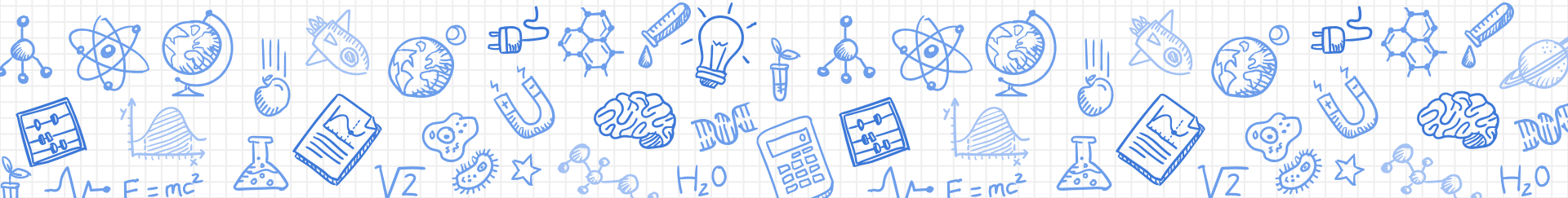
Análise de Dados e Inferência Estatística com Python

Aula de hoje!

- ✘ One-Hot Encoding
- ✘ Análise de correlação

One-Hot Encoding



One-Hot Encoding

Alguns algoritmos conseguem trabalhar diretamente com variáveis categóricas, maioria dos casos dos modelos de árvore de decisão por exemplo. Entretanto, em sua maioria, os modelos necessitam que sejam transformados em valor. E para isso, devemos nos atentar em duas etapas basicamente: Integer Encoding e One-Hot Encoding.

One-Hot Encoding

Cada categoria deverá ser renomeada por um número, exemplo de um modelo para Banco. Precisa classificar entre cliente muito bom, bom, ruim e muito ruim. Logo, terá 4 números (1, 2, 3 e 4), essa transformação é chamada de label encoding ou um integer encoding.

One-Hot Encoding

Contudo, usando label encoding pode resultar uma performance ruim em seu modelo. Nesses casos, devemos one-hot-encoding que significa em transformá-las em variáveis(colunas) e binárias.

One-Hot Encoding

One Hot Encoding: Entenda porque você precisa transformar dados categóricos.

Às vezes o sucesso de um projeto de Data Science está na capacidade de trabalhar nos dados.

Muitas vezes ficamos presos em como um algoritmo de Machine Learning funciona, ou ainda, em como fazer um tunning deste.

Mas, muitas vezes, o que realmente faz a diferença é o pré-processamento dos dados.

Dados Categóricos

Dados categóricos podem ser de tipos nominais ou ordinais.

Um exemplo de variável nominal, seria a variável “sexo”, esta poderia ter valores como “Masculino” ou “Feminino”.

Ou, ainda, a variável “cor”, esta poderia ter valores como “Branco”,”Preto”,”Amarelo”,”Vermelho” e por ai vai..

Variáveis ordinais, seguem uma ordenação entre as categorias, um exemplo seria a variável escolaridade, esta poderia ter valores como “1º,2º,3º graus”

Outro exemplo seria a variável “mês”, esta poderia ter valores como “Janeiro”, “Fevereiro”, “Março” ...

Mas como fazer isso?

Podemos usar o utilitário LabelEncoder da biblioteca scikit-learn para fazer isso.

Basicamente o que esse utilitário faz é converter os valores categórios em valores numéricos.

Por exemplo, imagine que temos uma variável chamada “cor” que recebe a lista de possíveis valores: (‘branco’,’preto’,’amarelo’,’vermelho’).

Este deve nos retornar algo como [0,1,2,3], ou seja, o valor ‘amarelo’ foi transformado em 0, o valor “branco” foi transformado em 1, o valor ‘preto’ transformado em 2 e por fim o valor “vermelho” foi transformado em 3.

Mas como fazer isso?

Vamos aplicar o LabelEncoder na lista de valores, veja:

cor = ['branco','preto','amarelo','amarelo','vermelho','branco','preto','preto']

Podemos ver que a lista de tamanhos tem vários valores repetidos, vamos ver os valores únicos:

set(cor)

O comando set elimina valores repetidos da lista, podemos ver que temos 4 valores distintos de cores.

Em seguida use o LabelEncoder para fazer a codificação e transformar isso em inteiro.

Mas como fazer isso?

O primeiro a fazer é importar o módulo de pré-processamento:

from sklearn.preprocessing import LabelEncoder

Após isso, instancie um objeto do tipo LabelEncoder.

label_encoder = LabelEncoder()

Agora faça a transformação dos dados com o método fit_transform(), veja:

valores_numericos = label_encoder.fit_transform(cor)

A variável valores_numericos recebe a lista de valores já codificados, veja:

valores_numericos

Mas como fazer isso?

```
In [17]: valores_numericos
Out[17]: array([1, 2, 0, 0, 3, 1, 2, 2])
```

Mas como fazer isso?

Veja que interessante, para os 4 valores distintos na lista “cor” o LabelEncoder atribuiu um valor (0,1,2,3).

set(valores_numericos)

É possível inverter a codificação, para isso use o método inverse_transform() e passe o valor codificado, veja:

valor_real = label_encoder.inverse_transform(2)

print(valor_real)

Esse método é importante, pois, as vezes precisamos retornar o valor na sua forma real.

No exemplo acima passei o valor 2 e o retorno foi o valor “preto”.

One Hot Encoding???

O que fizemos com o LabelEncoder foi transformar valores categóricos em valores inteiros, porém, isso pode ser perigoso.

Alguns algoritmos de Machine Learning podem aprender uma informação que não é verdadeira nos valores gerados.

Por exemplo, imagine que os valores da variável valores_numericos fossem features para um classificador, este poderia aprender que 4 é maior que 3 e isso poderia inferir no processo de aprendizado.

Ou ainda, o algoritmo poderia calcular a média desses valores de alguma forma e dar pesos específicos para essa feature.

Esses valores podem prejudicar a eficiência do algoritmo, haja visto, que estes números não significam nada, apenas foram codificados para representar valores categóricos.

One Hot Encoding???

Nossa variável é nominal lembra? Não tem sentido de ordenação entre os valores, concorda?

Mas como podemos resolver esse problema?

Para resolver isso usaremos o One Hot Encoding.

O One Hot Enconding irá gerar vetores binários para cada valor inteiro.

Por exemplo, os valores inteiros [0,1,2] podem ser transformados em vetores como:

[1, 0, 0]

[0, 1, 0]

[0, 0, 1]

Dessa forma resolvemos o problema que o LabelEncoding criou, pois, os vetores binários não significam nada.

Cada valor inteiro, agora é representado por um vetor binário único.

One Hot Encoding???

Veja como implementar o One Hot Encoding.

Primeiramente vamos criar um objeto chamado “onehot_encoder” do tipo OneHotEncoding.

onehot_encoder = OneHotEncoder(sparse=False)

Acima, usamos o parâmetro sparse=False para gerar a matriz de vetores.

Em seguida, vamos transformar a lista de valores em uma lista de 1 dimensão, para então passar para o onehot_encoder.

inteiros = valores_numericos.reshape(len(valores_numericos),1)

Agora passamos os valores para o objeto onehot_encoder fazer a transformação.

vetores_binarios = onehot_encoder.fit_transform(inteiros)

Veja como ficou os dados transformados pelo One Hot Encoding.

One Hot Encoding???

```
In [21]: vetores_binarios

Out[21]: array([[ 0.,  1.,  0.,  0.],
                [ 0.,  0.,  1.,  0.],
                [ 1.,  0.,  0.,  0.],
                [ 1.,  0.,  0.,  0.],
                [ 0.,  0.,  0.,  1.],
                [ 0.,  1.,  0.,  0.],
                [ 0.,  0.,  1.,  0.],
                [ 0.,  0.,  1.,  0.]])
```

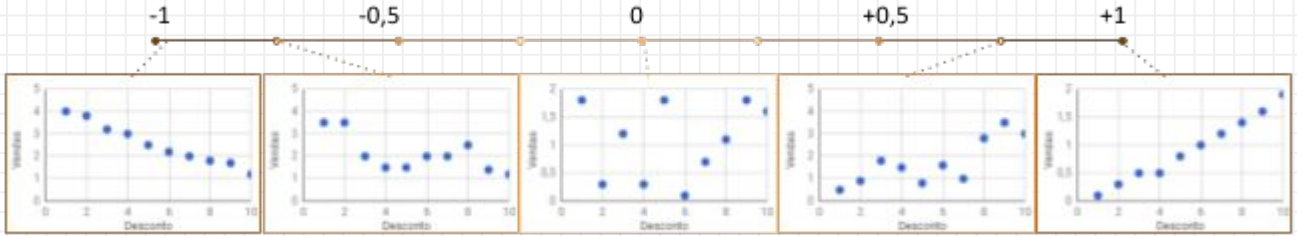
Análise de correlação

É uma análise descritiva que mede se há e qual o grau de dependência entre duas variáveis (desconto e vendas), como no exemplo simplificado à seguir:

- Se o desconto e as vendas aumentam e diminuem quase sempre juntos: há correlação positiva.
- Se as vendas caem quase sempre que o desconto aumenta, ou vice-versa: há correlação negativa.
- Se os aumentos e quedas nos descontos não tem efeito sobre o volume de vendas: não há correlação.

Análise de correlação

O indicativo que diz se há ou não há correlação, ou se ela é positiva ou negativa, é o coeficiente de correlação, que é um número que varia de -1 a +1, como na régua à seguir:



Quanto mais próximo de 1 for o valor do coeficiente, sendo positivo ou negativo, mais forte é a evidência de que há uma relação entre as duas variáveis.

Quando fazer análise de correlação?

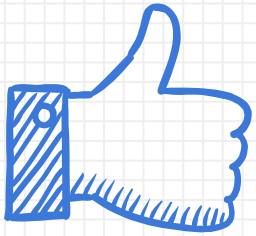
Quando você tem uma hipótese de que o aumento ou queda em uma variável estão associados à evolução de outra variável, por exemplo, se aumentar o desconto, as vendas também aumentam.

Quando fazer análise de correlação?

Quando você tem uma hipótese de que o aumento ou queda em uma variável estão associados à evolução de outra variável, por exemplo, se aumentar o desconto, as vendas também aumentam.

Próxima Aula

Teremos testes de correlação e como fazer em Python.



Obrigado!