DL_01

March 3, 2021

Deep Learning

Rogério de Oliveira

---

1 Neurônio simples

Um neurônio artificial do tipo **perceptron** faz uma combinação linear de entradas e aplica uma função de ativação como a função *sign*, *tanh* ou *relu* para produzir uma saída.

$$f(X) = sign(w_0 + w_1 x_1 + ... + w_n x_n)$$

O treinamento do neurônio é feito ajustando-se os pesos w_n de acordo com base em uma função de custo (por exemplo uma medida do erro e predição) obtido para se estimar a saída $f(X) \cong y$.

$$\min_W \sum ||f(X) - y||$$

Desse modo, você pode entender o aprendizado de um neurônio como um problema de otimização.

2 Redes neurais

Um único neurônio entretanto tem uma capacidade bastante limitada de aprendizado, restringindo-se a problemas de separação linear. Desse modo, por exemplo, ele não consegue aprender uma função como a função **XOR**.

Função $XOR(X) \rightarrow y$:

```
  X      y
 0 0     0
 0 1     1
 1 0     1
 1 0     0
```

que é não linearmente separável.

Para resolver essa limitação podemos então trabalhar com múltiplos neurônios em camadas. As saídas dos neurônios de uma camada são então empregadas como entradas para a camada seguinte.

As camadas entre a camada inicial de neurônios (de entrada) e a camada final (de saída) constituem as camadas ocultas da rede.

O treinamento da rede segue o mesmo princípio, embora mais complexo, ajustando os pesos w_n de acordo com o erro de predição obtido para se estimar a saída $f(X) \cong y$.

$$\min_W \sum ||f(X) - y||$$

Chamamos esse aprendizado de *backpropagation* ou *retropropagação*.

Acesse agora http://playground.tensorflow.org/ para uma demonstração.

3 Esquema Geral para Modelos Supervisionados e MLP

Modelos de Aprendizado Supervisionado seguem todos um esquema bastante geral no `scikit learn` e em muitos frameworks. Modelos Supervisionados de Redes Neurais podem ser implementados com o `scikit learn` e seguem a mesma estrutura.

4 Toy-example: `Remain or Leave?`

Abaixo um *toy-example*, um conjunto de dados de notas de alunos, de 0 a 5, para as disciplinas `A, B, C, D` e se o aluno `R` (*remain*) permanece no curso no final do semestre ou `L` (*leave*) deixa o curso.

Apesar de um exemplo simpls, muitos outros problemas interesse se encaixam nessa mesma tipologia como problemas de *churn* de clientes, *fraud/non-fraud*, *credit/non-credit*, *defect/not-defect*, *benign/malign* etc.

Fig. 1.Esquema Geral para Modelos Supervisionados, aqui empregando uma Árvore de Decisão.

5 Obtenção dos Dados

[1]:
```
import pandas as pd
students = pd.DataFrame({'A':[3, 5, 1, 1, 4, 2, 1, 5, 2, 4, 4, 2, 2, 2, 5, 5,␣
↪3, 4, 5, 4],
                        'B':[1, 1, 4, 5, 2, 4, 2, 3, 2, 1, 4, 3, 5, 2, 4, 1,␣
↪3, 2, 3, 2],
                        'C':[2, 1, 5, 3, 3, 1, 3, 4, 1, 4, 2, 4, 4, 1, 4, 1,␣
↪5, 1, 4, 3],
                        'D':[2, 3, 3, 1, 1, 2, 3, 4, 2, 4, 3, 5, 4, 2, 3, 1,␣
↪2, 2, 3, 2],
                        'status':['R', 'L', 'R', 'L', 'L', 'L', 'R', 'R', 'L',␣
↪'R', 'L', 'R', 'R', 'L', 'L', 'L', 'R', 'L', 'L', 'L']})
print(students)
new_students = pd.DataFrame({'A':[5, 4, 4, 3],
                            'B':[1, 4, 1, 3],
                            'C':[1, 1, 2, 3],
```

```
                                    'D':[2, 4, 1, 3],
                                    'status':['?', '?', '?', '?']})
print(new_students)
```

```
    A  B  C  D status
0   3  1  2  2      R
1   5  1  1  3      L
2   1  4  5  3      R
3   1  5  3  1      L
4   4  2  3  1      L
5   2  4  1  2      L
6   1  2  3  3      R
7   5  3  4  4      R
8   2  2  1  2      L
9   4  1  4  4      R
10  4  4  2  3      L
11  2  3  4  5      R
12  2  5  4  4      R
13  2  2  1  2      L
14  5  4  4  3      L
15  5  1  1  1      L
16  3  3  5  2      R
17  4  2  1  2      L
18  5  3  4  3      L
19  4  2  3  2      L
   A  B  C  D status
0  5  1  1  2      ?
1  4  4  1  4      ?
2  4  1  2  1      ?
3  3  3  3  3      ?
```

6 Definição das Variáveis Preditoras `X` e Dependente `y`

Neste ponto é esperado que os dados já estejam prontos para a aplicação do modelo tendo sido analisados e transformados nas fases de Entendimento e Preparação dos Dados (tratamento de nulos, hot encode, normalização e outras transformações necessárias).

[2]:
```
X = students[['A','B','C','D']]
y = students.status
```

7 Separação dos Conjuntos de Treinamento e Teste X_train, X_test, y_train, y_test

[3]:
```
from sklearn.model_selection import train_test_split
seed = 1984

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

8 Declaração do Modelo clf

[4]:
```
from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier(criterion='gini',
                             max_depth=None,
                             random_state=seed)
```

9 Treinamento do modelo fit()

[5]:
```
clf.fit(X_train, y_train)
```

[5]:
```
DecisionTreeClassifier(random_state=1984)
```

10 Predição do Conjunto de Teste predict()

[6]:
```
y_pred = clf.predict(X_test)
```

11 Avaliação das métricas do modelo

Dependente do tipo de modelo. Aqui empregamos apenas a acuracidade.

[7]:
```
from sklearn.metrics import accuracy_score

accuracy = accuracy_score(y_pred, y_test)
print(f'Accuracy: {accuracy :0.3f} %')
```

```
Accuracy: 1.000 %
```

12 Predição dos novos casos

[8]:
```
X_new = new_students[['A','B','C','D']]

y_pred = clf.predict(X_new)

print(y_pred)
```

```
['L' 'R' 'L' 'R']
```

Nota O esquema apresentado é aqui é um esquema introdutório com propósitos unicamente didáticos e é, portanto, um modelo bastante simplificado. Particularmete ele não leva em consideração múltiplas execuções dos modelos ou técnicas como Cross Validation e também emprega uma única métrica de resultados, a acuracidade. O seguimento desse tema deveria incluir o Cross Validation e outras métricas (ROC, F1-score, recall, overfitting etc.).

13 Esquema Geral Completo

Colocando todas as células em um único código temos o seguinte:

[9]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

from sklearn.tree import DecisionTreeClassifier

seed = 1984

X = students[['A','B','C','D']]
y = students.status

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪stratify=y, random_state=seed)

clf = DecisionTreeClassifier(criterion='gini',
                             max_depth=None,
                             random_state=seed)

#
# Por exemplo, alternativamente, poderíamos empregar um outro modelo como um␣
↪SVC no lugar a Árvore de Decisão
#
# from sklearn import svm
# clf = svm.SVC()
#
# Todos as demais instruções, poderiam ser mantidas sem qualquer alteração.
#
```

```
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

accuracy = accuracy_score(y_pred, y_test)
print(f'Accuracy: {accuracy :0.3f} %')

X_new = new_students[['A','B','C','D']]

y_pred = clf.predict(X_new)

print(y_pred)
```

```
Accuracy: 1.000 %
['L' 'R' 'L' 'R']
```

Note que

> “‘ from sklearn.tree import DecisionTreeClassifier clf = DecisionTreeClassifier(criterion=‘gini’, max_depth=None, random_state=seed)

```
Poderia ser substituído por qualquer outro modelo, por exemplo uma Support Vector Machine:

>```
from sklearn import svm
clf = svm.SVC()
```

Com todas as demais instruções mantidas sem qualquer mudança.

14 Mais Métricas (opcional)

[10]:
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score

y_pred = clf.predict(X_test)

cm = confusion_matrix(y_pred, y_test)
print(cm)

accuracy = accuracy_score(y_pred, y_test)
print(accuracy)

print(classification_report(y_pred,y_test))
```

```
[[4 0]
 [0 2]]
1.0
              precision    recall  f1-score   support
```

```
           L       1.00      1.00      1.00         4
           R       1.00      1.00      1.00         2

    accuracy                           1.00         6
   macro avg       1.00      1.00      1.00         6
weighted avg       1.00      1.00      1.00         6
```

Alternativamente

[11]:
```python
clf.score(X_test, y_test)
```

[11]:
```
1.0
```

15 Exercício

Altere o esquema geral apresentado para aplicação de um modelo de rede neural MLP.

```
from sklearn.neural_network import MLPClassifier

clf = MLPClassifier(solver='lbfgs',
              alpha = ...,
              max_iter = ...,
              hidden_layer_sizes = ...,
              random_state = ...)
```

Nota

> Função de Ativação: * activation{'identity', 'logistic', 'tanh', 'relu'}, default='relu'
>
> Solvers: * L-BFGS: Use para pequenos conjuntos de dados. * Adam: Use para grandes conjuntos de dados (default). * SGD: Gradiente estocástico, requer definir corretamente parâmetros como taxa de aprendizado, momentum etc.

[12]:
```python
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

from sklearn.neural_network import MLPClassifier

seed = 1984

X = students[['A','B','C','D']]
y = students.status

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪stratify=y, random_state=seed)

clf = MLPClassifier(solver='lbfgs',
```

```
                alpha = 0.01,
                max_iter = 100,
                hidden_layer_sizes = (2,2),
                random_state = seed)

clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

accuracy = accuracy_score(y_pred, y_test)
print(f'Accuracy: {accuracy :0.3f} %')

X_new = new_students[['A','B','C','D']]

y_pred = clf.predict(X_new)

print(y_pred)
```

```
Accuracy: 1.000 %
['L' 'L' 'L' 'R']
```

```
/home/pbraga/.local/lib/python3.8/site-
packages/sklearn/neural_network/_multilayer_perceptron.py:471:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

16 Exibindo os pesos da MLP

Este código executará sobre o modelo o exercício anterior (`clf`).

[13]:
```
import numpy as np

print(f'\nCamadas : ')
for i in range(len(clf.coefs_)):
  print(np.shape(clf.coefs_[i]))

for i in range(len(clf.coefs_)):
  print(f'\nW{i}: ')
  print(clf.coefs_[i])
  print(f'\nB{i}: ')
  print(clf.intercepts_[i])
```

```

Camadas :
```

```
(4, 2)
(2, 2)
(2, 1)

W0:
[[ 1.0703103  -0.27709637]
 [ 1.53423257 -0.42054466]
 [-1.0931328   0.30892174]
 [-1.50550167  0.46162624]]

B0:
[1.16101666 0.33004816]

W1:
[[ 2.66789684e+00  7.15011937e-04]
 [-7.21939314e-01 -3.02613406e-02]]

B1:
[ 0.41857034 -0.8053998 ]

W2:
[[-2.77532442]
 [-0.03938361]]

B2:
[10.97377198]
```

17 Exercício

Neste exercício você fará a predição de diagnóstico de câncer de mama a partir de features já extraídas de imagens para diagnóstico. Você pode usar os dados pré-formatados aqui, evitando assim empregar os dados brutos da fonte original.

Decision Tree Model in the Diagnosis of Breast Cancer

[Breast Cancer Data](https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

Analise os dados e, antes de criar seu modelo neural, verifique a necessidade tratamentos prévios dos dados como:

- *Feature selection* com a exclusão de atributos para o treinamento ( )
- Tratamento dos dados faltantes ( )
- *Hot encode* para conversão de dados categóricos ( )
- Normalização dos dados ( )

Apresente os resultados de acuracidade do seu modelo para as variáveis preditoras normalizadas e não normalizadas. Qual a sua conclusão?

[101]:
```python
import pandas as pd
breast = pd.read_csv('http://meusite.mackenzie.br/rogerio/DLA2021S1/
↪breast_cancer.csv')
breast.head()
```

[101]:
```
         id diagnosis  radius_mean  texture_mean  perimeter_mean  area_mean  \
0    842302         M        17.99         10.38          122.80     1001.0
1    842517         M        20.57         17.77          132.90     1326.0
2  84300903         M        19.69         21.25          130.00     1203.0
3  84348301         M        11.42         20.38           77.58      386.1
4  84358402         M        20.29         14.34          135.10     1297.0

   smoothness_mean  compactness_mean  concavity_mean  concave points_mean  \
0          0.11840           0.27760          0.3001              0.14710
1          0.08474           0.07864          0.0869              0.07017
2          0.10960           0.15990          0.1974              0.12790
3          0.14250           0.28390          0.2414              0.10520
4          0.10030           0.13280          0.1980              0.10430

   …  radius_worst  texture_worst  perimeter_worst  area_worst  \
0  …         25.38          17.33           184.60      2019.0
1  …         24.99          23.41           158.80      1956.0
2  …         23.57          25.53           152.50      1709.0
3  …         14.91          26.50            98.87       567.7
4  …         22.54          16.67           152.20      1575.0

   smoothness_worst  compactness_worst  concavity_worst  concave points_worst  \
0            0.1622             0.6656           0.7119                0.2654
1            0.1238             0.1866           0.2416                0.1860
2            0.1444             0.4245           0.4504                0.2430
3            0.2098             0.8663           0.6869                0.2575
4            0.1374             0.2050           0.4000                0.1625

   symmetry_worst  fractal_dimension_worst
0          0.4601                  0.11890
1          0.2750                  0.08902
2          0.3613                  0.08758
3          0.6638                  0.17300
4          0.2364                  0.07678

[5 rows x 32 columns]
```

[84]:
```python
# Seu código

# Mapeando a feature 'diagnosis' para '0' ou '1'
diag = {"M":0, "B":1}
breast['diagnosis'] = breast['diagnosis'].map(diag)
```

[85]:
```
breast.head()
```

[85]:
```
         id  diagnosis  radius_mean  texture_mean  perimeter_mean  area_mean  \
0    842302          0        17.99         10.38          122.80     1001.0
1    842517          0        20.57         17.77          132.90     1326.0
2  84300903          0        19.69         21.25          130.00     1203.0
3  84348301          0        11.42         20.38           77.58      386.1
4  84358402          0        20.29         14.34          135.10     1297.0

   smoothness_mean  compactness_mean  concavity_mean  concave points_mean  \
0          0.11840           0.27760          0.3001              0.14710
1          0.08474           0.07864          0.0869              0.07017
2          0.10960           0.15990          0.1974              0.12790
3          0.14250           0.28390          0.2414              0.10520
4          0.10030           0.13280          0.1980              0.10430

   …  radius_worst  texture_worst  perimeter_worst  area_worst  \
0  …         25.38          17.33           184.60      2019.0
1  …         24.99          23.41           158.80      1956.0
2  …         23.57          25.53           152.50      1709.0
3  …         14.91          26.50            98.87       567.7
4  …         22.54          16.67           152.20      1575.0

   smoothness_worst  compactness_worst  concavity_worst  concave points_worst  \
0            0.1622             0.6656           0.7119                0.2654
1            0.1238             0.1866           0.2416                0.1860
2            0.1444             0.4245           0.4504                0.2430
3            0.2098             0.8663           0.6869                0.2575
4            0.1374             0.2050           0.4000                0.1625

   symmetry_worst  fractal_dimension_worst
0          0.4601                  0.11890
1          0.2750                  0.08902
2          0.3613                  0.08758
3          0.6638                  0.17300
4          0.2364                  0.07678

[5 rows x 32 columns]
```

[86]:
```
breast.corr().style.background_gradient().set_precision(2)
```

[86]:
```
<pandas.io.formats.style.Styler at 0x7f60aa1badf0>
```

[87]:
```
fList=[]
for i in breast:
    fList.append(i)
```

[88]:
```python
# Removendo features com correlação menor que [.1]
cVal=0
for j in fList:
    cVal = breast['diagnosis'].corr(breast[j])
    if cVal < 0:
        cVal*=-1
    if cVal < 0.1:
        breast = breast.drop([j], axis = 1)
```

[89]:
```python
breast.head()
```

[89]:
```
   diagnosis  radius_mean  texture_mean  perimeter_mean  area_mean  \
0          0        17.99         10.38          122.80     1001.0
1          0        20.57         17.77          132.90     1326.0
2          0        19.69         21.25          130.00     1203.0
3          0        11.42         20.38           77.58      386.1
4          0        20.29         14.34          135.10     1297.0

   smoothness_mean  compactness_mean  concavity_mean  concave points_mean  \
0          0.11840           0.27760          0.3001              0.14710
1          0.08474           0.07864          0.0869              0.07017
2          0.10960           0.15990          0.1974              0.12790
3          0.14250           0.28390          0.2414              0.10520
4          0.10030           0.13280          0.1980              0.10430

   symmetry_mean  …  radius_worst  texture_worst  perimeter_worst  \
0         0.2419  …         25.38          17.33           184.60
1         0.1812  …         24.99          23.41           158.80
2         0.2069  …         23.57          25.53           152.50
3         0.2597  …         14.91          26.50            98.87
4         0.1809  …         22.54          16.67           152.20

   area_worst  smoothness_worst  compactness_worst  concavity_worst  \
0      2019.0            0.1622             0.6656           0.7119
1      1956.0            0.1238             0.1866           0.2416
2      1709.0            0.1444             0.4245           0.4504
3       567.7            0.2098             0.8663           0.6869
4      1575.0            0.1374             0.2050           0.4000

   concave points_worst  symmetry_worst  fractal_dimension_worst
0                0.2654          0.4601                  0.11890
1                0.1860          0.2750                  0.08902
2                0.2430          0.3613                  0.08758
3                0.2575          0.6638                  0.17300
4                0.1625          0.2364                  0.07678

[5 rows x 26 columns]
```

[94]:
```python
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

from sklearn.neural_network import MLPClassifier

seed = 1984

X = breast.loc[:, breast.columns != 'diagnosis']
y = breast.loc[:, breast.columns == 'diagnosis']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪stratify=y, random_state=seed)

clf = MLPClassifier(solver='lbfgs',
              alpha = 0.01,
              max_iter = 100,
              hidden_layer_sizes = (2,2),
              random_state = seed)

clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

accuracy = accuracy_score(y_pred, y_test)
print(f'Accuracy: {accuracy :0.3f} %')
```

```
Accuracy: 0.626 %
```

```
/home/pbraga/.local/lib/python3.8/site-packages/sklearn/utils/validation.py:72:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  return f(**kwargs)
```

[91]:
```python
# Normalizando os dados
from sklearn.preprocessing import MinMaxScaler

# create a scaler object
scaler = MinMaxScaler()
# fit and transform the data
breastNorm = pd.DataFrame(scaler.fit_transform(breast), columns=breast.columns)
```

[96]:
```python
X = breastNorm.loc[:, breastNorm.columns != 'diagnosis']
y = breastNorm.loc[:, breastNorm.columns == 'diagnosis']

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

```
clf = MLPClassifier(solver='lbfgs',
              alpha = 0.01,
              max_iter = 100,
              hidden_layer_sizes = (2,2),
              random_state = seed)

clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

accuracy = accuracy_score(y_pred, y_test)
print(f'Accuracy: {accuracy :0.3f} %')
```

```
Accuracy: 0.959 %
```

```
/home/pbraga/.local/lib/python3.8/site-packages/sklearn/utils/validation.py:72:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  return f(**kwargs)
/home/pbraga/.local/lib/python3.8/site-
packages/sklearn/neural_network/_multilayer_perceptron.py:471:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

Após a normalização dos dados a precisão do preditor melhora consideravelmente pois as features são colocadas em uma mesma escala.

18 Aplicando Cross-Validation

Este código executará sobre o modelo e conjuntos de dados do exercício anterior (`clf, X_train, y_train, X_test, y_test`).

[97]:
```
from sklearn.model_selection import cross_val_score

cv = cross_val_score(clf, X_train, y_train, cv=10)
test_score = clf.fit(X_train, y_train).score(X_test, y_test)

print('CV accuracy score: %0.3f' % np.mean(cv))
print('Test accuracy score: %0.3f' % (test_score))
```

```
/home/pbraga/.local/lib/python3.8/site-packages/sklearn/utils/validation.py:72:
DataConversionWarning: A column-vector y was passed when a 1d array was
```

```
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  return f(**kwargs)
/home/pbraga/.local/lib/python3.8/site-packages/sklearn/utils/validation.py:72:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  return f(**kwargs)
/home/pbraga/.local/lib/python3.8/site-
packages/sklearn/neural_network/_multilayer_perceptron.py:471:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/home/pbraga/.local/lib/python3.8/site-packages/sklearn/utils/validation.py:72:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  return f(**kwargs)
/home/pbraga/.local/lib/python3.8/site-
packages/sklearn/neural_network/_multilayer_perceptron.py:471:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/home/pbraga/.local/lib/python3.8/site-packages/sklearn/utils/validation.py:72:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  return f(**kwargs)
/home/pbraga/.local/lib/python3.8/site-
packages/sklearn/neural_network/_multilayer_perceptron.py:471:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/home/pbraga/.local/lib/python3.8/site-packages/sklearn/utils/validation.py:72:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  return f(**kwargs)
/home/pbraga/.local/lib/python3.8/site-
```

```
packages/sklearn/neural_network/_multilayer_perceptron.py:471:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/home/pbraga/.local/lib/python3.8/site-packages/sklearn/utils/validation.py:72:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  return f(**kwargs)
/home/pbraga/.local/lib/python3.8/site-
packages/sklearn/neural_network/_multilayer_perceptron.py:471:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/home/pbraga/.local/lib/python3.8/site-packages/sklearn/utils/validation.py:72:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  return f(**kwargs)
/home/pbraga/.local/lib/python3.8/site-
packages/sklearn/neural_network/_multilayer_perceptron.py:471:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/home/pbraga/.local/lib/python3.8/site-packages/sklearn/utils/validation.py:72:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  return f(**kwargs)
/home/pbraga/.local/lib/python3.8/site-
packages/sklearn/neural_network/_multilayer_perceptron.py:471:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/home/pbraga/.local/lib/python3.8/site-packages/sklearn/utils/validation.py:72:
DataConversionWarning: A column-vector y was passed when a 1d array was
```

```
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  return f(**kwargs)
/home/pbraga/.local/lib/python3.8/site-
packages/sklearn/neural_network/_multilayer_perceptron.py:471:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/home/pbraga/.local/lib/python3.8/site-packages/sklearn/utils/validation.py:72:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  return f(**kwargs)
/home/pbraga/.local/lib/python3.8/site-
packages/sklearn/neural_network/_multilayer_perceptron.py:471:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/home/pbraga/.local/lib/python3.8/site-packages/sklearn/utils/validation.py:72:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  return f(**kwargs)
```

```
CV accuracy score: 0.932
Test accuracy score: 0.959
```

```
/home/pbraga/.local/lib/python3.8/site-
packages/sklearn/neural_network/_multilayer_perceptron.py:471:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

19 Vários Modelos Comparados (opcional)

[98]:
```python
models = [ ]

from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(criterion='gini',
```

```
                              max_depth=None,
                              random_state=seed)
models.append(['Decision Tree', clf])

from sklearn import neighbors
n_neighbors = 5
clf = neighbors.KNeighborsClassifier(n_neighbors)

models.append(['Knn', clf])

from sklearn.naive_bayes import GaussianNB, BernoulliNB
clf = BernoulliNB()

models.append(['Naive Bayes', clf])

from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(n_estimators=10)

models.append(['Random Forest', clf])

from sklearn import svm
clf = svm.SVC()

models.append(['Support Vector Machines', clf])

from sklearn.neural_network import MLPClassifier

clf = MLPClassifier(solver='lbfgs',
              alpha = 1e-5,
              max_iter = 10000,
              hidden_layer_sizes = (3, 10, 2),
              random_state = seed)

models.append(['MLP Neural Network', clf])

from sklearn.metrics import accuracy_score
for model in models:
    clf = model[1]
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_pred, y_test)
    print('Model ' + model[0] + ' accuracy: {:0.3f} %'.format(accuracy))
```

```
Model Decision Tree accuracy: 0.906 %
Model Knn accuracy: 0.971 %
Model Naive Bayes accuracy: 0.632 %
Model Random Forest accuracy: 0.947 %
```

```
Model Support Vector Machines accuracy: 0.965 %
Model MLP Neural Network accuracy: 0.626 %
```

```
<ipython-input-98-2b59bc4d4739>:43: DataConversionWarning: A column-vector y was
passed when a 1d array was expected. Please change the shape of y to (n_samples,
), for example using ravel().
  clf.fit(X_train, y_train)
/home/pbraga/.local/lib/python3.8/site-packages/sklearn/utils/validation.py:72:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  return f(**kwargs)
<ipython-input-98-2b59bc4d4739>:43: DataConversionWarning: A column-vector y was
passed when a 1d array was expected. Please change the shape of y to
(n_samples,), for example using ravel().
  clf.fit(X_train, y_train)
/home/pbraga/.local/lib/python3.8/site-packages/sklearn/utils/validation.py:72:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  return f(**kwargs)
/home/pbraga/.local/lib/python3.8/site-packages/sklearn/utils/validation.py:72:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  return f(**kwargs)
```