Exemplo de Regressão

1) Identificação do Problema

Nome da base: Boston Housing
Qtde instâncias: 506
Atributos de entrada: 13 valores numéricos, a saber:

- **CRIM**: Per capita crime rate by town
- **ZN**: Proportion of residential land zoned for lots over 25,000 sq. ft
- **INDUS**: Proportion of non-retail business acres per town
- **CHAS**: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise
- **NOX**: Nitric oxide concentration (parts per 10 million)
- **RM**: Average number of rooms per dwelling
- **AGE**: Proportion of owner-occupied units built prior to 1940
- **DIS**: Weighted distances to five Boston employment centers
- **RAD**: Index of accessibility to radial highways
- **TAX**: Full-value property tax rate per $10,000
- **PTRATIO**: Pupil-teacher ratio by town
- **B**: $1000(Bk - 0.63)^2$, where Bk is the proportion of [people of African American descent] by town
- **LSTAT**: Percentage of lower status of the population
- **MEDV**: Median value of owner-occupied homes in $1000s

Atributo alvo: valor contínuo representando o preço médio de casas ocupadas na cidade de Bolton (em unidades de 1000)

Descrição do problema: predição do valor de uma casa dado os atributos de entrada.
Levantamento de dados feito em Boston (USA);
Link para a base de dados e descrição detalhada: https://www.kaggle.com/vikrishnan/boston-house-prices

2) Descrição dos experimentos

Objetivo deste exemplo é apresentar a tarefa de regressão e como podemos executá-la usando Python e sua biblioteca de Machine Learning (Scikit Learn). Criaremos árvores de regressão 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: coeficiente de determinação (R2 score), uma das formas de avaliar a qualidade do ajuste do modelo criado.

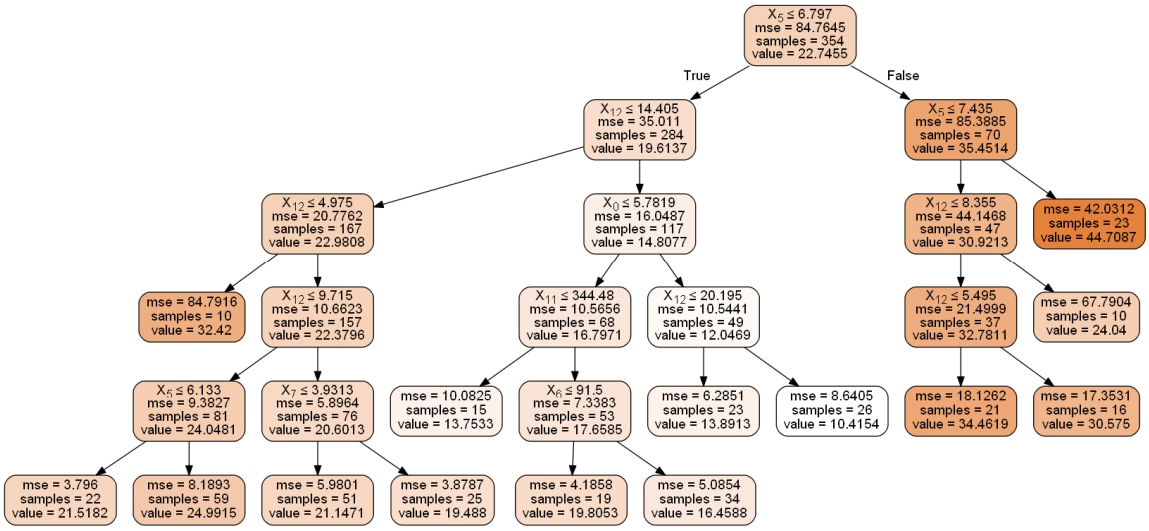
3) Script Python Utilizado

```
# Este exemplo carrega a base Boston e treina uma árvore de regressão usando
# holdout e outra usando validação cruzada com 3 pastas.

from sklearn.datasets import load_boston
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import r2_score
from sklearn import  cross_validation
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 a base Boston
boston = load_boston()
X, y = boston['data'], boston['target']

print("Predição do valor de residências em Boston (Regressão)")
print("Quantidade de instâncias e atributos")
print(X.shape)

# Holdout 70% treinamento e 30% para teste
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=.3, random_state=0)

# Declara a árvore de regressão
regressor = DecisionTreeRegressor(max_depth=5, min_samples_split=30, min_samples_leaf=10,
random_state=0)

# Treina a árvore de regressão
regressor.fit(X_train, y_train)
output_prediction = regressor.predict(X_test)
```

```
# Imprime os resultados do experimento baseado em holdout
print("\nAvaliação utilizando Holdout com 30% para teste")
print("R2 score: %.2f" % r2_score(y_test, output_prediction))

# Validação Cruzuzada com 3 folds
#declara uma nova árvore de regressão
regressor1 = DecisionTreeRegressor(max_depth=5, min_samples_split=30, min_samples_leaf=10,
random_state=0)

# define a quantidade de folds ou pastas e executa a validação cruzada
folds=3
result = cross_validation.cross_val_score(regressor1, X, y, cv=folds)

# Imprime os resultados da validação cruzada
print("\nAvaliação usando Validação Cruzada com %d folds" % folds)
print("R2 scores = " + str(result))
print("R2 score médio: %.2f" % (result.mean()))

# Imprime a árvore de regressão gerada
print("\nÁrvore Gerada no experimento baseado em Holdout")
dot_data = StringIO()
export_graphviz(regressor, 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

4.1) Árvore Gerada (experimento holdout)



Explicaremos o processo de geração (algoritmo indutor) nas próximas aulas. Conforme é possível observar diferente do processo de classificação aqui a árvore tem nas folhas o valor a ser informado ou predito (*value*). Mse é o erro médio quadrático atrelado à predição.

4.2) Avaliação

Resultados baseados em Holdout 70/30
Coeficiente de determinação (R2 score) = 0.73

Resultados baseados em Validação Cruzada (3 folds)
Coeficiente de determinação por fold (R2 score) = [0.70706397 0.43778255 0.32465065]
Coeficiente médio = 0.49

Análise: Considerando os resultados da validação cruzada (normalmente recomendada para problemas com poucas instâncias na base), observamos coeficiente de determinação de 0.73. Isto significa que 73% da variável dependente consegue ser explicada pelo regressor. Este valor vai de 0 a 1, podendo ser negativo. O valor zero significa que o modelo criado não representa adequadamente os dados, e o valor 1 é o ideal. Analogamente, quando um coeficiente próximo a -1 indica uma associação negativa e forte entre duas variáveis.

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.