01

Random Forest e Bagging

Transcrição

Nessa aula vamos usar os datasets: movies_multilinear_reg.csv (https://s3.amazonaws.com/caelum-online-public/machine-learning-aprendizado-supervisionado/movies_multilinear_reg.csv) e avaliacoes_usuario.csv (https://s3.amazonaws.com/caelum-online-public/machine-learning-aprendizado-supervisionado/avaliacoes_usuario.csv).

[00:00] Nesse último conjunto de aulas nós vamos ver como nós vamos solucionar o problema que nós vimos na última aula.

[00:11] Conforme eu falei, as árvores isoladas tendem a funcionar piores do que outros métodos mais tradicionais, como a regressão linear ou o naive bayes, por exemplo. Além de ela tender a overfitar os nossos dados e por isso que nós partimos pra algumas técnicas de podagem ou pruning, que foi o que nós vimos no caso do max depth, por exemplo, que nós acabamos setando.

[00:36] A ideia nesse conjunto de aulas é nós vermos como que nós podemos combinar essas árvores de maneiras diferentes conforme elas estão sendo construídas e tudo mais, pra criar um novo modelo em cima disso, porque se nós conseguimos combinar várias árvores, nós tendemos a reduzir a variância, porque nós estamos pegando a média, e se nós pegamos a média, nós talvez generalizemos isso um pouco melhor.

[01:02] A ideia, como é que eu posso explicar? Vamos pensar que, e se nós tivéssemos, ao invés de uma única árvore que pode ser criada com profundidade N, em cima de um dado de treinamento, nós temos infinitos dados de treino, e pra cada desses dados de treino, nós treinamos uma árvore diferente.

[01:23] E com várias árvores treinadas, nós pegamos a média dos valores dessa árvore ou no caso de uma regressão, por exemplo, então nós temos os dados de filmes, que é o nosso caso, que é o problema que nós estamos querendo trabalhar, e nós queremos calcular a bilheteria em cima desse conjunto de filmes.

[01:41] E se nós então calculamos uma árvore diferente, pra um conjunto de filmes, sei lá, que nós temos os 10 mil primeiros filmes, nós treinamos uma árvore, de 10 mil a 200 mil filmes nós treinamos com árvores nós fazendo isso repetidas vezes, centenas de vezes, nós vamos ter 100 árvores diferentes, e nós falamos: “pro dado X vamos fazer a previsão pra esse cara?”, beleza, pra esse dado X pra essa árvore respondeu tanto, pra aquela árvore respondeu tanto, e nós pegamos a média desses valores pra ser o valor da nossa regressão.

[02:13] Só que nós não temos infinitos dados de treino, nós não temos centenas de dados de treino diferentes, nós só temos um, então como que nós fazemos isso? Nós pegamos amostras. Essas amostras podem, inclusive, ser com reposição de informação. Então vamos pensar numa forma mais esquemática?

[02:31] Nós temos aqui o nosso conjunto de filmes, que é aquele conjunto de filmes que nós vimos. Vamos abri-lo aqui pra nós darmos uma olhada? Eu vou entrar na minha pasta, estou entrando no meu curso. Eu tenho aqui o curso, eu tenho meu dataset, nós lembramos o nosso que nós demos? Foi esse daqui, movies_multilinear_reg, que nós fizemos a regressão no primeiro caso.

[02:56] Ele está abrindo aqui, está abrindo o nosso querido Excel, e enquanto ele abre, qual é a ideia? Nós o abrimos aqui, eu vou aumentar a tela, venho no view, vou dar um zoom. Enquanto ele abre a aba pra dar um zoom, vamos voltar pro nosso caso?

[03:19] Eu tenho esse filme, imagina que são todos esses filmes. E nós queremos pegar uma amostra, então nós pegamos, sei lá, 20 caras desses filmes, pega Jumanji, pega Leaving Las Vegas, pega Babe e assim por diante, até completar 20, de forma aleatória. Vai ser a nossa primeira amostra.

[03:40] Nós vamos olhar de novo esses caras, vamos pegar outros 20 caras, incluindo ou não os primeiros caras, então Babe de novo, o Richard III, pegamos de novo Friday, até fechar mais 20 caras, essa vai ser nossa segunda amostra. E nós repetimos o processo pra uma terceira amostra. Nós fazemos isso N vezes, ou seja, até completar o número máximo de árvores, você tem centenas de vezes, não tem um número máximo, na verdade.

[04:07] E pra cada um desses caras nós treinamos uma árvore. Aqui é uma árvore pra esse cara, nós vamos ter a árvore pra esse cara, nós vamos ter a árvore pra esse cara. E as árvores vão ser treinadas em cima desses dados de treino. E nós jogamos e falamos, pro nosso dado X, nosso dado de teste, vamos pegar o Zootopia. Pra Zootopia, o que essa árvore responde? Qual é a bilheteria estimada pra esse cara? E qual é a bilheteria estimada pra esse cara? E qual é a bilheteria estimada pra esse cara?

[04:35] E então nós vamos ter diferentes valores de regressão, exatamente como nós vimos no caso de árvore de decisão e nós simplesmente fazemos a média. E nós fizemos a média ali, temos o nosso resultado, nesse caso bagging. Esse processo é chamado de bagging, ele vai fazendo uma mochila, uma sacola pra cada um dos nossos conjuntos de árvores diferentes.

[05:05] E qual é a ideia? No caso da regressão, nós pegamos a média. E no caso da classificação? Nós vamos ter uma classificação, novamente, esse conjunto de dados é se o usuário gostou ou não gostou de um filme, joga numa árvore, não gostou, joga nessa árvore daqui, gostou, joga nessa árvore aqui, não gostou. Considerando nosso conjunto deu não gostou, gostou, não gostou. Não gostou. Nós levamos em consideração o conjunto final, a maioria dos votos.

[05:31] O processo de construção é muito parecido, tanto pra regressão quanto pra classificação, o que muda são as regras, eu não vou entrar novamente nos detalhes dessas regras, no split dessa divisão da árvore, porque o scikit-learn faz essa divisão pra nós. Então falando de scikit-learn, como que nós codamos isso?

[05:49] Eu vou abrir aqui, esse foi o código que nós já escrevemos, então eu tenho lá o Pandas, eu vou ter o Matplotlib pra nós vermos as coisas, a divisão de treino e teste. Aqui nós fazemos a criação do modelo.

[06:06] Nós não vamos precisar entrar nos detalhes da criação do modelo aqui, porque nós vamos fazer outro modelo. Só que todo o resto nós podemos reaproveitar, então pra mostrar isso, eu vou copiar e não vou escrever de novo na mão, porque foi exatamente o que nós fizemos na aula passada. Eu estou copiando todos esses dados, inclusive os imports, estou dando um copy, eu vou entrar no meu terminal.

[06:31] Eu estou entrando no meu curso que, se eu não me engano está aqui. Então eu tenho aqui meu curso. Eu entrei no Python, estou colando o código aqui. Colei. Agora eu tenho meu “treino.shape”, o meu “teste.shape”, o meu “teste_labels.shape”, e o meu “treino_labels.shape”. Como nós podemos ver, nós temos treino, treino, mesmo tamanho, o que muda são o número de colunas em cima disso.

[07:19] E agora o que nós vamos fazer? Nós vamos usar o “from sklearn.ensemble”, de juntar, porque ela combina, “import BaggingRegressor”, nós vamos fazer a regressão. Eu vou até copiar esse cara para o nosso querido Atom. Na verdade eu vou criar um novo arquivo e vou copiar exatamente tudo o que nós acabamos de fazer, porque é a mesma coisa basicamente. Aqui é mais pra nós nos guiarmos. Então “Command + V”, tudo o que nós já fizemos.

[08:00] Vou até salvar aqui pra ficar mais visível, eu vou chamar aqui de bagging. Vamos chamar de “ensemble.py”, porque nós vamos juntar todos os nossos métodos aqui.

[08:19] Agora nós criamos um modelo, nós só precisamos dar, o modelo recebe "BaggingRegressor.fit(treino, treino_labels)". Ele deu um erro, porque nós não criamos o modelo propriamente, comi bola aqui um pouco. Criei o modelo.

[08:44] Agora nós já fazemos modelo recebe, vamos ver, pegando aqui nos meus esquemas, "modelo.fit(treino, treino_labels)". Criamos o curso. Ele deu um warning aqui, porque eu passei um vetor coluna quando ele estava esperando array de uma única dimensão.

[09:14] Se nós olharmos o nosso column vector, no caso foi o que eu passei pro nosso treino labels, ele pede pra nós usamos o ravel, porque é isso que ele está falando aqui. Vou até dar uma diminuída aqui pra nós darmos uma olhada melhor. Por favor mude isso usando – eu vou limpar a tela pra ficar mais fácil de ver – por favor usando isso, usando ravel. Por quê? O que vai acontecer?

[09:43] Se nós dermos olhada no nosso treino labels, ele está com esse formato, se eu der um ".ravel" vai ficar tudo num conjunto só. Então criamos agora o nosso bagging regressor, ele está aqui feliz e contente.

[10:00] Agora o que nós precisamos fazer? Nós só precisamos fazer o nosso score, então eu estou justamente copiando porque é a mesma ideia. Criamos aqui o nosso treino e teste, "Command + V", "Alt + Tab", vou fazer meu "modelo.fit" passando já o ".ravel". "Command + V", ".ravel" aqui. Vou copiar tudo pra mostrar que funciona, "Command + S" pra salvar, "Command + V" agora, deu certo.

[10:41] Para os nossos dados do treino deu 96 e aqui deu quase 80%, ou seja, já deu muito próximo do nosso caso de regressão linear. E se eu não me engano, nós o temos aqui. Eu vou importar, eu vou fazer exatamente a mesma comparação pra nós podermos comparar os dados, vamos fazer aqui. Olha só, 82 ali. 79 pra 82. Já chegou muito próximo logo de primeira, o que já é muito bom.

[11:07] Uma coisa muito legal também do bagging é que nós não precisamos mais nos preocupar com o limite da árvore, o quanto que ela pode crescer, ela meio que pode crescer à vontade, nós podemos passar alguns limites pra evitar esse overfitting, mas como nós pegamos a média agora das árvores, a tendência é que essa média acaba reduzindo a variância no modelo e nós acabamos não pegando tanto overfitting, é uma lida um pouco melhor que overfitting.

[11:36] No caso aqui ele está usando o número de estimadores igual 10, que são 10 árvores. Nós podemos aumentar isso pra ver o que acontece, só por curiosidade. Estou apertando pra cima pra nós voltarmos, eu vou usar "n_estimators=20", porque agora eu quero que sejam 20 árvores. E eu vou fazer a mesma coisa de antes, "modelo.fit", ok. Criei agora, vamos fazer esse score? Agora estou curioso.

[12:06] Nosso dado de treino não mudou, nosso dado de teste aumentou muito pouco, ou seja, quase não fez diferença. Então nós já termos essas 10 árvores no total já resolve bem para o nosso caso. Combinou, funciona bem, chegou perto da regressão linear, não foi melhor, mas já foi muito melhor do que a decision tree simples, sem nós nos preocuparmos com essa podagem inicial, essa podagem de profundidade máxima, ela já foi mais direta e resolveu bem. Querendo ou não, 79 pra 81 é relativamente próximo.

[12:48] E agora como que nós fazemos para o caso de classificação? No caso da regressão, ela vai pegar a média, no caso da classificação ela vai pegar a maioridade dos votos, exatamente como eu falei.

[13:04] A construção é praticamente a mesma, então novamente aqui, eu vou copiar os dados como nós já fizemos antes e vou até fazer toda essa parte daqui, que nós chamamos a nossa regressão logística, só pra nós relembrarmos. Eu vou reexecutar os dados novamente, mas por uma questão de nós lembrarmos o que está acontecendo, lembrarmos dos dados e tudo mais, estou copiando pra provar que funciona exatamente da forma como nós vimos.

[13:36] Venho aqui, dei paste, ok. Fiz as previsões, só chegar aqui, “previsoes”, ok, são as previsões que ele deu, muito bizarro, na verdade. E se nós chegarmos aqui, fizermos as previsões dos nossos dados de teste, vamos fazer a acurácia, vamos aplicar aqui também a acurácia, a nossa querida acurácia. Ok. Vamos limpar aqui, estamos colando aqui. A nossa acurácia para o caso da regressão logística foi de 82%.

[14:12] Agora o que nós precisamos fazer? Novamente a mesma ideia, “from sklearn.ensemble import BaggingClassifier”. Agora nós fazemos “modelo” recebe “BaggingClassifier”, abre e fecha. “modelo.fit(treino, treino_labels)”. Criamos o classificador.

[14:44] Agora a coisa aqui é mais simples, já criamos o classificador, nós fazemos a avaliação desse cara em cima dos nossos dados de teste, vamos aqui, paste, previsões, e simplesmente calculamos a nossa acurácia, como nós já estamos acostumados, já fizemos milhares de vezes. Nesse caso novamente vamos ver quanto que deu. Deu praticamente a mesma coisa. Olha só que loucura.

[15:14] Claro, na verdade não é que deu a mesma coisa, no caso da regressão logística nós estávamos usando model, agora aqui nós chamamos de modelo. Quando os dois dados estão exatamente iguais ali, suspeite disso, porque provavelmente você errou, que foi o caso que aconteceu aqui comigo.

[15:34] Então fiz agora a acurácia, eu estou passando previsões e o teste, agora acredito que esteja tudo certo, acurácia, 74%. Foi um pouco pior, mas é justamente mais um caso pra nós vermos um modelo de classificação que funciona bem, acima de 70% ali, quase 75, seria outro indicativo ok num modelo bom. Pra esse conjunto de dados a regressão logística funcionou, acabou indo melhor. Mas pra outro conjunto de dados nós poderíamos ver um caso que o bagging funcionaria bem.

[16:07] E agora qual é a ideia? Vamos supor que nós queiramos, não vamos supor, é uma melhoria que nós podemos fazer em cima desse modelo, que e se ao invés de nós considerarmos só esse corte, esse conjunto de amostras, e montar as árvores considerando todas as nossas features, todas as nossas características, nós só considerássemos um número aleatório de características, de forma que nós reduzíssemos ainda mais esse viés?

[16:39] O que eu quero dizer com isso? Se nós chegássemos aqui, nós temos essas N características, se nós formos ver, se eu não me engano, são 15, que são esses 15 dados pra gerar essa previsão. E se nós, ao invés de escolhermos esses 15 dados, nós escolhêssemos o conjunto menor do que esses 15 dados pra fazer cada divisão da nossa arvorezinha lá?

[17:01] O que eu quero dizer com isso? É se ao invés de 15 nós selecionássemos um subconjunto de quatro dados e esses quatro poderiam ser, por exemplo, romance, terror, a duração e o filme ser de aventura. Nós pegamos esses quatro dados de forma aleatória, e considerando esses quatro dados, nós criamos uma regra de divisão. Depois nós pegamos outros quatro dados, se ele é de animação, se ele é de comédia, a duração e o investimento também.

[17:35] Então dado o meu conjunto de características pra cada divisão, eu seleciono um conjunto aleatório de características e esse conjunto aleatório de características é utilizado pra fazer a minha regra de divisão entre os meus dados, em geral. Então eu vou ter duas coisas, eu vou ter uma coisa que já aconteceu, que é o caso do bagging, que já acontece no bagging, que é como que eu vou pegar os meus dados, vou criar várias subamostras, dessas subamostras eu crio uma árvore.

[18:14] Só que agora eu vou ter um caso a mais, na hora de construir essa árvore eu considero um subconjunto de features, as minhas features pra selecionar, pra fazer as minhas divisões. Quando eu falo features são as características, se é documentário, sci-fi, mistério, horror, e eu considero um conjunto aleatório desses a cada vez que eu estou dividindo a minha árvore.

[18:35] Esse outro algoritmo, na verdade, é o que nós chamamos de random forest e, novamente, não vou entrar nas partes mais complicadas, mas a ideia geral dele é esse, que é mais um adicional, é outra forma de você construir essa

árvore, combinando as árvores dessa forma.

[18:49] Vamos fazer aqui como que nós criamos o random forest. Novamente eu estou usando ".ensemble import RandomForestRegressor". Agora que eu importei eu vou só incorporar o modelo, o modelo recebe "RandomForestRegressor". Antes do modelo.fit vamos recopiar os dados, porque eu mudei os nomes de treino e teste para o nosso dataset daqui. Agora são os nossos dados de treino e teste, estou copiando-o pra recuperá-los. Agora é "modelo.fit(treino, treino_labels)".

[19:51] Novamente, o mesmo erro quando nós estávamos trabalhando com o bagging regressor. Então dei "Ctrl + L" pra nós limparmos, ".ravel" aqui. Criamos o nosso modelo. O max features auto é, no máximo, quantas características eu estou solucionando para fazer a regra de divisão da minha árvore.

[20:14] No caso ele está setado como auto, é porque ele pega a raiz do número de características que eu tenho. Se são 15, no caso são 15, ele seleciona no máximo 4 ali, 3 e alguma coisa, 4 por divisão da minha árvore, pra selecionar e fazer a divisão da minha árvore. Número de estimadores, novamente, é o máximo de árvores que eu tenho.

[20:34] E a ideia, você vai construir essa floresta dessa maneira aleatória. Por que aleatória? Porque as decisões que eu monto pra ir montando minha árvore são regras aleatórias e é esse resultado final que acaba tendo, por isso eu vou ter várias árvores, essas árvores são construídas de formas aleatórias, por isso random forest.

[20:55] E agora eu já fiz o modelo.fit, vou copiar exatamente, como nós já vimos, o score e o caso. Deu 78% nesse caso. Vamos passar aqui um ".ravel" só por desencargo, não mudou, e aqui um ".ravel" só por desencargo, não mudou. Deu 78%. Também ali comparado deu um pouco pior do que o bagging, mas também foi bem próximo do nosso caso final, mas funcionou bem. Nós passamos diretamente, ela funcionou de uma forma aceitável.

[21:32] Novamente, pra esse conjunto de dados, uma regressão linear funciona melhor. Pode ter outro conjunto de dados que talvez esse modelo acabe sendo a melhor decisão a se fazer. Vamos fazer a mesma coisa pro nosso caso de gostos?

[21:44] Nós importamos basicamente tudo, como nós já fizemos antes. Eu estou dando aqui um paste, fizemos nossos dados de treino e dados de teste. Agora aqui nós vamos ter "from sklearn.ensemble import RandomForestClassifier". Então nós temos o modelo, ele recebe "RandomForestClassifier", ok, "modelo.fit(treino, treino_labels)". Criamos.

[22:27] Novamente o critério de divisão foi o Gini, inclusive da regra de decisão dos classificadores, como nós já vimos. Mesmo ponto aqui, seleção de features máxima, número de estimadores, coisas que nós vimos.

[22:42] Agora o que nós precisamos fazer? Nós já fizemos o fit, nós só precisamos nos preocupar, novamente estou copiando, mas porque nós já fizemos isso antes. Na verdade vou até escrever de novo, "previsoes" recebe "modelo", inclusive porque lá estava model, ".predict(teste)". E "acuracia" recebe "accuracy_score", se eu não me engano são teste labels e previsões. Labels e previsões.

[23:14] No caso então, a acurácia desse modelo não foi muito boa, ou seja, vimos também mais um caso em que uma floresta não funcionou tão bem quando o bagging, então essa escolha meio aleatória do número máximo de features não foi muito legal.

[23:31] Uma coisa que nós podemos ver é se ao invés de quatro nós selecionarmos até mesmo mais, apenas na curiosidade de ver o que acontece. Então vamos olhar aqui? "max_features" ao invés de 3 ali, alguma coisa, vamos selecionar que são 5, no máximo 5. O que isso daqui pode empatar? Na verdade, vamos até selecionar mais um parâmetro, aqui também tem o máximo de profundidade, no caso desse algoritmo, que nós vamos setar também como 5, só pra ver o que acontece.

[24:00] Nós vamos ter então “modelo.fit”, criamos aqui, como nós vimos, max depth 5, max feature 5. “modelo.fit”, fizemos a previsão em cima dos nossos dados de teste. Acurácia aqui, qual foi nossa acurácia agora? 75%.

[24:17] Agora só com essas mudanças, nuances que nós fizemos, alguns parâmetros tunáveis, eu selecionei esses valores porque nós temos uma quantidade de colunas muito grande, e uma raiz de 15, que também é um valor meio baixo. E também quis selecionar esse número de profundidade mais por esse ponto de viés. Conforme nós vimos, se nós fizermos esses dados para os dados de treino, agora vai dar um pouco melhor, mas provavelmente estava dando algum tipo de overfitting em cima desses dados.

[24:51] O ponto legal dessa aula é que nós vimos como nós podemos combinar, existem algoritmos de você combinar diferentes árvores de decisão, na verdade em alguns desses modelos podem ser usados outros tipos de algoritmo, focamos aqui só em árvores de decisão, pra você construir novos modelos em cima disso. E existem muitos algoritmos de casos reais que usam, por exemplo, random forests como coração, assim, alguns no caso de reconhecimento de face, por exemplo, em aplicações bem interessantes.

[25:22] E o legal da próxima aula é que nós vimos que esse modelo considera muito a média como fator mãe pra você construir, no caso das árvores regressoras. Nós vamos ver outros tipos de conjuntos de algoritmos que também são utilizados pra combinar, mas eles não levam em consideração a média, as árvores são construídas de uma forma um pouco mais abstrata e um pouco diferente, mas também funcionam. Isso é sobre o que nós vamos ver na aula seguinte.