0   0     3     4
## Merc 240D            24.4    4  146.7   62  3.69  ...  20.00   1   0     4     2
## Merc 230             22.8    4  140.8   95  3.92  ...  22.90   1   0     4     2
## Merc 280             19.2    6  167.6  123  3.92  ...  18.30   1   0     4     4
## Merc 280C            17.8    6  167.6  123  3.92  ...  18.90   1   0     4     4
## Merc 450SE           16.4    8  275.8  180  3.07  ...  17.40   0   0     3     3
## Merc 450SL           17.3    8  275.8  180  3.07  ...  17.60   0   0     3     3
## Merc 450SLC          15.2    8  275.8  180  3.07  ...  18.00   0   0     3     3
## Cadillac Fleetwood   10.4    8  472.0  205  2.93  ...  17.98   0   0     3     4
## Lincoln Continental  10.4    8  460.0  215  3.00  ...  17.82   0   0     3     4
## Chrysler Imperial    14.7    8  440.0  230  3.23  ...  17.42   0   0     3     4
## Fiat 128             32.4    4   78.7   66  4.08  ...  19.47   1   1     4     1
## Honda Civic          30.4    4   75.7   52  4.93  ...  18.52   1   1     4     2
## Toyota Corolla       33.9    4   71.1   65  4.22  ...  19.90   1   1     4     1
## Toyota Corona        21.5    4  120.1   97  3.70  ...  20.01   1   0     3     1
## Dodge Challenger     15.5    8  318.0  150  2.76  ...  16.87   0   0     3     2
## AMC Javelin          15.2    8  304.0  150  3.15  ...  17.30   0   0     3     2
## Camaro Z28           13.3    8  350.0  245  3.73  ...  15.41   0   0     3     4
## Pontiac Firebird     19.2    8  400.0  175  3.08  ...  17.05   0   0     3     2
## Fiat X1-9            27.3    4   79.0   66  4.08  ...  18.90   1   1     4     1
## Porsche 914-2        26.0    4  120.3   91  4.43  ...  16.70   0   1     5     2
## Lotus Europa         30.4    4   95.1  113  3.77  ...  16.90   1   1     5     2
## Ford Pantera L       15.8    8  351.0  264  4.22  ...  14.50   0   1     5     4
## Ferrari Dino         19.7    6  145.0  175  3.62  ...  15.50   0   1     5     6
## Maserati Bora        15.0    8  301.0  335  3.54  ...  14.60   0   1     5     8
## Volvo 142E           21.4    4  121.0  109  4.11  ...  18.60   1   1     4     2
##
## [32 rows x 11 columns]
```

```
X = data.drop(columns=["mpg"])
y = data["mpg"]
```

```
from sklearn.linear_model import ElasticNet
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=4)
```

```
enet = ElasticNet(alpha=0.5, l1_ratio=0.5)

enet.fit(X_train, y_train)
```

```
## ElasticNet(alpha=0.5)
```

```
y_pred = enet.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

```
## Mean Squared Error: 16.912766490726845
```

Exercício 04

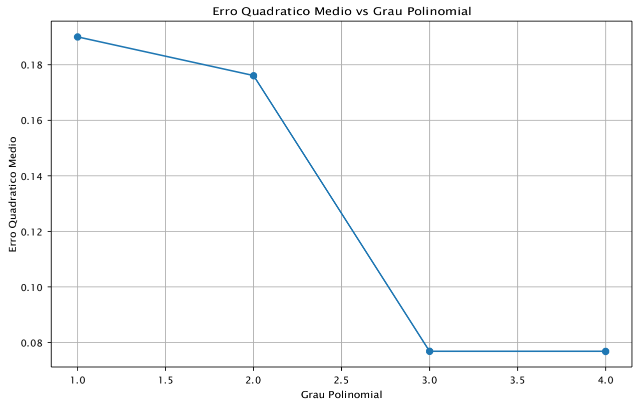
```
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(0)
X = np.linspace(-1, 1, 100)
Y = 2*X**3 + X + 10 + np.random.normal(0, 0.3, 100)
```

```
graus = range(1, 6)
```

```
erros = []
for grau in graus:
    coeffs = np.polyfit(X, Y, grau)
    poly_fit = np.poly1d(coeffs)
    predicted_values = poly_fit(X)
    erro = np.mean((predicted_values - Y)**2)
    erros.append(erro)
```

```
plt.clf()
plt.figure(figsize=(10, 6))
plt.plot(graus, erros, marker="o")
plt.xlabel("Grau Polinomial")
plt.ylabel("Erro Quadratico Medio")
plt.title("Erro Quadratico Medio vs Grau Polinomial")
plt.grid(True)
plt.show()
```


Erro Quadratico Medio vs Grau Polinomial
Erro Quadratico Medio
0.18
0.16
0.14
0.12
0.10
0.08
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Grau Polinomial