02

Árvores Classificadores

Transcrição

Seguem os dados do filme Zootopia:

```
movieId,Titulo,Documentary,Sci-Fi,Mystery,Horror,Romance,Thriller,Crime,Fantasy,Comedy,Animatio
9999999,Zootopia,0,0,0,0,0,0,1,1,1,1,0,1,110,27.74456356,?????
```

[00:00] Nós já entendemos o processo de criação dessa árvore. Se nós voltarmos aqui, vou copiar todos os dados que nós escrevemos, estou dando um “Command + C” em tudo o que nós já fizemos, só pra nós darmos uma revisada. Estou copiando, estou colando, está importando tudo, deu até vários enters.

[00:19] E agora então, se nós chegarmos aqui, vamos voltar lá pra onde nós queremos, dei um “Command + L”, eu vou aqui, modelo.score os dados de treino, modelo.score dados de teste. O que nós podemos tirar de insumo nesse primeiro momento? É que a regra de decisão que a nossa árvore vai fazendo se baseia com como ela divide da melhor forma possível os nossos dados de treino.

[00:47] E conforme mais complexos são os nossos dados, mais complexas são essas regras de decisão, esses detalhes que a árvore vai se emaranhando, ela começa a fazer divisões, obviamente, não lineares, não bem obviamente, mas a árvore de decisão funciona bem com regras não lineares, porque quanto mais não linear, ela tende – com base nessas regras – a pegar essas decisões, essas nuâncias que os nossos dados acabam tendo.

[01:20] E o que acaba acontecendo como consequência disso? Como ela se preocupa em aprender bem com base nos nossos dados de treino, ela acaba pegando regras muito específicas da divisão desses nós.

[01:33] E quando eu falo da divisão desses nós, se nós voltarmos nos nossos antigos slides, é como se fosse, aqui é uma regra de divisão, aqui é outra regra de divisão, então quando ela está com dados muito complexos, ela pode falar, se o investimento for menor do que 4 milhões e o filme for de romance, mas não for de terror, então o investimento foi 2 milhões 327 mil.

[02:00] São detalhes muito pequenos e ela aprende muito bem para os dados de teste, ela overfita em cima, fita de aprender em cima, ela se encaixa e ela superencaixa, overfitting, em cima nesses dados de treinamento. E quando nós vamos pro nosso caso de teste, que é pra nós generalizarmos, nós acabamos tendo uma diferença muito grande, o que não aconteceu quando nós já vimos no caso da regressão linear, que eu fiz treino e teste, 82, 82.

[02:28] O que nós podemos fazer como medida disso? Como medida pra nós combatermos isso? Nós podemos setar um número máximo de profundidade que eu quero, que meus nós vão, e isso acontece quando eu seto o modelo aqui, eu estou subindo aqui, “max_depth”, que no caso, vamos supor que seja 5, eu faço meu fit, olha o que vai acontecer agora.

[02:52] Eu fiz o meu fit, vou dar um “Command + L”, eu tenho um modelo.score treino, caiu pra mais ou menos 80%. E o modelo.score de teste deu 78%. Já ficou um pouco mais próximo, inclusive um do outro, quanto do valor final da nossa regressão.

[03:20] Vamos ver como seria, por exemplo, no caso, pra nós prevermos o nosso Zootopia. Então nesse caso nós conseguimos esse ponto um pouco mais ideal. É só fazer ali o modelo.predict pegando os dados do Zootopia. E se eu não me engano, eu já tenho esses dados separados aqui.

[03:37] Vou pegar o Zootopia completo, que são todos esses carinhas, estou copiando todos esses dados, eu vou jogar ali no Atom, vou chamar esse dado de Zootopia, desci tudo, então nós podemos chamar esse "zootopia", ele vai ser uma lista, coloquei esses dados na lista e eu só preciso trocar, "Command + D", apaguei, vírgula, espaço, essa é a nossa lista de Zootopia. "Command + C" pra copiar, dei um copy, aqui colei, Zootopia, e "modelo.predict(zootopia)".

[04:28] Esse erro, deprecation warning, é porque ele pede uma lista, no caso, então nós só precisamos fazer assim, não é uma lista única, esse seria o nosso valor da nossa bilheteria final.

[04:48] Agora nós vimos que os dados se encaixaram um pouco melhor. Os nossos dados de treino tanto dados de teste com relação a R quadrado, que é esse score, eles se aproximaram um pouco melhor um do outro, você não tem o overfitting e eles se aproximaram, inclusive, melhor do caso de regressão múltipla que nós fizemos. Não foi o melhor, mas nós já vamos entender também por que isso acontece.

[05:16] Nós vimos um caso de regressão, como que nós vemos um caso de classificação? Vamos voltar pro nosso exemplo. É muito parecido, o processo de criação da árvore é basicamente o mesmo, nós vamos ter regra de decisão que são pontos que nós fazemos essa divisão, o processo de construção da árvore é muito parecido. Então é como se nós estivéssemos aqui, aqui é a parte vermelha, não gostou; aqui é a parte verde, indiferente; aqui é a parte azul, gostou.

[05:39] E como é que nós pegamos esse critério de divisão? A diferença, na verdade, pra árvore de regressão da de classificação, é o critério que nós usamos aqui. Por exemplo, um deles é o índice Gini, que eu não vou entrar muito nos detalhes de como ele funciona e tudo mais, porque novamente, assim como no caso da regressão, o scikit-learn já fez isso pra nós. Então vamos um pouquinho código de novo?

[06:05] Qual é a diferença para a nossa situação? Aqui eu vou, só pra nós não perdermos, nós temos aqui Zootopia. E agora nós não queremos mais esse cara, nós vamos fazer exatamente todas essas mesmas.

[06:24] Pra facilitar nós vamos fazer exatamente o mesmo trabalho que nós fizemos quando nós mexemos com regressão logística, então nós temos também o gostos.py, e eu simplesmente quero ler esse cara, vou dar um copy, vamos chamar de "gostos" mesmo. Voltamos aqui, "Command + V" pra colar, voltamos, Ctrl pra cima. Separamos as características e as labels, ok. Fiz a divisão de treino e teste, split, ok. Estamos voltando para cá, Ctrl pra cima pra voltar, ida e volta. Características aqui, labels aqui.

[07:08] Vamos copiar esse cara pra ver como que eles ficaram no nosso terminal? Copiamos todo mundo, "Command + C", desci um pouco pra ver exatamente o que está acontecendo. "Command + C", terminal, "Ctrl + L", "Command + V". Então nosso treino são as colunas que o nosso determinado usuário gostou.

[07:31] Ctrl pra cima pra nós voltarmos pra cá, nós só estamos interessados em copiar pra nós ganharmos um pouco de tempo. É exatamente a mesma situação. Então eu vou copiar, inclusive, pra nós lembrarmos as acurácias. Estou copiando exatamente tudo o que nós já fizemos na aula passada. Estou dando um "Alt + Tab" aqui, Ctrl pra cima pra ir. Aqui voltei. Aqui enter, "Command + V", ok. Vamos copiar então esses caras.

[08:06] Nós temos as características, as labels, o treino e teste. Aqui eu estou transformando o meu series do Pandas no array do numpy, novamente pro teste. Aqui eu estou transformando o meu label numa lista única, então eu estou pegando os valores aqui, o ravel, pra transformar os valores do meu series numa lista. Aqui no caso, inclusive, eu acho que é um dataframe e aqui a mesma coisa pro dataframe. Aqui regressão logística, fitei, fiz a previsão e calculei a acurácia.

[08:43] Vamos lembrar quanto que deu para o caso da regressão logística? Eu já fiz aqui separação em treino e teste, vamos fazer com um pouco mais de calma. Estou aqui, Shift, "Command + C", "Command + V", previsões não está definindo. Porque, novamente, o mesmo erro da última vez, nós importamos aqui tudo certo, mas esquecemos de importar o método. Então "from sklearn.linear_model import LogisticRegression".

[09:27] Novamente nós copiamos, “Command + C” pra copiar, “Ctrl + L” pra limpar a tela, “Command + V” pra colar, novamente previsões não está definido, porque eu não importei de novo. Eu venho aqui, “sklearn.linear_model”, copiei, “Ctrl + L”, “Command + V”, invalid syntax porque eu coloquei aqui. Agora sim temos certeza que vai dar tudo certo. Não encontrou, é porque é “sklearn”. Agora sim eu tenho certeza que vai dar certo. Eu aposto isso.

[10:05] Deu certo. Então novamente, aqui vai ser “model.fit”, na verdade é porque eu chamei aqui de modelo, mas é “model.fit”, vamos chamar todos aqui de model, porque depois eu quero chamar de modelo. Model, previsões. Ok. Então fomos aqui, ok, copiamos tudo, “Command + C” pra copiar, “Ctrl + L”, “Command + V”. Previsões is not defined. Por que not defined? Porque tem um O aqui. Então subi “previoes”, só pra nós vermos a acurácia. 81% para o caso de regressão logística. Aqui eu já vou renomear pra nós não nos perdermos.

[10:54] Como que nós fazemos para o caso da árvore em si? Novamente é muito simples. Como que nós fizemos pra nós subirmos nossa árvore da primeira vez? Nós temos o nosso tree, e agora, ao invés de nós fazermos decision tree regressor, nós queremos o modelo que seja decision tree classifier. Então vou dar um “Command + C”, vou dar um “Command + V”. Nós queremos um que seja “DecisionTreeClassifier”.

[11:24] E agora nós fazemos o “modelo.fit”. Nós chamamos de treino e teste, se eu não me engano. Isso, treino e treino labels. E todo o resto é literalmente igual pra nós compararmos. Eu vou até reescrever só pra nós acompanharmos o que eu estou fazendo, “previsoes” recebe “modelo.predict(teste)” . Aqui eu já importei, então a nossa acurácia, nada mais é, acurácia recebe “accuracy_score(teste_labels, previsoes)”.

[12:15] Agora eu copiei esses e vou simplesmente copiar pra nós vermos o que está acontecendo. “Ctrl + L” pra limpar, “Command + V” limpa. Deu erro por algum motivo. Vamos por partes pra nós entendermos o que está acontecendo.

[12:31] Eu copiei o meu modelo, consegui copiar. Eu fitei o meu modelo. É porque eu estou dando vírgula aqui, é “treino_labels”, então eu fiz o fit do modelo, copy, paste, “Command + V” pra colar. Criei a árvore decision tree classifier, fiz as previsões em cima dos dados de teste, previsões feitas e calculei a acurácia. “Command + C” pra copiar, “Command + V” para colar. A acurácia aqui, 59%, é superbaixo.

[13:05] Exatamente pelo mesmo motivo que nós discutimos na árvore, no caso da regressão. O comportamento das duas é muito parecido. Vamos ver por que eu falo isso, vamos ver qual foi nossa acurácia de treino só pra nós darmos uma olhada? Então treino, passando treino_labels e as nossas previsões. Acurácia 1. Exatamente a mesma coisa que aconteceu. Esse ponto ela foi encontrado regras muito complexas e ela foi quebrando, ela foi se dividindo lá embaixo.

[13:40] E qual é a ideia? Vamos limitar isso pra ela não sair quebrando tudo e vamos ver se melhora um pouco isso. Então quando eu criar aqui eu vou criar um “max_depth=5”, pode ser 2 ou pode ser 3, eu fiz novamente meu fit, criei. Vamos ver agora no caso de treino? Só pra nós darmos uma olhada. Acurácia, estou apertando sempre pra cima pra nós voltarmos e vermos o que está acontecendo, caiu um pouco melhor a acurácia. Aqui eu quero previsões, no caso, de teste, estou subindo aqui, eu vou ter meu teste labels, e agora minha acurácia, 72%.

[14:26] Ainda não está tão boa do que a nossa, se eu não me engano é o model, que é o da Logistic Regression. Se nós fizermos “previoes”, nós chamamos de não é por acaso justamente por conta disso. Então nós temos aqui o “previoes”. Nossa acurácia, aqui no caso, da regressão logística, 81%. Ou seja, é um pouco melhor, mas ainda não bateu a nossa regressão logística, por quê?

[14:58] Porque as árvores, em geral, elas isoladas resolvem bem problemas não lineares, mas elas isolados tendem a se comportar um pouco pior de que outros modelos, do que Naive Bayes, regressão logística ou regressão linear em geral.

[15:13] E pra isso existem métodos que se preocupam em combinar a árvore, combinar regras de decisões dessas árvores pra reduzir também, no caso, a variância, ou o viés que a árvore pode dar, como também elas combinadas

tendem a dar resultados melhores, e elas isoladas, enquanto elas isoladas não dão. E é exatamente isso que nós vamos aprender, como isso funciona na nossa próxima aula.