notebook04-avaliando_a_generalizacao_de_algoritmos

November 24, 2020

1 Avaliando a generalização de algoritmos

1.1 Dados do Curso

Instituição: IFES

Curso: Mestrado Profissional Computação Aplicada

Professor: Francisco de Assis Boldt

Aluno: Arthur Chisté Lucas

1.2 Ambiente

IDE: MS Visual Studio Code

Versão Python: 3.8.3 64bits com anaconda 2020.07

1.3 Introdução

Nesta tarefa, será utilizado um dataset contendo a classificação de celulares obtido na Kaggle

https://www.kaggle.com/iabhishekofficial/mobile-price-classification

O dataset foi baixado e armazenado em meu GitHub público, sendo acessado diretamente de lá, conforme a URL abaixo:

https://github.com/arthurclucas/ReconhecimentoPadroes/blob/main/data/mobile_price_classification/train.csv

Nesta tarefa, faremos a transformação de dados por meio dos scalers StandardScaler, RobustScaler e MinMaxScaler dentro e fora do pipeline e o cross validation dos dados com os cross validators TimeSeriesSplit, KFold, ShuffleSplit, StratifiedKFold, StratifiedShuffleSplit.

[99]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import pylab as pl
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import (cross_validate, TimeSeriesSplit, KFold,␣
 ↪ShuffleSplit, StratifiedKFold, StratifiedShuffleSplit)
from sklearn.preprocessing import (StandardScaler, RobustScaler, MinMaxScaler)
from sklearn.pipeline import Pipeline
from sklearn.neighbors import KNeighborsClassifier
```

[100]:
```
url = 'https://github.com/arthurclucas/ReconhecimentoPadroes/blob/main/data/
 ↪mobile_price_classification/train.csv?raw=true'
dados = pd.read_csv(url)
dados.head(5)
```

[100]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	\
0	842	0	2.2	0	1	0	7	0.6	
1	1021	1	0.5	1	0	1	53	0.7	
2	563	1	0.5	1	2	1	41	0.9	
3	615	1	2.5	0	0	0	10	0.8	
4	1821	1	1.2	0	13	1	44	0.6	

	mobile_wt	n_cores	…	px_height	px_width	ram	sc_h	sc_w	talk_time	\
0	188	2	…	20	756	2549	9	7	19	
1	136	3	…	905	1988	2631	17	3	7	
2	145	5	…	1263	1716	2603	11	2	9	
3	131	6	…	1216	1786	2769	16	8	11	
4	141	2	…	1208	1212	1411	8	2	15	

	three_g	touch_screen	wifi	price_range
0	0	0	1	1
1	1	1	0	2
2	1	1	0	2
3	1	0	0	2
4	1	1	0	1

[5 rows x 21 columns]

[101]:
```
dados.info()
```

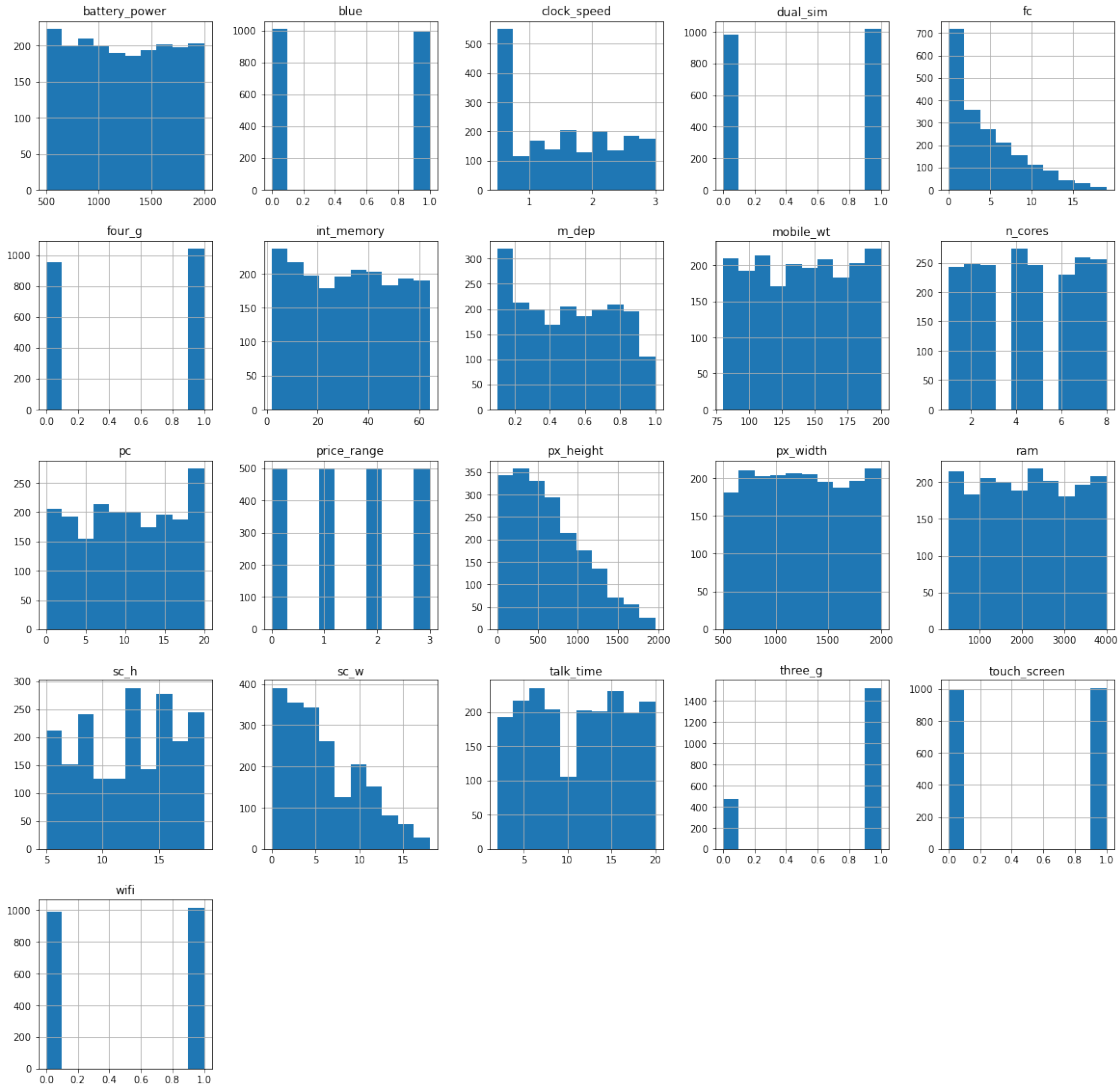
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   battery_power  2000 non-null   int64
 1   blue           2000 non-null   int64
 2   clock_speed    2000 non-null   float64
 3   dual_sim       2000 non-null   int64
 4   fc             2000 non-null   int64
 5   four_g         2000 non-null   int64
 6   int_memory     2000 non-null   int64
 7   m_dep          2000 non-null   float64
 8   mobile_wt      2000 non-null   int64
 9   n_cores        2000 non-null   int64
 10  pc             2000 non-null   int64
 11  px_height      2000 non-null   int64
 12  px_width       2000 non-null   int64
```

```
 13  ram            2000 non-null   int64
 14  sc_h           2000 non-null   int64
 15  sc_w           2000 non-null   int64
 16  talk_time      2000 non-null   int64
 17  three_g        2000 non-null   int64
 18  touch_screen   2000 non-null   int64
 19  wifi           2000 non-null   int64
 20  price_range    2000 non-null   int64
dtypes: float64(2), int64(19)
memory usage: 328.2 KB
```

Segue histograma de como os dados se encontram na base

[102]:
```
fig = plt.figure(figsize = (20,20))
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
ax = fig.gca()
dados.hist(ax = ax)
```

[102]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000001566A4E6580>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000015667185760>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001566A88ED30>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001566A9B1820>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000156668A6B50>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000001566A345040>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001566A345FD0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001566ACE94C0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001566AB32CD0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001566B9D5160>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x00000156667395B0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001566B842A00>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001566B4D7E50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001566B3F82E0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001566B57D730>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000001566B6EDB80>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001566BA6CFD0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001566B50B460>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001566B5598B0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001566B5E2D00>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000001566B695190>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001566B71A520>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001566B79D7F0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001566B8A6F70>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001566B8BC730>]],
      dtype=object)
```



Declara as instâncias dos Scalers, Cross Validator e modelos de classificação que serão utilizados.

A variável **Xs** terá uma lista de tuplas no formado (descrição, dados) gerados após tratamento pelos Scalers

[103]:
```
n_splits=5

cvs = [KFold(n_splits=n_splits), ShuffleSplit(n_splits=n_splits),␣
 ↪StratifiedKFold(n_splits=n_splits),␣
 ↪StratifiedShuffleSplit(n_splits=n_splits),␣
 ↪TimeSeriesSplit(n_splits=n_splits)]
cv_names = []

for cv in cvs:
```

```
    cv_names.append(type(cv).__name__)

scs = [StandardScaler(), RobustScaler(), MinMaxScaler()]
sc_names = []
sc_names.append('NoScaler')
for sc in scs:
    sc_names.append(type(sc).__name__)

modelos = [LogisticRegression(), KNeighborsClassifier(n_neighbors=7)]
model_names = []
for modelo in modelos:
    model_names.append(type(modelo).__name__)

Xs = []

final = []
final_cols = ['modelo', 'cross_validator', 'scaler', 'mean', 'pipeline']

y = dados['price_range']
X = dados.drop('price_range', axis = 1)
```

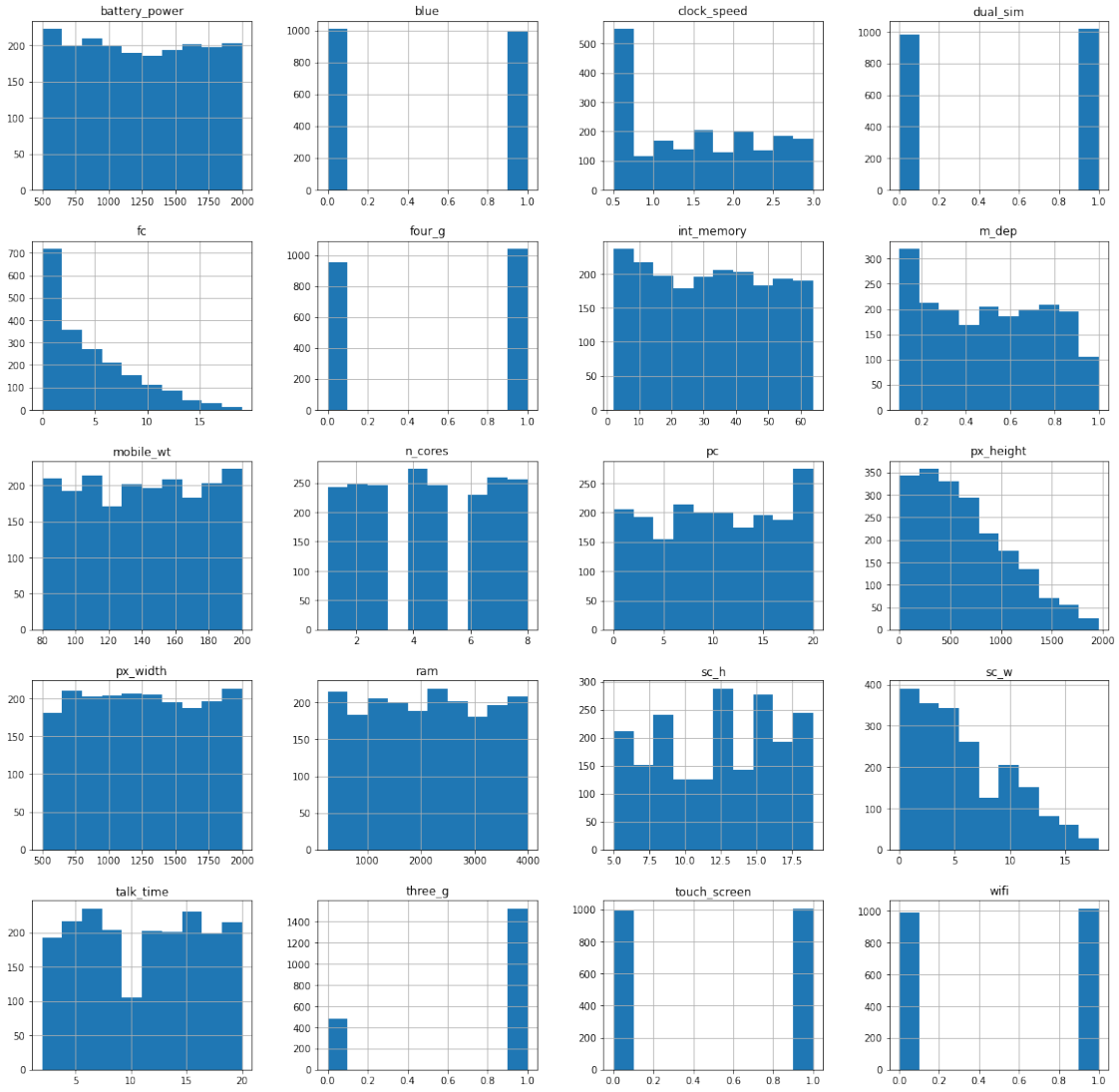
[104]:
```
Xs.clear()
Xs.append(('NoScaler', X))

for sc in scs:
    Xs.append((type(sc).__name__, sc.fit_transform(X)))
```

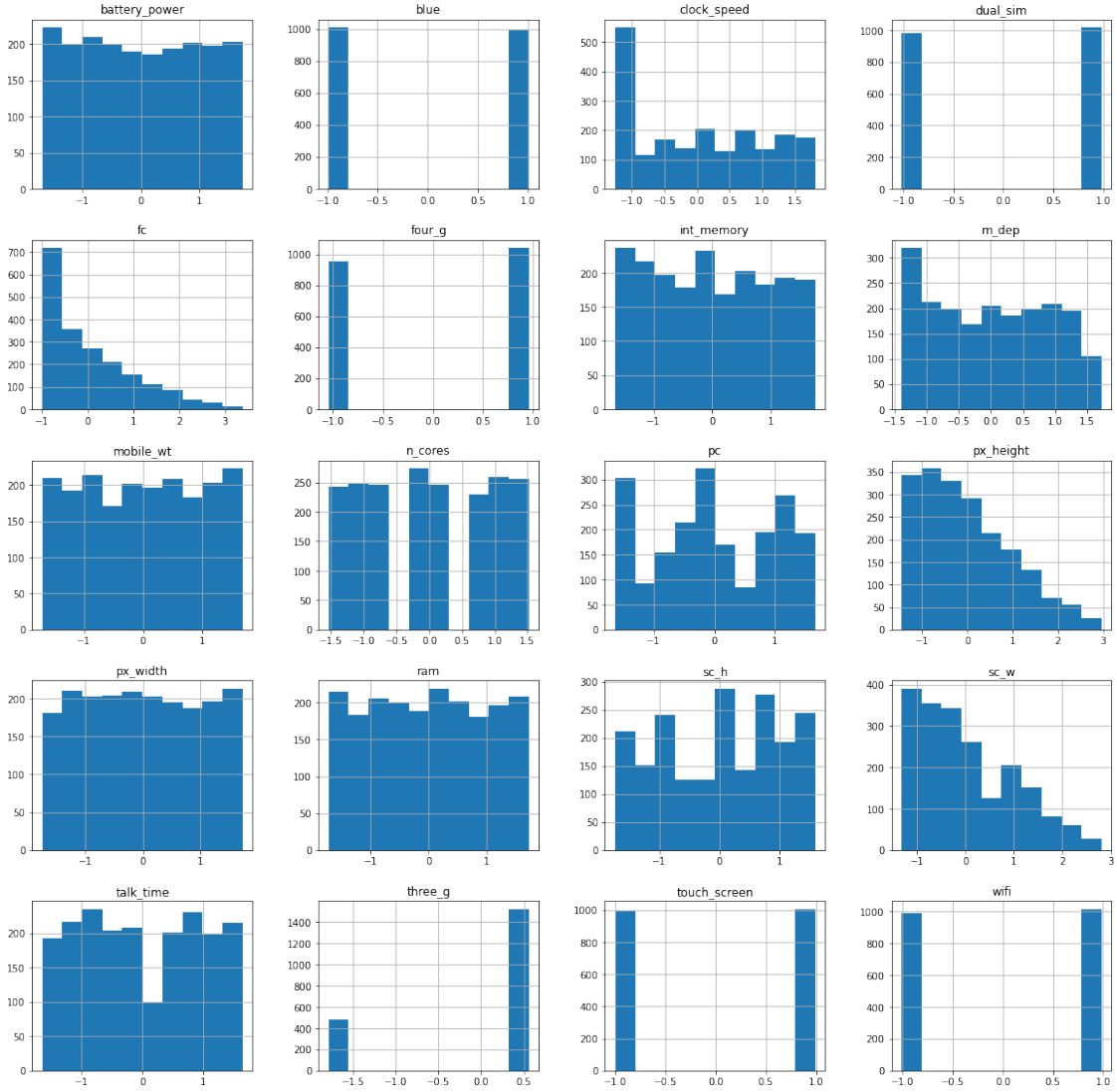
Imprime os histogramas dos dados tratados por cada um dos scalers

[105]:
```
columns = dados.columns.drop('price_range')
for Xi in Xs:
    df = pd.DataFrame(data=Xi[1], columns=columns)
    fig = plt.figure(figsize = (20,20))
    plt.xticks(fontsize=12)
    plt.yticks(fontsize=12)
    ax = fig.gca()
    df.hist(ax = ax)
    pl.suptitle(Xi[0])
```

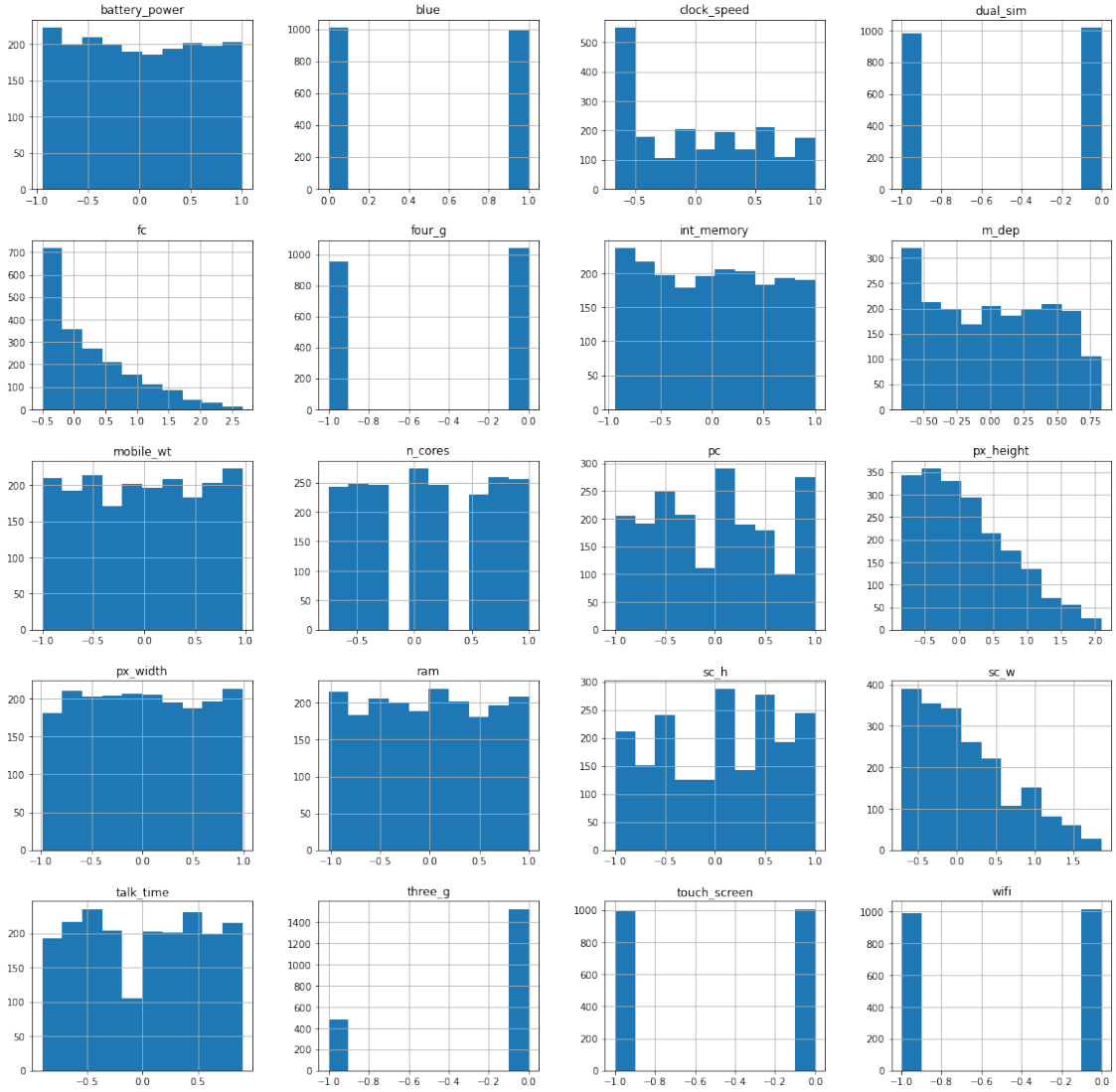
NoScaler


battery_power
blue
clock_speed
dual_sim
fc
four_g
int_memory
m_dep
mobile_wt
n_cores
pc
px_height
px_width
ram
sc_h
sc_w
talk_time
three_g
touch_screen
wifi

StandardScaler



RobustScaler



[106]:
```
final.clear()
for cv in cvs:
    for modelo in modelos:
        for Xi in Xs:
            mean = 0
            for i in range(10):
                scores = cross_validate(modelo, Xi[1], y, cv=cv)
                mean += np.mean(scores['test_score'])
            mean = mean/10
```

```
            final.append((type(modelo).__name__, type(cv).__name__, Xi[0],␣
 ↪mean, False))

for cv in cvs:
    for modelo in modelos:
        for sc in scs:
            mean = 0
            for i in range(10):
                pipeline =  Pipeline([("padronizacao", sc), ("classificador",␣
 ↪modelo)])
                scores = cross_validate(modelo, X, y, cv=cv)
                mean += np.mean(scores['test_score'])
            mean = mean /10
            final.append((type(modelo).__name__, type(cv).__name__, type(sc).
 ↪__name__, mean, True))
```

[107]:
```
df = pd.DataFrame(data=final, columns=final_cols)
```

[108]:
```
df1 = df[df['pipeline']]
df1.head(50)
```

[108]:

	modelo	cross_validator	scaler	mean	\
40	LogisticRegression	KFold	StandardScaler	0.628000	
41	LogisticRegression	KFold	RobustScaler	0.628000	
42	LogisticRegression	KFold	MinMaxScaler	0.628000	
43	KNeighborsClassifier	KFold	StandardScaler	0.924000	
44	KNeighborsClassifier	KFold	RobustScaler	0.924000	
45	KNeighborsClassifier	KFold	MinMaxScaler	0.924000	
46	LogisticRegression	ShuffleSplit	StandardScaler	0.635700	
47	LogisticRegression	ShuffleSplit	RobustScaler	0.630200	
48	LogisticRegression	ShuffleSplit	MinMaxScaler	0.632400	
49	KNeighborsClassifier	ShuffleSplit	StandardScaler	0.925500	
50	KNeighborsClassifier	ShuffleSplit	RobustScaler	0.925200	
51	KNeighborsClassifier	ShuffleSplit	MinMaxScaler	0.925400	
52	LogisticRegression	StratifiedKFold	StandardScaler	0.637000	
53	LogisticRegression	StratifiedKFold	RobustScaler	0.637000	
54	LogisticRegression	StratifiedKFold	MinMaxScaler	0.637000	
55	KNeighborsClassifier	StratifiedKFold	StandardScaler	0.925000	
56	KNeighborsClassifier	StratifiedKFold	RobustScaler	0.925000	
57	KNeighborsClassifier	StratifiedKFold	MinMaxScaler	0.925000	
58	LogisticRegression	StratifiedShuffleSplit	StandardScaler	0.638600	
59	LogisticRegression	StratifiedShuffleSplit	RobustScaler	0.640500	
60	LogisticRegression	StratifiedShuffleSplit	MinMaxScaler	0.627600	
61	KNeighborsClassifier	StratifiedShuffleSplit	StandardScaler	0.924800	
62	KNeighborsClassifier	StratifiedShuffleSplit	RobustScaler	0.932400	
63	KNeighborsClassifier	StratifiedShuffleSplit	MinMaxScaler	0.925200	
64	LogisticRegression	TimeSeriesSplit	StandardScaler	0.622823	

```
65    LogisticRegression         TimeSeriesSplit    RobustScaler  0.622823
66    LogisticRegression         TimeSeriesSplit    MinMaxScaler  0.622823
67  KNeighborsClassifier         TimeSeriesSplit  StandardScaler  0.925526
68  KNeighborsClassifier         TimeSeriesSplit    RobustScaler  0.925526
69  KNeighborsClassifier         TimeSeriesSplit    MinMaxScaler  0.925526

    pipeline
40      True
41      True
42      True
43      True
44      True
45      True
46      True
47      True
48      True
49      True
50      True
51      True
52      True
53      True
54      True
55      True
56      True
57      True
58      True
59      True
60      True
61      True
62      True
63      True
64      True
65      True
66      True
67      True
68      True
69      True
```

[109]:
```python
df2 = df[df['pipeline'] == False]
df2.head(50)

#final_cols = ['i', 'modelo', 'cross_validator', 'scaler', 'mean', 'pipeline']
# for cv in cvs:
#     for modelo in modelos:
#         for sc in scs:
```

```
#             print(df[ (df['modelo'] == type(modelo).__name__) &
↪(df['cross_validator'] == type(cv).__name__) & (df['scaler'] == type(sc).
↪__name__)])
```

[109]:

	modelo	cross_validator	scaler	mean	\
0	LogisticRegression	KFold	NoScaler	0.628000	
1	LogisticRegression	KFold	StandardScaler	0.962500	
2	LogisticRegression	KFold	RobustScaler	0.954000	
3	LogisticRegression	KFold	MinMaxScaler	0.922000	
4	KNeighborsClassifier	KFold	NoScaler	0.924000	
5	KNeighborsClassifier	KFold	StandardScaler	0.518000	
6	KNeighborsClassifier	KFold	RobustScaler	0.544000	
7	KNeighborsClassifier	KFold	MinMaxScaler	0.400000	
8	LogisticRegression	ShuffleSplit	NoScaler	0.631500	
9	LogisticRegression	ShuffleSplit	StandardScaler	0.961800	
10	LogisticRegression	ShuffleSplit	RobustScaler	0.958700	
11	LogisticRegression	ShuffleSplit	MinMaxScaler	0.923000	
12	KNeighborsClassifier	ShuffleSplit	NoScaler	0.925000	
13	KNeighborsClassifier	ShuffleSplit	StandardScaler	0.521900	
14	KNeighborsClassifier	ShuffleSplit	RobustScaler	0.551600	
15	KNeighborsClassifier	ShuffleSplit	MinMaxScaler	0.401700	
16	LogisticRegression	StratifiedKFold	NoScaler	0.637000	
17	LogisticRegression	StratifiedKFold	StandardScaler	0.962500	
18	LogisticRegression	StratifiedKFold	RobustScaler	0.954500	
19	LogisticRegression	StratifiedKFold	MinMaxScaler	0.922500	
20	KNeighborsClassifier	StratifiedKFold	NoScaler	0.925000	
21	KNeighborsClassifier	StratifiedKFold	StandardScaler	0.514500	
22	KNeighborsClassifier	StratifiedKFold	RobustScaler	0.552000	
23	KNeighborsClassifier	StratifiedKFold	MinMaxScaler	0.393500	
24	LogisticRegression	StratifiedShuffleSplit	NoScaler	0.631600	
25	LogisticRegression	StratifiedShuffleSplit	StandardScaler	0.961500	
26	LogisticRegression	StratifiedShuffleSplit	RobustScaler	0.959200	
27	LogisticRegression	StratifiedShuffleSplit	MinMaxScaler	0.924300	
28	KNeighborsClassifier	StratifiedShuffleSplit	NoScaler	0.924300	
29	KNeighborsClassifier	StratifiedShuffleSplit	StandardScaler	0.517800	
30	KNeighborsClassifier	StratifiedShuffleSplit	RobustScaler	0.555400	
31	KNeighborsClassifier	StratifiedShuffleSplit	MinMaxScaler	0.410500	
32	LogisticRegression	TimeSeriesSplit	NoScaler	0.622823	
33	LogisticRegression	TimeSeriesSplit	StandardScaler	0.939940	
34	LogisticRegression	TimeSeriesSplit	RobustScaler	0.909910	
35	LogisticRegression	TimeSeriesSplit	MinMaxScaler	0.860060	
36	KNeighborsClassifier	TimeSeriesSplit	NoScaler	0.925526	
37	KNeighborsClassifier	TimeSeriesSplit	StandardScaler	0.512913	
38	KNeighborsClassifier	TimeSeriesSplit	RobustScaler	0.528529	
39	KNeighborsClassifier	TimeSeriesSplit	MinMaxScaler	0.415015	

pipeline

```
0      False
1      False
2      False
3      False
4      False
5      False
6      False
7      False
8      False
9      False
10     False
11     False
12     False
13     False
14     False
15     False
16     False
17     False
18     False
19     False
20     False
21     False
22     False
23     False
24     False
25     False
26     False
27     False
28     False
29     False
30     False
31     False
32     False
33     False
34     False
35     False
36     False
37     False
38     False
39     False
```