Relatório Reconhecimento de Padrões: Atividade 04 − 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.

Inicialmente para todas as letras eu carreguei a base com o pandas, separei a classe dos atributos, e usei o StratifiedKFold para realizar o 100-fold cross validation estratificado. Então fiz então o treinamento com o Knn guardando os scores nas variáveis score_fmetric_1 e score_fmetric_2.

```
data = pd.read_csv('Skin_NonSkin.txt', header=None, names=['v1', 'v2', 'v3', 'class'], sep='\t')

data_x = data[['v1', 'v2', 'v3']].to_numpy()
data_y = data['class'].to_numpy()

skfold = StratifiedKFold(n_splits=100)
n_splits = skfold.get_n_splits(data_x, data_y)

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

for train_index, test_index in skfold.split(data_x, data_y):
  data_x_train, data_x_teste = data_x[train_index], data_x[test_index]
  data_y_train, data_y_teste = data_y[train_index], data_y[test_index]

  knn = KNeighborsClassifier(n_neighbors = 1, weights="distance", metric="euclidean")
  knn.fit(data_x_train, data_y_train)
  predict = knn.predict(data_x_teste)

  _, _, score_fmetric, _ = precision_recall_fscore_support(data_y_teste, 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.

Nessa letra eu calculei o máximo usando a função max, a média usando o numpy.average e o mínimo com a função min, nos dois últimos também usei a função round para arredondar em 5 casas decimais.

```
#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'Medida-F Classe 2\nMax: {max_2}\nMed: {med_2}\nMin: {min_2}')
```

O resultado foi:

```
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
```

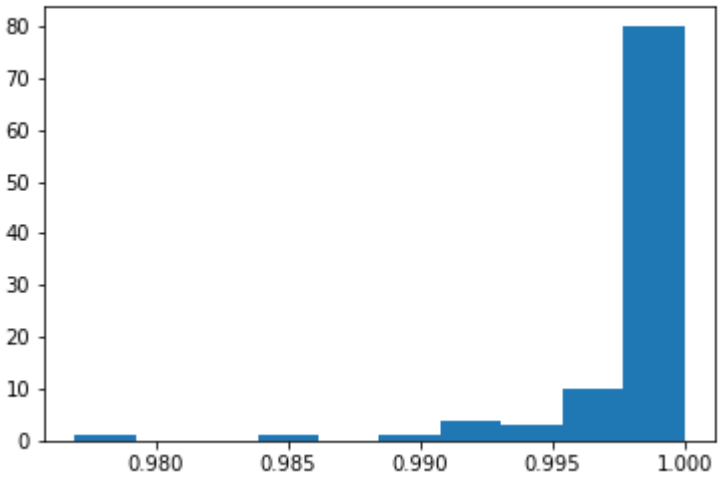
b) Mostre o histograma da medida-F.

Para criar a figura do histograma eu usei a biblioteca matplotlib.pyplot com a função hist, que gera o histograma. Depois utilizei a função savefig para salvar a figura.

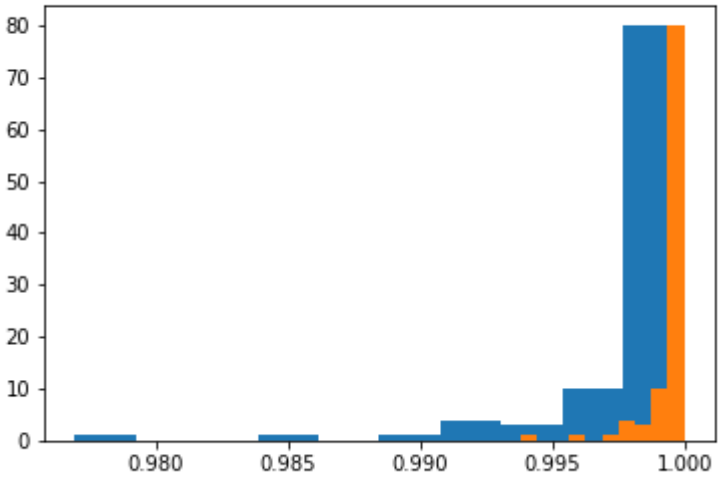
```
#Letra B
plt.hist(score_fmetric_1)
plt.savefig('hist_class_1.png')
plt.hist(score_fmetric_2)
plt.savefig('hist_class_2.png')
```

O resultado:

Classe 1



Classe 2



c) Calcule o intervalo de confiança da medida-F.
Para o cálculo do intervalo de confiança da medida-F primeiro eu obtive o desvio padrão, utilizando a função numpy.std. Depois eu calculei o intervalo de confiança utilizando a fórmula dada em aula, e arredondei novamente em 5 casas decimais.

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

O resultado

```
Intervalo de confiança da Classe 1: 0.99166 - 1.00496
Intervalo de confiança 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.

Inicialmente para todas as letras eu carreguei a base com o pandas, separei a classe dos atributos, nas duas versões da base. Então fiz o treinamento com Knn e guardei os scores nas variáveis data_scores, para a base completa, e data_scores_without, para a base sem a última coluna.

```
cols = ['class', 'alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash', 'magnesium', 'total_phenols', 'flavanoids', 'nonflavanoid_phenols', 'proanthocyanins', 'color_intensity', 'hue', 'OD28

data = pd.read_csv('wine.data', header=None, names=['class', 'alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash', 'magnesium', 'total_phenols', 'flavanoids', 'nonflavanoid_phenols', 'proan

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

data_without = pd.read_csv('wine.data', header=None, names=['class', 'alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash', 'magnesium', 'total_phenols', 'flavanoids', 'nonflavanoid_phenols'

data_without_x = data[cols[1:-1]]
data_without_y = data['class']

data_scores = []
data_scores_without = []

for i in range(100):
  data_train_x, data_test_x, data_train_y, data_test_y = train_test_split(data_x, data_y, test_size=0.5)

  knn = KNeighborsClassifier(n_neighbors = 1, weights="distance", metric="euclidean")
  knn.fit(data_train_x, data_train_y)

  predict = knn.predict(data_test_x)
  data_score = knn.score(data_test_x, data_test_y)
  data_scores = np.append(data_scores, data_score)

  data_without_train_x, data_without_test_x, data_without_train_y, data_without_test_y = train_test_split(data_without_x, data_without_y, test_size=0.5)

  knn_without = KNeighborsClassifier(n_neighbors = 1, weights="distance", metric="euclidean")
  knn_without.fit(data_without_train_x, data_without_train_y)

  predict_without = knn_without.predict(data_without_test_x)
  data_score_without = knn_without.score(data_without_test_x, data_without_test_y)
  data_scores_without = np.append(data_scores_without, data_score_without)
```

a) Calcule a diferença das 100 taxas de acerto.
Para essa letra eu apenas fiz o print das variáveis da taxa de acerto, colocando uma menos a outra para obter o resultado.

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

Resultado:

```
Diferença das 100 taxas de acerto:
[-0.1011236  -0.15730337 -0.08988764 -0.11235955 -0.07865169 -0.19101124
 -0.13483146 -0.07865169 -0.11235955 -0.11235955 -0.14606742  0.03370787
 -0.16853933 -0.21348315 -0.08988764 -0.15730337  0.03370787 -0.03370787
 -0.1011236  -0.25842697 -0.05617978 -0.19101124 -0.15730337 -0.11235955
 -0.13483146 -0.11235955 -0.1011236  -0.1011236  -0.06741573 -0.15730337
 -0.08988764 -0.21348315 -0.1011236  -0.11235955 -0.07865169 -0.06741573
 -0.04494382 -0.12359551  0.02247191 -0.07865169 -0.15730337 -0.14606742
 -0.02247191 -0.08988764 -0.04494382 -0.08988764 -0.12359551 -0.01123596
 -0.06741573 -0.14606742 -0.1011236  -0.08988764 -0.08988764 -0.11235955
 -0.1011236  -0.08988764 -0.03370787 -0.25842697 -0.11235955 -0.16853933
 -0.16853933 -0.08988764 -0.16853933 -0.15730337 -0.13483146 -0.1011236
 -0.12359551 -0.12359551 -0.07865169 -0.21348315 -0.13483146 -0.1011236
 -0.14606742 -0.12359551 -0.20224719 -0.16853933 -0.16853933 -0.23595506
 -0.03370787 -0.12359551 -0.1011236  -0.1011236  -0.12359551 -0.13483146
 -0.11235955 -0.03370787 -0.12359551 -0.16853933 -0.11235955 -0.06741573
 -0.15730337 -0.03370787 -0.01123596 -0.17977528 -0.14606742 -0.05617978
 -0.07865169 -0.16853933 -0.1011236  -0.08988764]
```

b) Calcule o intervalo de confiança destas diferenças.
Para fazer o cálculo do intervalo de confiança das diferenças eu novamente utilizei a função numpy.average, para obter a média, e a

numpy.std, para obter o desvio padrão, sempre colocando uma menos a outra pois é a diferença. Depois calculo o intervalo de confiança, como na questão anterior.

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

confidence_interval = round(med - (1.96 * dev), 5)
confidence_interval_w = round(med + (1.96 * dev), 5)

print(f'Intervalo de confiança: {confidence_interval}   {confidence_interval_w}')
```

Resultado:

```
Intervalo de confiança: -0.23994   0.00061
```

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.
Como o 0 está fora do intervalo, rejeitamos o H_0, ou seja, os classificadores não tem o mesmo erro. Analisando 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.
Dessa vez como foi pedido para cada versão da base, eu calculei igual a questão anterior, mas apenas de cada base separada.

```
#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_without = np.average(data_scores_without)
dev_without = data_scores_without.std()

confidence_interval_without_1_n = round(med_without - (1.96 * dev_without), 5)
confidence_interval_without_1_p = round(med_without + (1.96 * dev_without), 5)

print(f'Intervalo de confiança da taxa de acerto da base sem a ultima coluna: {confidence_interval_without_1_n}   {confidence_interval_without_1_p}')
```

Resultado:

```
Intervalo de confiança da taxa de acerto da base completa: 0.6307   0.79919
Intervalo de confiança da taxa de acerto da base sem a ultima coluna: 0.75781   0.91141
```

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.

Inicialmente eu carreguei a base com o pandas, separei a classe dos atributos, e fui removendo a base de acordo com o comentário do código. Depois fiz o Knn, guardando os scores a cada repetição, depois calculei o intervalo de confiança como feito nas outras questões.

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

data = pd.read_csv('iris.data', header=None, names=['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'class'])

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):
  data_train_x, data_test_x, data_train_y, data_test_y = train_test_split(data_x, data_y, test_size=0.5)

  knn = KNeighborsClassifier(n_neighbors = 1, weights="distance", metric="euclidean")
  knn.fit(data_train_x, data_train_y)

  predict = knn.predict(data_test_x)
  data_score = knn.score(data_test_x, data_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(f'Intervalo de confiança: {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: [0.92012 0.99135]

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: [0.87729 0.97338]

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: [0.64748 0.81812]

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: [0.49659 0.72874]

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.

Inicialmente eu carreguei a base com o pandas, separei a classe dos atributos, utilizei o Knn até 15 com 100 repetições e holdout 50/50, e fiz como nas outras questões guardando o score e printando a média de acertos e o intervalo de confiança.

```
cols = ['class', 'alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash', 'magnesium', 'total_phenols', 'flavanoids', 'nonflavanoid_phenols', 'proanthocyanins', 'color_inte

data = pd.read_csv('wine.data', header=None, names=['class', 'alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash', 'magnesium', 'total_phenols', 'flavanoids', 'nonflavan

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

data_scores = []

for i in range(100):
  data_train_x, data_test_x, data_train_y, data_test_y = train_test_split(data_x, data_y, test_size=0.5)

  knn = KNeighborsClassifier(n_neighbors = 1, weights="distance", metric="euclidean") #aumentar o n_neighbors de acordo com o número de vizinhos que queremos
  knn.fit(data_train_x, data_train_y)

  predict = knn.predict(data_test_x)
  data_score = knn.score(data_test_x, data_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(f'Média de acertos: {round(med, 5)}')
print(f'Intervalo de confiança: [{confidence_interval_n}    {confidence_interval_p}]')
```

1-nn
Média de acertos: 0.71506
Intervalo de confiança: [0.62015 0.80996]

2-nn
Média de acertos: 0.72472
Intervalo de confiança: [0.63793 0.81151]

3-nn
Média de acertos: 0.70449
Intervalo de confiança: [0.62537 0.78362]

4-nn
Média de acertos: 0.70124
Intervalo de confiança: [0.6193 0.78317]

5-nn
Média de acertos: 0.71191
Intervalo de confiança: [0.63211 0.79171]

6-nn
Média de acertos: 0.71169
Intervalo de confiança: [0.6387 0.78467]

7-nn
Média de acertos: 0.71416
Intervalo de confiança: [0.63804 0.79027]

8-nn
Média de acertos: 0.70652
Intervalo de confiança: [0.63182 0.78122]

9-nn
Média de acertos: 0.71146
Intervalo de confiança: [0.63152 0.7914]

10-nn
Média de acertos: 0.71056

Intervalo de confiança: [0.63054 0.79058]

11-nn
Média de acertos: 0.7091
Intervalo de confiança: [0.62536 0.79284]

12-nn
Média de acertos: 0.70281
Intervalo de confiança: [0.62015 0.78547]

13-nn
Média de acertos: 0.71371
Intervalo de confiança: [0.64231 0.7851]

14-nn
Média de acertos: 0.70528
Intervalo de confiança: [0.62342 0.78714]

15-nn
Média de acertos: 0.71798
Intervalo de confiança: [0.64672 0.78924]

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