Aprendizagem de Máquina: Atividade 05 – Comparação de Classificadores

Carlos Emmanuel Pereira Alves
Curso de Bacharelado em Ciência da Computação
Universidade Federal do Agreste de Pernambuco (UFAPE)
Garanhuns, Brasil
carlos.emmanuel.236@gmail.com

1) Realize 100-fold cross validation estratificado na base Skin Segmentation utilizando o classificador 1-NN com distância Euclidiana então realize os procedimentos abaixo.

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import StratifiedKFold
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import precision_recall_fscore_support

labels = ['v1', 'v2', 'v3', 'class']

data = pd.read_csv('Skin_NonSkin.txt', header=None, names=labels, sep='\t')

data_x = data[labels[:-1]].to_numpy()
data_y = data['class'].to_numpy()

skfold = StratifiedKFold(n_splits=100)

score_fmetric_1 = np.array([])
score_fmetric_2 = np.array([])

for train_index, test_index in skfold.split(data_x, data_y):
  X_train, X_test = data_x[train_index], data_x[test_index]
  y_train, y_test = data_y[train_index], data_y[test_index]

  knn = KNeighborsClassifier(n_neighbors=1)
  knn.fit(X_train, y_train)
  predict = knn.predict(X_test)

  _, _, score_fmetric, _ = precision_recall_fscore_support(y_test, predict)

  score_fmetric_1 = np.append(score_fmetric_1, score_fmetric[0])
  score_fmetric_2 = np.append(score_fmetric_2, score_fmetric[1])
```

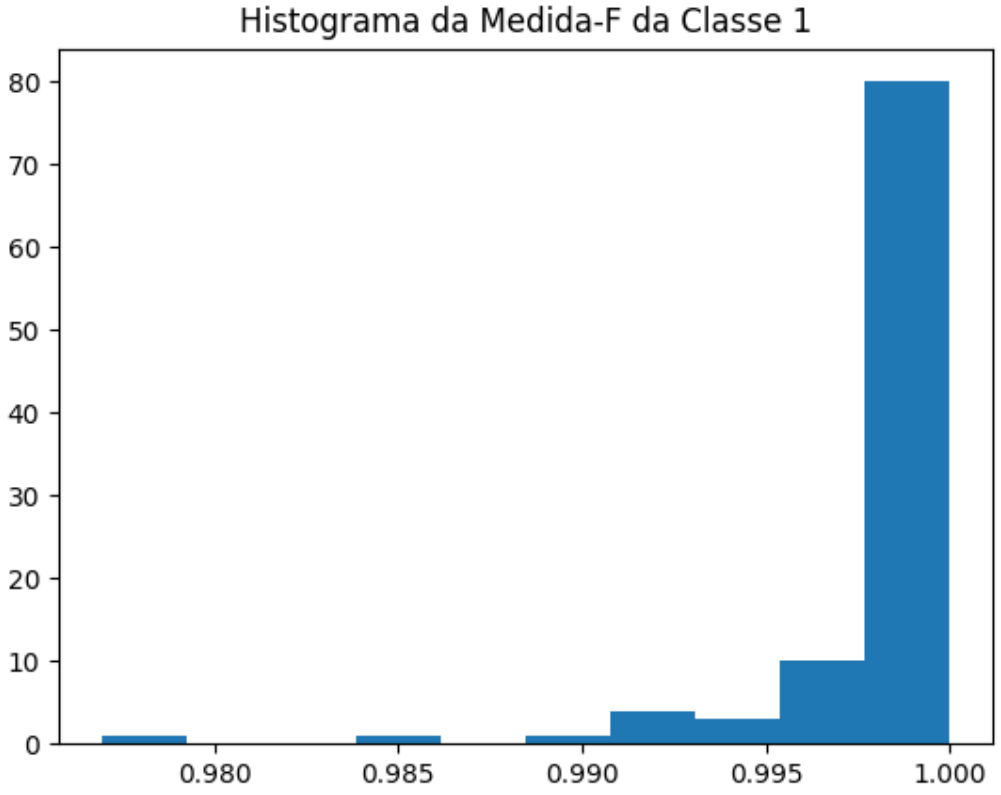
a) Mostre a média, o máximo e o mínimo da medida-F.

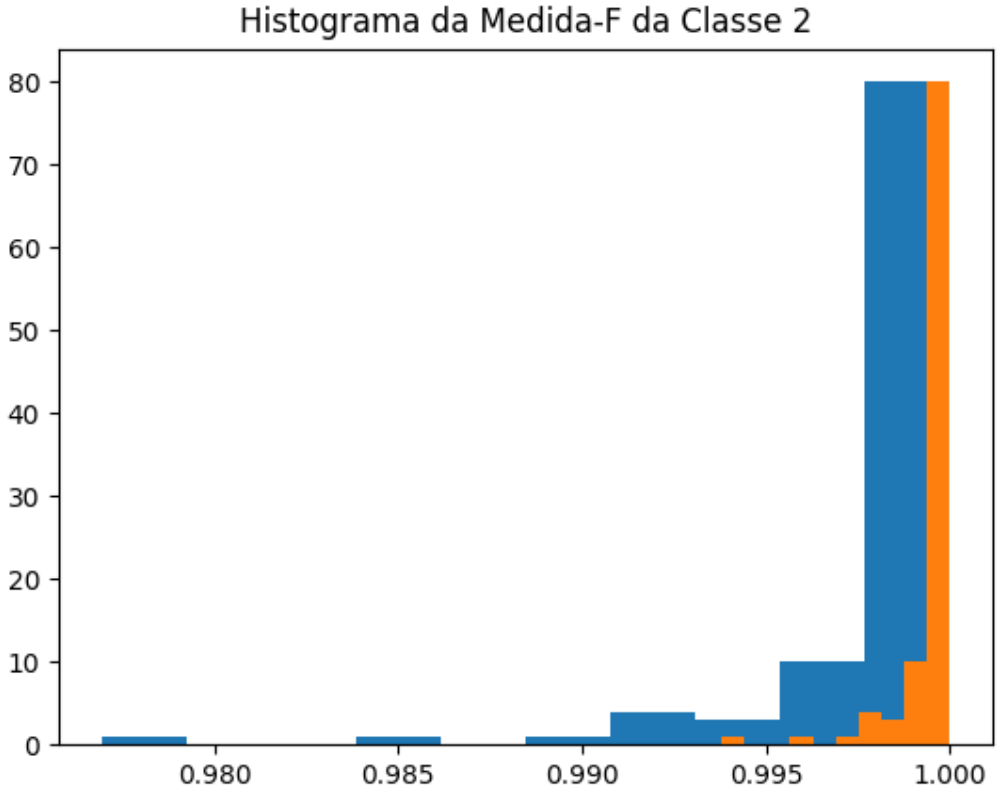
```
#Letra A
max_1 = score_fmetric_1.max()
med_1 = round(np.average(score_fmetric_1), 5)
min_1 = round(score_fmetric_1.min(), 5)

max_2 = score_fmetric_2.max()
med_2 = round(np.average(score_fmetric_2), 5)
min_2 = round(score_fmetric_2.min(), 5)

print(f'Medida-F Classe 1\nMax: {max_1}\nMed: {med_1}\nMin: {min_1}')
print(f'\nMedida-F Classe 2\nMax: {max_2}\nMed: {med_2}\nMin: {min_2}')
```

```
Medida-F Classe 1
Max: 1.0
Med: 0.99831
Min: 0.97692

Medida-F Classe 2
Max: 1.0
Med: 0.99955
Min: 0.99378
```

b) Mostre o histograma da medida-F.

```
#Letra B
plt.hist(score_fmetric_1)
plt.title('Histograma da Medida-F da Classe 1')
plt.savefig('hist_class_1.png')
plt.title('Histograma da Medida-F da Classe 2')
plt.hist(score_fmetric_2)
plt.savefig('hist_class_2.png')
```


Histograma da Medida-F da Classe 1
80
70
60
50
40
30
20
10
0
0.980
0.985
0.990
0.995
1.000


Histograma da Medida-F da Classe 2
80
70
60
50
40
30
20
10
0
0.980
0.985
0.990
0.995
1.000

c) Calcule o intervalo de confiança da medida-F.

```
#Letra C
dev_1 = np.std(score_fmetric_1)
dev_2 = np.std(score_fmetric_2)

confidence_interval_1_n = round(med_1 - (1.96 * dev_1), 5)
confidence_interval_1_p = round(med_1 + (1.96 * dev_1), 5)
confidence_interval_2_n = round(med_2 - (1.96 * dev_2), 5)
confidence_interval_2_p = round(med_2 + (1.96 * dev_2), 5)

print(f'Intervalo de confiança da medida-F da Classe 1: {confidence_interval_1_n} - {confidence_interval_1_p}')
print(f'Intervalo de confiança da medida-F da Classe 2: {confidence_interval_2_n} - {confidence_interval_2_p}')
```

```
Intervalo de confiança da medida-F da Classe 1: 0.99166 - 1.00496
Intervalo de confiança da medida-F da Classe 2: 0.99778 - 1.00132
```

d) Qual a medida-F mínima que você espera ao aplicar este classificador, sob as mesmas condições de treinamento, para dados nunca vistos?
A medida-F mínima esperada vai ser de 99,166% para a Classe 1 e de 99,778% para a Classe 2, que é o menor limite calculado no intervalo de confiança. Temos o nível de confiança em 95%, por isso podemos ter um número menor que não foi previsto em nenhum dos intervalos.

e) Qual a medida-F esperada para o classificador quando aplicada a dados nunca antes vistos.
Para dados não vistos antes teremos a média de 99,831% para a Classe 1 e de 99,955% para a Classe 2.

2) Realize um experimento pareado com 100 repetições de Holdout 50/50 utilizando o classificador 1-NN com distância Euclidiana. Utilize duas versões da base Wine archive.ics.uci.edu/ml/datasets/Wine para este experimento, a primeira versão é a base original, a segunda versão é a base sem a última coluna. Após calcular 100 taxas de acerto para cada uma das versões da base, realize os procedimentos abaixo.

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier

cols = ['class', 'alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash', 'magnesium', 'total_phenols', 'flavanoids',
        'nonflavanoid_phenols', 'proanthocyanins', 'color_intensity', 'hue', 'OD280/OD315_of_diluted_wines', 'proline']

data = pd.read_csv('wine.data', header=None, names=cols)

data_x = data[cols[1:]]
data_y = data['class']

data_alt_x = data[cols[1:-1]]
data_alt_y = data['class']

data_scores = []
data_alt_scores = []

for i in range(100):
  train_X, test_X, train_y, test_y = train_test_split(data_x, data_y, test_size=0.5)

  knn = KNeighborsClassifier(n_neighbors=1)
  knn.fit(train_X, train_y)

  data_score = knn.score(test_X, test_y)
  data_scores = np.append(data_scores, data_score)

  alt_train_X, alt_test_X, alt_train_y, alt_test_y = train_test_split(data_alt_x, data_alt_y, test_size=0.5)

  knn_without = KNeighborsClassifier(n_neighbors=1)
  knn_without.fit(alt_train_X, alt_train_y)

  data_alt_score = knn_without.score(alt_test_X, alt_test_y)
  data_alt_scores = np.append(data_alt_scores, data_alt_score)
```

a) Calcule a diferença das 100 taxas de acerto.

```python
#Letra A
print(f'Diferença das 100 taxas de acerto:\n{data_scores - data_alt_scores}')
```

```
Diferença das 100 taxas de acerto:
[-0.12359551 -0.1011236  -0.11235955 -0.03370787 -0.13483146 -0.17977528
 -0.08988764 -0.12359551 -0.07865169 -0.13483146 -0.12359551 -0.13483146
 -0.06741573 -0.14606742 -0.2247191  -0.19101124 -0.1011236  -0.11235955
 -0.12359551 -0.11235955 -0.14606742 -0.06741573 -0.1011236  -0.07865169
 -0.03370787 -0.05617978 -0.15730337 -0.21348315 -0.04494382 -0.12359551
 -0.21348315 -0.20224719 -0.21348315 -0.07865169 -0.11235955 -0.12359551
 -0.06741573 -0.1011236  -0.11235955 -0.07865169 -0.12359551 -0.23595506
 -0.11235955 -0.08988764 -0.12359551 -0.13483146 -0.14606742 -0.05617978
 -0.08988764 -0.1011236  -0.11235955 -0.20224719 -0.07865169 -0.05617978
 -0.20224719 -0.1011236  -0.06741573 -0.12359551 -0.06741573 -0.20224719
 -0.07865169 -0.06741573 -0.16853933 -0.08988764 -0.12359551 -0.30337079
 -0.05617978 -0.14606742 -0.14606742 -0.15730337 -0.12359551 -0.06741573
 -0.11235955 -0.1011236  -0.04494382 -0.06741573 -0.05617978 -0.13483146
 -0.08988764 -0.08988764 -0.05617978 -0.14606742 -0.19101124 -0.07865169
 -0.19101124 -0.1011236  -0.1011236  -0.05617978 -0.03370787 -0.14606742
 -0.21348315 -0.19101124 -0.05617978 -0.04494382 -0.12359551 -0.05617978
 -0.15730337 -0.14606742  0.02247191 -0.21348315]
```

b) Calcule o intervalo de confiança destas diferenças.

```
#Letra B
med = np.average(data_scores - data_alt_scores)
dev = np.std(data_scores - data_alt_scores)

confidence_interval_n = round(med - (1.96 * dev), 5)
confidence_interval_p = round(med + (1.96 * dev), 5)

print(f'Intervalo de confiança: {confidence_interval_n}   {confidence_interval_p}')
```

```
Intervalo de confiança: -0.2252   -0.00896
```

c) Realize o teste de hipótese sobre estas diferenças para verificar se a diferença da taxa de acerto é significativa entre as duas versões. Mostre sua conclusão para o teste.

	Completa	Sem a última coluna	Diferença
0	0.685393	0.842697	-0.157303
1	0.662921	0.842697	-0.179775
2	0.662921	0.831461	-0.168539
3	0.719101	0.775281	-0.056180
4	0.719101	0.853933	-0.134831
5	0.696629	0.898876	-0.202247
6	0.685393	0.865169	-0.179775
7	0.685393	0.820225	-0.134831
8	0.696629	0.808989	-0.112360
9	0.730337	0.842697	-0.112360
10	0.730337	0.831461	-0.101124
11	0.685393	0.842697	-0.157303
12	0.617978	0.831461	-0.213483
13	0.786517	0.887640	-0.101124
14	0.651685	0.876404	-0.224719
15	0.674157	0.865169	-0.191011
16	0.719101	0.820225	-0.101124
17	0.662921	0.831461	-0.168539
18	0.719101	0.808989	-0.089888
19	0.775281	0.820225	-0.044944
20	0.696629	0.876404	-0.179775
21	0.685393	0.842697	-0.157303
22	0.696629	0.808989	-0.112360
23	0.674157	0.865169	-0.191011
24	0.741573	0.808989	-0.067416
25	0.786517	0.820225	-0.033708
26	0.685393	0.853933	-0.168539
27	0.651685	0.865169	-0.213483
28	0.719101	0.764045	-0.044944
29	0.752809	0.865169	-0.112360
30	0.662921	0.865169	-0.202247
31	0.707865	0.887640	-0.179775
32	0.719101	0.853933	-0.134831
33	0.674157	0.808989	-0.134831
34	0.696629	0.842697	-0.146067
35	0.685393	0.820225	-0.134831
36	0.674157	0.842697	-0.168539
37	0.674157	0.797753	-0.123596
38	0.696629	0.786517	-0.089888
39	0.707865	0.831461	-0.123596
40	0.707865	0.831461	-0.123596
41	0.775281	0.876404	-0.101124
42	0.719101	0.842697	-0.123596
43	0.730337	0.786517	-0.056180
44	0.764045	0.831461	-0.067416
45	0.696629	0.820225	-0.123596
46	0.685393	0.808989	-0.123596
47	0.685393	0.786517	-0.101124
48	0.707865	0.842697	-0.134831
49	0.651685	0.853933	-0.202247
50	0.696629	0.808989	-0.112360
51	0.730337	0.876404	-0.146067
52	0.741573	0.775281	-0.033708
53	0.674157	0.808989	-0.134831
54	0.674157	0.853933	-0.179775
55	0.719101	0.853933	-0.134831
56	0.808989	0.775281	0.033708
57	0.741573	0.865169	-0.123596
58	0.640449	0.853933	-0.213483
59	0.764045	0.876404	-0.112360
60	0.752809	0.831461	-0.078652
61	0.662921	0.842697	-0.179775
62	0.730337	0.831461	-0.101124
63	0.674157	0.853933	-0.179775
64	0.617978	0.831461	-0.213483
65	0.651685	0.955056	-0.303371
66	0.640449	0.820225	-0.179775
67	0.629213	0.853933	-0.224719
68	0.640449	0.842697	-0.202247
69	0.696629	0.865169	-0.168539
70	0.696629	0.842697	-0.146067
71	0.730337	0.775281	-0.044944
72	0.730337	0.842697	-0.112360
73	0.685393	0.786517	-0.101124
74	0.707865	0.764045	-0.056180
75	0.662921	0.808989	-0.146067
76	0.741573	0.820225	-0.078652
77	0.707865	0.853933	-0.146067
78	0.741573	0.786517	-0.044944
79	0.674157	0.786517	-0.112360
80	0.707865	0.752809	-0.044944
81	0.685393	0.853933	-0.168539
82	0.685393	0.842697	-0.157303
83	0.741573	0.853933	-0.112360
84	0.696629	0.865169	-0.168539
85	0.617978	0.808989	-0.191011
86	0.696629	0.842697	-0.146067
87	0.741573	0.797753	-0.056180
88	0.674157	0.786517	-0.112360
89	0.674157	0.808989	-0.134831
90	0.662921	0.887640	-0.224719
91	0.696629	0.910112	-0.213483
92	0.730337	0.797753	-0.067416
93	0.640449	0.786517	-0.146067
94	0.674157	0.808989	-0.134831
95	0.707865	0.831461	-0.123596
96	0.606742	0.898876	-0.292135
97	0.651685	0.842697	-0.191011
98	0.696629	0.797753	-0.101124
99	0.674157	0.887640	-0.213483

Como mostrado na letra B, o 0 está fora do intervalo, então rejeitamos o H_0, ou seja, os classificadores não tem o mesmo erro. Analisando os dados acima vemos que, a base sem a última coluna tem uma taxa de acerto significativamente maior.

d) Calcule o intervalo de confiança da taxa de acerto para cada versão da base.

```
#Letra D
med_1 = np.average(data_scores)
dev_1 = data_scores.std()

confidence_interval_1_n = round(med_1 - (1.96 * dev_1), 5)
confidence_interval_1_p = round(med_1 + (1.96 * dev_1), 5)

print(f'Intervalo de confiança da taxa de acerto da base completa: {confidence_interval_1_n}   {confidence_interval_1_p}')

med_alt = np.average(data_alt_scores)
dev_alt = data_alt_scores.std()

confidence_interval_alt_1_n = round(med_alt - (1.96 * dev_alt), 5)
confidence_interval_alt_1_p = round(med_alt + (1.96 * dev_alt), 5)

print(f'Intervalo de confiança da taxa de acerto da base sem a ultima coluna: {confidence_interval_alt_1_n}   {confidence_interval_alt_1_p}')
```

```
Intervalo de confiança da taxa de acerto da base completa: 0.64296   0.79075
Intervalo de confiança da taxa de acerto da base sem a ultima coluna: 0.76418   0.90368
```

e) Realize o teste de hipótese de sobreposição dos intervalos de confiança. Mostre sua conclusão para o teste.
Existe sobreposição entre os intervalos de confiança, portanto, não podemos afirmar se os classificadores possuem ou não taxas de acerto diferentes.

3) Qual o número máximo de características que podem ser removidas da base Iris archive.ics.uci.edu/ml/datasets/iris sem reduzir significativamente a taxa de acerto? Defina a metodologia utilizada para justificar sua resposta.

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import precision_recall_fscore_support as score

cols =['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'class']

data = pd.read_csv('iris.data', header=None, names=cols)

data_x = data[cols[:-1]] #-4 para tirar 3 colunas, -3 para tirar 2 colunas, -2 para tirar 1 coluna e -1 para a base completa
data_y = data['class']

data_scores = []

for i in range(100):
  train_X, test_X, train_y, test_y = train_test_split(data_x, data_y, test_size=0.5)

  knn = KNeighborsClassifier(n_neighbors = 1)
  knn.fit(train_X, train_y)

  data_score = knn.score(test_X, test_y)
  data_scores = np.append(data_scores, data_score)

med = np.average(data_scores)
dev = np.std(data_scores)

confidence_interval_n = round(med - (1.96 * dev), 5)
confidence_interval_p = round(med + (1.96 * dev), 5)
```

```python
print('Utilizando o 1-nn com distância euclidiana, 100 repetições e holdout 50/50, e a base completa:\n')
print(f'Intervalo de confiança da taxa de acerto: {confidence_interval_n}   {confidence_interval_p}')
```

```
Utilizando o 1-nn com distância euclidiana, 100 repetições e holdout 50/50, e a base completa:

Intervalo de confiança da taxa de acerto: 0.91741   0.98926
```

```python
print('Utilizando o 1-nn com distância euclidiana, 100 repetições e holdout 50/50, e a base sem a última coluna:\n')
print(f'Intervalo de confiança da taxa de acerto: {confidence_interval_n}   {confidence_interval_p}')
```

```
Utilizando o 1-nn com distância euclidiana, 100 repetições e holdout 50/50, e a base sem a última coluna:

Intervalo de confiança da taxa de acerto: 0.88587   0.96986
```

```python
print('Utilizando o 1-nn com distância euclidiana, 100 repetições e holdout 50/50, e a base sem as duas últimas colunas:\n')
print(f'Intervalo de confiança da taxa de acerto: {confidence_interval_n}   {confidence_interval_p}')
```

```
Utilizando o 1-nn com distância euclidiana, 100 repetições e holdout 50/50, e a base sem as duas últimas colunas:

Intervalo de confiança da taxa de acerto: 0.63988   0.79746
```

```python
print('Utilizando o 1-nn com distância euclidiana, 100 repetições e holdout 50/50, e a base sem as três últimas colunas:\n')
print(f'Intervalo de confiança da taxa de acerto: {confidence_interval_n}   {confidence_interval_p}')
```

```
Utilizando o 1-nn com distância euclidiana, 100 repetições e holdout 50/50, e a base sem as três últimas colunas:

Intervalo de confiança da taxa de acerto: 0.48342   0.72058
```

Podemos ver que retirando 1 coluna o intervalo de confiança da taxa de acerto não diminui tanto, ainda temos sobreposição. Mas quando retiramos 2 colunas o intervalo de confiança da taxa de acerto cai significativamente. E,

ainda, quando retiramos 3 colunas o intervalo de confiança da taxa de acerto cai drasticamente. Então o número máximo de colunas que podem ser retiradas sem afetar significativamente o intervalo de confiança da taxa de acerto é 1.

4) Utilizando o classicador k-NN na base Wine archive.ics.uci.edu/ml/datasets/Wine, teste os valores k = 1, . . . , 15. Para qual valor de k o classificador apresenta uma taxa de acerto significativamente maior? Defina a metodologia utilizada para justificar sua resposta.

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier

cols = ['class', 'alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash', 'magnesium', 'total_phenols', 'flavanoids',
        'nonflavanoid_phenols', 'proanthocyanins', 'color_intensity', 'hue', 'OD280/OD315_of_diluted_wines', 'proline']

data = pd.read_csv('wine.data', header=None, names=cols)

data_x = data[cols[1:]]
data_y = data['class']

data_scores = []

for i in range(100):
  train_X, test_X, train_y, test_y = train_test_split(data_x, data_y, test_size=0.5)

  knn = KNeighborsClassifier(n_neighbors=1) #aumentar o n_neighbors de acordo com o número de vizinhos que queremos
  knn.fit(train_X, train_y)

  data_score = knn.score(test_X, test_y)
  data_scores = np.append(data_scores, data_score)

med = np.average(data_scores)
dev = np.std(data_scores)

confidence_interval_n = round(med - (1.96 * dev), 5)
confidence_interval_p = round(med + (1.96 * dev), 5)

print(knn.n_neighbors,'-nn', sep='')
print(f'Média de acertos: {round(med, 5)}')
print(f'Intervalo de confiança: [{confidence_interval_n}    {confidence_interval_p}]')
```

```
1-nn
Média de acertos: 0.71551
Intervalo de confiança: [0.6358    0.79521]
```

```
2-nn
Média de acertos: 0.67079
Intervalo de confiança: [0.59068    0.75089]
```

```
3-nn
Média de acertos: 0.6918
Intervalo de confiança: [0.60985    0.77374]
```

```
4-nn
Média de acertos: 0.68596
Intervalo de confiança: [0.603    0.76891]
```

```
5-nn
Média de acertos: 0.69584
Intervalo de confiança: [0.62372    0.76796]
```

```
6-nn
Média de acertos: 0.68865
Intervalo de confiança: [0.61059    0.76672]
```

```
7-nn
Média de acertos: 0.68596
Intervalo de confiança: [0.60945    0.76246]
```

```
8-nn
Média de acertos: 0.6773
Intervalo de confiança: [0.60281    0.7518]
```

```
9-nn
Média de acertos: 0.68539
Intervalo de confiança: [0.61458    0.75621]
```

```
10-nn
Média de acertos: 0.6827
Intervalo de confiança: [0.61222    0.75318]
```

```
11-nn
Média de acertos: 0.69213
Intervalo de confiança: [0.61969    0.76458]
```

```
12-nn
Média de acertos: 0.68719
Intervalo de confiança: [0.60317    0.77121]
```

```
13-nn
Média de acertos: 0.69528
Intervalo de confiança: [0.62084    0.76972]
```

```
14-nn
Média de acertos: 0.68708
Intervalo de confiança: [0.61811    0.75605]
```

```
15-nn
Média de acertos: 0.69708
Intervalo de confiança: [0.61997    0.77418]
```

Vemos que todos os intervalos se sobrepõem, então não podemos rejeitar o H_0.
Devemos olhar outras métricas para tentar classificá-los.