Algortimo Multinomial NB

Temos o X(dados) e o Y(marcações), mas e agora? O que faremos com eles? Lembra que no exemplo anterior, nós tinhamos os nossos dados e marcacoes:

dados = [porco1, porco2, porco3, cachorro4, cachorro5, cachorro6]

marcacoes = [1, 1, 1, -1, -1, -1]

O que fazíamos mesmo? Rodávamos o modelo! Então agora vamos importar o algoritmo MultinomialNB:

from dados import carregar\_acessos

X, Y = carregar\_acessos()

from sklearn.naive\_bayes import MultinomialNB

E o que fazíamos com esse algoritmo? Pedíamos para ele criar um modelo e adaptá-lo (fit) com os nossos dados(X), e marcações(Y):

from dados import carregar\_acessos

X, Y = carregar\_acessos()

from sklearn.naive\_bayes import MultinomialNB

modelo = MultinomialNB()

modelo.fit(X, Y)

Agora precisamos testar o nosso modelo! Vamos verificar esse novo usuário:

[1, 0, 1]

Observe que ele entrou na home, não entrou na página de como funciona e entrou na página de contato. E agora? Ele comprou ou não? Vamos pedir para que o nosso modelo preveja (predict) para nós:

from dados import carregar\_acessos

X, Y = carregar\_acessos()

from sklearn.naive\_bayes import MultinomialNB

modelo = MultinomialNB()

modelo.fit(X, Y)

modelo.predict([1,0,1])

Porém, lembra que o predict espera um array de arrays para não mostrar aquele *warning*? Então vamos adicionar um array:

modelo.predict([[1,0,1]])

Por fim, vamos imprimir o predict para verificar o resultado:

from dados import carregar\_acessos

X, Y = carregar\_acessos()

from sklearn.naive\_bayes import MultinomialNB

modelo = MultinomialNB()

modelo.fit(X, Y)

print(modelo.predict([[1,0,1]]))

Testando o nosso algoritmo:

> python classifica\_acessos.py

> [1]

Ele imprimiu 1 então isso significa que esse usuário vai comprar! Estamos acreditando que ele vai comprar de acordo com o nosso modelo. Mas e se tivéssemos mais um usuário que caiu direto na página de como funciona e não entrou em contato ([0,1,0])? Ele simplesmente entrou na página de como funciona e viu apenas quais são os nossos planos, nem chegou a ver quais são os nossos produtos ou serviços, e agora? Esse usuário vai comprar ou não vai comprar? Vamos adicioná-lo ao nosso modelo e pedir para ele prever para nós!

print(modelo.predict([[1,0,1],[0,1,0]]))

Verificando o resultado desse novo usuário:

> python classifica\_acessos.py

> [1 0]

Observe que o nosso modelo está prevendo pra nós que apenas o primeiro usuário vai comprar e o segundo não... Será que apenas esses dois testes já é o suficiente? Com certeza precisamos verificar outros cenários para garantir que está funcionando!

Além de testar apenas 2 casos, podemos testar muito mais, vamos adicionar um terceiro que só acessou apenas a página home([1,0,0]):

print(modelo.predict([[1,0,1],[0,1,0],[1,0,0]]))

E agora? Será que ele compra? Vamos pedir para que o nosso algoritmo preveja:

> python classifica\_acessos.py

> [1 0 0]

Também não compra... Vamos testar mais um caso em que o usuário entra na página home e na página de como funciona, porém não entra na página de contato([1,1,0]):

print(modelo.predict([[1,0,1],[0,1,0],

[1,0,0], [1,1,0]]))

Será que agora ele vai comprar? Vamos verificar o resultado:

> python classifica\_acessos.py

> [1 0 0 0]

Nenhum desses usuários compraram... Provavelmente existe alguma característica em comum entre eles!

Mas e aquele usuário que entra na página home, entra na página de como funciona e também entra na página de contato ([1,1,1]), ele compra ou não?

print(modelo.predict([[1,0,1],[0,1,0],

[1,0,0], [1,1,0], [1,1,1]]))

Vejamos o resultado:

> python classifica\_acessos.py

> [1 0 0 0 0]

Também não compra... Esse usuário acessou todas as páginas, mas, mesmo assim, provavelmente não comprará.

Observe que o modelo está dizendo apenas o que ele acredita! Essa prevenção está boa ou ruim? O algoritmo está chutando bem ou mal? O nível de acerto está alto ou baixo? Como podemos fazer para saber todas essas informações? Podemos testar o nosso modelo da mesma forma que fizemos quando estávamos classificando os porcos e cachorros! Faremos isso no próximo vídeo!