Solução Lista 03

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

28 outubro, 2024

Exercício 01

```
import pandas as pd
import statsmodels.api as sm
import seaborn as sns
import numpy as np
```

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

```
##                    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
##
## [5 rows x 11 columns]
```

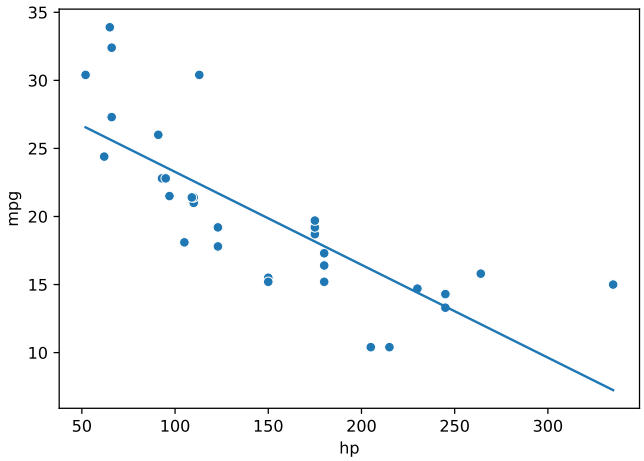
```
rl = sm.formula.ols("mpg ~ hp", data=df).fit()
print(rl.summary())
```

```
##                             OLS Regression Results
##    ==============================================================================
## Dep. Variable:                    mpg   R-squared:                       0.602
## Model:                            OLS   Adj. R-squared:                  0.589
## Method:                 Least Squares   F-statistic:                     45.46
## Date:                dom, 02 jul 2023   Prob (F-statistic):           1.79e-07
## Time:                        22:13:51   Log-Likelihood:                -87.619
## No. Observations:                  32   AIC:                             179.2
## Df Residuals:                      30   BIC:                             182.2
## Df Model:                           1
## Covariance Type:            nonrobust
##    ==============================================================================
##                  coef    std err          t      P>|t|      [0.025      0.975]
```

```
## ------------------------------------------------------------------------------
## Intercept     30.0989      1.634     18.421      0.000      26.762      33.436
## hp            -0.0682      0.010     -6.742      0.000      -0.089      -0.048
## ==============================================================================
## Omnibus:                        3.692   Durbin-Watson:                   1.134
## Prob(Omnibus):                  0.158   Jarque-Bera (JB):                2.984
## Skew:                           0.747   Prob(JB):                        0.225
## Kurtosis:                       2.935   Cond. No.                         386.
## ==============================================================================
##
## Notes:
## [1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

```
sns.scatterplot(x='hp', y='mpg', data=df)

sns.lineplot(x=df['hp'], y=rl.predict(df['hp']))
```




```
from statsmodels.stats.outliers_influence import variance_inflation_factor

fator = pd.DataFrame()
fator["VAR"] = df.columns[1:]
fator["FATOR"] = [variance_inflation_factor(df.values, i) for i in range(1, df.shape[1])]

print(fator)
```

```
##     VAR       FATOR
## 0   cyl  113.712389
## 1  disp  101.586486
## 2    hp   58.387781
## 3  drat  136.544038
## 4    wt  218.046496
## 5  qsec  414.597687
## 6    vs    8.755929
## 7    am    8.104772
## 8  gear  123.142338
## 9  carb   32.367561
```

Exercício 02

```
import pandas as pd
import re
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor

pd.set_option('display.max_columns', None)
```

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

```
##    Unnamed: 0              Name  Age  \
## 0           0  Cristiano Ronaldo   32
## 1           1           L. Messi   30
## 2           2             Neymar   25
## 3           3          L. Suárez   30
## 4           4           M. Neuer   31
##
##                                        Photo Nationality  \
## 0  https://cdn.sofifa.org/48/18/players/20801.png    Portugal
## 1  https://cdn.sofifa.org/48/18/players/158023.png   Argentina
## 2  https://cdn.sofifa.org/48/18/players/190871.png      Brazil
## 3  https://cdn.sofifa.org/48/18/players/176580.png     Uruguay
## 4  https://cdn.sofifa.org/48/18/players/167495.png     Germany
##
##                                Flag  Overall  Potential  \
## 0  https://cdn.sofifa.org/flags/38.png       94         94
## 1  https://cdn.sofifa.org/flags/52.png       93         93
## 2  https://cdn.sofifa.org/flags/54.png       92         94
## 3  https://cdn.sofifa.org/flags/60.png       92         92
## 4  https://cdn.sofifa.org/flags/21.png       92         92
##
##                  Club                                  Club Logo   Value  \
## 0       Real Madrid CF  https://cdn.sofifa.org/24/18/teams/243.png  €95.5M
## 1         FC Barcelona  https://cdn.sofifa.org/24/18/teams/241.png   €105M
## 2  Paris Saint-Germain   https://cdn.sofifa.org/24/18/teams/73.png   €123M
## 3         FC Barcelona  https://cdn.sofifa.org/24/18/teams/241.png    €97M
## 4     FC Bayern Munich   https://cdn.sofifa.org/24/18/teams/21.png    €61M
##
##     Wage  Special Acceleration Aggression Agility Balance Ball control  \
## 0  €565K     2228           89         63      89      63           93
## 1  €565K     2154           92         48      90      95           95
## 2  €280K     2100           94         56      96      82           95
## 3  €510K     2291           88         78      86      60           91
## 4  €230K     1493           58         29      52      35           48
##
##   Composure Crossing Curve Dribbling Finishing Free kick accuracy GK diving  \
## 0        95       85    81        91        94                 76         7
## 1        96       77    89        97        95                 90         6
## 2        92       75    81        96        89                 84         9
```

```
## 3        83       77    86        86        94                 84        27
## 4        70       15    14        30        13                 11        91
##
##   GK handling GK kicking GK positioning GK reflexes Heading accuracy  \
## 0          11         15             14          11                88
## 1          11         15             14           8                71
## 2           9         15             15          11                62
## 3          25         31             33          37                77
## 4          90         95             91          89                25
##
##   Interceptions Jumping Long passing Long shots Marking Penalties Positioning  \
## 0            29      95           77         92      22        85           95
## 1            22      68           87         88      13        74           93
## 2            36      61           75         77      21        81           90
## 3            41      69           64         86      30        85           92
## 4            30      78           59         16      10        47           12
##
##   Reactions Short passing Shot power Sliding tackle Sprint speed Stamina  \
## 0        96            83         94             23           91      92
## 1        95            88         85             26           87      73
## 2        88            81         80             33           90      78
## 3        93            83         87             38           77      89
## 4        85            55         25             11           61      44
##
##   Standing tackle Strength Vision Volleys   CAM    CB   CDM    CF    CM  \
## 0              31       80     85      88  89.0  53.0  62.0  91.0  82.0
## 1              28       59     90      85  92.0  45.0  59.0  92.0  84.0
## 2              24       53     80      83  88.0  46.0  59.0  88.0  79.0
## 3              45       80     84      88  87.0  58.0  65.0  88.0  80.0
## 4              10       83     70      11   NaN   NaN   NaN   NaN   NaN
##
##        ID   LAM    LB   LCB   LCM   LDM    LF    LM    LS    LW   LWB  \
## 0   20801  89.0  61.0  53.0  82.0  62.0  91.0  89.0  92.0  91.0  66.0
## 1  158023  92.0  57.0  45.0  84.0  59.0  92.0  90.0  88.0  91.0  62.0
## 2  190871  88.0  59.0  46.0  79.0  59.0  88.0  87.0  84.0  89.0  64.0
## 3  176580  87.0  64.0  58.0  80.0  65.0  88.0  85.0  88.0  87.0  68.0
## 4  167495   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
##
##   Preferred Positions   RAM    RB   RCB   RCM   RDM    RF    RM    RS    RW  \
## 0              ST LW   89.0  61.0  53.0  82.0  62.0  91.0  89.0  92.0  91.0
## 1                 RW   92.0  57.0  45.0  84.0  59.0  92.0  90.0  88.0  91.0
## 2                 LW   88.0  59.0  46.0  79.0  59.0  88.0  87.0  84.0  89.0
## 3                 ST   87.0  64.0  58.0  80.0  65.0  88.0  85.0  88.0  87.0
## 4                 GK    NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
##
##     RWB    ST
## 0  66.0  92.0
## 1  62.0  88.0
## 2  64.0  84.0
## 3  68.0  88.0
## 4   NaN   NaN
```

```
selected_columns = ['Age', 'Overall', 'Potential',  'Wage', 'Special', 'Acceleration', 'Aggression', 'Ag
                    'Balance', 'Ball control', 'Composure', 'Crossing', 'Curve', 'Dribbling', 'Finishing
```

```
                  'Positioning', 'Stamina', 'Interceptions', 'Strength', 'Vision', 'Volleys', 'Jumping
                  'Penalties', 'Shot power', 'Sprint speed', 'Heading accuracy', 'Long passing', 'Shor

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

```
##    Age  Overall  Potential   Wage  Special Acceleration Aggression Agility  \
## 0   32       94         94  €565K     2228           89         63      89
## 1   30       93         93  €565K     2154           92         48      90
## 2   25       92         94  €280K     2100           94         56      96
## 3   30       92         92  €510K     2291           88         78      86
## 4   31       92         92  €230K     1493           58         29      52
##
##   Balance Ball control Composure Crossing Curve Dribbling Finishing  \
## 0      63           93        95       85    81        91        94
## 1      95           95        96       77    89        97        95
## 2      82           95        92       75    81        96        89
## 3      60           91        83       77    86        86        94
## 4      35           48        70       15    14        30        13
##
##   Positioning Stamina Interceptions Strength Vision Volleys Jumping Penalties  \
## 0          95      92            29       80     85      88      95        85
## 1          93      73            22       59     90      85      68        74
## 2          90      78            36       53     80      83      61        81
## 3          92      89            41       80     84      88      69        85
## 4          12      44            30       83     70      11      78        47
##
##   Shot power Sprint speed Heading accuracy Long passing Short passing
## 0         94           91               88           77            83
## 1         85           87               71           87            88
## 2         80           90               62           75            81
## 3         87           77               77           64            83
## 4         25           61               25           59            55
```

```
df.describe()
```

```
##                 Age       Overall     Potential       Special
## count  17981.000000  17981.000000  17981.000000  17981.000000
## mean      25.144541     66.247984     71.190813   1594.095100
## std        4.614272      6.987965      6.102199    272.151435
## min       16.000000     46.000000     46.000000    728.000000
## 25%       21.000000     62.000000     67.000000   1449.000000
## 50%       25.000000     66.000000     71.000000   1633.000000
## 75%       28.000000     71.000000     75.000000   1786.000000
## max       47.000000     94.000000     94.000000   2291.000000
```

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

```
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
trata_modificadores('34-2')
```

```
## 32
```

```
from tqdm import tqdm
```

```
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, 14.71it/s] 14%|#4        |
```

```
for i in selected_columns:
  print(i)
  print(df[i].sort_values().unique())
```

```
## Age
## [16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
##  40 41 43 44 47]
## Overall
## [46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69
##  70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93
##  94]
## Potential
## [46 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70
##  71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94]
## Wage
## [  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
##   18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35
##   36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53
##   54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71
##   72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89
##   90  91  92  94  95  96  97  98  99 100 105 110 115 120 125 130 135 140
##  145 150 155 160 165 170 175 180 185 190 195 200 205 210 215 220 225 230
##  235 240 250 260 265 275 280 285 295 310 325 340 355 370 510 565]
## Special
## [ 728  736  755 ... 2278 2286 2291]
## Acceleration
## [11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34
##  35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58
##  59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82
##  83 84 85 86 87 88 89 90 91 92 93 94 95 96]
```

```
## Aggression
## [ 11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28
##   29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46
##   47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64
##   65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82
##   83  84  85  86  87  88  89  90  91  92  93  94  95  96 106]
## Agility
## [14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37
##  38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61
##  62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85
##  86 87 88 89 90 91 92 93 94 95 96]
## Balance
## [11 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37
##  38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61
##  62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85
##  86 87 88 89 90 91 92 93 94 95 96]
## Ball control
## [ 8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
##  32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55
##  56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79
##  80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95]
## Composure
## [  5  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27
##   28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45
##   46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63
##   64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81
##   82  83  84  85  86  87  88  89  90  91  92  95  96 100]
## Crossing
## [  5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22
##   23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
##   41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58
##   59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76
##   77  78  79  80  81  82  83  84  85  86  87  88  89  90  91 101]
## Curve
## [ 6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
##  30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53
##  54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77
##  78 79 80 81 82 83 84 85 86 87 88 89 90 92]
## Dribbling
## [ 2  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
##  27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
##  51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
##  75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 96 97]
## Finishing
## [ 2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
##  26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
##  50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73
##  74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 94 95]
## Positioning
## [ 2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
##  26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
##  50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73
##  74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 95]
## Stamina
```

```
## [ 12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29
##   30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47
##   48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65
##   66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83
##   84  85  86  87  88  89  90  91  92  93  94  95  97 116]
## Interceptions
## [-1  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
##  27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
##  51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
##  75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92]
## Strength
## [12 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42
##  43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66
##  67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90
##  91 92 93 94 95 96 98]
## Vision
## [10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33
##  34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57
##  58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81
##  82 83 84 85 86 87 88 89 90 91 92 94]
## Volleys
## [ 4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
##  28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51
##  52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75
##  76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91]
## Jumping
## [13 15 16 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45
##  46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69
##  70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93
##  94 95]
## Penalties
## [ 5  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
##  30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53
##  54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77
##  78 79 80 81 82 83 84 85 86 87 88 89 90 91 92]
## Shot power
## [ 3  6  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
##  30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53
##  54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77
##  78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 98]
## Sprint speed
## [ 11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28
##   29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46
##   47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64
##   65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82
##   83  84  85  86  87  88  89  90  91  92  93  94  95  96  97 110]
## Heading accuracy
## [ 4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
##  28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51
##  52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75
##  76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94]
## Long passing
## [ 7  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
##  32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55
```

```
##  56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79
##  80 81 82 83 84 85 86 87 88 89 90 92 93]
## Short passing
## [10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33
##  34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57
##  58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81
##  82 83 84 85 86 87 88 89 90 91 92]
```

```
df = df.astype(int)
df = df.dropna()
X = df.drop('Overall', axis=1)
y = df['Overall']
np.asarray(df)
```

```
## array([[32, 94, 94, ..., 88, 77, 83],
##        [30, 93, 93, ..., 71, 87, 88],
##        [25, 92, 94, ..., 62, 75, 81],
##        ...,
##        [17, 46, 61, ..., 47, 30, 42],
##        [18, 46, 64, ..., 40, 44, 49],
##        [18, 46, 64, ..., 48, 24, 31]])
```

```
df.head()
```

```
##    Age  Overall  Potential  Wage  Special  Acceleration  Aggression  Agility  \
## 0   32       94         94   565     2228            89          63       89
## 1   30       93         93   565     2154            92          48       90
## 2   25       92         94   280     2100            94          56       96
## 3   30       92         92   510     2291            88          78       86
## 4   31       92         92   230     1493            58          29       52
##
##    Balance  Ball control  Composure  Crossing  Curve  Dribbling  Finishing  \
## 0       63            93         95        85     81         91         94
## 1       95            95         96        77     89         97         95
## 2       82            95         92        75     81         96         89
## 3       60            91         83        77     86         86         94
## 4       35            48         70        15     14         30         13
##
##    Positioning  Stamina  Interceptions  Strength  Vision  Volleys  Jumping  \
## 0           95       92             29        80      85       88       95
## 1           93       73             22        59      90       85       68
## 2           90       78             36        53      80       83       61
## 3           92       89             41        80      84       88       69
## 4           12       44             30        83      70       11       78
##
##    Penalties  Shot power  Sprint speed  Heading accuracy  Long passing  \
## 0         85          94            91                88            77
## 1         74          85            87                71            87
## 2         81          80            90                62            75
## 3         85          87            77                77            64
## 4         47          25            61                25            59
##
##    Short passing
```

```
## 0             83
## 1             88
## 2             81
## 3             83
## 4             55
```

```
X = sm.add_constant(X)

model = sm.OLS(y, X)
results = model.fit()

print(results.summary())
```

```
##                             OLS Regression Results
## ==============================================================================
## Dep. Variable:                Overall   R-squared:                       0.909
## Model:                            OLS   Adj. R-squared:                  0.909
## Method:                 Least Squares   F-statistic:                     6633.
## Date:                dom, 02 jul 2023   Prob (F-statistic):               0.00
## Time:                        22:14:02   Log-Likelihood:                -38934.
## No. Observations:               17981   AIC:                          7.792e+04
## Df Residuals:                   17953   BIC:                          7.814e+04
## Df Model:                          27
## Covariance Type:            nonrobust
## ==============================================================================
##                      coef    std err          t      P>|t|      [0.025      0.975]
## ------------------------------------------------------------------------------
## const            -17.3757      0.316    -54.963      0.000     -17.995     -16.756
## Age                0.6681      0.005    123.708      0.000       0.658       0.679
## Potential          0.6444      0.004    150.496      0.000       0.636       0.653
## Wage               0.0230      0.001     27.037      0.000       0.021       0.025
## Special            0.0293      0.000     62.081      0.000       0.028       0.030
## Acceleration      -0.0010      0.003     -0.319      0.750      -0.007       0.005
## Aggression        -0.0368      0.002    -20.830      0.000      -0.040      -0.033
## Agility           -0.0185      0.002     -7.910      0.000      -0.023      -0.014
## Balance           -0.0545      0.002    -25.816      0.000      -0.059      -0.050
## Ball control       0.0324      0.004      8.591      0.000       0.025       0.040
## Composure          0.0406      0.002     17.210      0.000       0.036       0.045
## Crossing          -0.0274      0.002    -12.519      0.000      -0.032      -0.023
## Curve             -0.0411      0.002    -19.124      0.000      -0.045      -0.037
## Dribbling         -0.0425      0.003    -13.440      0.000      -0.049      -0.036
## Finishing         -0.0123      0.002     -4.992      0.000      -0.017      -0.007
## Positioning       -0.0364      0.002    -14.982      0.000      -0.041      -0.032
## Stamina           -0.0083      0.002     -4.298      0.000      -0.012      -0.005
## Interceptions     -0.0818      0.002    -39.739      0.000      -0.086      -0.078
## Strength           0.0028      0.002      1.386      0.166      -0.001       0.007
## Vision            -0.0276      0.002    -12.062      0.000      -0.032      -0.023
## Volleys           -0.0248      0.002    -10.779      0.000      -0.029      -0.020
## Jumping           -0.0207      0.002    -11.840      0.000      -0.024      -0.017
## Penalties         -0.0475      0.002    -22.283      0.000      -0.052      -0.043
## Shot power        -0.0367      0.002    -16.933      0.000      -0.041      -0.032
## Sprint speed      -0.0057      0.003     -2.020      0.043      -0.011      -0.000
## Heading accuracy  -0.0043      0.002     -2.249      0.025      -0.008      -0.001
## Long passing      -0.0571      0.003    -20.428      0.000      -0.063      -0.052
```

```
## Short passing        0.0101      0.004      2.783      0.005       0.003       0.017
## ==============================================================================
## Omnibus:                      811.751   Durbin-Watson:                   1.739
## Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1035.298
## Skew:                          -0.471   Prob(JB):                    1.54e-225
## Kurtosis:                       3.703   Cond. No.                     3.30e+04
## ==============================================================================
##
## Notes:
## [1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
## [2] The condition number is large, 3.3e+04. This might indicate that there are
## strong multicollinearity or other numerical problems.
```

Exercício 03

```
import pandas as pd
import statsmodels.api as sm
from tqdm import tqdm

pd.set_option('display.max_columns', None)
file_url = "https://drive.google.com/uc?export=download&id=1jiWcGsl_tbqK5FOryUTq48kcDTKWTTuk"
df = pd.read_csv(file_url)

selected_columns = ['Age', 'Overall', 'Potential',  'Wage', 'Special', 'Acceleration', 'Aggression', 'Ag
                    'Balance', 'Ball control', 'Composure', 'Crossing', 'Curve', 'Dribbling', 'Finishing
                    'Positioning', 'Stamina', 'Interceptions', 'Strength', 'Vision', 'Volleys', 'Jumping
                    'Penalties', 'Shot power', 'Sprint speed', 'Heading accuracy', 'Long passing', 'Shor

df = df[selected_columns].copy()

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

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, 14.97it/s] 14%|#4        |
```

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

X = df.drop('Wage', axis=1)
y = df['Wage']
np.asarray(df)
```

```
## array([[32, 94, 94, ..., 88, 77, 83],
##        [30, 93, 93, ..., 71, 87, 88],
##        [25, 92, 94, ..., 62, 75, 81],
##        ...,
##        [17, 46, 61, ..., 47, 30, 42],
##        [18, 46, 64, ..., 40, 44, 49],
##        [18, 46, 64, ..., 48, 24, 31]])
```

```
df.head()
```

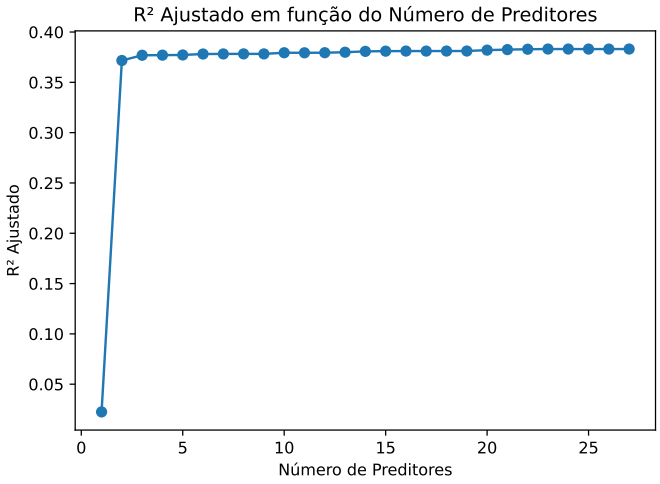
```
##    Age  Overall  Potential  Wage  Special  Acceleration  Aggression  Agility  \
## 0   32       94         94   565     2228            89          63       89
## 1   30       93         93   565     2154            92          48       90
## 2   25       92         94   280     2100            94          56       96
## 3   30       92         92   510     2291            88          78       86
## 4   31       92         92   230     1493            58          29       52
##
##    Balance  Ball control  Composure  Crossing  Curve  Dribbling  Finishing  \
## 0       63            93         95        85     81         91         94
## 1       95            95         96        77     89         97         95
## 2       82            95         92        75     81         96         89
## 3       60            91         83        77     86         86         94
## 4       35            48         70        15     14         30         13
##
##    Positioning  Stamina  Interceptions  Strength  Vision  Volleys  Jumping  \
## 0           95       92             29        80      85       88       95
## 1           93       73             22        59      90       85       68
## 2           90       78             36        53      80       83       61
## 3           92       89             41        80      84       88       69
## 4           12       44             30        83      70       11       78
##
##    Penalties  Shot power  Sprint speed  Heading accuracy  Long passing  \
## 0         85          94            91                88            77
## 1         74          85            87                71            87
## 2         81          80            90                62            75
## 3         85          87            77                77            64
## 4         47          25            61                25            59
##
##    Short passing
## 0             83
## 1             88
## 2             81
## 3             83
## 4             55
```

```
import matplotlib.pyplot as plt

num_preditores = range(1, X.shape[1] + 1)
r_quadrado_ajustado = []
for k in num_preditores:
    X_subset = X.iloc[:, :k]
    X_subset = sm.add_constant(X_subset)
    modelo = sm.OLS(y, X_subset)
    resultado = modelo.fit()
    r_quadrado_ajustado.append(resultado.rsquared_adj)

plt.clf()
plt.plot(num_preditores, r_quadrado_ajustado, marker='o')
plt.xlabel('Número de Preditores')
plt.ylabel('R² Ajustado')
plt.title('R² Ajustado em função do Número de Preditores')
```

```
plt.show()
```



Exercício 04

```
import pandas as pd
import numpy as np
import statsmodels.api as sm
from tqdm import tqdm

pd.set_option('display.max_columns', None)
file_url = "https://drive.google.com/uc?export=download&id=1jiWcGsl_tbqK5F0ryUTq48kcDTKWTTuk"
df = pd.read_csv(file_url)

selected_columns = ['Age', 'Overall', 'Potential',  'Wage', 'Special', 'Acceleration', 'Aggression', 'Ag
                    'Balance', 'Ball control', 'Composure', 'Crossing', 'Curve', 'Dribbling', 'Finishing
                    'Positioning', 'Stamina', 'Interceptions', 'Strength', 'Vision', 'Volleys', 'Jumping
                    'Penalties', 'Shot power', 'Sprint speed', 'Heading accuracy', 'Long passing', 'Shoo

df = df[selected_columns].copy()

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

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, 14.53it/s] 14%|#4        |
```

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

X = df.drop('Wage', axis=1)
y = df['Wage']
np.asarray(df)
```

```
## array([[32, 94, 94, ..., 88, 77, 83],
##        [30, 93, 93, ..., 71, 87, 88],
##        [25, 92, 94, ..., 62, 75, 81],
##        ...,
##        [17, 46, 61, ..., 47, 30, 42],
##        [18, 46, 64, ..., 40, 44, 49],
##        [18, 46, 64, ..., 48, 24, 31]])
```

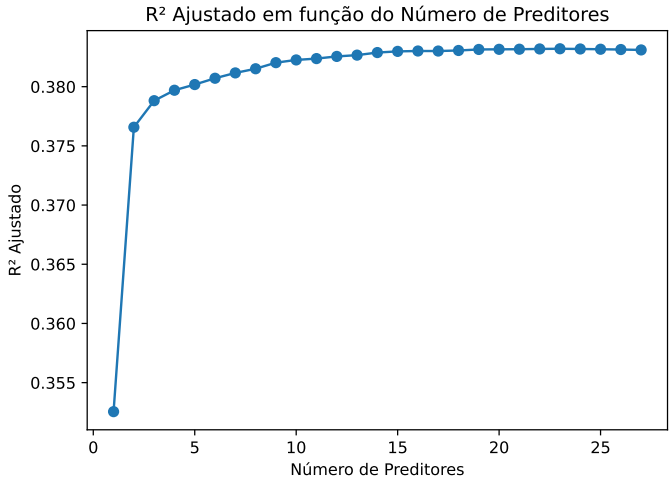
```
df.head()
```

```
##    Age  Overall  Potential  Wage  Special  Acceleration  Aggression  Agility  \
## 0   32       94         94   565     2228            89          63       89
## 1   30       93         93   565     2154            92          48       90
## 2   25       92         94   280     2100            94          56       96
## 3   30       92         92   510     2291            88          78       86
## 4   31       92         92   230     1493            58          29       52
##
##    Balance  Ball control  Composure  Crossing  Curve  Dribbling  Finishing  \
## 0       63            93         95        85     81         91         94
## 1       95            95         96        77     89         97         95
## 2       82            95         92        75     81         96         89
## 3       60            91         83        77     86         86         94
## 4       35            48         70        15     14         30         13
##
##    Positioning  Stamina  Interceptions  Strength  Vision  Volleys  Jumping  \
## 0           95       92             29        80      85       88       95
## 1           93       73             22        59      90       85       68
## 2           90       78             36        53      80       83       61
## 3           92       89             41        80      84       88       69
## 4           12       44             30        83      70       11       78
##
##    Penalties  Shot power  Sprint speed  Heading accuracy  Long passing  \
## 0         85          94            91                88            77
## 1         74          85            87                71            87
## 2         81          80            90                62            75
## 3         85          87            77                77            64
## 4         47          25            61                25            59
##
##    Short passing
## 0             83
## 1             88
## 2             81
## 3             83
## 4             55
```

```
num_preditores = range(1, X.shape[1] + 1)
preditores = X.columns.tolist()
preditores_selecionados = []
r_quadrado_ajustado = []
for k in num_preditores:
  melhor_r_quadrado_ajustado = -float('inf')
  melhor_preditor = None
  for preditor in preditores:
```

```
    if preditor not in preditores_selecionados:
      preditor_atual = preditores_selecionados + [preditor]
      X_subset = X[preditor_atual]
      X_subset = sm.add_constant(X_subset)
      modelo = sm.OLS(y, X_subset)
      resultado = modelo.fit()
      r_quadrado_ajustado_atual = resultado.rsquared_adj
      if r_quadrado_ajustado_atual > melhor_r_quadrado_ajustado:
        melhor_r_quadrado_ajustado = r_quadrado_ajustado_atual
        melhor_preditor = preditor
  preditores_selecionados.append(melhor_preditor)
  r_quadrado_ajustado.append(melhor_r_quadrado_ajustado)

plt.clf()
plt.plot(range(1, len(preditores_selecionados) + 1), r_quadrado_ajustado, marker='o')
plt.xlabel('Número de Preditores')
plt.ylabel('R² Ajustado')
plt.title('R² Ajustado em função do Número de Preditores')
plt.show()
```




```
melhor_modelo = preditores_selecionados[np.argmax(r_quadrado_ajustado)]
print('Melhor modelo (com base no R² ajustado):', melhor_modelo)
```

```
## Melhor modelo (com base no R² ajustado): Balance
```

Vantagens e desvantagens: O FSS constrói o modelo adicionando preditores de forma progressiva, o que reduz o risco de incluir preditores irrelevantes e ajuda a evitar overfitting, em comparação com o BSS, que considera todas as combinações possíveis. Embora seja mais eficiente, não garante encontrar a melhor combinação de preditores em termos de R^2 ajustado. Pode ser limitado se o verdadeiro melhor modelo incluir preditores que não são selecionados nas primeiras etapas.

Exercicio 5

```
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LinearRegression

scores = []
for n in num_preditores:
    model = LinearRegression()
    X_subset = X.iloc[:, :n]
    cv_scores = cross_val_score(model, X_subset, y, cv=10).mean()
    scores.append(cv_scores)

print("O número de preditores ideal é:", num_preditores[scores.index(max(scores))])
```

```
## O número de preditores ideal é: 1
```