Exemplo de Classificação

1) Identificação do Problema

Nome da base: Wine
Qtde instâncias: 178
Atributos de entrada: 13 valores numéricos, a saber:

1) Alcohol
2) Malic acid
3) Ash
4) Alcalinity of ash
5) Magnesium
6) Total phenols
7) Flavanoids
8) Nonflavanoid phenols
9) Proanthocyanins
10) Color intensity
11) Hue
12) OD280/OD315 of diluted wines
13) Proline

Atributo alvo: classe do vinho (1, 2 ou 3)
Número de instâncias por classe: 59, 71 e 48, respectivamente.
Descrição do problema: classificar vinhos com base em dados extraídos de análise química de vinhos produzidos em uma mesma região da Itália, porém derivados de 3 diferentes cultivos.
Link para a base de dados e descrição detalhada: https://archive.ics.uci.edu/ml/datasets/wine

2) Descrição dos experimentos

Objetivo deste exemplo é apresentar a tarefa de classificação e como podemos executá-la usando Python e sua biblioteca de Machine Learning (Scikit Learn). Criaremos árvores de decisão com base em dois protocolos experimentais que diferem na forma que utilizamos para avaliar a árvore criada.

- Protocolo Experimental 1
  - Estratégia de teste Holdout
    - 70% das instâncias para treinamento
    - 30% das instâncias para teste
    - Seleção de aleatória das instâncias de treinamento e teste
- Protocolo Experimental 2
  - Estratégia Validação Cruzada com 10 pastas
    - Base dividida aleatoriamente em 10 pastas/folds, em cada uma de 10 iterações, uma pasta diferente é usada para teste e as outras 9 para treinamento. A taxa de acerto final é calculada com base na média das 10 árvores criadas.
- Medida de desempenho: Taxa de acerto (%)
- Análise de erros via matriz de confusão.

3) Script Python Utilizado

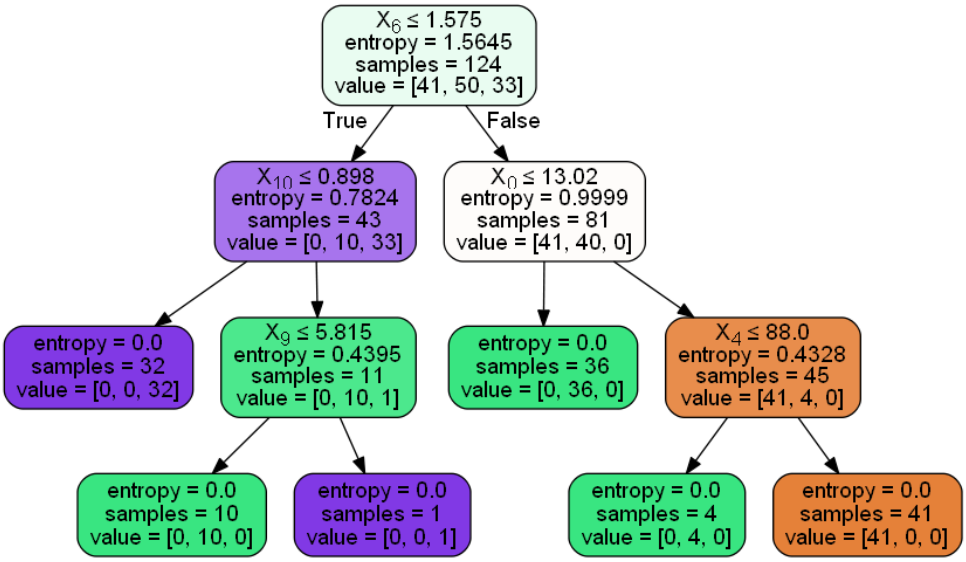
```
# Este exemplo carrega a base Wine da UCI, treina uma árvode de decisão usando
# holdout e outra usando validação cruzada com 10 pastas.
# Importa bibliotecas necessárias
import numpy as np
import urllib
from sklearn import tree
from sklearn import  cross_validation
from sklearn.metrics import confusion_matrix
from sklearn.cross_validation import train_test_split
from sklearn.externals.six import StringIO
from sklearn.tree import export_graphviz
from IPython.display import Image
from IPython.display import display
import pydotplus

# Carrega uma base de dados do UCI (neste exemplo a base wine)
# Cuidado este procedimento carrega apenas bases com atributos numéricos
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data"
raw_data = urllib.request.urlopen(url)

# Carrega arquivo como uma matriz
dataset = np.loadtxt(raw_data, delimiter=",")

# Imprime quantidade de instâncias e atributos da base
print("Instâncias e atributos")
print(dataset.shape)

# Coloca em X os 13 atributos de entrada e em y as classes
# Observe que na base Wine a classe é o primeiro atributo
X = dataset[:,1:13]
y = dataset[:,0]

# EXEMPLO USANDO HOLDOUT
# Holdout -> dividindo a base em treinamento (70%) e teste (30%). Tal divisão é aleatória e
estratificada, significa que os exemplos são selecionados aleatoriamente e a distribuição das
classes é respeitada para gerar as bases de treinamento e teste.
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=.3, random_state=42, stratify=y)
```

```
# declara o classificador, neste caso uma árvore de decisão será criada com base na teoria de ganho de informação de cada atributo estimado a partir de entropia.
clfa = tree.DecisionTreeClassifier(criterion='entropy')

# aqui ocorre o treinamento  o classificador
clfa = clfa.fit(X_train, y_train)

# avaliação usando a base de testes
predicted=clfa.predict(X_test)

# calcula a taxa de acerto na base de teste
score=clfa.score(X_test, y_test)

# calcula a matriz de confusão
matrix = confusion_matrix(y_test, predicted)

# apresenta os resultados
print("\nResultados baseados em Holdout 70/30")
print("Taxa de acerto = %.2f " % score)
print("Matriz de confusão:")
print(matrix)

# EXEMPLO USANDO VALIDAÇÃO CRUZADA
# declarada uma nova árvore (isto não significa treinar)
clfb = tree.DecisionTreeClassifier(criterion='entropy')
folds=10

# aqui sim ocorre o treinamento e teste usando validação cruzada. 10 Árvores são treinadas e testadas.
result = cross_validation.cross_val_score(clfb, X, y, cv=folds)

# Mostrando os resultados (taxa de acerto)
print("\nResultados baseados em Validação Cruzada")
print("Qtde folds: %d:" % folds)
print("Taxa de Acerto: %.2f" % result.mean())
print("Desvio padrão: %.2f" % result.std())

# matriz de confusão da validação cruzada
Z = cross_validation.cross_val_predict(clfb, X, y, cv=folds)
```

```
cm=confusion_matrix(y, Z)
print("Matriz de confusão:")
print(cm)

# Imprime a árvore gerada. Escolhemos imprimir a árvore do experimento holdout.
print("\nÁrvore Gerada no experimento baseado em Holdout")
dot_data = StringIO()
export_graphviz(clfa, out_file=dot_data,  filled=True, rounded=True, special_characters=True)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
im=Image(graph.create_png())
display(im)
```

4) Resultados do Processo de Classificação

4.1)Árvore Gerada (experimento holdout)



Explicaremos o processo de geração (algoritmo indutor) nas próximas aulas. Conforme é possível observar somente alguns atributos do problema foram utilizados (apenas os de maior ganho de informação). O mais discriminativo segundo a árvore é o X6 (corresponde na verdade ao atributo 7-Flavanoids), pois nesta base as instâncias apresentam a classe como primeiro atributo. Nós internos da árvore avaliam algum atributo, os nós folhas representam as classes do problema, os valores entre colchetes apontam a quantidade de instâncias de cada classe utilizadas nos cálculos daquele nó. Quanto maior a entropia maior a confusão entre classes naquele nó.

4.2)Taxas de acertos e Matriz de Confusão

Resultados baseados em Holdout 70/30
Taxa de acerto = 0.91 (91%)
Matriz de confusão:
[[16 2 0]
[1 19 1]
[0 1 14]]

Resultados baseados em Validação Cruzada (10 folds)
Taxa de Acerto: 0.92 (92%)
Desvio padrão: 0.06
Matriz de confusão:
[[54 5 0]
[2 68 1]
[3 1 44]]

Análise: Considerando os resultados da validação cruzada (normalmente recomendada para problemas com poucas instâncias na base), observamos uma taxa de acerto de 92%. Vale destacar que a árvore foi criada e nenhum ajuste fino de parâmetros foi realizado, este resultado pode sim ser ainda bem melhor. Na matriz de confusão temos na diagonal principal os acertos do classificador, enquanto nas demais posições temos seus erros. Por exemplo, na matriz da validação cruzada é possível observar que a classe de vinho 1 (primeira linha da matriz) apresenta 54 instâncias corretamente classificadas e 5 classificadas erroneamente como classe 2. Nenhum vinho da classe 1 foi confundido, ou classificado errado, como classe 3. A classe mais problemática foi a 1 com 5 erros.

Observações importantes:

i) A análise dos resultados é importante para definirmos os próximos passos que pode ser ajustar da árvore ou mesmo utilizar outros algoritmos indutores.

ii) Os resultados de holdout e validação cruzada não são comparáveis, pois representam protocolos experimentais diferentes.