01

Conhecendo DecisionTreeRegressor

Transcrição

Segue o link do arquivo movies_multilinear_reg.csv (https://s3.amazonaws.com/caelum-online-public/machine-learning-aprendizado-supervisionado/movies_multilinear_reg.csv).

[00:00] Uma coisa que nós vimos vendo nesse curso de Machine Learning é que não existe uma bala de prata ou um algoritmo canônico que nós vamos utilizar pra resolver o nosso problema, pra um problema A eu vou usar o eixo específico Y, por exemplo, pra um caso B eu vou usar outro algoritmo já pra resolver esse problema.

[00:25] Dependendo do domínio que nós estamos aplicando, dependendo dos nossos problemas, dependendo da distribuição dos nossos dados, um algoritmo específico pode resolver o problema de uma forma mais inteligente.

[00:36] Por exemplo, ainda usando esse exemplo, nós vimos em outro curso aqui da Alura que nós podemos usar Naive Bayes pra fazer classificação. Só que nós aprendemos na aula passada que eu posso fazer esse mesmo trabalho de classificação usando regressão logística. E eu tento encontrar a reta, a curva ideal pro meu problema, pra classificar corretamente os dados, caso tudo esteja de acordo com os conformes.

[01:06] E nessa aula de hoje aqui da Alura, de agora, nós vamos aprender outro algoritmo de aprendizado supervisionado, ou seja, tem a variável resposta e a variável independente, e eu tento generalizar isso pra quando eu receber uma variável independente, mas não souber a variável resposta. E esse algoritmo é usado tanto para classificação quanto para regressão.

[01:30] Só que antes de eu chegar e já tacar algoritmo e tudo mais, vamos entender um pouquinho da intuição por trás dele, qual é a ideia por trás desse algoritmo. Então vamos começar.

[01:43] Vamos relembrar, inicialmente falando de regressão, ao longo das nossas aulas lá de regressão, que eu falei no começo do curso. Indo aqui, nós vamos ter, por exemplo, no caso da regressão múltipla, nós temos nossos filmes, uma série de características dos filmes, a duração e o investimento são todas as variáveis independentes e eu tenho a bilheteria. Outro caso eu tenho simplesmente nome do filme, o investimento e a bilheteria, variável resposta em cima de cada uma dessas variáveis independentes.

[02:18] Só que e se aqui, ao invés de nós tentarmos encontrar uma reta, uma curva ou às vezes basear numa contagem e tudo mais, se nós baseássemos os nossos dados em uma série de regras de decisão? Então vamos pensar – ainda nessa ideia, pra eu explicar um pouco melhor esse caso – que nossos dados são distribuídos de uma forma um pouquinho diferente do que nós vimos lá no começo, eles estão distribuídos mais ou menos dessa forma.

[02:45] Aqui eu posso assumir que se a bilheteria for maior do que 0, maior positivo, a bilheteria positiva, o meu público gostou, ele entende bastante, mas é um valor que eu estou querendo prever. E se for abaixo de 0, o público não gostou, não vou ter muito sucesso. E meu investimento deu uma forma crescente, nós podemos ver que os dados se comportam um pouco diferente da primeira vez, quanto menor o meu investimento, eu tenho uma aceitação mais aceitável e quanto menor menos aceitável.

[03:11] E quando eu falo a regra de decisão, é se nós escolhêssemos pontos. Pontos que fossem dividindo os meus dados pra agrupar os meus dados da maneira mais inteligente possível. Então num primeiro momento eu vou olhar se eu tenho investimento, mas qual ponto do investimento que eu cruzo? Eu posso começar assim, então esses dados estão divididos em duas formas. Aqui são os dados que têm aquela ascensão, chegando quase no pico, aqui é o pico, e caindo também, mas pelo visto essa divisão também não está tão boa.

[03:43] E eu posso dividir então numa segunda regrinha aqui, que é usado numa área intermediária. Aqui são dados mais crescentes, aqui é mais o decrescente e aqui são os dados mais inferiores, propriamente ditos, parece até a bandeira da França as cores que eu acabei escolhendo.

[04:01] O que nós podemos ver? Eu tenho lá meu investimento e eu já tenho pontos específicos que eu posso fazer a divisão dos meus dados. Indo pra uns dados um pouco mais reais e se nós colocássemos regra de decisão mais ou menos assim? Eu tenho aqui meu investimento, se meu investimento for menor do que 11 milhões, a bilheteria esperada é essa. Se for maior do que 11 milhões, o valor que eu tenho é esse.

[04:28] Isso seria um caso simplesmente pra primeira situação, pra primeira divisão aqui, menor do que 11 milhões, 4 milhões, maior do que 11 milhões, 16 milhões. E você fala: "talvez eu queira dividir um pouquinho melhor, de uma forma um pouquinho mais inteligente", então você bota aqui o segundo caso. Ele está entre 11 milhões e 15 milhões? Então é 12 milhões a minha bilheteria esperada. E outros 16 milhões.

[04:47] Lembra que nós estamos falando de regressão, então a regressão justamente nesse caso pode ser definida como a média de todos os pontos que caíram nessa área. E eu tenho as minhas três divisões separadas nisso.

[05:03] Nós podemos ver, é que esses dados formam uma estrutura muito parecida com a de uma árvore. Aqui você tem uma árvore dentro do conceito de computação, então aqui eu tenho o desenho de uma árvore, aqui são os nós da minha árvore e aqui a ramificação que eu tenho d os meus nós. A árvore, regra de decisão, isso é o que nós chamamos de árvore de decisão.

[05:34] Qual é o ponto? Nós podemos olhar aqui e falar: "André, mas agora, beleza, eu entendi nesse caso, mas e pra um dado mais real? E pra um dado mais ou menos como o que eu tenho aqui, como que eu decido o investimento ideal?". Por exemplo, uma regra de decisão que eu posso ter é encontrar aquele split ideal que o valor que eu encontro aqui seja o menor e o quadrático em comparação com os dados que se encontram nessa área, no caso.

[06:04] Por exemplo, aqui são meus 4 milhões, eu tenho o menor erro quadrático, ou seja, o valor previsto em comparação com o valor obtido no fim das contas, com o valor previsto e o valor real. É um exemplo.

[06:18] Mas nós não precisamos nos preocupar em definir qual é a melhor regra de decisão, a regra de decisão inteligente, a não ser que nós queiramos pesquisar esse tipo de coisa ou estudar matemática com um pouco mais de nuância. Mas por que eu falo isso? Porque a ideia é que o scikit-learn consiga fazer isso pra nós. Na verdade ele não só consegue, ele faz isso para nós. Como que nós fazemos isso no scikit-learn?

[06:41] Num primeiro momento nós vamos sair daqui, nós vamos pra cá. Quais são as bibliotecas que eu vou precisar pra fazer isso daqui funcionar? As bibliotecas que nós estamos mais acostumados. Eu vou precisar do meu Pandas, vou mandar um zoom aqui pra nós podermos enxergar. Vou precisar do Matplotlib, "matplotlib.pyplot". O NumPy. O SKLEARN, nós vamos precisar da função tree, que é o que nós queremos fazer, e a divisão dos nossos dados em treino e teste, que nós já estamos acostumados, "sklearn.model_selection import train_test_split".

[07:38] Vamos ver se funciona? Logo no começo nós vamos aqui, vai abrir o terminal. Nós entramos aqui, vamos entrar no meu curso, André, ok, cd curso, ok. E aqui nós já podemos entrar no nosso querido Python. Entramos no Python. Vamos importar as bibliotecas, importamos todas as bibliotecas e vamos ver se vai tudo funcionar em ordem.

[08:12] Aqui já está importando a biblioteca pra nós ganharmos um pouquinho de tempo. Eu escrevi alguma coisa errada. Aqui é "from sklearn", from sklearn.linearmodel, "import numpy as np" aqui também, só pra nós não perdermos o costume. Importando as bibliotecas que faltam. Importamos todas elas aqui.

[08:36] Num primeiro momento o que é? É o famoso leitura de csv que nós já estamos acostumados. Pra esse exemplo, nós já trabalhamos ali com a manipulação dos dados numa regressão linear simples, eu vou direto para a regressão

linear múltipla. Então vou ter aqui meu “filmes”, ele vai receber um “pd.read_csv”. Nós botamos Pandas as pd? Não, “panda as pd” pra nós não termos problema. Eu entro na minha pasta “datasets/moves_multilinear_reg.csv”.

[09:26] E eu vou ter lá meu “filmes_caract”, ele recebe o “filmes”, que vai ser o data frame que eu estou criando, só que eu quero as colunas 2 a 17, que nós lembramos que são as colunas que vão ter todas as minhas variáveis dependentes.

[09:42] A primeira coluna, se nós formos ver aqui, é o meu movie ID, eu não estou interessado nisso, na verdade essa é a coluna 0, a 1 é o nome do filme, nós não estamos interessados nisso. 2 até a 17. Então viemos aqui, vamos importar tudo, na verdade, de novo, pra ver se funciona tudo. “filmes_caract” são os dados que nós queremos. Agora nós temos o “filmes_labels”, que é a variável resposta pra nós.

[10:15] É mais ou menos a mesma ideia do que nós acabamos de fazer, “filmes.columns” apenas da 17 até o final, que vai ser na prática basicamente 17. “filmes_labels”.

[10:42] Agora eu quero dividir meus dados em “treino”, “teste”, “treino_labels”,”teste_labels”, ele vai receber “train_test_spli”, que é nossa divisão que nós estamos acostumados a fazer, “filmes_caract” e nosso “filmes_labels”. Nós vamos aqui, nossos dados de treino. Se eu não me engano, ali, nós já temos agora 6800 caras, então tem “treino.size” e se eu jogar aqui “len(treino)” exatamente len treino que é o que nós queremos, sobre “len(filmes_caract), ali 75% de dados estimados para teste.

[11:44] Pra nós não perdermos nosso costume, aqui nós já fazemos também o reshape desses dados, pra já facilitar na hora que nós formos fitar os dados, então nós já damos um “np.array”, pra transformar tudo numa estrutura de array do numpy, nós passamos o próprio “(treino).reshape”, vou mostrar um pouquinho mais visível. O treino recebe o reshape, os dados de treino e aqui o número de atributos. Por que nós fazemos isso? Eu já vou mostrar.

[12:19] Se nós digitarmos aqui “type(treino)” nós temos aqui um dataframe fazendo isso agora, o que vai acontecer? type treino, eu mudo isso aqui para um array do numpy. Facilidade também na hora de tratar. Então a mesma coisa que nós vamos fazer é fazer com teste isso daqui, “np.array”, ele vai receber “(teste).reshape(len(teste),15”.

[12:50] A mesma coisa aqui, “len(teste)”, tem que dar aquele famoso 25%, caract, então eu estou vindo aqui, o len de teste tem que dar 25%. Nós temos aqui 100% dos dados, 25% separado pro teste e 75% separado pra treino.

[13:15] E nós precisamos basicamente também, pra uma questão de padronização, fazer a mesma coisa com as nossas labels. Eu vou ter aqui um “treino_label”, ele vai receber a mesma coisa, “np.array”, aqui eu já vou fazer os dois de uma vez, “(treino_labels).reshape(len(treino_labels), 1)”, porque é só 1 que nós temos e “teste_labels” recebe “np.array(teste_labels)”. Aqui é reshape, então aqui está errado, “(teste_labels).reshape(len(teste_labels),1).

[14:18] Agora vamos ver se rodou? Copio esses dois amiguinhos aqui, dou um “Ctrl + L”, “Command + V”, copiei e colei, felicidade. Agora que eu tenho nossos dados de treino e nossos dados de teste, o que nós precisamos fazer é criar a regressão, a nossa árvore de regressão pra fazer as previsões da forma como nós queremos.

[14:36] Eu vou gerar o meu modelo, meu modelo vai ser um “tree.DecisionTreeRegressor”. E uma vez que eu defini o meu modelo, eu só preciso fazer o “modelo. fit” em cima dos meus dados de treino. Então eu tenho essa opção de passar meu treino e meu treino labels. Uma vez feitos esses dados, depois nós só precisamos implementar o nosso score. É só fazer ali o “modelo.score(treino, treino_labels)” e “modelo.score(teste, teste_labels)”.

[15:30] Vamos ver como é que fica? Vamos copiar tudo, o ideal é ser sempre a mesma coisa, copiamos, dei aqueles famosos enters a mais. Deu aproximadamente 60%, 61% no nosso caso de teste e 100% para os nossos dados de treino. Vamos comparar com o que nós já fizemos antes, com o modelo de regressão linear que nós já fizemos antes.

[15:55] Como que nós fazemos isso? Nós vamos aqui “sklearn.linear_model import LinearRegressor”. Nós chamamos aqui “modelo_reg”, pode ficar assim, recebe “LinearRegressor”, ele vai receber o “modelo_reg.fit”, meus dados de treino

e treino labels. E nós já vamos fazer o nosso score. “modelo_reg.score(treino, treino_labels)”. Ok. E aqui a mesma coisa para o teste e teste labels.

[16:51] E nós simplesmente vamos aqui, copiamos, damos um copiar, vamos colar e deu erro por modelo reg não está definido. Linear Regression, não é Linear Regressor. Vamos só copiar aqui, “import LinearRegression”, eu confundi com o regressor da árvore. Regression. Vou ver se eu fiz certo. Fiz certo. Agora então se eu copiar esses dois caras. Deu 83% nos nossos dados de treino e 81% nos nossos dados de teste.

[17:37] O que nós conseguimos ver aqui é que o modelo, no caso da regressão linear, ali ela foi melhor do que nossa árvore, mas será que é assim? Será que de fato esse modelo é o pronto e será que nós conseguimos melhorar essa árvore, de alguma forma, pra ela ficar pelo menos melhor ou parecida com o nosso caso da regressão linear?

[17:58] E outra coisa, se nós voltarmos aqui, vou até rodar de novo esses dados, vou rodar mais uma vez pra nós enxergarmos um pouco melhor. Por que nós conseguimos 100% no caso de treino e 63% no caso de teste? Por que nós conseguimos essa discrepância tão grande, quando comparada com a da regressão linear nós tivemos 82% pro caso de treino e 82% também, aqui é 83% mais ou menos no caso de treino e 82% no caso de teste.

[18:32] Nós vamos entender um pouco mais a fundo como que a árvore de decisão funciona, um pouco mais das nuances por trás dela e também, nesse caso nós vimos um caso dela para a regressão, nós também vamos ver como usar a árvore de decisão para classificação.