

Nanodegree Engenheiro de Machine Learning

Modelo de Avaliação e Validação

Projeto 1: Estimando Preços dos Imóveis de Boston

Bem-vindo ao primeiro projeto do Nanodegree de Engenheiro de Machine Learning! Neste Notebook, alguns templates de código estão sendo fornecidos para você, e você irá precisar implementar funcionalidades adicionais para completar este projeto com sucesso. Você não vai precisar modificar o código que foi incluído além do que está sendo pedido. Seções que começam com '**Implementação**' no cabeçalho indicam que o bloco de código seguinte vai exigir que você providencie funcionalidade adicional. Instruções serão fornecidas para cada seção e as especificidades da implementação são marcadas no bloco de código com o comando 'TODO'. Não esqueça de ler as instruções atentamente!

Além do código implementado, haverá questões relacionadas com o projeto e sua implementação que você deve responder. Cada seção em que há uma questão para você responder, ela será precedida por '**Questão X**' no cabeçalho. Leia cada questão cuidadosamente e dê respostas completas no seguinte box de texto que contém '**Resposta:** '. O projeto enviado será avaliado com base nas respostas para cada uma das questões e a implementação que você nos forneceu.

Nota: Células de Código e de Markdown podem ser executadas utilizando o atalho de teclado **Shift + Enter**. Além disso, as células Markdown podem ser editadas ao clicar normalmente duas vezes na célula para entrar no modo de edição.

Começando

Neste projeto, você irá avaliar o desempenho e o poder de estimativa de um modelo que foi treinado e testado em dados coletados dos imóveis dos subúrbios de Boston, Massachusetts. Um modelo preparado para esses dados e visto como *bem ajustado* pode ser então utilizado para certas estimativas sobre um imóvel – em particular, seu valor monetário. Esse modelo seria de grande valor para alguém como um agente imobiliário, que poderia fazer uso dessas informações diariamente.

O conjunto de dados para este projeto se origina do repositório de Machine Learning da UCI (<https://archive.ics.uci.edu/ml/datasets/Housing>). Os dados de imóveis de Boston foram coletados em 1978 e cada uma das 489 entradas representa dados agregados sobre 14 atributos para imóveis de vários subúrbios de Boston. Para o propósito deste projeto, os passos de pré-processamento a seguir foram feitos para esse conjunto de dados:

- 16 observações de dados possuem um valor '**MEDV**' de 50.0. Essas observações provavelmente contêm **valores ausentes ou censurados** e foram removidas.
- 1 observação de dados tem um valor '**RM**' de 8.78. Essa observação pode ser considerada **aberrante** e foi removida.

- Os atributos 'RM', 'LSTAT', 'PTRATIO', and 'MEDV' são essenciais. O resto dos **atributos irrelevantes** foram excluídos.
- O atributo 'MEDV' foi **escalonado multiplicativamente** para considerar 35 anos de inflação de mercado.

Execute a célula de código abaixo para carregar o conjunto dos dados dos imóveis de Boston, além de algumas bibliotecas de Python necessárias para este projeto. Você vai saber que o conjunto de dados carregou com sucesso se o seu tamanho for reportado.

```
In [6]: # Importar as bibliotecas necessárias para este projeto
import numpy as np
import pandas as pd
import visuals as vs # Supplementary code
from sklearn.cross_validation import ShuffleSplit

# Formatação mais bonita para os notebooks
%matplotlib inline

# Executar o conjunto de dados de imóveis de Boston
data = pd.read_csv('housing.csv')
prices = data['MEDV']
features = data.drop('MEDV', axis = 1)

# Êxito
print "O conjunto de dados de imóveis de Boston tem {} pontos com {} variáveis"
```

O conjunto de dados de imóveis de Boston tem 489 pontos com 4 variáveis e m cada.

Explorando os Dados

Na primeira seção deste projeto, você fará uma rápida investigação sobre os dados de imóveis de Boston e fornecerá suas observações. Familiarizar-se com os dados durante o processo de exploração é uma prática fundamental que ajuda você a entender melhor e justificar seus resultados.

Dado que o objetivo principal deste projeto é construir um modelo de trabalho que tem a capacidade de estimar valores dos imóveis, vamos precisar separar os conjuntos de dados em **atributos** e **variável alvo**. O **atributos**, 'RM', 'LSTAT' e 'PTRATIO', nos dão informações quantitativas sobre cada ponto de dado. A **variável alvo**, 'MEDV', será a variável que procuramos estimar. Eles são armazenados em `features` e `prices`, respectivamente.

Implementação: Calcular Estatísticas

Para a sua primeira implementação de código, você vai calcular estatísticas descritivas sobre preços dos imóveis de Boston. Dado que o `numpy` já foi importado para você, use essa biblioteca para executar os cálculos necessários. Essas estatísticas serão extremamente importantes depois para analisar várias estimativas resultantes do modelo construído.

Na célula de código abaixo, você precisará implementar o seguinte:

- Calcular o mínimo, o máximo, a média, a mediana e o desvio padrão do 'MEDV', que está armazenado em `prices`.
 - Armazenar cada cálculo em sua respectiva variável.

```
In [21]: minimum_price = np.min(prices)

maximum_price = np.max(prices)

mean_price = np.mean(prices)

median_price = np.median(prices)

std_price = np.std(prices)

print "Estatísticas para os dados dos imóveis de Boston:\n"
print "Preço mínimo: ${:,.2f}".format(minimum_price)
print "Preço máximo: ${:,.2f}".format(maximum_price)
print "Preço médio: ${:,.2f}".format(mean_price)
print "Preço mediano: ${:,.2f}".format(median_price)
print "Desvio padrão dos preços: ${:,.2f}".format(std_price)
```

Estatísticas para os dados dos imóveis de Boston:

Preço mínimo: \$105,000.00
Preço máximo: \$1,024,800.00
Preço médio: \$454,342.94
Preço mediano: \$438,900.00
Desvio padrão dos preços: \$165,171.13

Questão 1 - Observação de Atributos

Para lembrar, estamos utilizando três atributos do conjunto de dados dos imóveis de Boston: 'RM', 'LSTAT' e 'PTRATIO'. Para cada observação de dados (vizinhança):

- 'RM' é o número médio de quartos entre os imóveis na vizinhança.
- 'LSTAT' é a porcentagem de proprietários na vizinhança considerados de "classe baixa" (proletariado).
- 'PTRATIO' é a razão de estudantes para professores nas escolas de ensino fundamental e médio na vizinhança.

Utilizando sua intuição, para cada um dos atributos acima, você acha que um aumento no seu valor poderia levar a um **aumento** no valor do 'MEDV' ou uma **diminuição** do valor do 'MEDV'? Justifique sua opinião para cada uma das opções.

Dica: Você espera que um imóvel que tem um valor 'RM' de 6 custe mais ou menos que um imóvel com valor 'RM' de 7?

Resposta: Intuitivamente eu diria que se o número de quartos na vizinhança for maior, a faixa econômica da vizinhança também o será. Em certa medida, as features `LSTAT` e `RM` podem possuir algum tipo de relação entre si. Contudo, o modelo fica mais robusto quanto maior o número de features relevantes que puderem ser utilizadas no treinamento.

Desenvolvendo um Modelo

Na segunda seção deste projeto, você vai desenvolver ferramentas e técnicas necessárias para um modelo que faz estimativas. Ser capaz de fazer avaliações precisas do desempenho de cada modelo através do uso dessas ferramentas e técnicas ajuda a reforçar a confiança que você tem em suas estimativas.

Implementação: Definir uma Métrica de Desempenho

É difícil medir a qualidade de um modelo dado sem quantificar seu desempenho durante o treinamento e teste. Isso é geralmente feito utilizando algum tipo de métrica de desempenho, através do cálculo de algum tipo de erro, qualidade de ajuste, ou qualquer outra medida útil. Para este projeto, você irá calcular o coeficiente de determinação (<https://pt.wikipedia.org/wiki/R%C2%B2>), R^2 , para quantificar o desempenho do seu modelo. O coeficiente da determinação para um modelo é uma estatística útil em análise regressa, como se ele frequentemente descrevesse como "good" a capacidade do modelo de fazer estimativas.

Os valores para R^2 têm um alcance de 0 a 1, que captura a porcentagem da correlação ao quadrado entre a estimativa e o valor atual da **variável alvo**. Um modelo R^2 de valor 0 sempre falha ao estimar a variável alvo, enquanto que um modelo R^2 de valor 1, estima perfeitamente a variável alvo. Qualquer valor entre 0 e 1 indica qual a porcentagem da variável alvo, ao utilizar esse modelo, ele pode ser explicado pelos **atributos**. *Um modelo pode dar também um R^2 negativo, que indica que o modelo não é melhor do que aquele que estima ingenuamente a média da variável alvo.*

Para a função 'performance_metric' na célula de código abaixo, você irá precisar implementar o seguinte:

- Utilizar o `r2_score` do `sklearn.metrics` para executar um cálculo de desempenho entre `y_true` e `y_predict`.
- Atribuir a pontuação do desempenho para a variável `score`.

```
In [8]: from sklearn.metrics import r2_score

def performance_metric(y_true, y_predict):
    """ Calcular e retornar a pontuação de desempenho entre
        valores reais e estimados baseado na métrica escolhida. """

    score = r2_score(y_true, y_predict)

    return score
```

Questão 2 - Qualidade do Ajuste

Admita que um conjunto de dados que contém cinco observações de dados e um modelo fez a seguinte estimativa para a variável alvo:

Valores Reais	Estimativa
---------------	------------

Valores Reais	Estimativa
3.0	2.5
-0.5	0.0
2.0	2.1
7.0	7.8
4.2	5.3

Você consideraria que esse modelo foi capaz de capturar a variação da variável alvo com sucesso? Por que ou por que não?

Executar a célula de código abaixo para usar a função `performance_metric` e calcular o coeficiente de determinação desse modelo.

```
In [9]: # Calcular o desempenho deste modelo
score = performance_metric([3, -0.5, 2, 7, 4.2], [2.5, 0.0, 2.1, 7.8, 5.3])
print "O coeficiente de determinação, R^2, do modelo é {:.3f}.".format(score)
```

O coeficiente de determinação, R², do modelo é 0.923.

Resposta: Considerando que o R² calcula a capacidade do modelo de explicar os dados, o valor de 92.3% parece significativo, contudo, um olhar mais atento no conjunto de dados revela que o modelo está superestimando o valor em 80% dos casos. Apesar do modelo explicar mais de 90% dos valores, eu diria que não é um sucesso. Além disso, para afirmar que o modelo é um sucesso ou não, acredito que seja necessário definir que tipo de problemas estamos tratando. Se no problema tratado, esses 7.7% faltantes forem relevantes, talvez seja necessário um modelo mais acurado e neste caso não daria para afirmar que o modelo teve sucesso.

Implementação: Misturar e Separar os Dados

Sua próxima implementação exige que você pegue o conjunto de dados de imóveis de Boston e divida os dados em subconjuntos de treinamento e de teste. Geralmente os dados são também misturados em uma ordem aleatória ao criar os subconjuntos de treinamento e de teste para remover qualquer viés (ou erro sistemático) na ordenação do conjunto de dados.

Para a célula de código abaixo, você vai precisar implementar o seguinte:

- Utilize `train_test_split` do `sklearn.cross_validation` para misturar e dividir os dados de `features` e `prices` em conjuntos de treinamento e teste.
 - Divida os dados em 80% treinamento e 20% teste.
 - Mude o `random_state` do `train_test_split` para um valor de sua escolha. Isso garante resultados consistentes.
- Atribuir a divisão de treinamento e teste para `X_train`, `X_test`, `y_train` e `y_test`.

```
In [10]: from sklearn.cross_validation import train_test_split

X_train, X_test, y_train, y_test = train_test_split(features, price,
                                                    test_size=0.2,
                                                    random_state=0)

# Êxito
print "Separação entre treino e teste feita com êxito."
```

Separação entre treino e teste feita com êxito.

Questão 3 - Treinamento e Teste

Qual o benefício de separar o conjunto de dados em alguma relação de subconjuntos de treinamento e de teste para um algoritmo de aprendizagem?

Dica: O que pode dar errado se não houver uma maneira de testar seu modelo?

Resposta: Separar o conjunto de teste do conjunto de treinamento garante que seu modelo não fica enviesado pelos casos de teste. Usar os casos de teste para treinar o modelo pode acarretar que você tenha uma falsa impressão sobre a capacidade de generalização do seu modelo.

Analisando o Modelo de Desempenho

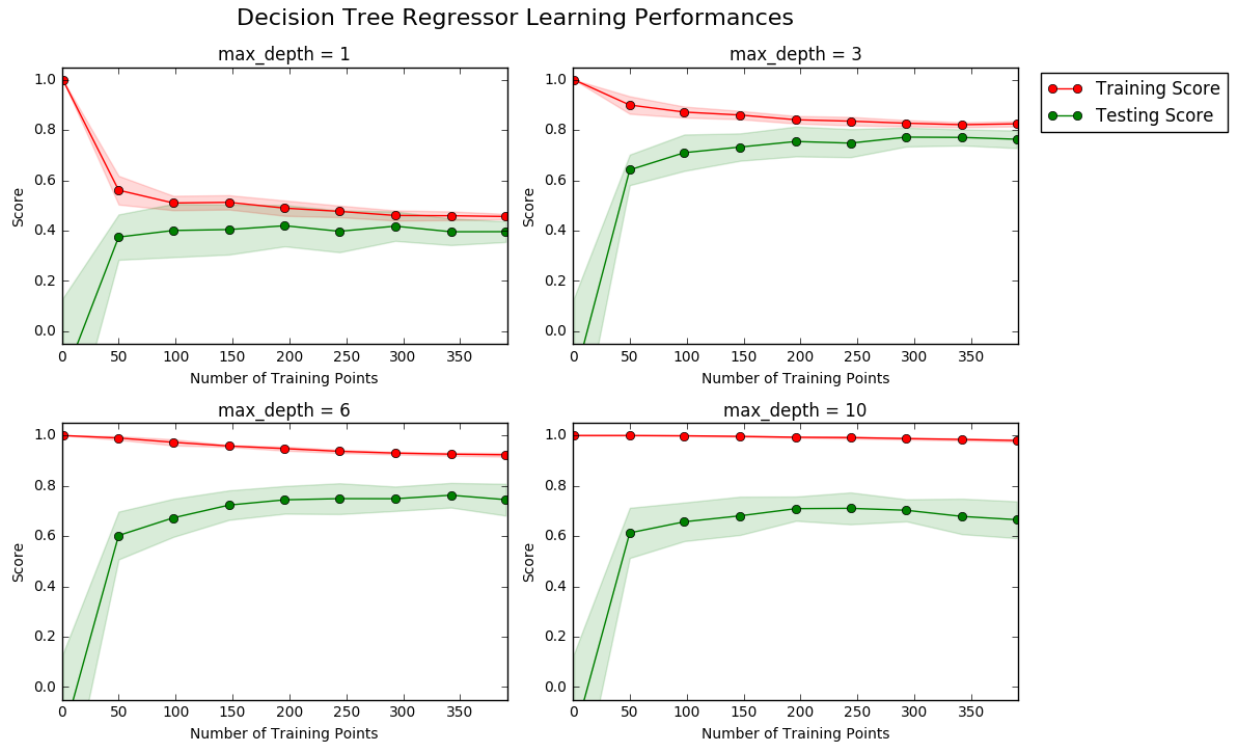
Na terceira parte deste projeto, você verá o desempenho em aprendizagem e teste de vários modelos em diversos subconjuntos de dados de treinamento. Além disso, você irá investigar um algoritmo em particular com um parâmetro `'max_depth'` (profundidade máxima) crescente, em todo o conjunto de treinamento, para observar como a complexidade do modelo afeta o desempenho. Plotar o desempenho do seu modelo baseado em critérios diversos pode ser benéfico no processo de análise, por exemplo: para visualizar algum comportamento que pode não ter sido aparente nos resultados sozinhos.

Curvas de Aprendizagem

A célula de código seguinte produz quatro gráficos para um modelo de árvore de decisão com diferentes níveis de profundidade máxima. Cada gráfico visualiza a curva de aprendizagem do modelo para ambos treinamento e teste, assim que o tamanho do conjunto treinamento aumenta. Note que a região sombreada da curva de aprendizagem denota a incerteza daquela curva (medida como o desvio padrão). O modelo é pontuado em ambos os conjuntos treinamento e teste utilizando R^2 , o coeficiente de determinação.

Execute a célula de código abaixo e utilizar esses gráficos para responder as questões a seguir.

```
In [11]: # Criar curvas de aprendizagem para tamanhos de conjunto de treinamento var.
vs.ModelLearning(features, prices)
```



Questão 4 - Compreendendo os Dados

Escolha um dos gráficos acima e determine a profundidade máxima para o modelo. O que acontece com a pontuação da curva de treinamento se mais pontos de treinamento são adicionados? E o que acontece com a curva de teste? Ter mais pontos de treinamento beneficia o modelo?

Dica: As curvas de aprendizagem convergem para uma pontuação em particular?

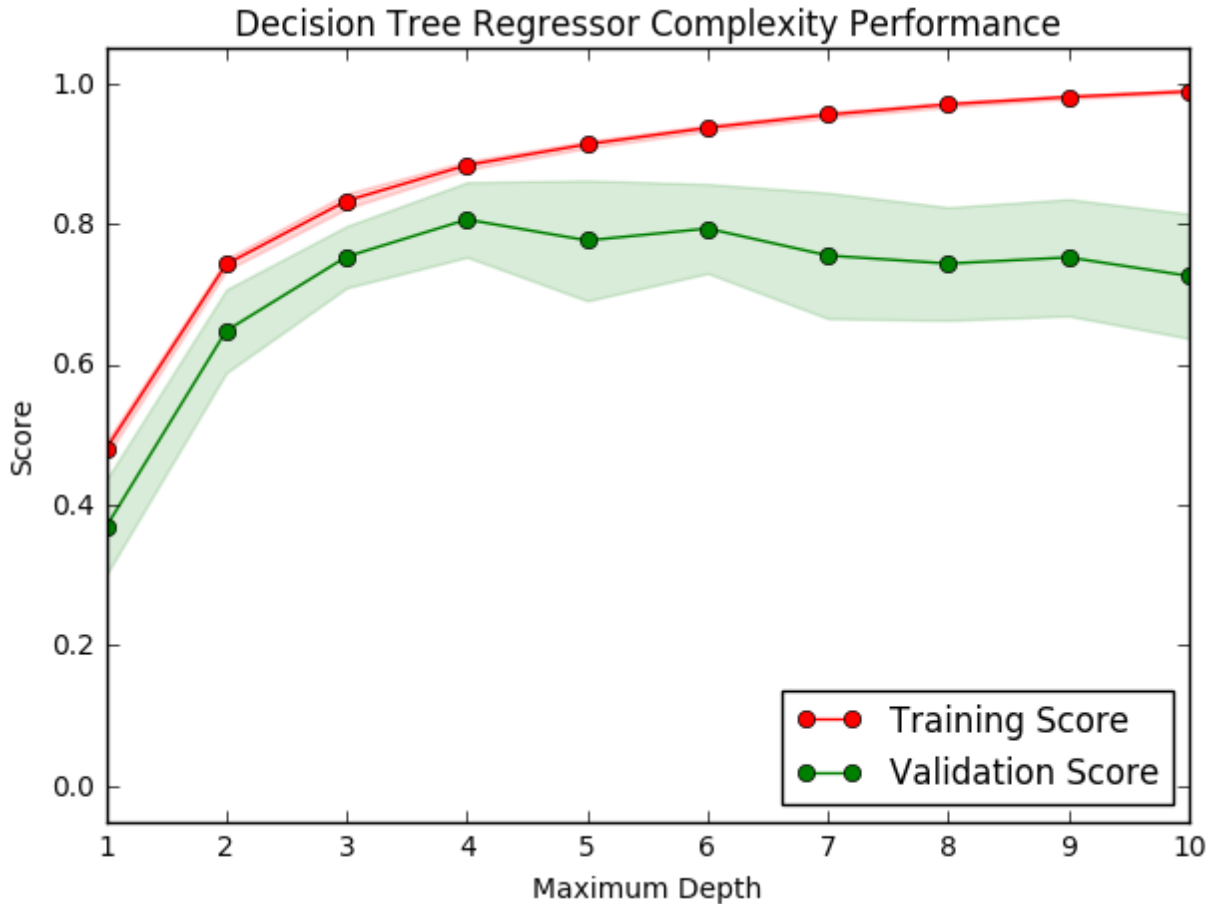
Resposta: A profundidade máxima 3 parece adequada. Apesar da curva de treinamento ser menor que as curvas de 6 e 10 níveis de profundidade, o valor para a curva de teste (de profundidade 3) está mais alto. Adicionar mais profundidade do que 3 aparentemente coloca o modelo em overfitting porque o score da curva de teste fica menor para as profundidades 6 e 10. Neste aspecto, a adição de mais pontos de treinamento não beneficia o modelo, porque a capacidade de aprendizagem do modelo está reduzindo. Se essa capacidade está reduzindo, os novos pontos adicionados podem não agregar valor nenhum ao modelo.

Curvas de Complexidade

A célula de código a seguir produz um gráfico para um modelo de árvore de decisão que foi treinada e validada nos dados de treinamento utilizando profundidades máximas diferentes. O gráfico produz duas curvas de complexidade – uma para o treinamento e uma para a validação. Como a **curva de aprendizagem**, a área sombreada de ambas as curvas de complexidade denota uma incerteza nessas curvas, e o modelo pontuou em ambos os conjuntos de treinamento e validação utilizando a função `performance_metric`.

Execute a célula de código abaixo e utilize o gráfico para responder as duas questões a seguir.

```
In [22]: vs.ModelComplexity(X_train, y_train)
```



Questão 5 - Equilíbrio entre viés e variância

Quando o modelo é treinado com o profundidade máxima 1, será que o modelo sofre mais de viés (erro sistemático) ou variância (erro aleatório)? E o que acontece quando o modelo é treinado com profundidade máxima 10? Quais pistas visuais existem no gráfico para justificar suas conclusões?

Dica: Como você sabe que um modelo está experimentando viés alto ou variância alta?

Resposta: Na profundidade 1 o modelo sofre mais com viés (underfitting) porque ele não possui complexidade suficiente pra gerar uma generalização do conjunto de dados. De acordo com que se aproxima da profundidade 10 ele passa a sofrer de variância (overfitting) porque o modelo fica muito específico para a amostra de treinamento. As pistas gráficas são as comparações entre a curva de treinamento e a curva de validação. Na medida que aumentamos a complexidade, observamos que os scores de treinamento e validação crescem juntos até próximo da árvore de profundidade 4. Após isso, apesar do score de treinamento continuar subindo, o score de validação começa a cair.

Questão 6 - Modelo Ótimo de Melhor Suposição

Qual profundidade máxima ('max_depth') você acredita que resulta em um modelo que melhor generaliza um dado desconhecido? Que intuição te levou a essa resposta?

Resposta: Profundidade 4. Porque é a profundidade que representa o ápice dos scores de treinamento e validação. A partir desse profundidade, acredito que o modelo começa a sofrer de overfitting porque apesar de aumentar o score de treinamento, o modelo começa a perder score de validação.

Avaliando o Desempenho do Modelo

Nesta parte final do projeto, você irá construir um modelo e fazer uma estimativa de acordo com o conjunto de atributos do cliente utilizando um modelo otimizado a partir de `fit_model`.

Questão 7 - Busca em Matriz

O que é a técnica de busca em matriz (grid search) e como ela pode ser aplicada para otimizar um algoritmo de aprendizagem?

Resposta: A técnica de busca em matriz é a construção de uma matriz com parametros de diferentes valores para avaliar o desempenho do modelo com cada um desses parâmetros. Ela pode ser utilizada para fazer o ajuste fino de um algoritmo de aprendizagem.

Questão 8 - Validação Cruzada

O que é a técnica de treinamento de validação-cruzada k-fold? Quais benefícios essa técnica proporciona para busca em matriz ao otimizar um modelo?

Dica: Assim como há um raciocínio por trás de utilizar um conjunto de teste, o que poderia dar errado ao utilizar busca em matriz sem um conjunto de validação cruzada?

Resposta: A técnica de validação cruzada particiona os as instancias de teste em grupos menores que podem ser utilizados para amenizar o viés das instancias de teste. Esse tipo de particionamento auxilia durante a busca em matriz porque ameniza o viés dos modelos.

Implementação: Ajustar um Modelo

Na sua última implementação, você vai precisar unir tudo o que foi aprendido e treinar um modelo utilizando o **algoritmo de árvore de decisão**. Para garantir que você está produzindo um modelo otimizado, você treinará o modelo utilizando busca em matriz para otimizar o parâmetro de profundidade máxima (`'max_depth'`) para uma árvore de decisão. Esse parâmetro pode ser entendido como o número de perguntas que o algoritmo de árvore de decisão pode fazer sobre os dados antes de fazer uma estimativa. Árvores de decisão são parte de uma classe de algoritmos chamados *algoritmos de aprendizagem supervisionada*.

Para a função `fit_model` na célula de código abaixo, você vai precisar implementar o seguinte:

- Utilize o `DecisionTreeRegressor` (<http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html>) do

- sklearn.tree para gerar um objeto regressor de árvore de decisão.
 - Atribua esse objeto à variável 'regressor'.
- Gere um dicionário para 'max_depth' com os valores de 1 a 10 e atribua isso para a variável 'params'.
- Utilize o `make_scorer` (http://scikit-learn.org/stable/modules/generated/sklearn.metrics.make_scorer.html) do `sklearn.metrics` para gerar um objeto de função de pontuação.
 - Passe a função `performance_metric` como um parâmetro para esse objeto.
 - Atribua a função de pontuação à variável 'scoring_fnc'.
- Utilize o `GridSearchCV` (http://scikit-learn.org/stable/modules/generated/sklearn.grid_search.GridSearchCV.html) do `sklearn.grid_search` para gerar um objeto de busca por matriz.
 - Passe as variáveis 'regressor', 'params', 'scoring_fnc' and 'cv_sets' como parâmetros para o objeto.
 - Atribua o objeto `GridSearchCV` para a variável 'grid'.

```
In [18]: from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import make_scorer
from sklearn.grid_search import GridSearchCV

def fit_model(X, y):
    """ Desempenhar busca em matriz sobre o parâmetro the 'max_depth' para u
        árvore de decisão de regressão treinada nos dados de entrada [X, y].

    # Gerar conjuntos de validação-cruzada para o treinamento de dados
    cv_sets = ShuffleSplit(X.shape[0], n_iter = 10, test_size = 0.20, random

    regressor = DecisionTreeRegressor(random_state=0)

    max_depths = [x for x in range(1,10)]
    params = {'max_depth': max_depths}

    # TODO: Transformar 'performance_metric' em uma função de pontuação utili
    scoring_fnc = make_scorer(performance_metric)

    # TODO: Gerar o objeto de busca em matriz
    grid = GridSearchCV(regressor, params, scoring=scoring_fnc)

    # Ajustar o objeto de busca em matriz com os dados para calcular o model
    grid = grid.fit(X, y)

    # Devolver o modelo ótimo depois de realizar o ajuste dos dados
    return grid.best_estimator_
```

Fazendo Estimativas

Uma vez que o modelo foi treinado em conjunto de dados atribuído, ele agora pode ser utilizado para fazer estimativas em novos conjuntos de entrada de dados. No caso do *regressor da árvore de decisão*, o modelo aprendeu *quais são as melhores perguntas sobre a entrada de dados*, e pode

responder com uma estimativa para a **variável alvo**. Você pode utilizar essas estimativas para conseguir informações sobre os dados dos quais o valor da variável alvo é desconhecida – por exemplo, os dados dos quais o modelo não foi treinado.

Questão 9 - Modelo Ótimo

_Qual profundidade máxima do modelo ótimo? Como esse resultado se compara com a sua suposição na **Questão 6**?

Executar a célula de código abaixo para ajustar o regressor da árvore de decisão com os dados de treinamento e gerar um modelo ótimo.

```
In [19]: # Ajustar os dados de treinamento para o modelo utilizando busca em matriz
reg = fit_model(X_train, y_train)

# Produzir valores para 'max_depth'
print "O parâmetro 'max_depth' é {} para o modelo ótimo.".format(reg.get_pai

O parâmetro 'max_depth' é 4 para o modelo ótimo.
```

Resposta: O parâmetro 'max_depth' é 4 para o modelo ótimo. Na questão 6 só haviam as opções de 3 ou 6 níveis de profundidade. O modelo ótimo vai de encontro com as análises realizadas a partir das curvas de aprendizagem que apontavam algo próximo da profundidade 3 como sendo o ápice do aprendizado/teste do modelo.

Questão 10 -Estimando Preços de Venda

Imagine que você era um corretor imobiliário na região de Boston ansioso para utilizar esse modelo que ajuda os imóveis que seus clientes desejam vender. Você coletou as seguintes informações de três dos seus clientes:

Atributos	Cliente 1	Cliente 2	Cliente 3
Número total de quartos em um imóvel	5 quartos	4 quartos	8 quartos
Nível de pobreza da vizinhança (em %)	17%	32%	3%
Razão estudante:professor das escolas próximas	15-to-1	22-to-1	12-to-1

Qual valor você sugeriria para cada um dos seus clientes para a venda de suas casas? Esses preços parecem razoáveis dados os valores para cada atributo?

Dica: Utilize as estatísticas que você calculou na seção **Explorando Dados** para ajudar a justificar sua resposta.

Execute a célula de códigos abaixo para que seu modelo otimizado faça estimativas para o imóvel de cada um dos clientes.

```
In [20]: # Gerar uma matriz para os dados do cliente
client_data = [[5, 17, 15], # Cliente 1
               [4, 32, 22], # Cliente 2
               [8, 3, 12]] # Cliente 3

# Mostrar estimativas
for i, price in enumerate(reg.predict(client_data)):
    print "Preço estimado para a casa do cliente {}: ${:,.2f}".format(i+1, price)
```

```
Preço estimado para a casa do cliente 1: $393,093.75
Preço estimado para a casa do cliente 2: $230,865.96
Preço estimado para a casa do cliente 3: $1,021,650.00
```

Resposta:

```
Preço estimado para a casa do cliente 1: `393,093.75`
Preço estimado para a casa do cliente 2: `230,865.96`
Preço estimado para a casa do cliente 3: `1,021,650.00`
```

Estatística descritiva do dataset:

```
Preço mínimo: 105,000.00
Preço máximo: 1,024,800.00
Preço médio: 454,342.94
Preço mediano: 438,900.00
Desvio padrão dos preços: 165,171.13
```

Considerando os valores da estatística descritiva para o dataset, os valores previstos pelo modelo parecem ser consistentes com a média de valores imobiliários a época do dataset uma vez que eles estão dentro dos limites de valores mínimo e máximo.

Sensibilidade

Um modelo ótimo não é necessariamente um modelo robusto. Às vezes, um modelo é muito complexo ou muito simples para generalizar os novos dados. Às vezes, o modelo pode utilizar um algoritmo de aprendizagem que não é apropriado para a estrutura de dados especificado. Outras vezes, os próprios dados podem ter informação excessiva ou exemplos insuficientes para permitir que o modelo apreenda a variável alvo – ou seja, o modelo não pode ser ajustado. Execute a célula de código abaixo para rodar a função `fit_model` dez vezes com diferentes conjuntos de treinamento e teste para ver como as estimativas para um cliente específico mudam se os dados foram treinados.

```
In [22]: vs.PredictTrials(features, prices, fit_model, client_data)
```

```
Trial 1: $411,000.00  
Trial 2: $411,417.39  
Trial 3: $415,800.00  
Trial 4: $428,316.00  
Trial 5: $413,334.78  
Trial 6: $411,931.58  
Trial 7: $399,663.16  
Trial 8: $407,232.00  
Trial 9: $402,531.82  
Trial 10: $413,700.00
```

Range in prices: \$28,652.84

Questão 11 - Aplicabilidade

Em poucas linhas, argumente se o modelo construído deve ou não ser utilizado de acordo com as configurações do mundo real.

Dica: Algumas questões para responder:

- *Quão relevante dados coletados em 1978 podem ser nos dias de hoje?*
- *Os atributos presentes são suficientes para descrever um imóvel?*
- *Esse modelo é robusto o suficiente para fazer estimativas consistentes?*
- *Dados coletados em uma cidade urbana como Boston podem ser aplicados para uma cidade rural?*

Resposta: - Após a bolha imobiliária e a crise de 2008 o mercado imobiliário mudou bastante. Neste sentido, os dados estão desatualizados e não poderiam ser utilizados para uma estimativa nos dias de hoje. Além disso, a estrutura socio-familiar mudou bastante e não se vê muitos casais com vários filhos para justificar casas de mais de 3 quartos, por exemplo.

Temos ainda que o modelo poderia conter informações como distância do centro da cidade, existencia de estações de metrô nas proximidades e taxa de desemprego da região poderiam ser de grande ajuda para construir um modelo mais robusto e preciso. Ainda assim, mesmo construindo um modelo que se adeque á realidade de uma cidade, é preciso observar que o modelo pode diferir de uma cidade para outra. Como uma cidade urbana é bem diferente de uma cidade rural, é preciso que o modelo seja treinado com dados que façam sentido para a realidade da cidade alvo das estimativas. Mesmo entre as cidades grandes é possível que existam diferenças entre os modelos devido á cultura e características sociais dos cidadãos daquela localidade.

```
In [ ]:
```

```
In [ ]:
```

