Projeto da disciplina Aprendizagem de Máquina

Professores: Evandro Costa e Xu Yang

Alunas: Ana Correia e Helynne Lima

O banco de dados escolhido para trabalhar neste projeto foi o Bank Marketing Data Set, disponível aqui.

```
import pandas as pd

url = 'https://github.com/mhelynne/bank-project-am/blob/master/bank/bank-num.csv?raw=true'
data = pd.read_csv(url, sep=",", index_col=0)

data.head()
```

	age	education	default	balance	housing	loan	day	month	duration	campaign	pday
0	58	3	0	2143	1	0	5	5	261	1	-
1	44	2	0	29	1	0	5	5	151	1	-
2	33	2	0	2	1	1	5	5	76	1	-
3	47	0	0	1506	1	0	5	5	92	1	-
4	33	0	0	1	0	0	5	5	198	1	-

Imports e configurações iniciais

```
import math
import numpy as np
from time import time
from sklearn.preprocessing import StandardScaler

from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier
from sklearn.svm import SVC
from sklearn.model_selection import cross_validate


target = data.pop('y')
cv = 10

scoring = ['accuracy', 'f1_weighted']
```

Função de treinamento e teste

```
def train_test(model, data= data, target= target, cv= cv):

  train_log = ""
  t0 = time()

  scores = cross_validate(model, data, target, cv= cv,
                          scoring=scoring, return_train_score=False)

  train_log  += "Tempo gasto: " + str(round(time()-t0, 3)) + "s\n"

  fit_time = np.mean(scores['fit_time'])
  train_log  += "\nTempo médio de treinamento: " + str(round(fit_time, 3))

  score_time = np.mean(scores['score_time'])
  train_log  += "\nTempo médio de teste: " + str(round(score_time, 3))

  total_time = fit_time + score_time
  train_log  += "\nTempo total médio: " + str(round(score_time, 3))

  test_accuracy = np.mean(scores['test_accuracy'])
  train_log  += "\nAcurácia média: " + str(round(test_accuracy, 3))

  test_f1_weighted = np.mean(scores['test_f1_weighted'])
  train_log  += "\nF1-score médio de teste: " + str(round(test_f1_weighted, 3))

  return train_log
```

KNN

k = raiz quadrada de n

```
n = data.shape[0]
sqrt_n = math.sqrt(n)
neigh = KNeighborsClassifier(n_neighbors= int(sqrt_n))
log = train_test(neigh)
print(log)
```

```
Tempo gasto: 9.905s

Tempo médio de treinamento: 0.24
Tempo médio de teste: 0.748
Tempo total médio: 0.748
Acurácia média: 0.884
F1-score médio de teste: 0.847
```

k = 10

```
neigh10 = KNeighborsClassifier(n_neighbors= 10)

log = train_test(neigh10)
print(log)
```

```
Tempo gasto: 5.124s

Tempo médio de treinamento: 0.231
Tempo médio de teste: 0.28
Tempo total médio: 0.28
```

Randon Forest

```
rdnf = RandomForestClassifier(criterion='entropy', max_depth= 5, random_state=42)

log = train_test(rdnf)
print(log)
```

```
Tempo gasto: 17.347s

Tempo médio de treinamento: 1.697
Tempo médio de teste: 0.036
Tempo total médio: 0.036
Acurácia média: 0.883
F1-score médio de teste: 0.842
```

Naive Bayes

```
gnb = GaussianNB()

log = train_test(gnb)
print(log)
```

```
Tempo gasto: 0.379s

Tempo médio de treinamento: 0.031
Tempo médio de teste: 0.005
Tempo total médio: 0.005
Acurácia média: 0.785
F1-score médio de teste: 0.802
```

Multi-layer Perceptron

```
mlp = MLPClassifier(solver='lbfgs', hidden_layer_sizes=(3, ), random_state=1)

log = train_test(mlp)
print(log)
```

```
Tempo gasto: 4.527s

Tempo médio de treinamento: 0.444
Tempo médio de teste: 0.007
Tempo total médio: 0.007
Acurácia média: 0.388
F1-score médio de teste: 0.344
```

Ajuste de parâmetros da Rede Neral

```
'''
mlp_aj = MLPClassifier(max_iter=100)

#definição do espaço de hiperparâmetros
parameter_space = {
    'hidden_layer_sizes': [(50,50,50), (50,100,50), (100,)],
    'activation': ['tanh', 'relu'],
    'solver': ['sgd', 'adam', 'lbfgs'],
    'alpha': [0.0001, 0.05],
    'learning_rate': ['constant','adaptive'],
}

# A execucao desse bloco de codigo pode demorar.

from sklearn.model_selection import GridSearchCV

clf = GridSearchCV(mlp_aj, parameter_space, n_jobs=-1, cv=cv)
clf.fit(data, target)

# Best paramete set
print('Best parameters found:\n', clf.best_params_)
'''
```

O resultado do GridSearchCV foi:

```
Best parameters found:
{'activation': 'relu', 'alpha': 0.05, 'hidden_layer_sizes': (50, 100, 50), 'learning_rate': 'adapti
```

Aplicando os parâmetros encontrados:

```
mlp = MLPClassifier(activation='relu', alpha=0.05,
                    hidden_layer_sizes=(50, 100, 50),
                    learning_rate='adaptive',
                    solver='sgd', random_state=42)

log = train_test(mlp)
print(log)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/neural_network/_multilayer_perceptron.p
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/neural_network/_multilayer_perceptron.p
  % self.max_iter, ConvergenceWarning)
Tempo gasto: 756.482s

Tempo médio de treinamento: 75.628
Tempo médio de teste: 0.017
Tempo total médio: 0.017
Acurácia média: 0.882
F1-score médio de teste: 0.85
/usr/local/lib/python3.6/dist-packages/sklearn/neural_network/_multilayer_perceptron.p
  % self.max_iter, ConvergenceWarning)
```

SVM

```
svm = SVC(max_iter=10000)
log = train_test(svm)
print(log)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/svm/_base.py:231: ConvergenceWarning: S
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/svm/_base.py:231: ConvergenceWarning: S
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/svm/_base.py:231: ConvergenceWarning: S
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/svm/_base.py:231: ConvergenceWarning: S
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/svm/_base.py:231: ConvergenceWarning: S
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/svm/_base.py:231: ConvergenceWarning: S
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/svm/_base.py:231: ConvergenceWarning: S
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/svm/_base.py:231: ConvergenceWarning: S
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/svm/_base.py:231: ConvergenceWarning: S
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/svm/_base.py:231: ConvergenceWarning: S
  % self.max_iter, ConvergenceWarning)
Tempo gasto: 477.897s

Tempo médio de treinamento: 44.923
Tempo médio de teste: 2.864
Tempo total médio: 2.864
Acurácia média: 0.882
F1-score médio de teste: 0.829
```

Com dados padronizados:

```
svm = SVC(max_iter=10000)

ss = StandardScaler()
data_std = pd.DataFrame(ss.fit_transform(data), columns = data.columns)

log = train_test(svm, data_std)
print(log)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/svm/_base.py:231: ConvergenceWarning: S
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/svm/_base.py:231: ConvergenceWarning: S
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/svm/_base.py:231: ConvergenceWarning: S
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/svm/_base.py:231: ConvergenceWarning: S
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/svm/_base.py:231: ConvergenceWarning: S
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/svm/_base.py:231: ConvergenceWarning: S
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/svm/_base.py:231: ConvergenceWarning: S
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/svm/_base.py:231: ConvergenceWarning: S
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/svm/_base.py:231: ConvergenceWarning: S
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/svm/_base.py:231: ConvergenceWarning: S
  % self.max_iter, ConvergenceWarning)
Tempo gasto: 396.719s

Tempo médio de treinamento: 36.891
Tempo médio de teste: 2.78
Tempo total médio: 2.78
Acurácia média: 0.871
F1-score médio de teste: 0.849
```

Aa .* log = train_test(neigh) pNão encontrado Substituir por SUBSTITUIR