4.5 - Featuring engineering: criando novas variáveis para facilitar a análise (parte 1)

00;00;00;18 - 00;00;26;27
Bem-vindes à nossa aula sobre Featuring Engineering. Hoje vamos explorar como transformar e criar novas variáveis a partir dos nossos dados. A engenharia de recursos ou Featuring Engineering é uma habilidade essencial para qualquer análise de dados, pois nos permite extrair insights mais profundos e fazer previsões mais precisas. Vamos lá. Bom, vamos aqui primeiro no nosso notebook já criar uma nova célula de texto pra gente poder começar uma nova aula.

00;00;27;12 - 00;00;57;20
Então vamos lá... Nosso jogo da velha, Featuring Engineering. Ok. E aqui a gente já pode começar. Bom, essa engenharia de recursos é o processo de selecionar, criando e transformando características, que são os "Features", de dados brutos, para torná-los mais úteis para algoritmos de machine learning. Em termos simples, é como preparar os ingredientes antes de cozinhar. Imagine que você está fazendo um bolo.

00;00;57;27 - 00;01;24;07
Você precisa de ingredientes como farinha, ovos e açúcar. A gente até aprendeu isso em outras aulas. Mas se você quiser um bolo ainda mais saboroso, você pode adicionar ingredientes extras, como chocolate ou frutas ou cenoura em bolo de cenoura. Essas adições extras são como Features Engineering. Elas melhoram a qualidade do resultado final. Bom, um bom exemplo da engenharia de recursos é criar novas variáveis com base em combinações das variáveis existentes.

00;01;24;21 - 00;01;45;04
Lembra lá no módulo 2 que nós criamos uma coluna chamada Novo Nível? Vamos criar uma coluna novo nível aqui, mas a gente vai fazer algo chique agora. Vamos fazer uma função que vai receber os dados de gestor e de nível, ou seja, os dados da coluna gestor e da coluna de nível, e vai retornar para a gente se essa pessoa é gestora ou se tem um valor de nível já definido, fechou?

00;01;45;17 - 00;02;17;07
Vamos lá... Vamos chamar essa função de "preencher nível". Então vamos lá. Função aqui a gente coloca o quê? Def, de definition, de função, preencher underline nível. Bom, a gente vai colocar aqui esse parênteses e essa função vai receber dois parâmetros que a gente já falou, que é gestor e nível. Então, dentro dos parênteses a gente vai colocar os parâmetros gestor vírgula nível e depois dos parênteses a gente coloca os dois pontos.

00;02;17;08 - 00;02;34;28
Ou seja, a partir desses dois pontos, vai ser relacionado à nossa função. A gente vai ficar atento para a questão da indentação. Bom, se o parâmetro de gestor que a função receber foi igual a 1, então vai retornar o valor "pessoa gestora".

00;02;35;16 - 00;02;58;29
Assim. O legal do Colab é isso também, a gente vai ficar atento à intentação, mas no Colab, automaticamente se eu coloquei aqui o def, o nome da função, dois pontos, quando a gente dá enter, ele já vai fazer indentação pra gente, que é fácil, maravilhoso. Então tá, a ideia é: se a pessoa for gestora, então vai retornar para a gente “pessoa gestora”.

00;02;59;25 - 00;03;27;21
Se a pessoa for gestora, então a gente usa o “if”. If gestor igual igual a 1, ou seja, se gestor foi igual a 1, dois pontos, o que vai acontecer? Olha a indentação maravilhosa de novo que já tem aqui pra gente. E vai ter o return: return pessoa gestora. Beleza. Então a gente está aqui fazendo a função.

00;03;27;21 - 00;04;03;00
Dentro da função, tem: se gestor igual a 1, retorna a pessoa gestora. E caso contrário? Caso contrário, o retorno vai ser o próprio valor de nível, porque caso a pessoa não seja gestora, ela vai ser júnior, pleno ou sênior, beleza? Então a gente vai colocar aqui ó, vai dar o enter... Só que aí o enter aqui vai seguir a indentação. Só que agora a gente tem um caso contrário, que é o else. O else tem que ficar no mesmo nível do if. Então a gente dá um backspace aqui e a gente coloca else.

00;04;03;16 - 00;04;30;14
Else dois pontos, porque no caso contrário aqui a gente não tem nenhuma condição, porque ou é uma coisa ou é outra. A gente dá um enter, return nível. Essa é a nossa função, temos aqui pronta. Vamos executar essa célula para ter a função pronta para uso e na próxima célula vamos aplicar ela. Beleza. Para aplicar, vamos usar uma função do pandas chamada apply.

00;04;30;17 - 00;04;51;07
O apply é uma função que, aplicada a uma série ou uma tabela, permite que a gente aplique uma função específica ao longo das linhas ou colunas da tabela. A gente usa assim: a gente começa com a nossa tabela, que no caso é dados ponto apply, que é uma função e o parêntese. Porém, nesse caso, dentro do apply nós usamos a expressão lambda.

00;04;51;19 - 00;05;08;29
Essa expressão é uma forma concisa de definir uma função sem precisar nomear e tudo mais. Então a gente vai fazer uma função, mas a gente não vai precisar nomear, a gente vai só usar o lambda. Normalmente a gente usa o lambda quando a gente vai usar uma função simples, mas só é usado uma vez, a gente não vai ter que usar várias vezes.

00;05;09;16 - 00;05;37;21
Então a gente coloca assim: lambda X dois pontos... Nesse formato, nós vamos aplicar alguma função, ou seja, com o uso do lambda para cada linha, o uso do apply, na nossa tabela de dados, que é representada pelo X. Ok? Então esse lambda X diz que a função sem nome que vamos criar recebe esse argumento X e esse argumento X é a nossa tabela de dados.

00;05;38;10 - 00;06;02;10
E qual vai ser a realização dessa função? Vai ser o seguinte: preencher nível. Ok? Então a gente está aplicando a função preencher nível na nossa tabela de dados. Aí você pode perguntar: Uai, Carol, a gente está criando um função para executar uma função? Porque o lambda já é uma função e o preencher nível é outra função. E é isso mesmo.

00;06;02;17 - 00;06;22;18
Como a gente precisa que a função preencher nível seja executada em todas as linhas da nossa tabela, a gente utiliza o apply e o lambda para isso. Então a gente não precisa criar uma função para executar outra função linha a linha. A gente usa o lambda e tudo certo. Então, dentro do preencher nível, nós colocamos os parâmetros. Quais são os parâmetros?

00;06;22;25 - 00;06;47;29
Os parâmetros são a coluna gestor e a coluna nível que, no caso, a nossa tabela está sendo representada por X. Então X, que é a nossa tabela e a gente coloca a nossa coluna. Poxa, esqueci qual é o nome da coluna. A gente pode criar aqui uma célula de código antes: dados (a nossa tabela), ponto columns e aí a gente vê como está escrito exatamente. A primeira coluna é gestor.

00;06;48;04 - 00;07;23;13
Então aqui está: gestor, a gente copia igual, coloca lá dentro entre aspas simples ou duplas. E o segundo parâmetro é a coluna de nível. Então X nível e ok. Vamos apagar essa linha de código aqui em cima só pra não ficar poluindo. Então a gente está aplicando a função preencher nível na nossa tabela e a gente está mandando nessa função a coluna de gestor e a coluna de nível.

00;07;23;13 - 00;07;55;21
O resultado disso aí a gente quer criar uma nova coluna que é a coluna novo nível. Então vamos atribuir o resultado disso aqui a uma nova coluna. Então vamos colocar assim: dados, novo nível, fecha aqui. Então aqui a gente tem essa nova coluna que vai receber os valores aí da nossa função. Beleza, vamos executar.

00;07;55;21 - 00;08;23;04
Executando, deu um erro, olha aqui: erro de gestor, como se não tivesse a coluna de gestor. Só que a gente deu lá os dados ponto columns, a gente copiou exatamente igual. O que está acontecendo? Está acontecendo que o apply aplica a função por colunas no default, no padrão dele, mas no caso a gente quer aplicar nas linhas. Então a gente dá uma vírgula aqui e coloca um parâmetro chamado axis e em um modo padrão é axis igual a zero.

00;08;23;11 - 00;08;53;18
Mas como a gente quer o contrário, a gente coloca a axis igual 1. E aí sim a gente executa e vai dar certo. Numa nova célula de código, a gente pode até dar uma olhada nessa coluna de novo nível. A gente já tem: júnior, pessoa gestora, pleno, sênior, enfim, deu tudo certo. Podemos até fazer um value counts aqui pra gente poder saber a quantidade de nível das pessoas nessa coluna de novo nível.

00;08;53;24 - 00;09;37;02
Vamos colocar assim: ponto value counts no final. E aí, olha que legal, a gente viu que temos 713 pessoas que são gestoras, facilitou pra nós aqui. Numa coluna só a gente já tem várias respostas. Outro exemplo dessa engenharia de recursos é converter variáveis categóricas em variáveis indicadoras. Como assim? Por exemplo, a gente tem a coluna de nível, então vou colocar aqui mais uma célula de código: dados nível, só pra gente poder dar uma olhadinha aqui.

00;09;38;12 - 00;10;05;15
Nessa coluna de nível temos os dados categóricos: pleno, júnior e sênior. Contudo, e se a gente quisesse uma coluna só para júnior, uma só pra pleno e uma só pra sênior? Transformar essas variáveis categóricas em um indicador, se a pessoa é ou não, sabe? Por exemplo, uma coluna de júnior, onde pra cada linha em que a pessoa fosse júnior, houvesse um indicador de true e quando a pessoa não fosse júnior, o valor seria false.

00;10;06;15 - 00;10;29;06
A gente consegue fazer isso de um jeito bem tranquilo usando a função Get Dummies, do Pandas. Basta ter uma célula e o seguinte código, vamos lá! Mais uma célula de código, pd (que é a do pandas) ponto get dummies (get underline dummies, com dois Ms). A gente abre parênteses e dentro dos parênteses a gente coloca os parâmetros.

00;10;29;20 - 00;10;59;03
O primeiro parâmetro é a nossa tabela: tabela dados, vírgula e qual é a coluna que a gente está querendo pegar os valores. No caso é columns, coluna em inglês, e aí, qual é a coluna que a gente está querendo? Coluna nível. Ok. E ao executar esse código, a gente tem isso aqui como resposta. Inicialmente, nada mudou, mas se a gente arrastar aqui lá para o final da nossa tabela, a gente vê três colunas: nível júnior, nível pleno e nível sênior, e em cada coluna a gente tem o true ou false aqui.

00;10;59;03 - 00;11;23;15
Isso é importante porque os algoritmos de análise e aprendizado de máquina trabalham com dados numéricos. Por isso, dados categóricos frequentemente são transformados em numéricos ou booleanos. Um true ou false é como o 1 e zero para máquina, ok? Uma utilidade na utilização do get dummies para a análise de dados é permitir a criação de gráficos e tabelas que mostram a contagem ou a proporção de cada categoria.

00;11;24;12 - 00;11;54;27
Para adicionar essas três colunas criadas à nossa tabela, a gente atribui o resultado, que é uma tabela, à nova tabela dados. Então assim, aqui a gente já teve como resultado a tabela, a gente coloca aqui “dados” que vai receber essa tabela agora com mais três tabelas. E prontinho. Se a gente criar aqui mais uma célula de código abaixo e der uma olhada nas colunas, a gente vai ver que temos todas as colunas e as últimas são: nível júnior, nível pleno e nível sênior.

00;11;55;23 - 00;12;20;10
Nós conseguimos fazer o contrário também, transformar as variáveis contínuas em discretas. Vamos supor que queremos transformar a coluna de idade em variáveis discretas que nos diga qual geração essa pessoa pertence: geração X, geração Z, geração millennial ou outra geração. Vamos criar uma função que realize essa determinação de geração pra gente. Vamos criar mais uma célula de código aqui.

00;12;20;22 - 00;12;48;01
Vamos chamar essa função de determinar geração: def determinar underline geração. E aí essa função vai receber o quê? O valor de idade. Então: idade. E agora a gente vai fazer algumas condicionais. Eu pesquisei já no Google e vi que a geração X, no momento em que a gente está gravando essa aula, são de pessoas com idade entre 39 e 58 anos.

00;12;48;13 - 00;13;18;26
Então, aqui na minha função, vou colocar dois pontos aqui embaixo, if idade (então é o nosso valor de idade que vai chegar aí, que é um número)... Se esse número estiver entre 39 e 58, a gente vai retornar a geração X. Então a gente coloca aqui: 39 idade 58. Então, se a idade for maior que 39, mas menor que 58... Na verdade tem que ser menor ou igual para incluir as pessoas de 58 anos.

00;13;19;00 - 00;13;48;15
Então, se for maior que 39, mas menor ou igual a 58, a gente retorna que a pessoa é da geração X. Então: return geração X. Ok? Bom, pessoas com idade entre 29 e 39 anos são millennials e os 13 a 29 anos são a geração Z. Vamos colocar essas condicionais. Vamos lá, lembrando sempre das indentações, então, a gente coloca mais um if.

00;13;48;16 - 00;14;30;04
Na verdade, a gente pode usar o elif, que é meio que: não encaixou no primeiro if, vai para um segundo if. Elif a gente coloca 29 maior ou igual idade menor ou igual 39. Ok, e se a idade estiver nesse intervalo, retorna millennial: return millennial. Ok. E elif é meio que um else if. Elif maior ou igual a 13 idade menor ou igual a 29, então return geração Z.

00;14;30;11 - 00;14;57;03
Bom, eu olhando aqui, eu já vi que eu errei porque eu coloquei aqui no primeiro a idade maior que 39 e menor ou igual a 58, e no segundo aqui eu coloquei maior ou igual 29, mas depois coloquei maior ou igual 29 de novo. O maior ou igual é sempre na maior idade aqui. Então vou tirar o igual daqui e o igual daqui.

00;14;58;05 - 00;15;24;03
Então agora a gente tem certinho. Se a idade for maior que 39, menor ou igual a 58, então geração X. Maior que 29 e menor ou igual a 39: millennial. Maior que 13 e menor ou igual a 29, geração Z. Ok? Bom, e vamos colocar uma condição de caso contrário: se não encaixou em nenhuma dessas, a gente vai retornar em outra geração.

00;15;24;16 - 00;15;49;28
Então vamos colocar um else return outra geração. Beleza? Ok, algumas coisas aqui: em alguns momentos eu coloquei espaço em outros momentos eu não coloquei. Não tem problema, mas é óbvio que fica mais bonitinho se a gente tiver um padrão, ok? Em alguns momentos eu coloquei aspas duplas, em outros momentos eu coloquei aspas simples.

00;15;50;18 - 00;16;21;06
Não tem problema, é óbvio que é bom ter um padrão, mas não tem problema nenhum. Então a gente tem a nossa função aqui, prontíssima. Vamos executar essa célula, criar uma nova célula de código embaixo e vamos criar uma coluna chamada Geração. E essa coluna vai receber a resposta da nossa função. Então vamos lá dados geração. Mas como a gente pode aplicar uma função em uma coluna inteira mesmo?

00;16;21;24 - 00;16;42;18
Hoje a gente acabou de aprender a usar o apply. Então a gente vai lá usar o apply. Então vai ser: qual a coluna que a gente está usando agora? Dessa vez vai ser mais fácil porque na outra função a gente tinha dois parâmetros, a gente tinha nível e gestor. Agora a gente tem um parâmetro só, então a gente não precisa mandar a tabela toda,.

00;16;42;18 - 00;17;18;08
A gente está usando o apply para uma coluna. Então é dados idade ponto apply. Não precisamos usar o lambda, olha: facinho. Determinar geração, ok? Como a função tem só um parâmetro, fica bem mais fácil pra gente. A gente executa e aqui numa próxima célula de código, a gente pode dar uma olhada nessa geração.

00;17;18;08 - 00;17;44;22
Vamos usar o value counts para a gente poder ver a quantidade de pessoas por geração: value counts e parênteses vazios. E olha aí a gente tem muitas pessoas da geração Z que está entre 13 e 29 anos e até faz sentido porque não sei se vocês lembram quando a gente calculou moda, a moda deu 27 anos. Então a gente tem uma grande quantidade de pessoas aí com 27 anos.

00;17;45;06 - 00;18;01;22
Então está aí na geração Z. A gente tem em segundo lugar o millennial. A gente tem só 511 pessoas da geração X, que é de 39 a 58 anos. Ok? Interessante que a gente pode até pensar, então, que a nova geração ela tem se interessado bastante pela área de TI, né?