KNN em python

October 18, 2022

[1]:
```python
from IPython.display import Image
%matplotlib inline
```

[3]:
```python
from sklearn import datasets
import numpy as np

iris = datasets.load_iris()
X = iris.data[:, [2, 3]]
y = iris.target
```

[4]:
```python
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
X, y, test_size=0.3, random_state=1, stratify=y)
```

[5]:
```python
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

sc.fit(X_train)
```

[5]:
```
StandardScaler()
```

[6]:
```python
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)
```

[12]:
```python
X_test_std
```

[12]:
```
array([[ 0.89820289,  1.44587881],
       [-1.16537974, -1.04507821],
       [-1.33269725, -1.17618121],
       [ 0.39625036,  0.65926081],
       [ 0.34047786,  0.2659518 ],
       [ 0.11738784,  0.1348488 ],
       [ 1.12129291,  0.79036381],
       [ 0.39625036,  0.3970548 ],
       [ 0.84243039,  0.92146681],
       [-1.38846976, -1.04507821],
```

```
       [-1.27692475, -1.04507821],
       [ 0.61934037,  0.79036381],
       [-1.33269725, -1.30728421],
       [-0.27301968, -0.2584602 ],
       [-1.33269725, -1.30728421],
       [ 0.56356787,  0.2659518 ],
       [ 0.73088538,  1.44587881],
       [ 0.39625036,  0.3970548 ],
       [ 0.28470535,  0.1348488 ],
       [ 0.78665788,  1.05256981],
       [ 1.17706541,  1.18367281],
       [-1.33269725, -1.43838721],
       [ 0.34047786,  0.2659518 ],
       [ 0.61934037,  1.05256981],
       [ 0.22893285,  0.1348488 ],
       [ 0.50779537,  0.5281578 ],
       [-0.4403372 , -0.1273572 ],
       [ 1.0655204 ,  1.70808482],
       [-1.22115225, -0.78287221],
       [ 0.67511288,  1.05256981],
       [-1.22115225, -1.30728421],
       [-1.33269725, -1.30728421],
       [ 0.11738784, -0.2584602 ],
       [ 0.11738784,  0.1348488 ],
       [ 1.40015543,  0.79036381],
       [ 0.9539754 ,  1.18367281],
       [-1.33269725, -1.43838721],
       [-1.22115225, -1.30728421],
       [-1.33269725, -1.30728421],
       [ 0.50779537,  0.2659518 ],
       [ 1.0655204 ,  1.44587881],
       [ 0.73088538,  0.79036381],
       [ 0.45202286,  0.3970548 ],
       [-1.27692475, -1.30728421],
       [-1.27692475, -1.43838721]])
```

[14]:
```
X_combined_std = np.vstack((X_train_std, X_test_std))
y_combined_std = np.hstack((y_train, y_test))
```

[8]:
```
from matplotlib.colors import ListedColormap
import matplotlib.pyplot as plt

# To check recent matplotlib compatibility
import matplotlib
from distutils.version import LooseVersion
```

```python
def plot_decision_regions(X, y, classifier, test_idx=None, resolution=0.02):

    # setup marker generator and color map
    markers = ('s', 'x', 'o', '^', 'v')
    colors = ('red', 'blue', 'lightgreen', 'gray', 'cyan')
    cmap = ListedColormap(colors[:len(np.unique(y))])

    # plot the decision surface
    x1_min, x1_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    x2_min, x2_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, resolution),
                           np.arange(x2_min, x2_max, resolution))
    Z = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)
    Z = Z.reshape(xx1.shape)
    plt.contourf(xx1, xx2, Z, alpha=0.3, cmap=cmap)
    plt.xlim(xx1.min(), xx1.max())
    plt.ylim(xx2.min(), xx2.max())

    for idx, cl in enumerate(np.unique(y)):
        plt.scatter(x=X[y == cl, 0],
                    y=X[y == cl, 1],
                    alpha=0.8,
                    color=colors[idx],
                    marker=markers[idx],
                    label=cl,
                    edgecolor='black')

    # highlight test examples
    if test_idx:
        # plot all examples
        X_test, y_test = X[test_idx, :], y[test_idx]


        if LooseVersion(matplotlib.__version__) < LooseVersion('0.3.4'):
            plt.scatter(X_test[:, 0],
                        X_test[:, 1],
                        c='',
                        edgecolor='black',
                        alpha=1.0,
                        linewidth=1,
                        marker='o',
                        s=100,
                        label='test set')
        else:
            plt.scatter(X_test[:, 0],
                        X_test[:, 1],
                        c='none',
```

```
                                edgecolor='black',
                                alpha=1.0,
                                linewidth=1,
                                marker='o',
                                s=100,
                                label='test set')
```

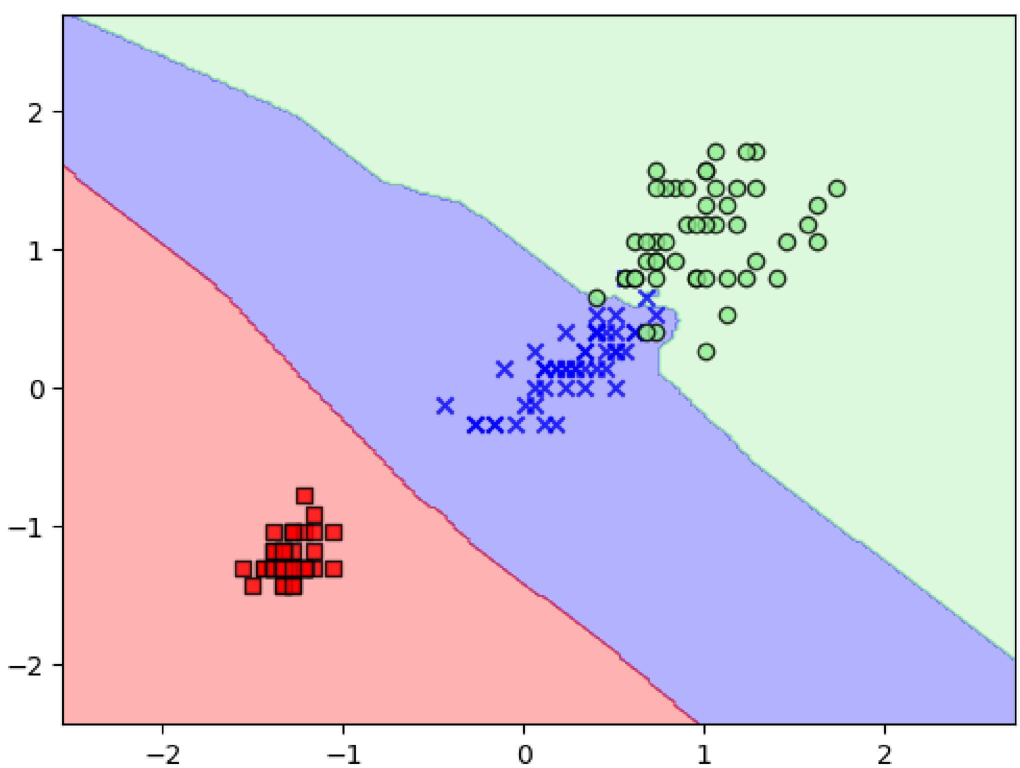
[9]:
```
from sklearn.neighbors import KNeighborsClassifier

KNN = KNeighborsClassifier(n_neighbors=5, p=2, metric='minkowski')
```

[10]:
```
KNN.fit(X_train_std, y_train)
```

[10]:
```
KNeighborsClassifier()
```

[15]:
```
plot_decision_regions(X_combined_std, y_combined_std, classifier=KNN)
```

```
/tmp/ipykernel_178950/3366577703.py:28: UserWarning: You passed a
edgecolor/edgecolors ('black') for an unfilled marker ('x').  Matplotlib is
ignoring the edgecolor in favor of the facecolor.  This behavior may change in
the future.
  plt.scatter(x=X[y == cl, 0],
```



[16]:
```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

[31]:
```python
data = pd.read_csv('ClassificaDados', index_col=0)
```

[32]:
```python
data.head(20)
```

[32]:
```
         WTT       PTI       EQW       SBI       LQE       QWG       FDJ  \
0   0.913917  1.162073  0.567946  0.755464  0.780862  0.352608  0.759697
1   0.635632  1.003722  0.535342  0.825645  0.924109  0.648450  0.675334
2   0.721360  1.201493  0.921990  0.855595  1.526629  0.720781  1.626351
3   1.234204  1.386726  0.653046  0.825624  1.142504  0.875128  1.409708
4   1.279491  0.949750  0.627280  0.668976  1.232537  0.703727  1.115596
5   0.833928  1.523302  1.104743  1.021139  1.107377  1.010930  1.279538
6   0.944705  1.251761  1.074885  0.286473  0.996440  0.428860  0.910805
7   0.816174  1.088392  0.895343  0.243860  0.943123  1.045131  1.146536
8   0.776551  1.463812  0.783825  0.337278  0.742215  1.072756  0.880300
9   0.772280  0.515111  0.891596  0.940862  1.430568  0.885876  1.205231
10  1.284999  1.331018  0.618910  0.657017  1.037191  0.717346  0.778501
11  1.064356  1.414885  0.896798  0.629088  1.447704  0.791923  0.921676
12  0.682953  1.254723  0.998870  0.397701  1.011621  0.567863  1.335120
13  0.953318  1.318987  0.562921  0.905503  1.248314  0.677795  1.017305
14  0.801268  0.936390  0.696960  0.972440  0.851299  1.443119  1.194476
15  1.061691  1.044892  0.599729  0.465285  0.930288  0.974341  1.213450
16  0.715645  1.378594  0.997797  0.674996  1.228928  1.223293  0.589346
17  0.899792  1.225875  1.330887  0.335989  1.273570  1.100361  1.019617
18  0.883813  1.275891  0.924066  0.668814  1.269021  1.382093  0.728286
19  0.768311  1.394304  0.823118  0.612072  1.267414  0.881943  1.104178

         PJF       HQE       NXJ  TARGET CLASS
0   0.643798  0.879422  1.231409             1
1   1.013546  0.621552  1.492702             0
2   1.154483  0.957877  1.285597             0
3   1.380003  1.522692  1.153093             1
4   0.646691  1.463812  1.419167             1
5   1.280677  0.510350  1.528044             0
6   0.755305  1.111800  1.110842             0
7   1.341886  1.225324  1.425784             0
8   1.312951  1.118165  1.225922             0
9   0.596858  1.542580  0.981879             1
10  0.599317  1.245676  1.441695             1
11  1.237249  0.564281  1.423668             0
12  1.093735  0.847015  1.374779             0
13  0.528065  1.541712  1.118960             1
```

```
14  1.641496  1.118737  1.426573             0
15  1.247551  1.217625  1.623154             1
16  1.559900  0.845324  1.248714             0
17  1.223891  1.113441  1.490151             0
18  0.726723  0.975833  1.653815             0
19  1.660998  0.742885  1.098704             0
```

[33]:
```
data.info()
```

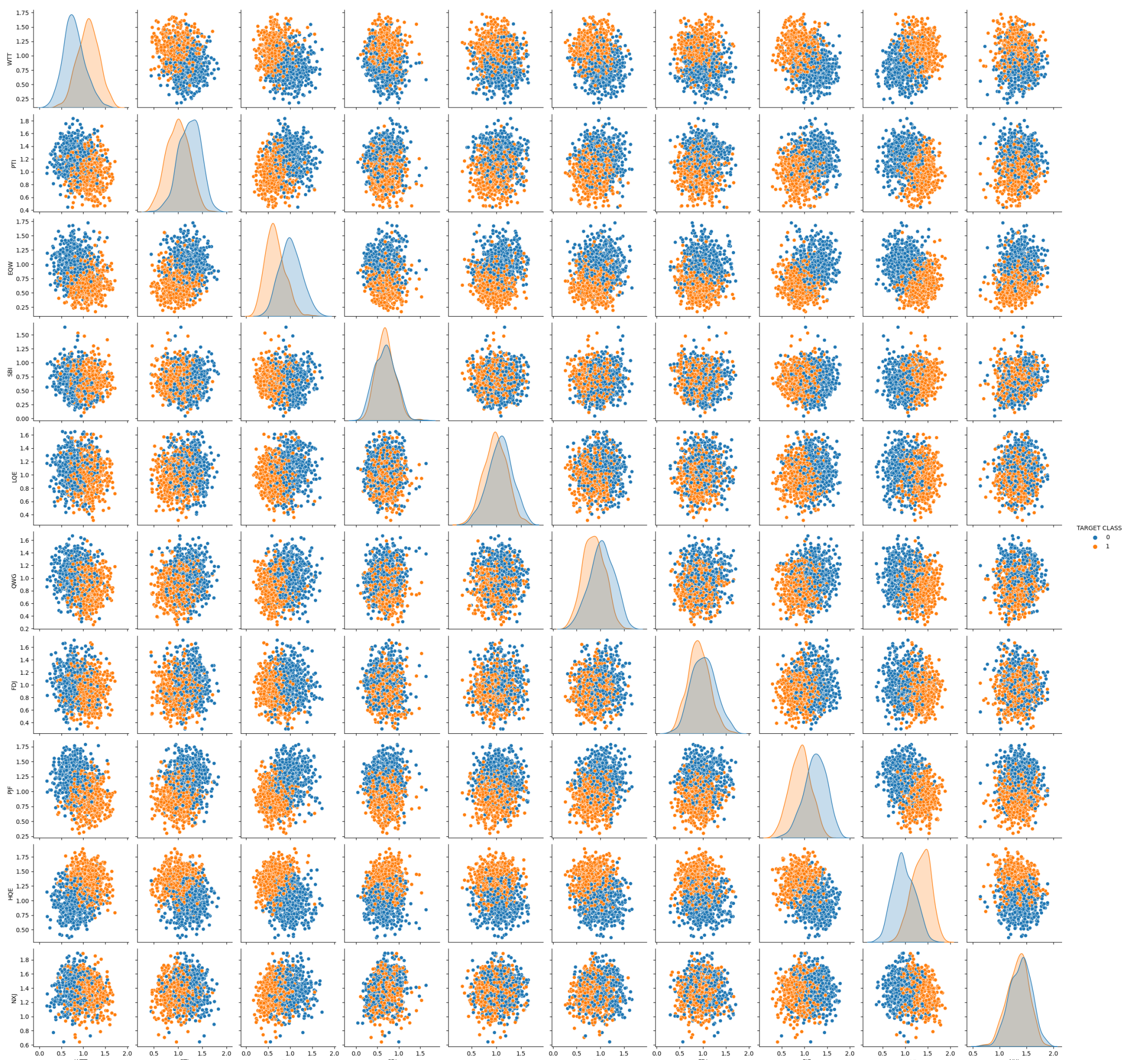
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1000 entries, 0 to 999
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   WTT           1000 non-null   float64
 1   PTI           1000 non-null   float64
 2   EQW           1000 non-null   float64
 3   SBI           1000 non-null   float64
 4   LQE           1000 non-null   float64
 5   QWG           1000 non-null   float64
 6   FDJ           1000 non-null   float64
 7   PJF           1000 non-null   float64
 8   HQE           1000 non-null   float64
 9   NXJ           1000 non-null   float64
 10  TARGET CLASS  1000 non-null   int64
dtypes: float64(10), int64(1)
memory usage: 93.8 KB
```

[34]:
```
data.columns
```

[34]:
```
Index(['WTT', 'PTI', 'EQW', 'SBI', 'LQE', 'QWG', 'FDJ', 'PJF', 'HQE', 'NXJ',
       'TARGET CLASS'],
      dtype='object')
```

[22]:
```
sns.pairplot(data, hue='TARGET CLASS')
```

[22]:
```
<seaborn.axisgrid.PairGrid at 0x7fc4d3712110>
```



[35]:
```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

sc.fit(data.drop('TARGET CLASS', axis=1))
```

[35]:
```
StandardScaler()
```

[36]:
```
dados=sc.transform(data.drop('TARGET CLASS', axis=1))
```

[26]:
```
dados
```

[26]:
```
array([[-0.12354188,  0.18590747, -0.91343069, …, -1.48236813,
        -0.9497194 , -0.64331425],
       [-1.08483602, -0.43034845, -1.02531333, …, -0.20224031,
```

```
        -1.82805088,  0.63675862],
       [-0.78870217,  0.33931821,  0.30151137, …,  0.28570652,
        -0.68249379, -0.37784986],
       …,
       [ 0.64177714, -0.51308341, -0.17920486, …, -2.36249443,
        -0.81426092,  0.11159651],
       [ 0.46707241, -0.98278576, -1.46519359, …, -0.03677699,
         0.40602453, -0.85567   ],
       [-0.38765353, -0.59589427, -1.4313981 , …, -0.56778932,
         0.3369971 ,  0.01034996]])
```

[37]:
```python
DF = pd.DataFrame(dados, columns = data.columns[:-1])
```

[38]:
```python
DF.head()
```

[38]:
```
        WTT       PTI       EQW       SBI       LQE       QWG       FDJ  \
0 -0.123542  0.185907 -0.913431  0.319629 -1.033637 -2.308375 -0.798951
1 -1.084836 -0.430348 -1.025313  0.625388 -0.444847 -1.152706 -1.129797
2 -0.788702  0.339318  0.301511  0.755873  2.031693 -0.870156  2.599818
3  0.982841  1.060193 -0.621399  0.625299  0.452820 -0.267220  1.750208
4  1.139275 -0.640392 -0.709819 -0.057175  0.822886 -0.936773  0.596782

        PJF       HQE       NXJ
0 -1.482368 -0.949719 -0.643314
1 -0.202240 -1.828051  0.636759
2  0.285707 -0.682494 -0.377850
3  1.066491  1.241325 -1.026987
4 -1.472352  1.040772  0.276510
```

0.1 Divisão Treino e Teste

[39]:
```python
from sklearn.model_selection import train_test_split
```

[41]:
```python
X_train, X_test, y_train, y_test = train_test_split(dados, data['TARGET CLASS'],
                                                    test_size=0.3,␣
 ↪random_state=1)
```

[42]:
```python
y_train
```

[42]:
```
731    1
716    0
640    1
804    1
737    1
      ..
767    1
72     1
```

```
908    1
235    1
37     1
Name: TARGET CLASS, Length: 700, dtype: int64
```

[43]:
```
X_train
```

[43]:
```
array([[ 2.05621551, -0.32655566,  0.2186274 , …, -0.41322254,
         0.09995974,  1.31436215],
       [-0.66162431,  1.2943079 , -0.15774655, …, -0.71699735,
        -1.3585909 , -0.56024653],
       [ 0.39523395, -2.09199987, -1.38109288, …, -0.67612905,
         0.21135605,  0.11393259],
       …,
       [ 0.90344929,  0.00424662, -1.01972037, …, -0.34549064,
        -0.09885529,  1.19270428],
       [-0.00962074,  1.24838194, -0.78177643, …, -1.4002669 ,
         0.49192358, -0.1189531 ],
       [ 0.10483098,  1.27975307, -0.13195213, …, -1.28654429,
         1.13589455, -0.29048815]])
```

[44]:
```
from sklearn.neighbors import KNeighborsClassifier
```

[45]:
```
KNN = KNeighborsClassifier(n_neighbors=1)
```

[46]:
```
KNN.fit(X_train, y_train)
```

[46]:
```
KNeighborsClassifier(n_neighbors=1)
```

[47]:
```
predicao = KNN.predict(X_test)
```

[48]:
```
predicao
```

[48]:
```
array([0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1,
       1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1,
       1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1,
       0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0,
       1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1,
       1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
       0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0,
       1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0,
       0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0,
       1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0,
       1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1])
```

0.2 Avaliando os resultados

[49]:
```
from sklearn.metrics import classification_report, confusion_matrix
```

[50]:
```
print(confusion_matrix(y_test, predicao))
```

```
[[147  14]
 [ 10 129]]
```

[51]:
```
prob=(147+129)/(24)
```

[52]:
```
prob
```

[52]:
```
11.5
```

[53]:
```
print(classification_report(y_test, predicao))
```

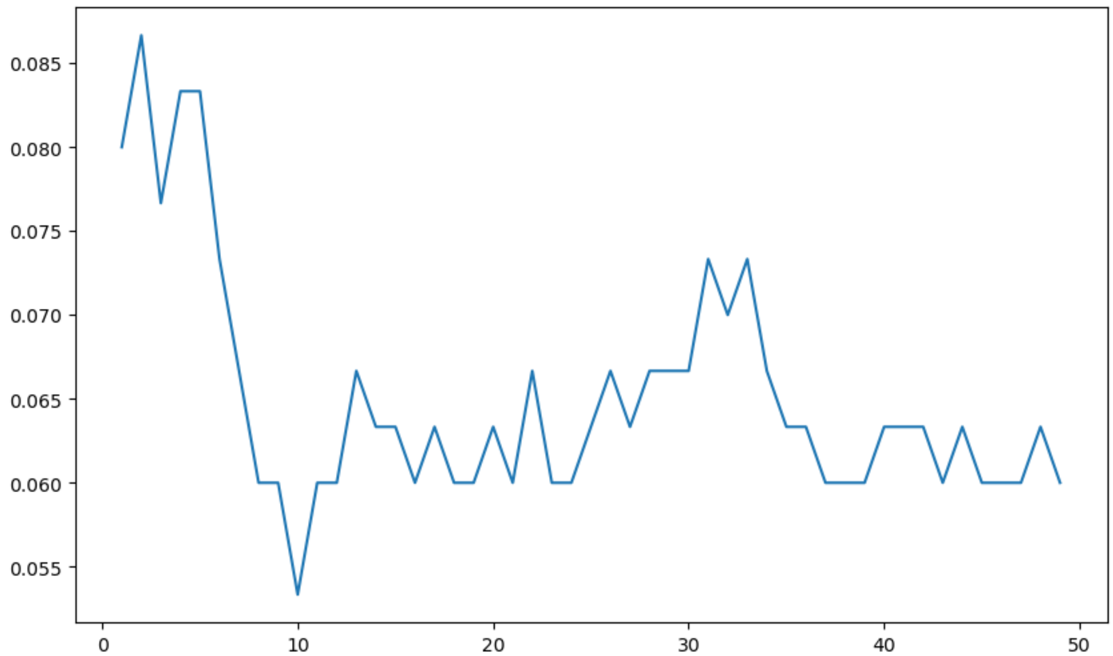
```
              precision    recall  f1-score   support

           0       0.94      0.91      0.92       161
           1       0.90      0.93      0.91       139

    accuracy                           0.92       300
   macro avg       0.92      0.92      0.92       300
weighted avg       0.92      0.92      0.92       300
```

0.3 Técnica do Cotovelo

[54]:
```
error_rate = []

for i in range(1, 50):
    KNN = KNeighborsClassifier(n_neighbors= i)
    KNN.fit(X_train, y_train)
    preditor_i = KNN.predict(X_test)
    error_rate.append(np.mean(preditor_i != Y_test))
```

[56]:
```
plt.figure(figsize=(10,6))
plt.plot(range(1, 50), error_rate)
```

[56]:
```
[<matplotlib.lines.Line2D at 0x7fc4cae473a0>]
```



[63]:
```
KNN = KNeighborsClassifier(n_neighbors=8)
KNN.fit(X_train, y_train)
predicao = KNN.predict(X_test)
print(confusion_matrix(y_test, predicao))
print(classification_report(y_test, predicao))
```

```
[[151  10]
 [  8 131]]
              precision    recall  f1-score   support

           0       0.95      0.94      0.94       161
           1       0.93      0.94      0.94       139

    accuracy                           0.94       300
   macro avg       0.94      0.94      0.94       300
weighted avg       0.94      0.94      0.94       300
```

0.4 Criando um grid de busca de parametros para KNN

[64]:
```
from sklearn.model_selection import GridSearchCV
```

[65]:
```
k_list = list(range(1,30))
weight_list = ['uniform', 'distance']
p_list=[1, 2]
```

[66]:
```
parametros = dict(n_neighbors=k_list, weights = weight_list, p = p_list)
```

[67]:
```
grid = GridSearchCV(KNN, parametros, cv=5, scoring='accuracy')
```

[68]:
```
grid.fit(dados, data['TARGET CLASS'])
```

[68]:
```
GridSearchCV(cv=5, estimator=KNeighborsClassifier(n_neighbors=8),
             param_grid={'n_neighbors': [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],
                         'p': [1, 2], 'weights': ['uniform', 'distance']},
             scoring='accuracy')
```

[69]:
```
grid.cv_results_
```

[69]:
```
{'mean_fit_time': array([0.00256987, 0.00158482, 0.00161514, 0.00160398,
0.00153265,
        0.00170612, 0.00157261, 0.00155387, 0.0016531 , 0.00157886,
        0.00158052, 0.00151086, 0.00161533, 0.00160589, 0.00166326,
        0.00155201, 0.0018312 , 0.00145826, 0.00162272, 0.00158086,
        0.00167294, 0.00145693, 0.00166698, 0.00160575, 0.00153966,
        0.0014771 , 0.00154181, 0.00157194, 0.00149183, 0.00142994,
        0.0014338 , 0.00144963, 0.00152264, 0.00159183, 0.0014524 ,
        0.00150628, 0.001477  , 0.00144978, 0.00149322, 0.00151243,
        0.00151029, 0.00157022, 0.00164232, 0.00162706, 0.00162735,
        0.00167584, 0.00159202, 0.00172853, 0.00148149, 0.00161514,
        0.00164971, 0.00157037, 0.00160623, 0.00164814, 0.00158591,
        0.00156779, 0.00172362, 0.00170765, 0.00156975, 0.00158496,
        0.00161757, 0.00176916, 0.00155387, 0.00164461, 0.00173125,
        0.0016912 , 0.00160313, 0.00173221, 0.00153127, 0.00161176,
        0.00158257, 0.001579  , 0.0017725 , 0.00157952, 0.00157762,
        0.00163813, 0.00151839, 0.00164914, 0.00163121, 0.00169916,
        0.00159836, 0.00157557, 0.00165982, 0.00158558, 0.00153351,
        0.00146618, 0.0016294 , 0.00147524, 0.0014411 , 0.00142436,
        0.00151167, 0.00157175, 0.00143981, 0.00143871, 0.00148878,
        0.00152249, 0.00154243, 0.0014421 , 0.00141749, 0.00144091,
        0.00144114, 0.00152035, 0.00162239, 0.00144076, 0.00151668,
        0.00143609, 0.00145249, 0.00143027, 0.00152912, 0.0014915 ,
        0.00159984, 0.00145822, 0.00145187, 0.00142717, 0.00156355,
        0.00146642]),
 'std_fit_time': array([1.19717073e-03, 2.89489249e-05, 8.85187020e-05,
1.06508237e-04,
        3.94836140e-05, 3.07664077e-04, 4.79597210e-05, 4.93554331e-05,
        1.53693471e-04, 3.98350002e-05, 5.80809190e-05, 1.92432006e-05,
        1.87769270e-04, 5.11269951e-05, 1.56936258e-04, 1.83782320e-04,
        3.13444676e-04, 4.00922233e-05, 2.07499151e-04, 1.20217693e-04,
        1.69230401e-04, 3.27463904e-05, 1.05252274e-04, 9.93689053e-05,
```

```
       3.23636514e-05, 4.61770936e-05, 8.38325833e-05, 2.42724231e-04,
       4.97250855e-05, 1.97496948e-05, 2.61241494e-05, 3.03335152e-05,
       8.96963015e-05, 1.50877832e-04, 2.30945778e-05, 1.08775262e-04,
       4.04457989e-05, 2.70973597e-05, 7.76618284e-05, 1.04932629e-04,
       6.07514867e-05, 1.11320287e-04, 1.55783626e-04, 1.93360237e-04,
       8.62433000e-05, 1.28541448e-04, 6.86472633e-05, 1.85423388e-04,
       4.24179820e-05, 2.29789682e-04, 7.92933885e-05, 3.52587466e-05,
       1.06547660e-04, 1.14668124e-04, 6.27243713e-05, 1.98792491e-05,
       7.09070803e-05, 3.99865918e-04, 1.17875747e-04, 5.98981118e-05,
       7.06397838e-05, 1.63744075e-04, 3.55953749e-05, 8.93421574e-05,
       1.21345604e-04, 1.02817370e-04, 1.08937829e-04, 2.15305342e-04,
       4.78426193e-05, 1.61097188e-04, 1.10643422e-04, 1.19201507e-04,
       2.76068067e-04, 5.43736556e-05, 5.67239840e-05, 1.80356770e-04,
       4.52913980e-05, 2.69161504e-04, 1.11431998e-04, 1.13597724e-04,
       1.08107828e-04, 1.00294011e-04, 1.87806684e-04, 5.82223052e-05,
       8.35937835e-05, 3.62043460e-05, 2.45980431e-04, 4.94323078e-05,
       2.50389911e-05, 1.69101812e-05, 6.77519325e-05, 8.04084605e-05,
       1.64235621e-05, 3.82070069e-05, 6.66231444e-05, 1.88312309e-04,
       4.56342378e-05, 2.03423449e-05, 9.61580880e-06, 1.55726577e-05,
       2.31875276e-05, 1.67086857e-04, 1.05069897e-04, 1.59786702e-05,
       1.18078746e-04, 2.75315551e-05, 4.18855779e-05, 7.08227596e-06,
       6.61666498e-05, 7.44466967e-05, 2.32808504e-04, 2.77605240e-05,
       1.56040179e-05, 7.95761074e-06, 8.10233191e-05, 3.25273551e-05]),
 'mean_score_time': array([0.02123842, 0.00655494, 0.01493597, 0.00455346,
0.0162878 ,
       0.0061615 , 0.01577864, 0.00500059, 0.01748219, 0.00617914,
       0.01549416, 0.00506358, 0.01673131, 0.00627923, 0.01589494,
       0.00575991, 0.01779428, 0.00599747, 0.01585493, 0.00533037,
       0.0174305 , 0.00633578, 0.01544175, 0.00533319, 0.01732826,
       0.00623798, 0.01620297, 0.00520644, 0.01611276, 0.00601664,
       0.01492729, 0.00518928, 0.01638737, 0.00677123, 0.01516838,
       0.00537424, 0.01603208, 0.006109  , 0.01570835, 0.00544925,
       0.01648264, 0.0070652 , 0.01637783, 0.00576992, 0.02108636,
       0.00674372, 0.01622086, 0.00569587, 0.01656804, 0.0066884 ,
       0.02022562, 0.00593677, 0.01710525, 0.00700507, 0.01637163,
       0.0059196 , 0.02151151, 0.00665135, 0.01628008, 0.00600123,
       0.01712065, 0.00728116, 0.01656694, 0.00651064, 0.01776567,
       0.00692205, 0.01669745, 0.00634546, 0.01687565, 0.00693436,
       0.01661086, 0.00667076, 0.0185977 , 0.00707493, 0.01656327,
       0.00588703, 0.01735554, 0.00727363, 0.01791983, 0.00652733,
       0.01736841, 0.00706325, 0.01712193, 0.00643377, 0.01737018,
       0.00668182, 0.01647301, 0.00602765, 0.01641994, 0.0066628 ,
       0.01644568, 0.00612211, 0.01640916, 0.00664916, 0.01665864,
       0.00606894, 0.01713305, 0.00669174, 0.01568294, 0.00603147,
       0.01646595, 0.00684714, 0.01671934, 0.00618944, 0.01679354,
       0.00677371, 0.01596284, 0.00614743, 0.0171936 , 0.00690536,
       0.01671023, 0.00631385, 0.01656651, 0.0069809 , 0.01652246,
```

```
        0.00625634]),
 'std_score_time': array([6.16602784e-03, 4.39725065e-04, 3.05829057e-04,
3.17232906e-04,
        5.49116653e-04, 1.66320254e-04, 1.70530502e-04, 4.49024490e-04,
        9.80477541e-04, 1.78930080e-04, 2.42536807e-04, 1.24297774e-04,
        1.25848736e-04, 4.27680242e-04, 7.57353561e-04, 9.66924228e-04,
        1.23696404e-03, 1.93427887e-04, 1.46944442e-03, 2.36822047e-04,
        9.40205163e-04, 6.65263124e-04, 9.58750566e-05, 1.72172752e-04,
        8.83245727e-04, 1.83280474e-04, 6.93706893e-04, 1.07135515e-04,
        3.80487172e-04, 4.06109081e-05, 7.23879955e-05, 5.94276853e-05,
        5.51351624e-04, 6.10674806e-04, 1.96359284e-04, 1.14123221e-04,
        3.60481972e-04, 8.20840232e-05, 5.92230150e-04, 1.80220438e-04,
        5.29266056e-04, 5.89726639e-04, 8.46332360e-05, 1.67673968e-04,
        7.00993989e-03, 1.70093796e-04, 4.44165671e-04, 2.08391226e-04,
        8.05591727e-04, 4.02332391e-04, 7.21267080e-03, 2.31428561e-04,
        1.15596539e-04, 5.51908548e-04, 1.34560759e-04, 1.93835718e-04,
        6.55312475e-03, 3.22378963e-04, 1.15373566e-04, 3.10414471e-04,
        2.03370707e-04, 6.20744400e-04, 2.36092589e-04, 4.40159405e-04,
        9.62343800e-04, 1.06141788e-04, 5.86510740e-04, 5.80991011e-04,
        3.59829787e-04, 2.48670802e-04, 1.03309717e-04, 5.19668533e-04,
        6.72912804e-04, 3.46199297e-04, 3.41598053e-04, 6.57331740e-05,
        5.74820037e-04, 5.58470703e-04, 9.71025789e-04, 4.88113830e-04,
        6.94816700e-05, 2.50388940e-04, 3.76349103e-04, 2.70503815e-04,
        8.71239289e-04, 1.10897562e-04, 8.51735942e-04, 2.34476274e-04,
        9.68246470e-05, 9.25838500e-05, 4.63521097e-04, 1.95334234e-04,
        7.50669519e-05, 1.87194875e-05, 1.14202617e-03, 1.04275140e-04,
        6.09629610e-04, 1.60116493e-05, 1.31137241e-04, 4.14715121e-05,
        1.26024498e-04, 1.92216601e-04, 8.08413805e-04, 9.14442144e-05,
        2.60894182e-04, 1.14609673e-05, 3.60429124e-04, 7.40177227e-05,
        5.22822957e-04, 1.40109931e-04, 9.26966473e-04, 1.42457637e-04,
        1.53429201e-04, 2.16787867e-04, 6.85228757e-04, 4.16724031e-05]),
 'param_n_neighbors': masked_array(data=[1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3, 4,
4, 4, 4, 5, 5,
                    5, 5, 6, 6, 6, 6, 7, 7, 7, 7, 8, 8, 8, 8, 9, 9, 9, 9,
                    10, 10, 10, 10, 11, 11, 11, 11, 12, 12, 12, 12, 13, 13,
                    13, 13, 14, 14, 14, 14, 15, 15, 15, 15, 16, 16, 16, 16,
                    17, 17, 17, 17, 18, 18, 18, 18, 19, 19, 19, 19, 20, 20,
                    20, 20, 21, 21, 21, 21, 22, 22, 22, 22, 23, 23, 23, 23,
                    24, 24, 24, 24, 25, 25, 25, 25, 26, 26, 26, 26, 27, 27,
                    27, 27, 28, 28, 28, 28, 29, 29, 29, 29],
              mask=[False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
```

```
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False],
        fill_value='?',
             dtype=object),
 'param_p': masked_array(data=[1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 2, 2,
1, 1,
                    2, 2, 1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 2, 2,
                    1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 2, 2, 1, 1,
                    2, 2, 1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 2, 2,
                    1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 2, 2, 1, 1,
                    2, 2, 1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 2, 2,
                    1, 1, 2, 2, 1, 1, 2, 2],
              mask=[False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False],
        fill_value='?',
             dtype=object),
 'param_weights': masked_array(data=['uniform', 'distance', 'uniform',
'distance',
                    'uniform', 'distance', 'uniform', 'distance',
                    'uniform', 'distance', 'uniform', 'distance',
                    'uniform', 'distance', 'uniform', 'distance',
                    'uniform', 'distance', 'uniform', 'distance',
                    'uniform', 'distance', 'uniform', 'distance',
                    'uniform', 'distance', 'uniform', 'distance',
                    'uniform', 'distance', 'uniform', 'distance',
                    'uniform', 'distance', 'uniform', 'distance',
                    'uniform', 'distance', 'uniform', 'distance',
                    'uniform', 'distance', 'uniform', 'distance',
```

```
                    'uniform', 'distance', 'uniform', 'distance',
                    'uniform', 'distance', 'uniform', 'distance',
                    'uniform', 'distance', 'uniform', 'distance',
                    'uniform', 'distance', 'uniform', 'distance',
                    'uniform', 'distance', 'uniform', 'distance',
                    'uniform', 'distance', 'uniform', 'distance',
                    'uniform', 'distance', 'uniform', 'distance',
                    'uniform', 'distance', 'uniform', 'distance',
                    'uniform', 'distance', 'uniform', 'distance',
                    'uniform', 'distance', 'uniform', 'distance',
                    'uniform', 'distance', 'uniform', 'distance',
                    'uniform', 'distance', 'uniform', 'distance',
                    'uniform', 'distance', 'uniform', 'distance',
                    'uniform', 'distance', 'uniform', 'distance',
                    'uniform', 'distance', 'uniform', 'distance',
                    'uniform', 'distance', 'uniform', 'distance',
                    'uniform', 'distance', 'uniform', 'distance',
                    'uniform', 'distance', 'uniform', 'distance'],
              mask=[False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False, False, False, False, False,
                    False, False, False, False],
        fill_value='?',
             dtype=object),
 'params': [{'n_neighbors': 1, 'p': 1, 'weights': 'uniform'},
  {'n_neighbors': 1, 'p': 1, 'weights': 'distance'},
  {'n_neighbors': 1, 'p': 2, 'weights': 'uniform'},
  {'n_neighbors': 1, 'p': 2, 'weights': 'distance'},
  {'n_neighbors': 2, 'p': 1, 'weights': 'uniform'},
  {'n_neighbors': 2, 'p': 1, 'weights': 'distance'},
  {'n_neighbors': 2, 'p': 2, 'weights': 'uniform'},
  {'n_neighbors': 2, 'p': 2, 'weights': 'distance'},
  {'n_neighbors': 3, 'p': 1, 'weights': 'uniform'},
  {'n_neighbors': 3, 'p': 1, 'weights': 'distance'},
  {'n_neighbors': 3, 'p': 2, 'weights': 'uniform'},
  {'n_neighbors': 3, 'p': 2, 'weights': 'distance'},
```

```
{'n_neighbors': 4, 'p': 1, 'weights': 'uniform'},
{'n_neighbors': 4, 'p': 1, 'weights': 'distance'},
{'n_neighbors': 4, 'p': 2, 'weights': 'uniform'},
{'n_neighbors': 4, 'p': 2, 'weights': 'distance'},
{'n_neighbors': 5, 'p': 1, 'weights': 'uniform'},
{'n_neighbors': 5, 'p': 1, 'weights': 'distance'},
{'n_neighbors': 5, 'p': 2, 'weights': 'uniform'},
{'n_neighbors': 5, 'p': 2, 'weights': 'distance'},
{'n_neighbors': 6, 'p': 1, 'weights': 'uniform'},
{'n_neighbors': 6, 'p': 1, 'weights': 'distance'},
{'n_neighbors': 6, 'p': 2, 'weights': 'uniform'},
{'n_neighbors': 6, 'p': 2, 'weights': 'distance'},
{'n_neighbors': 7, 'p': 1, 'weights': 'uniform'},
{'n_neighbors': 7, 'p': 1, 'weights': 'distance'},
{'n_neighbors': 7, 'p': 2, 'weights': 'uniform'},
{'n_neighbors': 7, 'p': 2, 'weights': 'distance'},
{'n_neighbors': 8, 'p': 1, 'weights': 'uniform'},
{'n_neighbors': 8, 'p': 1, 'weights': 'distance'},
{'n_neighbors': 8, 'p': 2, 'weights': 'uniform'},
{'n_neighbors': 8, 'p': 2, 'weights': 'distance'},
{'n_neighbors': 9, 'p': 1, 'weights': 'uniform'},
{'n_neighbors': 9, 'p': 1, 'weights': 'distance'},
{'n_neighbors': 9, 'p': 2, 'weights': 'uniform'},
{'n_neighbors': 9, 'p': 2, 'weights': 'distance'},
{'n_neighbors': 10, 'p': 1, 'weights': 'uniform'},
{'n_neighbors': 10, 'p': 1, 'weights': 'distance'},
{'n_neighbors': 10, 'p': 2, 'weights': 'uniform'},
{'n_neighbors': 10, 'p': 2, 'weights': 'distance'},
{'n_neighbors': 11, 'p': 1, 'weights': 'uniform'},
{'n_neighbors': 11, 'p': 1, 'weights': 'distance'},
{'n_neighbors': 11, 'p': 2, 'weights': 'uniform'},
{'n_neighbors': 11, 'p': 2, 'weights': 'distance'},
{'n_neighbors': 12, 'p': 1, 'weights': 'uniform'},
{'n_neighbors': 12, 'p': 1, 'weights': 'distance'},
{'n_neighbors': 12, 'p': 2, 'weights': 'uniform'},
{'n_neighbors': 12, 'p': 2, 'weights': 'distance'},
{'n_neighbors': 13, 'p': 1, 'weights': 'uniform'},
{'n_neighbors': 13, 'p': 1, 'weights': 'distance'},
{'n_neighbors': 13, 'p': 2, 'weights': 'uniform'},
{'n_neighbors': 13, 'p': 2, 'weights': 'distance'},
{'n_neighbors': 14, 'p': 1, 'weights': 'uniform'},
{'n_neighbors': 14, 'p': 1, 'weights': 'distance'},
{'n_neighbors': 14, 'p': 2, 'weights': 'uniform'},
{'n_neighbors': 14, 'p': 2, 'weights': 'distance'},
{'n_neighbors': 15, 'p': 1, 'weights': 'uniform'},
{'n_neighbors': 15, 'p': 1, 'weights': 'distance'},
{'n_neighbors': 15, 'p': 2, 'weights': 'uniform'},
```

```
{'n_neighbors': 15, 'p': 2, 'weights': 'distance'},
{'n_neighbors': 16, 'p': 1, 'weights': 'uniform'},
{'n_neighbors': 16, 'p': 1, 'weights': 'distance'},
{'n_neighbors': 16, 'p': 2, 'weights': 'uniform'},
{'n_neighbors': 16, 'p': 2, 'weights': 'distance'},
{'n_neighbors': 17, 'p': 1, 'weights': 'uniform'},
{'n_neighbors': 17, 'p': 1, 'weights': 'distance'},
{'n_neighbors': 17, 'p': 2, 'weights': 'uniform'},
{'n_neighbors': 17, 'p': 2, 'weights': 'distance'},
{'n_neighbors': 18, 'p': 1, 'weights': 'uniform'},
{'n_neighbors': 18, 'p': 1, 'weights': 'distance'},
{'n_neighbors': 18, 'p': 2, 'weights': 'uniform'},
{'n_neighbors': 18, 'p': 2, 'weights': 'distance'},
{'n_neighbors': 19, 'p': 1, 'weights': 'uniform'},
{'n_neighbors': 19, 'p': 1, 'weights': 'distance'},
{'n_neighbors': 19, 'p': 2, 'weights': 'uniform'},
{'n_neighbors': 19, 'p': 2, 'weights': 'distance'},
{'n_neighbors': 20, 'p': 1, 'weights': 'uniform'},
{'n_neighbors': 20, 'p': 1, 'weights': 'distance'},
{'n_neighbors': 20, 'p': 2, 'weights': 'uniform'},
{'n_neighbors': 20, 'p': 2, 'weights': 'distance'},
{'n_neighbors': 21, 'p': 1, 'weights': 'uniform'},
{'n_neighbors': 21, 'p': 1, 'weights': 'distance'},
{'n_neighbors': 21, 'p': 2, 'weights': 'uniform'},
{'n_neighbors': 21, 'p': 2, 'weights': 'distance'},
{'n_neighbors': 22, 'p': 1, 'weights': 'uniform'},
{'n_neighbors': 22, 'p': 1, 'weights': 'distance'},
{'n_neighbors': 22, 'p': 2, 'weights': 'uniform'},
{'n_neighbors': 22, 'p': 2, 'weights': 'distance'},
{'n_neighbors': 23, 'p': 1, 'weights': 'uniform'},
{'n_neighbors': 23, 'p': 1, 'weights': 'distance'},
{'n_neighbors': 23, 'p': 2, 'weights': 'uniform'},
{'n_neighbors': 23, 'p': 2, 'weights': 'distance'},
{'n_neighbors': 24, 'p': 1, 'weights': 'uniform'},
{'n_neighbors': 24, 'p': 1, 'weights': 'distance'},
{'n_neighbors': 24, 'p': 2, 'weights': 'uniform'},
{'n_neighbors': 24, 'p': 2, 'weights': 'distance'},
{'n_neighbors': 25, 'p': 1, 'weights': 'uniform'},
{'n_neighbors': 25, 'p': 1, 'weights': 'distance'},
{'n_neighbors': 25, 'p': 2, 'weights': 'uniform'},
{'n_neighbors': 25, 'p': 2, 'weights': 'distance'},
{'n_neighbors': 26, 'p': 1, 'weights': 'uniform'},
{'n_neighbors': 26, 'p': 1, 'weights': 'distance'},
{'n_neighbors': 26, 'p': 2, 'weights': 'uniform'},
{'n_neighbors': 26, 'p': 2, 'weights': 'distance'},
{'n_neighbors': 27, 'p': 1, 'weights': 'uniform'},
{'n_neighbors': 27, 'p': 1, 'weights': 'distance'},
```

```
  {'n_neighbors': 27, 'p': 2, 'weights': 'uniform'},
  {'n_neighbors': 27, 'p': 2, 'weights': 'distance'},
  {'n_neighbors': 28, 'p': 1, 'weights': 'uniform'},
  {'n_neighbors': 28, 'p': 1, 'weights': 'distance'},
  {'n_neighbors': 28, 'p': 2, 'weights': 'uniform'},
  {'n_neighbors': 28, 'p': 2, 'weights': 'distance'},
  {'n_neighbors': 29, 'p': 1, 'weights': 'uniform'},
  {'n_neighbors': 29, 'p': 1, 'weights': 'distance'},
  {'n_neighbors': 29, 'p': 2, 'weights': 'uniform'},
  {'n_neighbors': 29, 'p': 2, 'weights': 'distance'}],
 'split0_test_score': array([0.895, 0.895, 0.905, 0.905, 0.905, 0.895, 0.92 ,
0.905, 0.92 ,
        0.92 , 0.915, 0.915, 0.925, 0.905, 0.94 , 0.92 , 0.92 , 0.92 ,
        0.92 , 0.92 , 0.915, 0.92 , 0.91 , 0.915, 0.925, 0.925, 0.92 ,
        0.92 , 0.925, 0.92 , 0.93 , 0.93 , 0.925, 0.925, 0.93 , 0.93 ,
        0.935, 0.93 , 0.935, 0.935, 0.935, 0.935, 0.935, 0.935, 0.935,
        0.935, 0.935, 0.935, 0.93 , 0.93 , 0.94 , 0.94 , 0.93 , 0.93 ,
        0.935, 0.935, 0.935, 0.935, 0.945, 0.945, 0.935, 0.93 , 0.94 ,
        0.935, 0.925, 0.925, 0.935, 0.935, 0.93 , 0.925, 0.94 , 0.935,
        0.935, 0.935, 0.935, 0.935, 0.94 , 0.925, 0.94 , 0.94 , 0.925,
        0.925, 0.935, 0.935, 0.935, 0.93 , 0.94 , 0.94 , 0.935, 0.935,
        0.94 , 0.94 , 0.94 , 0.93 , 0.94 , 0.945, 0.93 , 0.93 , 0.94 ,
        0.94 , 0.93 , 0.93 , 0.935, 0.94 , 0.935, 0.935, 0.945, 0.945,
        0.935, 0.935, 0.94 , 0.94 , 0.935, 0.935, 0.94 , 0.94 ]),
 'split1_test_score': array([0.89 , 0.89 , 0.9  , 0.9  , 0.905, 0.89 , 0.915,
0.9  , 0.91 ,
        0.91 , 0.92 , 0.92 , 0.92 , 0.925, 0.93 , 0.92 , 0.915, 0.915,
        0.92 , 0.92 , 0.915, 0.915, 0.915, 0.92 , 0.915, 0.915, 0.925,
        0.925, 0.91 , 0.915, 0.925, 0.925, 0.915, 0.92 , 0.91 , 0.91 ,
        0.91 , 0.915, 0.915, 0.925, 0.91 , 0.91 , 0.925, 0.925, 0.91 ,
        0.91 , 0.92 , 0.925, 0.91 , 0.91 , 0.92 , 0.92 , 0.915, 0.92 ,
        0.92 , 0.92 , 0.92 , 0.92 , 0.915, 0.915, 0.92 , 0.915, 0.91 ,
        0.92 , 0.93 , 0.93 , 0.905, 0.91 , 0.925, 0.92 , 0.905, 0.905,
        0.92 , 0.92 , 0.905, 0.905, 0.915, 0.92 , 0.905, 0.905, 0.915,
        0.915, 0.905, 0.905, 0.92 , 0.92 , 0.905, 0.915, 0.915, 0.915,
        0.91 , 0.915, 0.915, 0.915, 0.91 , 0.91 , 0.91 , 0.91 , 0.915,
        0.92 , 0.915, 0.91 , 0.91 , 0.915, 0.915, 0.915, 0.92 , 0.92 ,
        0.915, 0.915, 0.92 , 0.92 , 0.91 , 0.91 , 0.92 , 0.92 ]),
 'split2_test_score': array([0.845, 0.845, 0.89 , 0.89 , 0.87 , 0.845, 0.895,
0.89 , 0.895,
        0.895, 0.905, 0.905, 0.89 , 0.875, 0.89 , 0.9  , 0.895, 0.895,
        0.89 , 0.89 , 0.92 , 0.885, 0.9  , 0.895, 0.92 , 0.92 , 0.91 ,
        0.91 , 0.91 , 0.89 , 0.915, 0.905, 0.915, 0.915, 0.91 , 0.91 ,
        0.92 , 0.915, 0.905, 0.9  , 0.92 , 0.92 , 0.92 , 0.92 , 0.92 ,
        0.91 , 0.905, 0.905, 0.93 , 0.93 , 0.915, 0.915, 0.92 , 0.91 ,
        0.91 , 0.905, 0.93 , 0.93 , 0.905, 0.905, 0.93 , 0.925, 0.92 ,
        0.905, 0.935, 0.935, 0.915, 0.915, 0.94 , 0.93 , 0.915, 0.905,
```

```
       0.935, 0.935, 0.915, 0.915, 0.93 , 0.925, 0.92 , 0.915, 0.93 ,
       0.93 , 0.915, 0.915, 0.925, 0.92 , 0.915, 0.915, 0.93 , 0.93 ,
       0.92 , 0.92 , 0.94 , 0.93 , 0.93 , 0.92 , 0.935, 0.935, 0.92 ,
       0.92 , 0.935, 0.925, 0.93 , 0.92 , 0.935, 0.935, 0.92 , 0.92 ,
       0.935, 0.93 , 0.93 , 0.92 , 0.935, 0.935, 0.925, 0.925]),
 'split3_test_score': array([0.885, 0.885, 0.93 , 0.93 , 0.9  , 0.885, 0.9  ,
0.93 , 0.925,
       0.925, 0.925, 0.925, 0.935, 0.915, 0.93 , 0.925, 0.935, 0.94 ,
       0.93 , 0.935, 0.92 , 0.935, 0.93 , 0.945, 0.925, 0.93 , 0.94 ,
       0.94 , 0.915, 0.93 , 0.945, 0.945, 0.935, 0.935, 0.94 , 0.94 ,
       0.935, 0.945, 0.935, 0.94 , 0.935, 0.935, 0.935, 0.935, 0.94 ,
       0.945, 0.94 , 0.94 , 0.935, 0.935, 0.94 , 0.94 , 0.93 , 0.935,
       0.94 , 0.94 , 0.94 , 0.94 , 0.935, 0.935, 0.93 , 0.945, 0.935,
       0.935, 0.94 , 0.94 , 0.935, 0.935, 0.955, 0.945, 0.94 , 0.94 ,
       0.94 , 0.94 , 0.945, 0.945, 0.94 , 0.945, 0.95 , 0.945, 0.945,
       0.945, 0.945, 0.945, 0.955, 0.945, 0.955, 0.94 , 0.945, 0.95 ,
       0.945, 0.945, 0.96 , 0.96 , 0.955, 0.95 , 0.95 , 0.955, 0.95 ,
       0.945, 0.95 , 0.955, 0.95 , 0.945, 0.95 , 0.95 , 0.945, 0.945,
       0.95 , 0.95 , 0.95 , 0.945, 0.945, 0.95 , 0.935, 0.935]),
 'split4_test_score': array([0.915, 0.915, 0.94 , 0.94 , 0.905, 0.915, 0.935,
0.94 , 0.93 ,
       0.93 , 0.965, 0.965, 0.935, 0.945, 0.975, 0.96 , 0.955, 0.955,
       0.96 , 0.96 , 0.965, 0.965, 0.965, 0.96 , 0.955, 0.96 , 0.97 ,
       0.975, 0.965, 0.97 , 0.97 , 0.975, 0.965, 0.97 , 0.96 , 0.965,
       0.96 , 0.975, 0.97 , 0.97 , 0.96 , 0.96 , 0.965, 0.97 , 0.97 ,
       0.965, 0.965, 0.97 , 0.965, 0.97 , 0.97 , 0.97 , 0.96 , 0.965,
       0.975, 0.97 , 0.96 , 0.965, 0.97 , 0.97 , 0.955, 0.965, 0.97 ,
       0.975, 0.96 , 0.965, 0.96 , 0.97 , 0.965, 0.965, 0.97 , 0.97 ,
       0.96 , 0.965, 0.965, 0.965, 0.96 , 0.965, 0.965, 0.965, 0.955,
       0.96 , 0.96 , 0.96 , 0.955, 0.965, 0.96 , 0.96 , 0.965, 0.97 ,
       0.965, 0.97 , 0.965, 0.965, 0.97 , 0.97 , 0.96 , 0.96 , 0.97 ,
       0.97 , 0.97 , 0.97 , 0.97 , 0.97 , 0.97 , 0.97 , 0.97 , 0.97 ,
       0.965, 0.97 , 0.975, 0.97 , 0.96 , 0.96 , 0.97 , 0.97 ]),
 'mean_test_score': array([0.886, 0.886, 0.913, 0.913, 0.897, 0.886, 0.913,
0.913, 0.916,
       0.916, 0.926, 0.926, 0.921, 0.913, 0.933, 0.925, 0.924, 0.925,
       0.924, 0.925, 0.927, 0.924, 0.924, 0.927, 0.928, 0.93 , 0.933,
       0.934, 0.925, 0.925, 0.937, 0.936, 0.931, 0.933, 0.93 , 0.931,
       0.932, 0.936, 0.932, 0.934, 0.932, 0.932, 0.936, 0.937, 0.935,
       0.933, 0.933, 0.935, 0.934, 0.935, 0.937, 0.937, 0.931, 0.932,
       0.936, 0.934, 0.937, 0.938, 0.934, 0.934, 0.934, 0.936, 0.935,
       0.934, 0.938, 0.939, 0.93 , 0.933, 0.943, 0.937, 0.934, 0.931,
       0.938, 0.939, 0.933, 0.933, 0.937, 0.936, 0.936, 0.934, 0.934,
       0.935, 0.932, 0.932, 0.938, 0.936, 0.935, 0.934, 0.938, 0.94 ,
       0.936, 0.938, 0.944, 0.94 , 0.941, 0.939, 0.937, 0.938, 0.939,
       0.939, 0.94 , 0.938, 0.939, 0.938, 0.941, 0.941, 0.94 , 0.94 ,
       0.94 , 0.94 , 0.943, 0.939, 0.937, 0.938, 0.938, 0.938]),
```

```
 'std_test_score': array([0.02289105, 0.02289105, 0.01886796, 0.01886796,
0.01363818,
        0.02289105, 0.0143527 , 0.01886796, 0.01240967, 0.01240967,
        0.02059126, 0.02059126, 0.01655295, 0.02315167, 0.02712932,
        0.01949359, 0.02009975, 0.02073644, 0.02244994, 0.02280351,
        0.01913113, 0.02615339, 0.02267157, 0.02293469, 0.014     ,
        0.01581139, 0.02088061, 0.02267157, 0.02073644, 0.02607681,
        0.01913113, 0.02332381, 0.01854724, 0.01964688, 0.01897367,
        0.02059126, 0.01691153, 0.02244994, 0.02227106, 0.02267157,
        0.01691153, 0.01691153, 0.0156205 , 0.01749286, 0.0204939 ,
        0.02111871, 0.02014944, 0.0212132 , 0.01772005, 0.01949359,
        0.01939072, 0.01939072, 0.0156205 , 0.01860108, 0.02222611,
        0.02177154, 0.0132665 , 0.0150333 , 0.02289105, 0.02289105,
        0.01157584, 0.0174356 , 0.0204939 , 0.02332381, 0.01208305,
        0.01392839, 0.01897367, 0.02111871, 0.0150333 , 0.01630951,
        0.02267157, 0.02437212, 0.0128841 , 0.01462874, 0.02135416,
        0.02135416, 0.01469694, 0.0168523 , 0.02130728, 0.02154066,
        0.01428286, 0.01581139, 0.01989975, 0.01989975, 0.01469694,
        0.01714643, 0.02167948, 0.01714643, 0.01661325, 0.01870829,
        0.01933908, 0.01964688, 0.01772005, 0.01923538, 0.02059126,
        0.02154066, 0.01720465, 0.01805547, 0.02009975, 0.01854724,
        0.01870829, 0.02158703, 0.02009975, 0.01964688, 0.01827567,
        0.01827567, 0.01870829, 0.01870829, 0.0167332 , 0.01870829,
        0.01886796, 0.01854724, 0.01630951, 0.01691153, 0.01749286,
        0.01749286]),
 'rank_test_score': array([114, 114, 108, 108, 113, 114, 112, 108, 106, 106,
94,  94, 105,
        108,  69,  96, 101,  96, 101,  96,  92, 101, 101,  92,  91,  88,
         69,  57,  96,  96,  39,  45,  84,  69,  88,  84,  77,  45,  77,
         57,  77,  77,  42,  33,  51,  69,  69,  51,  57,  51,  39,  39,
         84,  77,  45,  57,  33,  21,  57,  57,  57,  42,  51,  57,  29,
         14,  88,  69,   2,  33,  57,  84,  29,  14,  69,  69,  33,  42,
         45,  57,  57,  51,  77,  77,  21,  45,  51,  57,  21,   7,  45,
         29,   1,   7,   4,  17,  33,  21,  17,  17,   7,  21,  14,  29,
          4,   4,   7,   7,   7,   7,   2,  17,  33,  21,  21,  21],
       dtype=int32)}
```

[70]:
```
print("Melhores parametros {} com valor de acuracia {}".format(grid.
 ↪best_params_, grid.best_score_))
```

```
Melhores parametros {'n_neighbors': 24, 'p': 1, 'weights': 'uniform'} com valor
de acuracia 0.944
```

Parametros importantes em **KNN**:

- **n_neighbors**= Nũmero de vizinhos;
- **weights**= Peso da amostra dos vizinhos (padrão = uniform);
- **metric**= Métrica usada para o calculo da distância (padrão = minkowski);

- **p**= Parâmetro de poder da métrica (padrão = 2);
- **n_jobs** = Número de threads que podem ser paralelizados durante a busca dos vizinhos (padrão = 1).

```
[ ]:
```