regression-wine-2

August 19, 2020

[0]:

```

```

1 0. Introdução

Trabalho:

Aluno: Gabriel Luiz

Disciplina: Tópico em Aprendizado de Máquina

Objetivos :

- Escolha dois conjuntos de dados para trabalhar o problema de regressão. Separe cada dataset em conjunto de treinamento e conjunto de teste. Explique o seu critério de separação e o método utilizado.
- Você deverá implementar soluções para cada dataset usando:
- – regressão linear (ou regressão múltipla)
- – regressão polinomial
- – SVR (use os kernels linear, sigmoide, RBF e polinomial)
- – rede neural (MLP ou RBF).
- Descreva os parâmetros/arquiteturas de cada modelo.
- Compare os resultados (para treinamento e teste) com as medidas de desempenho SEQ, EQM, REQM, EAM e r^2 , e verifique qual a melhor opção dentre os métodos implementados que melhor se ajusta a seus dados.
- Você deverá fazer a visualização dos dados originais com os dados ajustados em cada experimento, tanto para o conjunto de treinamento quanto para o de teste. Os gráficos devem conter títulos nos eixos e legenda. Comente os resultados encontrados na visualização.

1.1 0.1 Dependências

Para realização da tarefa foram utilizados as seguintes bibliotecas:

[1]:
```
#Utils
import pandas as pd
import numpy as np
import pandas_profiling
import math

#Preprocess
from sklearn.preprocessing import StandardScaler

# Split
from sklearn.model_selection import train_test_split

# Regressores
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.neural_network import MLPRegressor

#Metricas
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error

#Visualização
import seaborn as sns
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

2 1. Dados

O conjunto de dados possui informações quimicas de vinhos Possui mais de 1500 registros e 12 atributos

Fonte: https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009

2.1 1.1 Informações sobre os dados:

Atributos: Input variables (based on physicochemical tests):

- fixed acidity
- volatile acidity
- citric acid
- residual sugar
- chlorides
- free sulfur dioxide

- total sulfur dioxide
- density
- pH
- sulphates
- alcohol

Output variable (based on sensory data): - quality (score between 0 and 10)

2.2 Importando Dataset

[2]:
```
dataset = './dataset/datasets_4458_8204_winequality-red.csv'

data_raw = pd.read_csv(dataset)
```

[3]:
```
data_raw.head()
```

[3]:
```
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0            7.4              0.70         0.00             1.9      0.076
1            7.8              0.88         0.00             2.6      0.098
2            7.8              0.76         0.04             2.3      0.092
3           11.2              0.28         0.56             1.9      0.075
4            7.4              0.70         0.00             1.9      0.076

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                 11.0                  34.0   0.9978  3.51       0.56
1                 25.0                  67.0   0.9968  3.20       0.68
2                 15.0                  54.0   0.9970  3.26       0.65
3                 17.0                  60.0   0.9980  3.16       0.58
4                 11.0                  34.0   0.9978  3.51       0.56

   alcohol  quality
0      9.4        5
1      9.8        5
2      9.8        5
3      9.8        6
4      9.4        5
```

[4]:
```
# wine_quality = []
# for quality in data_raw.quality:
#     if quality >= 6:
#         wine_quality.append(1)
#     else:
#         wine_quality.append(0)
#
# data_raw.quality = wine_quality
```

```
[5]: for col in data_raw:
         print(col, data_raw[col].unique())
```

```
fixed acidity [ 7.4  7.8 11.2  7.9  7.3  7.5  6.7  5.6  8.9  8.5  8.1  7.6  6.9
6.3
  7.1  8.3  5.2  5.7  8.8  6.8  4.6  7.7  8.7  6.4  6.6  8.6 10.2  7.
  7.2  9.3  8.   9.7  6.2  5.   4.7  8.4 10.1  9.4  9.   8.2  6.1  5.8
  9.2 11.5  5.4  9.6 12.8 11.  11.6 12.  15.  10.8 11.1 10.  12.5 11.8
 10.9 10.3 11.4  9.9 10.4 13.3 10.6  9.8 13.4 10.7 11.9 12.4 12.2 13.8
  9.1 13.5 10.5 12.6 14.  13.7  9.5 12.7 12.3 15.6  5.3 11.3 13.   6.5
 12.9 14.3 15.5 11.7 13.2 15.9 12.1  5.1  4.9  5.9  6.   5.5]
volatile acidity [0.7   0.88  0.76  0.28  0.66  0.6   0.65  0.58  0.5   0.615
0.61  0.62
 0.56  0.59  0.32  0.22  0.39  0.43  0.49  0.4   0.41  0.71  0.645 0.675
 0.685 0.655 0.605 0.38  1.13  0.45  0.67  0.52  0.935 0.29  0.31  0.51
 0.42  0.63  0.69  0.735 0.725 0.705 0.785 0.75  0.625 0.3   0.55  1.02
 0.775 0.9   0.545 0.575 0.33  0.54  1.07  0.695 1.33  0.745 1.04  0.715
 0.415 0.34  0.68  0.95  0.53  0.64  0.885 0.805 0.73  0.37  0.835 1.09
 0.57  0.44  0.635 0.82  0.48  1.    0.21  0.35  0.975 0.26  0.87  0.18
 0.27  0.2   0.36  0.83  0.46  0.47  0.77  0.815 0.795 0.665 0.765 0.24
 0.85  0.84  0.96  0.78  0.23  0.315 0.365 0.25  0.825 0.72  0.595 0.585
 0.915 0.755 0.845 1.24  0.8   0.98  1.185 0.92  1.035 1.025 0.565 0.74
 1.115 0.865 0.875 0.965 0.91  0.89  1.01  0.305 0.395 0.12  0.86  0.295
 1.005 0.19  0.955 0.16  1.58  0.79  1.18  0.475 0.81  0.895 0.855]
citric acid [0.   0.04 0.56 0.06 0.02 0.36 0.08 0.29 0.18 0.19 0.28 0.51 0.48
0.31
 0.21 0.11 0.14 0.16 0.24 0.07 0.12 0.25 0.09 0.3  0.2  0.22 0.15 0.43
 0.52 0.23 0.37 0.26 0.57 0.4  0.49 0.05 0.54 0.64 0.7  0.47 0.44 0.17
 0.68 0.53 0.1  0.01 0.55 1.   0.03 0.42 0.33 0.32 0.35 0.6  0.74 0.58
 0.5  0.76 0.46 0.45 0.38 0.39 0.66 0.62 0.67 0.79 0.63 0.61 0.71 0.65
 0.59 0.34 0.69 0.73 0.72 0.41 0.27 0.75 0.13 0.78]
residual sugar [ 1.9   2.6   2.3   1.8   1.6   1.2   2.    6.1   3.8   3.9   1.7
4.4
  2.4   1.4   2.5  10.7   5.5   2.1   1.5   5.9   2.8   2.2   3.    3.4
  5.1   4.65  1.3   7.3   7.2   2.9   2.7   5.6   3.1   3.2   3.3   3.6
  4.    7.    6.4   3.5  11.    3.65  4.5   4.8   2.95  5.8   6.2   4.2
  7.9   3.7   6.7   6.6   2.15  5.2   2.55 15.5   4.1   8.3   6.55  4.6
  4.3   5.15  6.3   6.    8.6   7.5   2.25  4.25  2.85  3.45  2.35  2.65
  9.    8.8   5.    1.65  2.05  0.9   8.9   8.1   4.7   1.75  7.8  12.9
 13.4   5.4  15.4   3.75 13.8   5.7  13.9 ]
chlorides [0.076 0.098 0.092 0.075 0.069 0.065 0.073 0.071 0.097 0.089 0.114
0.176
 0.17  0.368 0.086 0.341 0.077 0.082 0.106 0.084 0.085 0.08  0.105 0.083
 0.103 0.066 0.172 0.074 0.088 0.332 0.05  0.054 0.113 0.068 0.081 0.11
 0.07  0.111 0.079 0.115 0.094 0.093 0.104 0.464 0.401 0.062 0.107 0.045
 0.058 0.102 0.467 0.091 0.122 0.09  0.119 0.178 0.146 0.072 0.118 0.049
 0.06  0.117 0.087 0.236 0.61  0.095 0.1   0.36  0.067 0.27  0.099 0.046
```

```
 0.061 0.056 0.039 0.059 0.101 0.057 0.337 0.078 0.263 0.063 0.611 0.064
 0.096 0.358 0.343 0.186 0.112 0.213 0.214 0.121 0.128 0.052 0.12  0.116
 0.109 0.159 0.124 0.174 0.047 0.127 0.413 0.152 0.053 0.055 0.051 0.125
 0.2   0.171 0.226 0.25  0.108 0.148 0.143 0.222 0.157 0.422 0.034 0.387
 0.415 0.243 0.241 0.19  0.132 0.126 0.038 0.044 0.041 0.165 0.048 0.145
 0.147 0.012 0.194 0.161 0.123 0.414 0.216 0.043 0.042 0.369 0.166 0.136
 0.403 0.137 0.168 0.153 0.267 0.169 0.205 0.235 0.23 ]
free sulfur dioxide [11.  25.  15.  17.  13.   9.  16.  52.  51.  35.   6.  29.
23.  10.
 21.   4.  14.   8.  22.  40.   5.   3.   7.  12.  30.  33.  50.  19.
 20.  27.  18.  28.  34.  42.  41.  37.  32.  36.  24.  26.  39.  40.5
 68.  31.  38.  43.  47.   1.  54.  46.  45.   2.   5.5 53.  37.5 57.
 48.  72.  55.  66. ]
total sulfur dioxide [ 34.   67.   54.   60.   40.   59.   21.   18.  102.   65.
29.  145.
 148.  103.   56.   71.   37.   23.   11.   35.   16.   82.  113.   83.
  50.   15.   30.   19.   87.   46.   14.  114.   12.   96.  119.   73.
  45.   10.  110.   52.  112.   39.   27.   94.   43.   42.   80.   51.
  61.  136.   31.  125.   24.  140.  133.   85.  106.   22.   36.   69.
  64.  153.   47.  108.  111.   62.   28.   89.   13.   90.  134.   99.
  26.   63.  105.   20.  141.   88.  129.  128.   86.  121.  101.   44.
   8.   49.   38.  143.  144.  127.  126.  120.   55.   93.   95.   41.
  58.   72.   81.  109.   33.   53.   98.   48.   70.   25.  135.   92.
  74.   32.   77.  165.   75.  124.   78.  122.   66.   68.   17.   91.
  76.  151.  142.  116.  149.   57.  104.   84.  147.  155.  152.    9.
 139.  130.    7.  100.  115.    6.   79.  278.  289.  160.   77.5 131. ]
density [0.9978  0.9968  0.997   0.998   0.9964  0.9946  0.9959  0.9943  0.9974
 0.9986  0.9969  0.9982  0.9966  0.9955  0.9962  0.9972  0.9958  0.9993
 0.9957  0.9975  0.994   0.9976  0.9934  0.9954  0.9971  0.9956  0.9983
 0.9967  0.9961  0.9984  0.9938  0.9932  0.9965  0.9963  0.996   0.9973
 0.9988  0.9937  0.9952  0.9916  0.9944  0.9996  0.995   0.9981  0.9953
 0.9924  0.9948  0.99695 0.99545 0.99615 0.9994  0.99625 0.99585 0.99685
 0.99655 0.99525 0.99815 0.99745 0.9927  0.99675 0.99925 0.99565 1.00005
 0.9985  0.99965 0.99575 0.9999  1.00025 0.9987  0.99935 0.99735 0.99915
 0.9991  1.00015 0.9997  1.001   0.9979  1.0014  1.0001  0.99855 0.99845
 0.9998  0.99645 0.99865 0.9989  0.99975 0.999   1.0015  1.0002  0.9992
 1.0008  1.      1.0006  1.0004  1.0018  0.9912  1.0022  1.0003  0.9949
 0.9951  1.0032  0.9947  0.9995  0.9977  1.0026  1.00315 1.0021  0.9917
 0.9922  0.9921  0.99788 1.00024 0.99768 0.99782 0.99761 0.99803 0.99785
 0.99656 0.99488 0.99823 0.99779 0.99738 0.99701 0.99888 0.99938 0.99744
 0.99668 0.99727 0.99586 0.99612 0.99676 0.99732 0.99814 0.99746 0.99708
 0.99818 0.99639 0.99531 0.99786 0.99526 0.99641 0.99264 0.99682 0.99356
 0.99386 0.99702 0.99693 0.99562 1.00012 0.99462 0.99939 0.99632 0.99976
 0.99606 0.99154 0.99624 0.99417 0.99376 0.99832 0.99836 0.99694 0.99064
 0.99672 0.99647 0.99736 0.99629 0.99689 0.99801 0.99652 0.99538 0.99594
 0.99686 0.99438 0.99357 0.99628 0.99748 0.99578 0.99371 0.99522 0.99576
 0.99552 0.99664 0.99614 0.99517 0.99787 0.99533 0.99536 0.99824 0.99577
 0.99491 1.00289 0.99743 0.99774 0.99444 0.99892 0.99528 0.99331 0.99901
```

```
 0.99674 0.99512 0.99395 0.99504 0.99516 0.99604 0.99468 0.99543 0.99791
 0.99425 0.99509 0.99484 0.99834 0.99864 0.99498 0.99566 0.99408 0.99458
 0.99648 0.99568 0.99613 0.99519 0.99518 0.99592 0.99654 0.99546 0.99554
 0.99733 0.99669 0.99724 0.99643 0.99605 0.99658 0.99416 0.99712 0.99418
 0.99596 0.99556 0.99918 0.99697 0.99378 0.99162 0.99495 0.9928  0.99603
 0.99549 0.99722 0.99354 0.99635 0.99454 0.99598 0.99486 0.99007 0.99636
 0.99642 0.99584 0.99506 0.99822 0.99364 0.99514 0.99854 0.99739 0.99683
 0.99692 0.99756 0.99547 0.99859 0.99294 0.99634 0.99704 0.99258 0.99426
 0.99747 0.99784 0.99358 0.99572 0.99769 0.99534 0.99817 0.99316 0.99471
 0.99617 0.99529 0.99451 0.99479 0.99772 0.99666 0.99392 0.99388 0.99402
 0.9936  0.99374 0.99523 0.99593 0.99396 0.99698 0.9902  0.99252 0.99256
 0.99235 0.99352 0.99557 0.99394 0.9915  0.99379 0.99798 0.99341 0.9933
 0.99684 0.99524 0.99764 0.99588 0.99473 0.99616 0.99622 0.99544 0.99728
 0.99551 0.99434 0.99709 0.99384 0.99502 0.99667 0.99649 0.99716 0.99541
 0.99318 0.99346 0.99599 0.99478 0.99754 0.99439 0.99633 0.99419 0.99878
 0.99752 0.99428 0.99659 0.99677 0.99734 0.99678 0.99638 0.99922 0.99157
 0.99718 0.99621 0.99242 0.99494 0.99729 0.99414 0.99721 0.99627 0.99569
 0.99499 0.99437 0.99726 0.99456 0.99564 0.9908  0.99084 0.9935  0.99385
 0.99688 0.99619 0.99476 0.99328 0.99286 0.99914 0.99521 0.99362 0.99558
 0.99323 0.99191 0.99501 0.9929  0.99532 0.99796 0.99581 0.99608 0.99387
 0.99448 0.99589 0.99852 0.99472 0.99587 0.99332 0.99464 0.99699 0.99725
 0.99623 0.99609 0.99292 0.9942  1.00369 0.99713 0.99322 0.99706 0.99974
 0.99467 0.99236 0.99705 0.99334 0.99336 1.00242 0.99182 0.99808 0.99828
 0.99719 0.99542 0.99496 0.99344 0.99348 0.99459 0.99492 0.99508 0.99582
 0.99555 0.9941  0.99661 0.99842 0.99489 0.99665 0.99553 0.99714 0.99631
 0.99573 0.99717 0.99397 0.99646 0.99758 0.99306 0.99783 0.99765 0.99474
 0.99483 0.99314 0.99574 0.99651]
pH [3.51 3.2  3.26 3.16 3.3  3.39 3.36 3.35 3.28 3.58 3.17 3.11 3.38 3.04
 3.52 3.43 3.34 3.47 3.46 3.45 3.4  3.42 3.23 3.5  3.33 3.21 3.48 3.9
 3.25 3.32 3.15 3.41 3.44 3.31 3.54 3.13 2.93 3.14 3.75 3.85 3.29 3.08
 3.37 3.19 3.07 3.49 3.53 3.24 3.63 3.22 3.68 2.74 3.59 3.   3.12 3.57
 3.61 3.06 3.6  3.69 3.1  3.05 3.67 3.27 3.18 3.02 3.55 2.99 3.01 3.56
 3.03 3.62 2.88 2.95 2.98 3.09 2.86 3.74 2.92 3.72 2.87 2.89 2.94 3.66
 3.71 3.78 3.7  4.01 2.9 ]
sulphates [0.56 0.68 0.65 0.58 0.46 0.47 0.57 0.8  0.54 0.52 1.56 0.88 0.93 0.75
 1.28 0.5  1.08 0.53 0.91 0.63 0.59 0.55 0.66 0.6  0.73 0.48 0.83 0.51
 0.9  1.2  0.74 0.64 0.77 0.71 0.62 0.39 0.79 0.95 0.82 1.12 1.14 0.78
 1.95 1.22 1.98 0.61 1.31 0.69 0.67 0.7  0.49 0.92 2.   0.72 1.59 0.33
 1.02 0.97 0.85 0.43 1.03 0.86 0.76 1.61 1.09 0.84 0.96 0.45 1.26 0.87
 0.81 1.   1.36 1.18 0.89 0.98 1.13 1.04 1.11 0.99 1.07 0.44 1.06 1.05
 0.42 1.17 1.62 0.94 1.34 1.16 1.1  0.4  1.15 0.37 1.33 1.01]
alcohol [ 9.4         9.8        10.          9.5        10.5         9.2
  9.9         9.1         9.3         9.          9.7        10.1
 10.6         9.6        10.8        10.3        13.1        10.2
 10.9        10.7        12.9        10.4        13.         14.
 11.5        11.4        12.4        11.         12.2        12.8
 12.6        12.5        11.7        11.3        12.3        12.
 11.9        11.8         8.7        13.3        11.2        11.6
```

```
 11.1         13.4         12.1          8.4         12.7         14.9
 13.2         13.6         13.5         10.03333333  9.55         8.5
 11.06666667  9.56666667 10.55          8.8         13.56666667 11.95
  9.95        9.23333333  9.25          9.05        10.75       ]
quality [5 6 7 4 8 3]
```

2.3 Pré-processamento

[6]:
```python
# pandas_profiling.ProfileReport(data_raw)
```

2.4 Visualização

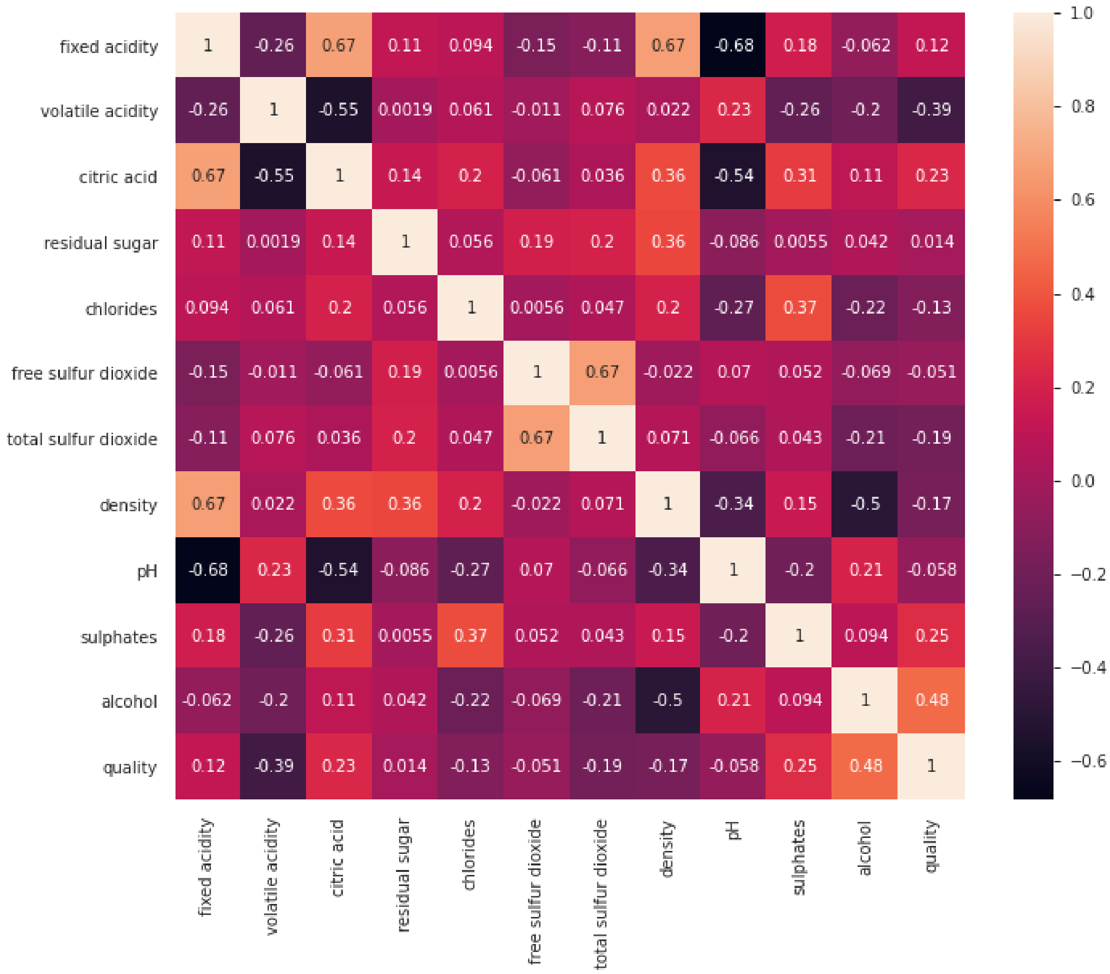
[7]:
```python
# sns.pairplot(data_raw)
```

[8]:
```python
plt.clf()
```

```
<Figure size 432x288 with 0 Axes>
```

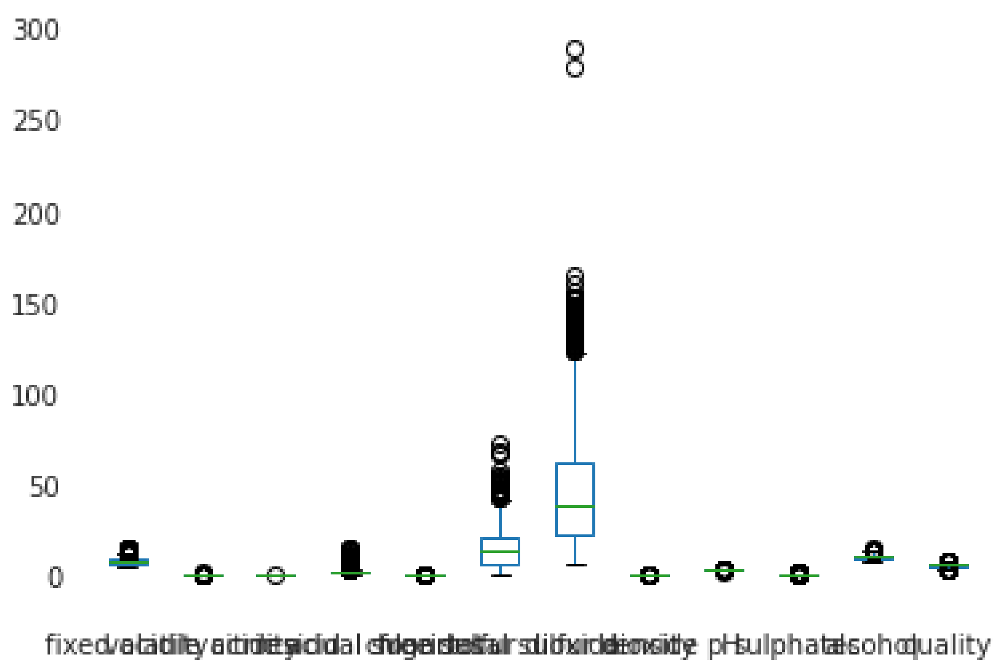
[9]:
```python
plt.subplots(figsize=(11, 9))
sns.heatmap(data_raw.corr(), annot=True)
```

[9]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f73df4fbb38>
```



[10]:
```
data_raw.plot.box()
```

[10]: <matplotlib.axes._subplots.AxesSubplot at 0x7f73dcbd6898>



2.5 Escalonando

[11]:
```
scaler = StandardScaler().fit(data_raw)
data_scaled = scaler.transform(data_raw)
```

[12]:
```
data_scaled_df = pd.DataFrame(data_scaled, columns=data_raw.columns)
```

[13]:
```
data_scaled_df.head()
```

[13]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	\
0	-0.528360	0.961877	-1.391472	-0.453218	-0.243707	
1	-0.298547	1.967442	-1.391472	0.043416	0.223875	
2	-0.298547	1.297065	-1.186070	-0.169427	0.096353	
3	1.654856	-1.384443	1.484154	-0.453218	-0.264960	
4	-0.528360	0.961877	-1.391472	-0.453218	-0.243707	

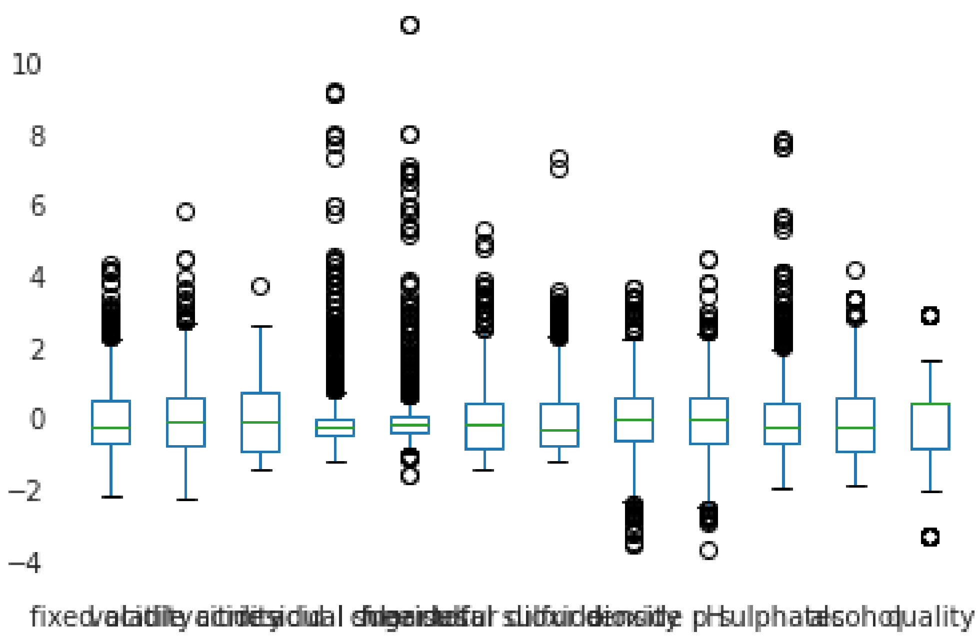
	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	\
0	-0.466193	-0.379133	0.558274	1.288643	-0.579207	
1	0.872638	0.624363	0.028261	-0.719933	0.128950	
2	-0.083669	0.229047	0.134264	-0.331177	-0.048089	
3	0.107592	0.411500	0.664277	-0.979104	-0.461180	
4	-0.466193	-0.379133	0.558274	1.288643	-0.579207	

alcohol	quality

```
0 -0.960246 -0.787823
1 -0.584777 -0.787823
2 -0.584777 -0.787823
3 -0.584777  0.450848
4 -0.960246 -0.787823
```

[14]:
```python
data_scaled_df.plot.box()
```

[14]: <matplotlib.axes._subplots.AxesSubplot at 0x7f73dc2a1518>



2.6 Utilidades

[15]:
```python
lista_metricas_treino = []
lista_metricas_teste = []
```

[16]:
```python
def metricas(y_true, y_pred, alg):
    r2 = r2_score(y_true, y_pred)
    eqm = mean_squared_error(y_true, y_pred)
    seq = len(y_true)*eqm
    reqm = math.sqrt(eqm)

    return {'Algoritmo':alg, 'R2':r2, 'EQM':eqm, 'REQM':reqm, 'SEQ':seq}
```

2.7 Separando conjuntos de Treino e Teste

Para a separação utilizou-se do train_test_split que divide o conjunto em treino e teste aleatóriamente

[17]:
```
test_attr = 'fixed acidity';
output_attr = 'quality';
train, test = train_test_split(data_scaled_df, test_size = 0.2, shuffle=True)

x_train = train.drop(columns=[output_attr])
y_train = train[output_attr]

x_test = test.drop(columns=[output_attr])
y_test = test[output_attr]
```

2.8 Aplicando a Regressão

2.8.1 Regressão Linear

[18]:
```
lire = LinearRegression()
```

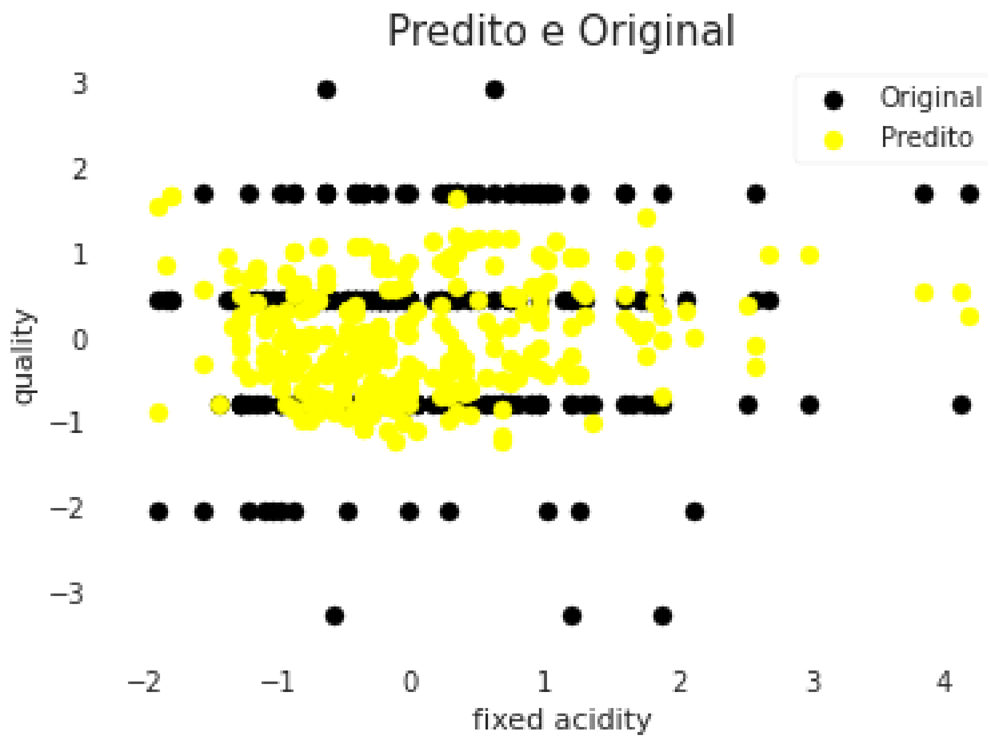
[19]:
```
lire.fit(x_train, y_train)
```

[19]:
```
LinearRegression()
```

2.9 Avaliação para Teste

[20]:
```
y_pred = lire.predict(x_test)
linear_metricas = metricas(y_test, y_pred, 'Regressão Linear - Teste')
lista_metricas_teste.append(linear_metricas)
```

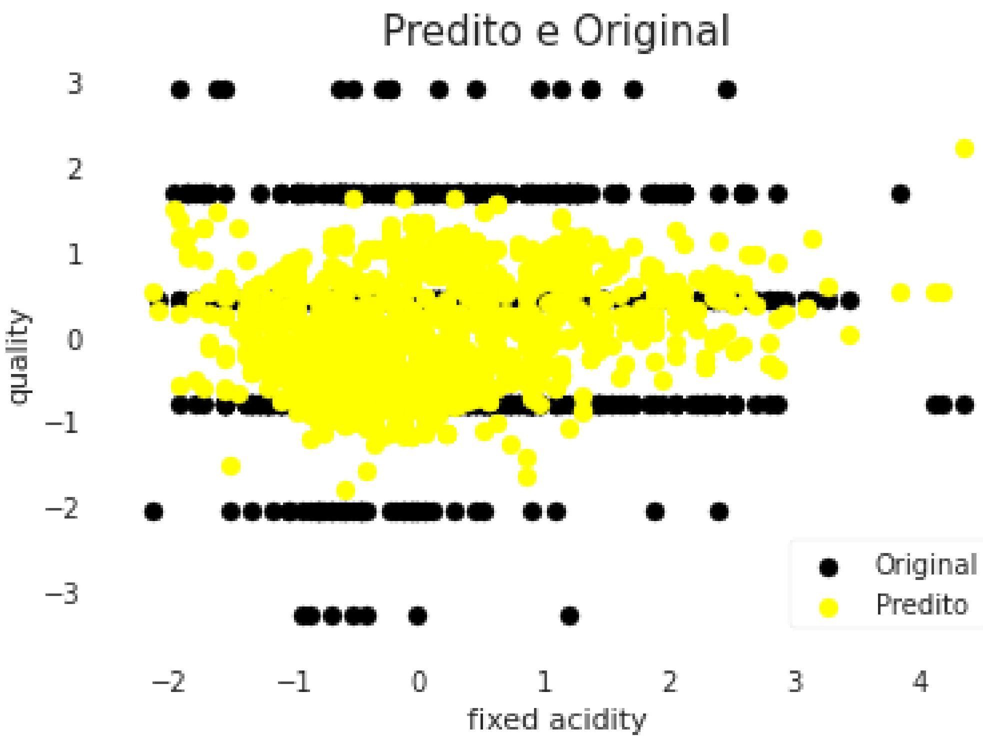
[21]:
```
plt.scatter(x_test[test_attr], y_test,  color='black')
plt.scatter(x_test[test_attr], y_pred, color='yellow')
plt.xlabel(test_attr)
plt.ylabel(output_attr)
plt.title('Predito e Original',fontsize=15)
plt.legend(['Original', 'Predito'])
plt.show()
```



2.10 Avaliação para Treino

[22]:
```
y_pred = lire.predict(x_train)
linear_metricas = metricas(y_train, y_pred, 'Regressão Linear - Treino')
lista_metricas_treino.append(linear_metricas)
```

[23]:
```
plt.scatter(x_train[test_attr], y_train,  color='black')
plt.scatter(x_train[test_attr], y_pred, color='yellow')
plt.xlabel(test_attr)
plt.ylabel(output_attr)
plt.title('Predito e Original',fontsize=15)
plt.legend(['Original', 'Predito'])
plt.show()
```



2.11 SVR

2.11.1 Kernel RBF

[24]:
```
svr_reg = SVR(kernel='rbf')
```

[25]:
```
svr_reg.fit(x_train, y_train)
```

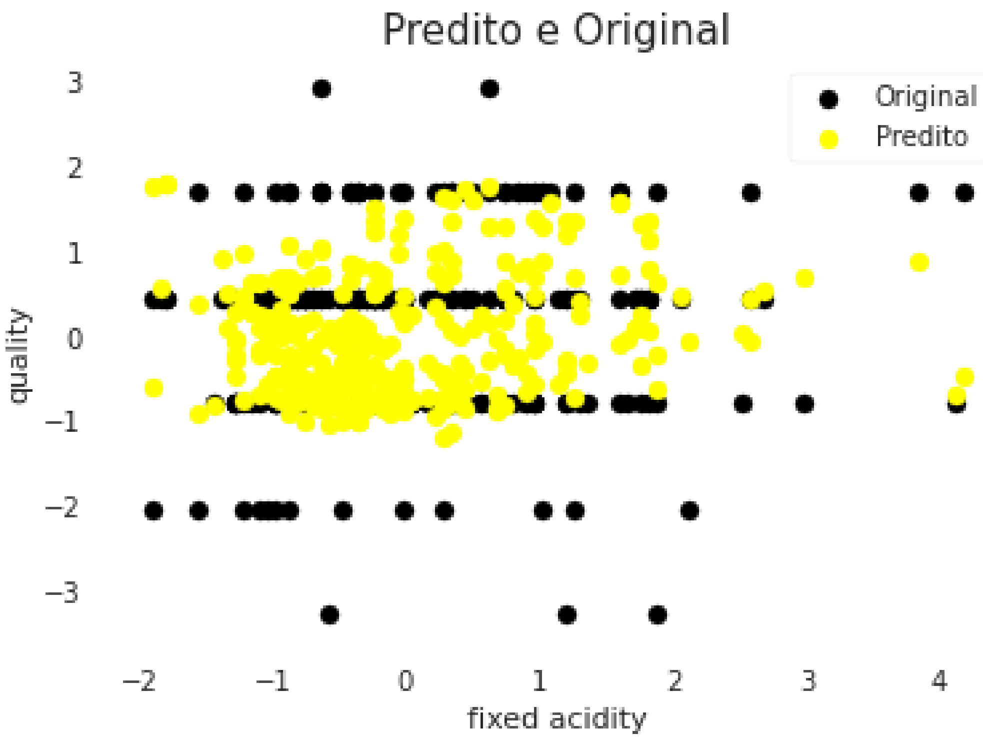
[25]: SVR()

2.12 Avaliação para Teste

[26]:
```
y_pred = svr_reg.predict(x_test)
svr_metricas = metricas(y_test, y_pred, 'SVR - RBF - Teste')
lista_metricas_teste.append(svr_metricas)
```

[27]:
```
plt.scatter(x_test[test_attr], y_test,  color='black')
plt.scatter(x_test[test_attr], y_pred, color='yellow')
plt.xlabel(test_attr)
plt.ylabel(output_attr)
```

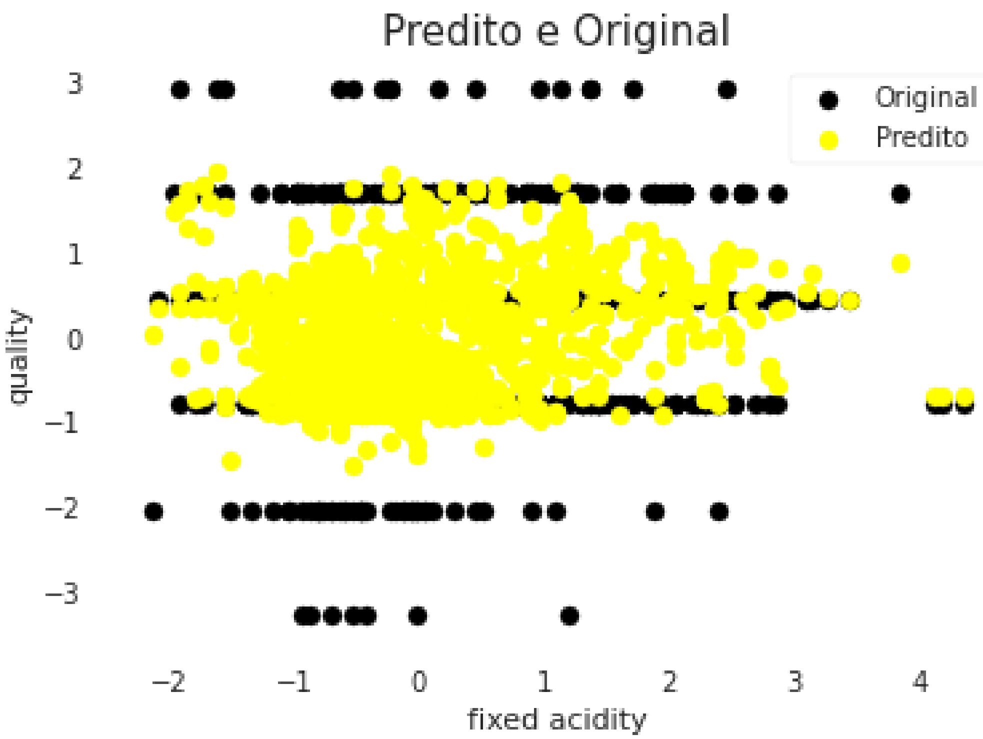
```
plt.title('Predito e Original',fontsize=15)
plt.legend(['Original', 'Predito'])
plt.show()
```



2.13 Avaliação para Treino

[28]:
```
y_pred = svr_reg.predict(x_train)
svr_metricas = metricas(y_train, y_pred, 'SVR - RBF - Treino')
lista_metricas_treino.append(svr_metricas)
```

[29]:
```
plt.scatter(x_train[test_attr], y_train,  color='black')
plt.scatter(x_train[test_attr], y_pred, color='yellow')
plt.xlabel(test_attr)
plt.ylabel(output_attr)
plt.title('Predito e Original',fontsize=15)
plt.legend(['Original', 'Predito'])
plt.show()
```



2.13.1 Kernel Linear

[30]:
```
svr_reg = SVR(kernel='linear')
```

[31]:
```
svr_reg.fit(x_train, y_train)
```

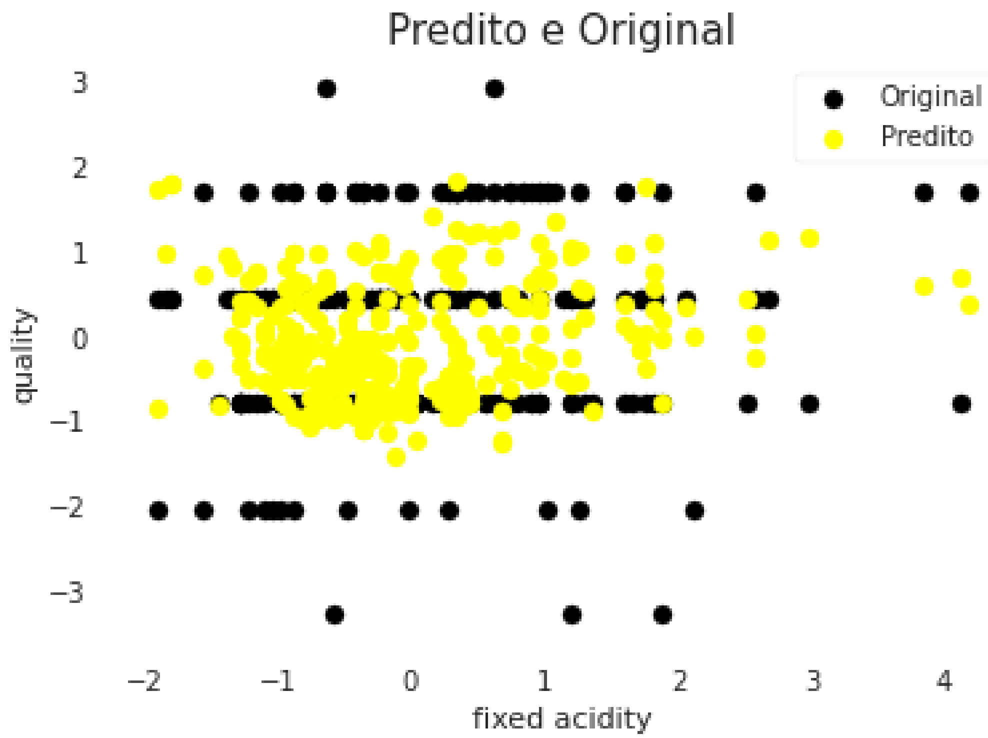
[31]:
```
SVR(kernel='linear')
```

2.14 Avaliação para Teste

[32]:
```
y_pred = svr_reg.predict(x_test)
metricas_svr = metricas(y_test, y_pred, 'SVR - Linear - Teste')
lista_metricas_teste.append(metricas_svr)
```

[33]:
```
plt.scatter(x_test[test_attr], y_test,  color='black')
plt.scatter(x_test[test_attr], y_pred, color='yellow')
plt.xlabel(test_attr)
plt.ylabel(output_attr)
plt.title('Predito e Original',fontsize=15)
plt.legend(['Original', 'Predito'])
```

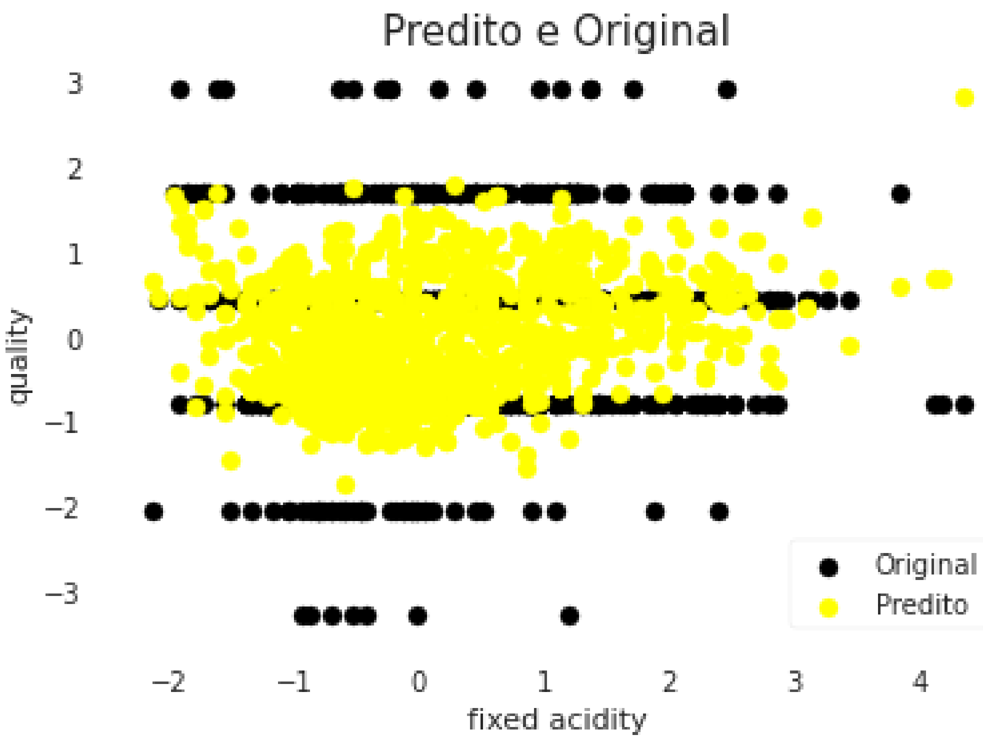
```
plt.show()
```



2.15 Avaliação para Treino

[34]:
```
y_pred = svr_reg.predict(x_train)
svr_metricas = metricas(y_train, y_pred, 'SVR - Linear - Treino')
lista_metricas_treino.append(svr_metricas)
```

[35]:
```
plt.scatter(x_train[test_attr], y_train,  color='black')
plt.scatter(x_train[test_attr], y_pred, color='yellow')
plt.xlabel(test_attr)
plt.ylabel(output_attr)
plt.title('Predito e Original',fontsize=15)
plt.legend(['Original', 'Predito'])
plt.show()
```



2.15.1 Kernel Sigmoide

[36]:
```
train, test = train_test_split(data_raw, test_size = 0.2, shuffle=True)

x_train_sig = train.drop(columns=[output_attr])
y_train_sig  = train[output_attr]

x_test_sig  = test.drop(columns=[output_attr])
y_test_sig  = test[output_attr]
```

[37]:
```
svr_reg = SVR(kernel='sigmoid')
```

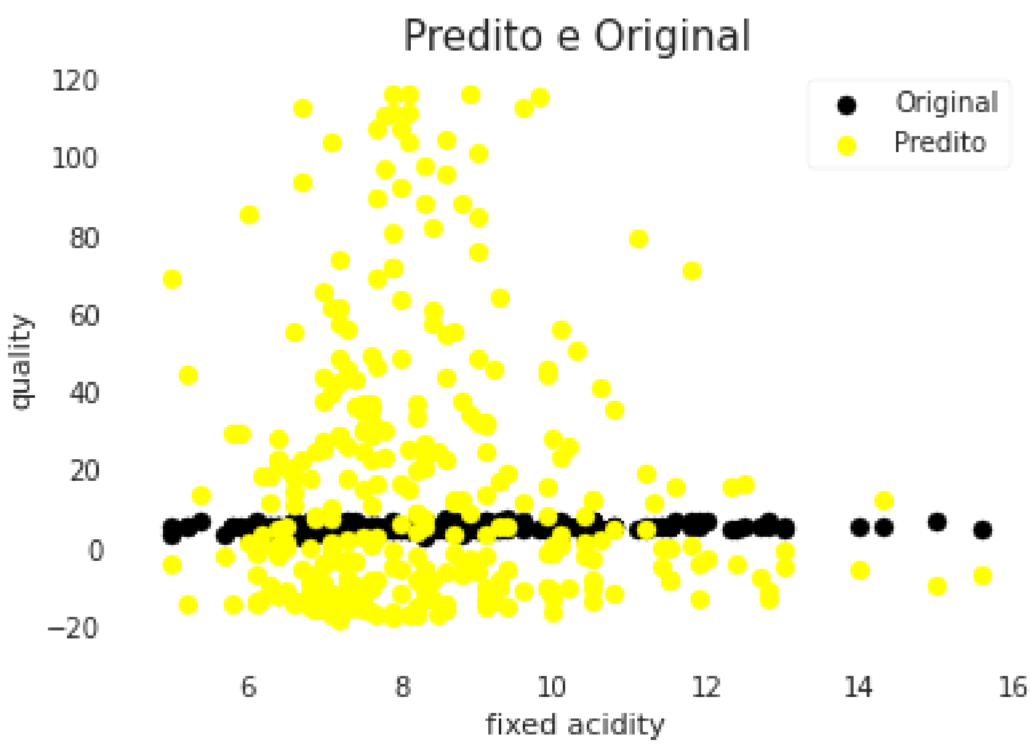
[38]:
```
svr_reg.fit(x_train_sig , y_train_sig )
```

[38]:
```
SVR(kernel='sigmoid')
```

2.16 Avaliação para Teste

[39]:
```
y_pred_sig  = svr_reg.predict(x_test_sig)
metricas_svr = metricas(y_test_sig , y_pred_sig , 'SVR - Sigmoide - Teste')
lista_metricas_teste.append(metricas_svr)
```

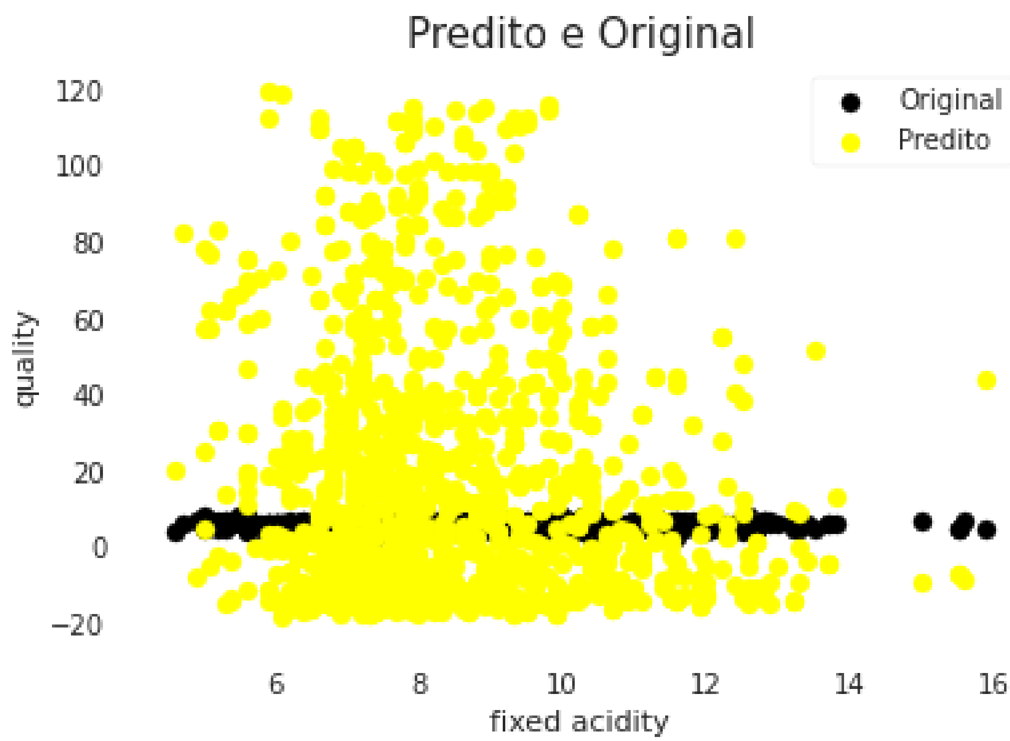
[40]:
```
plt.scatter(x_test_sig [test_attr], y_test_sig ,  color='black')
plt.scatter(x_test_sig [test_attr], y_pred_sig , color='yellow')
plt.xlabel(test_attr)
plt.ylabel(output_attr)
plt.title('Predito e Original',fontsize=15)
plt.legend(['Original', 'Predito'])
plt.show()
```



2.17 Avaliação para Treino

[41]:
```
y_pred_sig  = svr_reg.predict(x_train_sig)
svr_metricas = metricas(y_train_sig , y_pred_sig , 'SVR - Sigmoide - Treino')
lista_metricas_treino.append(svr_metricas)
```

[42]:
```
plt.scatter(x_train_sig [test_attr], y_train_sig ,  color='black')
plt.scatter(x_train_sig [test_attr], y_pred_sig , color='yellow')
plt.xlabel(test_attr)
plt.ylabel(output_attr)
plt.title('Predito e Original',fontsize=15)
plt.legend(['Original', 'Predito'])
plt.show()
```



2.17.1 Kernel Polinomial

[43]:
```
svr_reg = SVR(kernel='poly', degree=3)
```

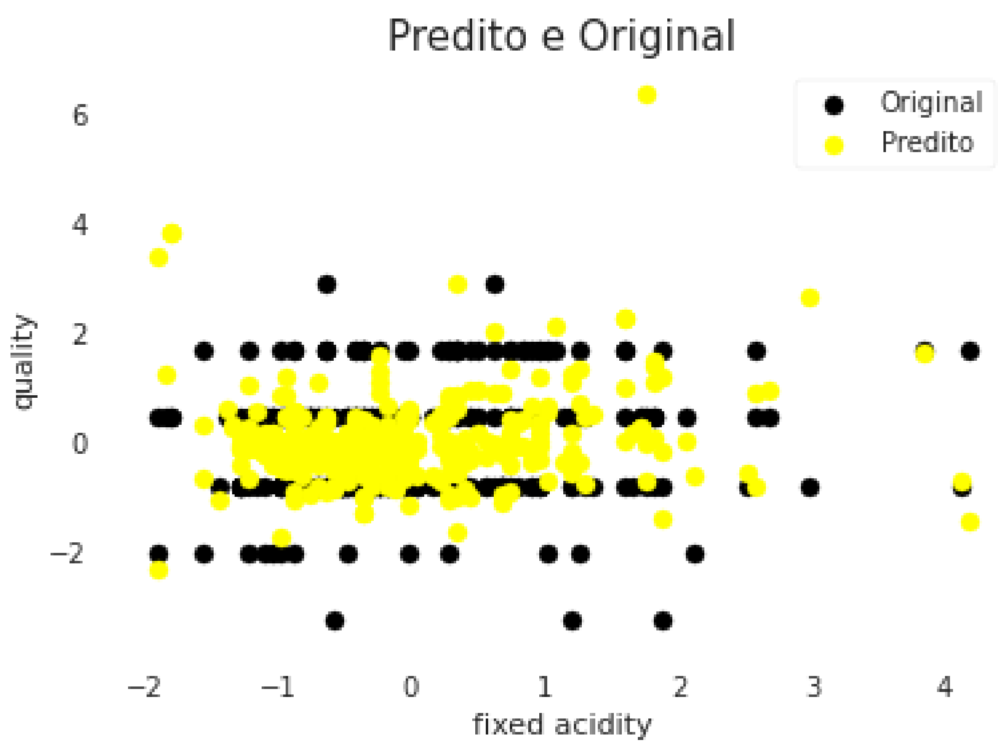
[44]:
```
svr_reg.fit(x_train, y_train)
```

[44]:
```
SVR(kernel='poly')
```

2.18 Avaliação para Teste

[45]:
```
y_pred = svr_reg.predict(x_test)
svr_metricas = metricas(y_test, y_pred, 'SVR - Polinomial - Teste')
lista_metricas_teste.append(svr_metricas)
```

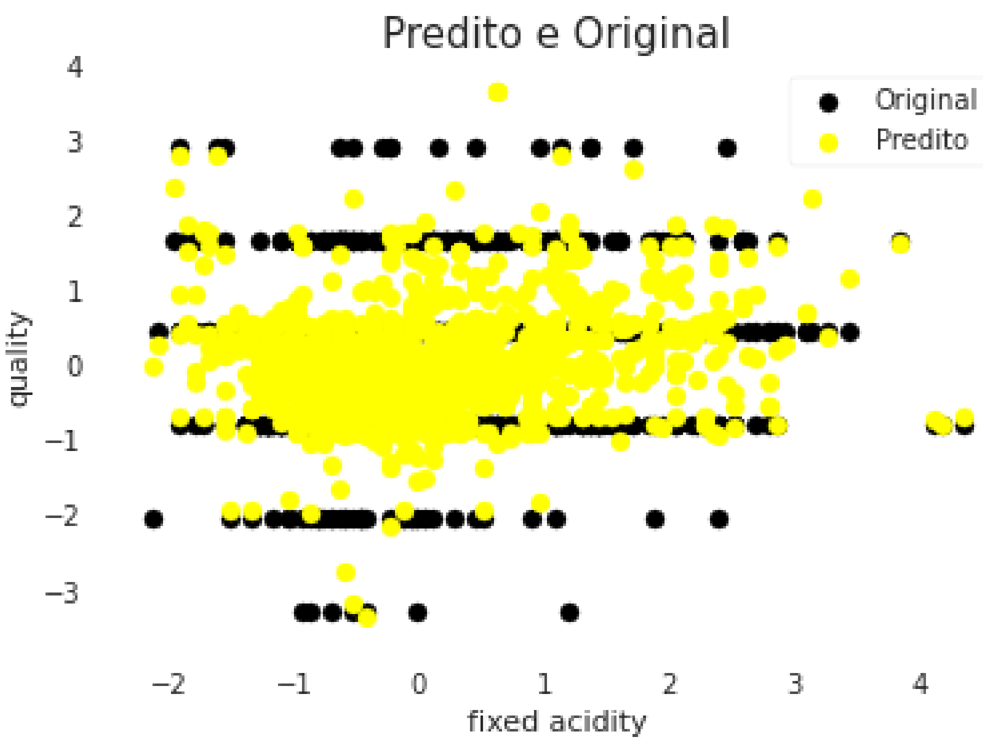
[46]:
```
plt.scatter(x_test[test_attr], y_test,  color='black')
plt.scatter(x_test[test_attr], y_pred, color='yellow')
plt.xlabel(test_attr)
plt.ylabel(output_attr)
plt.title('Predito e Original',fontsize=15)
plt.legend(['Original', 'Predito'])
plt.show()
```



2.19 Avaliação para Treino

[47]:
```
y_pred = svr_reg.predict(x_train)
svr_metricas = metricas(y_train, y_pred, 'SVR - Polinomial - Treino')
lista_metricas_treino.append(svr_metricas)
```

[48]:
```
plt.scatter(x_train[test_attr], y_train,  color='black')
plt.scatter(x_train[test_attr], y_pred, color='yellow')
plt.xlabel(test_attr)
plt.ylabel(output_attr)
plt.title('Predito e Original',fontsize=15)
plt.legend(['Original', 'Predito'])
plt.show()
```



2.20 Redes Neurais

2.20.1 Kernel Linear

[49]:
```
mlp_reg = MLPRegressor()
```

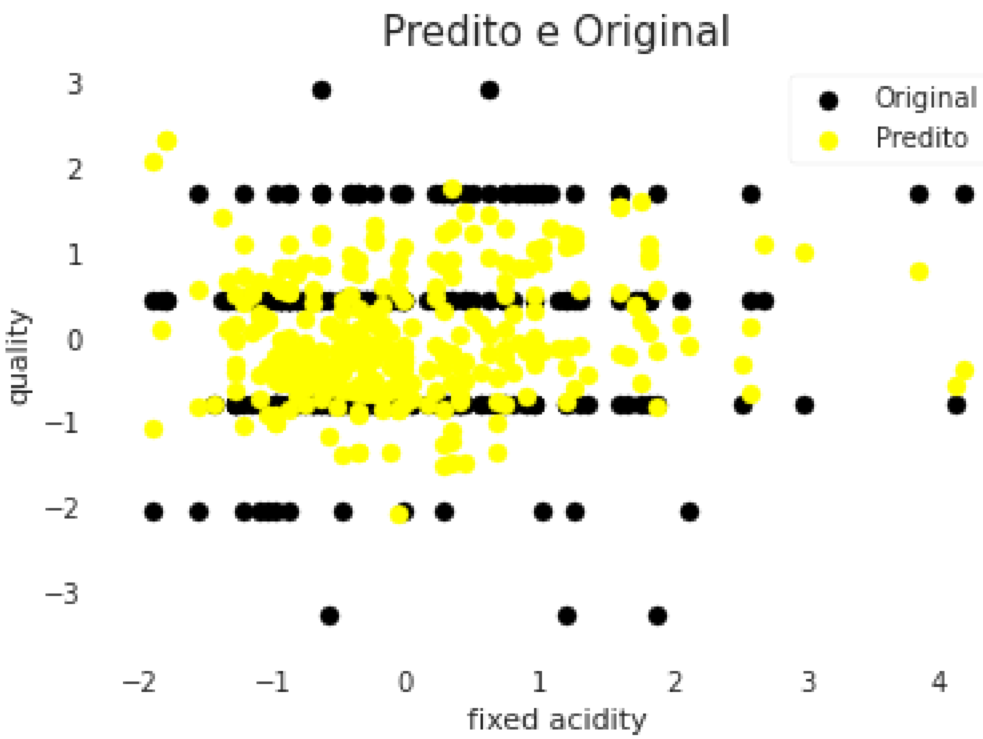
[50]:
```
mlp_reg.fit(x_train, y_train)
```

[50]:
```
MLPRegressor()
```

2.21 Avaliação para Teste

[51]:
```
y_pred = mlp_reg.predict(x_test)
mlp_metricas = metricas(y_test, y_pred, 'MLP - Teste')
lista_metricas_teste.append(mlp_metricas)
```

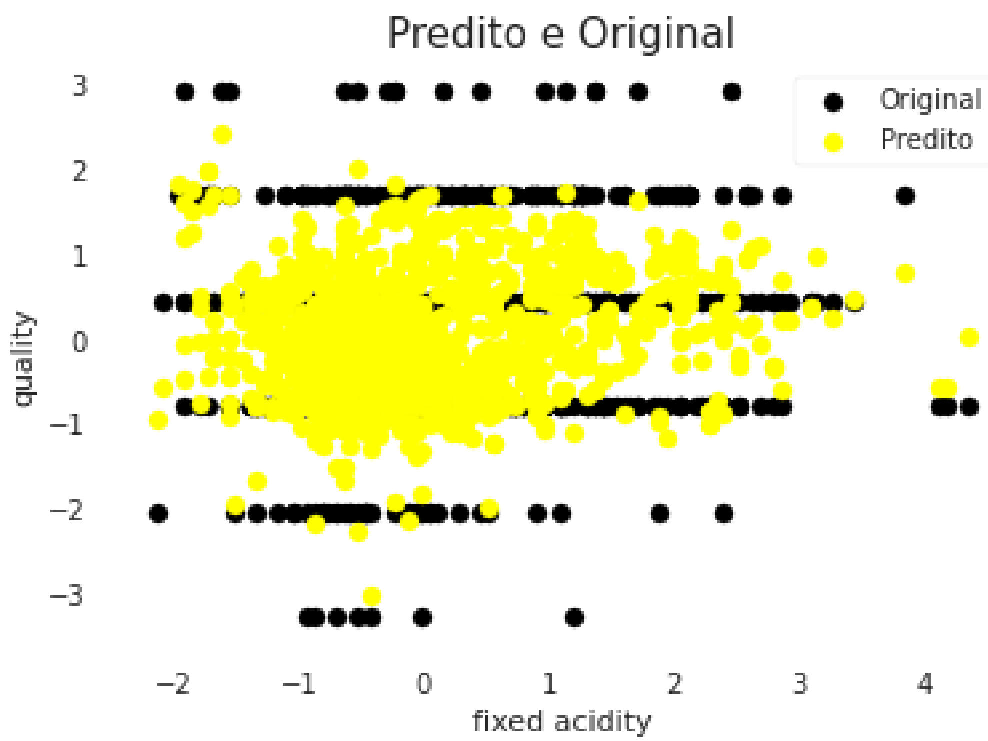
[52]:
```
plt.scatter(x_test[test_attr], y_test,  color='black')
plt.scatter(x_test[test_attr], y_pred, color='yellow')
plt.xlabel(test_attr)
plt.ylabel(output_attr)
plt.title('Predito e Original',fontsize=15)
plt.legend(['Original', 'Predito'])
plt.show()
```



2.22 Avaliação para Treino

[53]:
```
y_pred = mlp_reg.predict(x_train)
mlp_metricas = metricas(y_train, y_pred, 'MLP - Treino')
lista_metricas_treino.append(mlp_metricas)
```

[54]:
```
plt.scatter(x_train[test_attr], y_train,  color='black')
plt.scatter(x_train[test_attr], y_pred, color='yellow')
plt.xlabel(test_attr)
plt.ylabel(output_attr)
plt.title('Predito e Original',fontsize=15)
plt.legend(['Original', 'Predito'])
plt.show()
```



3 Resultados

[55]:
```
metricas_teste = pd.DataFrame(lista_metricas_teste)
metricas_teste
```

[55]:

	Algoritmo	R2	EQM	REQM	\
0	Regressão Linear - Teste	0.322617	0.708590	0.841778	
1	SVR - RBF - Teste	0.385657	0.642646	0.801652	
2	SVR - Linear - Teste	0.303653	0.728427	0.853480	
3	SVR - Sigmoide - Teste	-1955.626553	1316.117972	36.278340	
4	SVR - Polinomial - Teste	0.099509	0.941977	0.970555	
5	MLP - Teste	0.375016	0.653777	0.808565	

	SEQ
0	226.748957
1	205.646758
2	233.096752
3	421157.751023
4	301.432642
5	209.208543

[56]:
```
metricas_teste = round(metricas_teste, 3)
```

[57]:
```
metricas_teste
```

[57]:

	Algoritmo	R2	EQM	REQM	SEQ
0	Regressão Linear - Teste	0.323	0.709	0.842	226.749
1	SVR - RBF - Teste	0.386	0.643	0.802	205.647
2	SVR - Linear - Teste	0.304	0.728	0.853	233.097
3	SVR - Sigmoide - Teste	-1955.627	1316.118	36.278	421157.751
4	SVR - Polinomial - Teste	0.100	0.942	0.971	301.433
5	MLP - Teste	0.375	0.654	0.809	209.209

[58]:
```
metricas_teste.to_excel('wine_metricas_teste.xlsx')
```

[59]:
```
metricas_treino = pd.DataFrame(lista_metricas_treino)
metricas_treino
```

[59]:

	Algoritmo	R2	EQM	REQM	\
0	Regressão Linear - Treino	0.366947	0.625541	0.790911	
1	SVR - RBF - Treino	0.535704	0.458786	0.677337	
2	SVR - Linear - Treino	0.354539	0.637801	0.798625	
3	SVR - Sigmoide - Treino	-2007.247559	1298.276549	36.031605	
4	SVR - Polinomial - Treino	0.472925	0.520820	0.721679	
5	MLP - Treino	0.592636	0.402530	0.634453	

	SEQ
0	8.000668e+02
1	5.867869e+02
2	8.157479e+02
3	1.660496e+06
4	6.661287e+02
5	5.148359e+02

[60]:
```
metricas_treino = round(metricas_treino, 3)
```

[61]:
```
metricas_treino.to_excel('wine.xlsx')
```