PROCESSO DE DESENVOLVIMENTO DOCUMENTAÇÃO DE PROJETO		
Nome do curso: *Introdução à Machine Learning*	*Regressão*	Responsável: *Paulo Jarbas Camurça*

Roteiro

Regressão

Olá, seja bem-vindo(a)!

Você já deve conhecer as técnicas de aprendizado de máquina supervisionado e não supervisionado, e compreende, de forma geral, em que tipo de problemas elas podem ser aplicadas. Pois bem, chegou o momento de aprender um pouco mais sobre aprendizado supervisionado. Ele abrange, basicamente, dois tipos de problemas: regressão e classificação. Em linhas gerais, problemas de regressão buscam prever um valor contínuo com base em um conjunto de dados rotulados, ou seja, deseja-se prever um valor numérico, como, por exemplo, o consumo de um carro tendo dados de distância percorrida e velocidade. Já os de classificação busca-se associar variáveis de entrada em categorias distintas, como se uma pessoa pudesse, ou não, ter o seu crédito aprovado em uma análise bancária.

Neste curso, você vai conhecer os conceitos fundamentais de regressão, e como duas ou mais variáveis podem ser utilizadas para fazer previsões. Vamos lá?

Então, um modelo de regressão corresponde a uma equação matemática que descreve como duas ou mais variáveis estão relacionadas. Para entender essa relação, introduziremos o conceito de variáveis independentes e variáveis dependentes.

As variáveis independentes são aquelas que não dependem de outras variáveis para serem explicadas, como a idade de uma pessoa, por exemplo. Já as variáveis dependentes são aquelas nas quais se deseja prever, e precisam de outras variáveis para serem explicadas, como o valor do seguro de um automóvel, que depende do ano do carro; e também da idade do motorista.

Para modelar, matematicamente, as variáveis independentes, geralmente utiliza-se a variável X; já as variáveis dependentes, utiliza-se o Y. Então, pode-se expressar o Y como uma função de X. Para facilitar sua compreensão, acompanhe o exemplo a seguir:

Imagine que um vendedor de sorvetes deseja melhorar suas vendas e, para isso, ele irá utilizar um modelo de regressão que, a partir dos dados de temperatura, fará previsões do número de sorvetes vendidos na praia.

Suponha que no primeiro dia, a temperatura era de 30 graus, e ele vendeu 20 sorvetes. No segundo dia, a temperatura baixou para 25 graus, e ele vendeu apenas 12 sorvetes. Já no terceiro dia, aos 36 graus, ele conseguiu vender 50 sorvetes.

Assim, ao fazer esses registros dia após dia, ele pode utilizar um modelo de regressão que melhor se ajusta aos seus dados, e, então, estimar o número de sorvetes que serão vendidos no dia seguinte, tendo a temperatura como variável independente, e o número de sorvetes como variável dependente.

Mas qual modelo utilizar? quais são os modelos de regressão?

Os modelos de regressão basicamente são divididos em dois tipos, regressão simples e regressão múltipla, e a relação desses tipos de regressão podem ser: linear ou não linear. Acompanhe a seguir a definição de cada um desses tipos.

Em primeiro lugar, os modelos de regressão simples são aqueles em que uma única variável independente é usada para obter a variável dependente, como é o caso do nosso exemplo do vendedor de sorvetes.

Já na regressão múltipla, mais de uma variável independente, é usada para obter a variável dependente, como é o caso do exemplo do cálculo do valor do seguro de um carro, que precisava de duas variáveis independentes, que eram a idade do motorista e o ano do carro.

Para que você compreenda melhor como utilizar a regressão é necessário tentar entender, na prática, esse tipo de problema. Mas antes disso, acompanhe o exemplo a seguir:

Primeiro, será feita a demonstração de como importar as bibliotecas do python, como *numpy*, *pandas e matplotlib*. Para isso, você deve escrever uma linha de código para cada tipo de biblioteca com os comandos necessários para realizar essa importação. No caso, pode-se escrever, na primeira linha, “import numpy as np”; na segunda, “import pandas as pd”; e na última, “import matplotlib.pyplot as plt”.

Em seguida, definir as variáveis de temperatura e número de sorvetes. Para isso, crie um array, que é um vetor de uma dimensão. Para criar um array no python, utiliza-se o comando np.array. Então, você pode escrever “temperatura = np.array([30, 25, 36 ...])” e colocar os dados dos registros de temperatura diária. Para o número de sorvetes, escreva na linha seguinte: “numero_sorvetes = np.array([20, 12, 50, 10, ...])”, que representa os registros dos sorvetes vendidos diariamente.

Para organizar melhor os dados, será utilizada a classe *Dataframe* do pandas para criar uma tabela com os registros do número de sorvetes e das temperaturas observadas pelo vendedor

de sorvetes. Para criar o Dataframe, é necessário passar, em um dicionário do python, as variáveis “temperatura” e “numero_sorvetes”, como chaves desse dicionário. Para isso, escrevemos: “df = pd.Dataframe({‘temperatura’: temperatura, ‘numero_sorvetes’: numero_sorvetes})”, em que a chave com a string ‘temperatura’ está associada à variável temperatura, e a chave com a string “numero_sorvetes” está associada à variável numero_sorvetes.

Para mostrar as primeiras linhas da tabela criada, utiliza-se o método “head()” do pandas. Então, escreva df.head(), e será mostrado as quatro primeiras linhas da tabela, organizando os dados que você inseriu. No caso, temos uma tabela com duas colunas, uma com os valores da temperatura relacionados com a outra coluna, que apresenta a quantidade de sorvetes vendidos. Na primeira linha, temos a temperatura 30; interpretando a tabela é possível entender que, neste dia, foram vendidos 20 sorvetes; já na segunda linha, a temperatura é de 25 e a quantidade de sorvete vendia muda para 12. Analisando a tabela com um todo, é possível afirmar que, quanto mais quente a temperatura maior a quantidade de sorvetes vendidos, não é mesmo?! Mas, e se você tivesse muitos dados, como seria possível fazer essa mesma leitura de maneira mais rápida?

A melhor solução seria construir um gráfico. Pois bem, para fazer um gráfico com esses dados apresentados na tabela foi usado, no exemplo, a biblioteca matplotlib, que, nesse caso, já foi importada no início do código. Para isso, foi escrito na primeira linha de código “plt.plot(df['temperatura'], df['numero_sorvetes'], '*')”, em que, como primeiro argumento do plot, passamos o dataframe com as informações de temperatura; e, como segundo argumento, passamos o dataframe com os dados do número de sorvetes; o terceiro argumento é o estilo do plot que, no caso, foi usado um asterisco (*).

Em seguida, para plotar no gráfico o título do eixo x, escrevemos, na segunda linha de código, “plt.xlabel('Temperatura')”; e para plotar, no gráfico, o eixo e o título do eixo y, escrevemos na terceira linha “plt.ylabel('Sorvetes')”. Por fim, para exibir o gráfico, escrevemos, na quarta linha de código, plt.show().

Ao gerar o gráfico com essas informações, temos a apresentação da relação Temperatura x Sorvetes, no eixo x e y do gráfico, respectivamente. O eixo x, na horizontal, traz uma escala com nove temperaturas diferentes: 17,5; 20; 22,5; 25; 27,5; 30; 32,5; 35; 37,5. O eixo y, na vertical, mostra uma escala com cinco números, representando a quantidade de sorvetes vendidos: 10; 20; 30; 40; e 50. No centro do gráfico, desenhos de estrelas mostra o crescimento da venda de sorvetes, pois conforme a temperatura aumenta, a venda de sorvetes também aumenta. Isso confirma a análise feita apenas nas primeiras linhas de tabela dos dados que foi apresentada anteriormente, certo?

Bom, após definido na variável “df” os dados de temperatura e número de sorvetes, iremos construir o modelo. Em primeiro lugar, deve-se atribuir a variável independente “x” com os dados de temperatura, e a variável dependente “y”, como o número de sorvetes. Sendo assim, é necessário escrever “x = df['temperatura'].to_numpy()”, o método “to_numpy” converte os valores do dataframe para o tipo float, e para a variável y escrevemos abaixo “y = df['numero_sorvetes'].to_numpy()”

Para construir esse modelo de Machine Learning, é necessário dividir os dados em duas partes. Uma para treinar o modelo, e outra para testar o modelo com dados que ele não conhece. Para isso, iremos separar 80% dos dados para treino e 20% dos dados para teste. Então, será usada a classe *train_test_split* da biblioteca sklearn, que faz essa separação automaticamente. Em seguida, para importar a biblioteca escreva no código “from sklearn.model_selection import train_test_split”. A classe *train_test_split* recebe as variáveis “x” e “y” como entrada, e retorna quatro variáveis, **x_treino**, **x_teste**, **y_treino**, e **y_teste.** As duas primeiras correspondem aos dados de treino e testes para os dados de temperatura, já as duas últimas correspondem aos dados de treino e testes para o número de sorvetes. Então, para dividir os dados em treino e teste escrevemos em seguida “x_treino, x_teste, y_treino, y_teste = train_test_split(x, y, test_size=0.2)”.

Logo, **x_treino** e **y_treino** são usados para treinar o algoritmo, **x_teste** é a parte usada para o algoritmo fazer as predições depois de treinado, já **y_teste** é a parte das respostas usadas para testar o algoritmo.

Dando continuidade no processo, o modelo de regressão a ser utilizado será o módulo da biblioteca sklearn *LinearRegression,* e será do tipo linear. A variável “*modelo”* é criada ao instanciar o modelo de regressão. Para importar o módulo *LinearRegression*, escreva *“from sklearn.linear_model import LinearRegression” .* Em seguida, para instanciar o modelo, escreva *“modelo = LinearRegression().”.* Logo, a variável modelo irá conter o modelo de regressão linear que foi criado. Para realizar o treinamento, será usado o método *“fit”* desse módulo, usando as variáveis **x_treino** e **y_treino** como entrada. Para isso, deve ser escrito *“modelo.fit(x_treino.reshape(-1, 1), y_treino.reshape(-1, 1))”*, lembrando que o método reshape, que foi usado no código, serve para colocar os dados na forma de um vetor com n linhas e uma coluna.

Após o treinamento do modelo, podemos realizar as previsões da quantidade de sorvetes, definindo a variável “y_previsto”. Será usado o método *predict* da variável modelo, e, passando como entrada do método *predict,* a variável *x_test* representa os dados de temperatura que não

foram utilizados na etapa de treino. Para isso, deve ser escrito “y_previsto = modelo.predict(x_teste.reshape(-1,1))”.

Pronto, foi realizada uma predição, usando um modelo de regressão linear dentro do notebook jupyter. Após isso, é possível fazer um gráfico com o número de sorvetes vendidos e o número de sorvetes previstos pelo modelo apresentado. O Gráfico é gerado da mesma forma que o gráfico anterior, utilizando a biblioteca matplotlib através do comando “plt.plot(range(y_previsto.shape[0]), y_previsto, 'r--')” para plotar o valor previsto, e do comando “plt.plot(range(y_teste.shape[0]), y_teste, 'g--')” para plotar a variável de teste. O comando “plt.legend(['Sorvetes previstos', 'Sorvetes vendidos'])”, cria uma legenda para o gráfico, e os nomes das variáveis no gráfico são criadas pelos comandos “plt.xlabel('índice')” e “plt.ylabel('Sorvetes')”. Já o comando “plt.show()” mostra a figura.

Analisando esse segundo gráfico gerado, é possível perceber a relação Índice x Sorvetes, no eixo x e y do gráfico, respectivamente. O eixo x, na horizontal, traz uma escala com sete valores diferentes: 0.0; 0.5; 1.0; 1.5; 2.0; 2.5; e 3.0. O eixo y, na vertical, mostra uma escala com nove números: 32.5; 35; 37.5; 40; 42.5; 45; 47.5; 50; e 52.5. No centro do gráfico, há duas linhas pontilhadas, uma vermelha e outra verde, representando “Sorvetes previstos” e “Sorvetes vendidos”, respectivamente. A linha vermelha se mostra ascendente e quando chega no valor 42.5 do eixo y, começa a ficar estável. Já a linha verde se mostra ascendente até o valor 52.5 do eixo y, e logo após, começa a cair. A curva obtida com os dados previstos tende a subestimar o número de sorvetes vendidos. Isto pode ser explicado devido o fato que foram utilizados poucos dados para fazer o treinamento e previsão. Em Machine Learning, quanto mais dados são usados para treinamento, melhor o modelo consegue fazer previsões.

Agora que você pôde acompanhar o exemplo e compreender melhor como utilizar um modelo de regressão, deve estar se perguntando, “Como saber se a relação matemática de regressão é linear ou não linear?”.

Bom, uma regressão é dita linear quando apresenta linearidade em seus parâmetros, ou seja, a equação matemática que representa uma reta de regressão pode possuir termos constantes e termos multiplicando as variáveis independentes. Caso contrário, essa relação é dita não linear.

Um exemplo interessante é o caso da equação y = 10 + 5x, que representa uma equação linear, em que x é a variável independente e 10, e 5 são os termos constantes da equação.

Já a equação y = 1/(1 + exp(-x)), é uma equação não linear, pois apresenta a divisão de uma constante por uma soma com um exponencial. Essa equação não linear, que representa uma

curva com formato de **“S”**, é conhecida como função logística, e é utilizada em algoritmos de classificação, quando se deseja categorizar uma variável a partir de variáveis binárias ou classes, fornecendo os resultados em termos de probabilidade.

Dentre as mais variadas aplicações, a regressão logística pode ser utilizada em sistemas de análise de crédito, pois terá como resposta a probabilidade de um cliente ter ou não o crédito aprovado. Pode ser aplicada ainda para avaliar se os novos clientes de uma empresa, por exemplo, pagarão ou não as contas no prazo, de acordo com dados históricos de toda sua base de clientes.

Até aqui, você pode conhecer como modelos de regressão usam a relação entre variáveis dependentes e independentes para fazer previsões, assim como as definições de regressão simples e múltipla. A ideia apresentada aqui é deixar claro como são utilizados esses tipos de regressão e suas aplicações. Lembre-se de aprofundar seus conhecimentos a partir do que você já aprendeu, além de buscar compreender um pouco mais sobre os modelos de regressão que podem ser utilizados em Machine Learning. Bons estudos, resolva os exercícios propostos e até a próxima!