Uma discussão sobre o Algoritmo de Naive Bayes

Daniel Amorim Leite de Almeida, Ricardo Pires Mesquita

Centro Universitário Carioca (UniCarioca)
Rio de Janeiro – RJ – Brasil

Abstract. Artificial intelligence (IA), as we know it today, is present daily more than ever before. Through its integration with smartphones, cameras, operating systems, search queries, and work spaces. It's safe to say that nowadays it's a regular thing to have interactions with an AI. The means by which these artificial intelligence tools work is mathematics. In this article there will be a discussion of the mathematics behind one of the primary theorems of data classification. The focus will be on offering an introduction to the Naive Bayes algorithm, specifically the gaussian model. An application in Python will be presented in conjunction with scikit-learn.

Resumo. A inteligência artificial (IA), como é conhecida hoje, está presente no cotidiano mais do que nunca. Através da sua integração nos smartphones, sistemas operacionais, mecanismos de pesquisa, e nos ambientes de trabalho, é seguro dizer que a interação com IA nos tempos atuais é algo comum. O meio pelo qual estas ferramentas de inteligência artificial funcionam é a matemática. Neste artigo, será discutida a matemática envolvida em um dos principais teoremas de classificação de dados. O foco principal deste artigo é fornecer uma introdução ao algoritmo de *Naive Bayes*, especificamente o modelo gaussiano (distribuição normal). Será apresentada uma aplicação em *Python* em conjunto com *scikit-learn*.

1. Introdução

Antes de tudo, será definido o que é Inteligência Artificial (IA), *Machine Learning* (Aprendizado de Máquina) e onde o algoritmo de ***Naive Bayes*** apresentado aqui é utilizado. Inteligência Artificial é a área da Ciência da Computação responsável pelo desenvolvimento de sistemas que simulam a capacidade humana de resolver problemas. *Machine Learning* é um subconjunto da IA que se baseia no desenvolvimento de sistemas que aprendem, com base nas informações que consomem, utilizando para isso métodos matemáticos [Muller, J. et al 2019]. Aqui, em *Machine Learning,* é onde será trabalhado o algoritmo de *Naive Bayes*.

O teorema de ***Naive Bayes*** é basicamente um método probabilístico que pressupõe uma independência entre os atributos dos dados. É muito utilizado em *Machine Learning* para classificação de dados. Uma vantagem em utilizar esse modelo é que, por supor uma independência entre os atributos, ele é capaz de fazer o treinamento de um modelo com poucas amostras [Harrison. 2019]. Os algoritmos de classificação podem ser encontrados em vários sistemas modernos, tais como: filtragem de spam em e-mails, classificação de textos, sistemas de recomendação, avaliação para crédito financeiro, análise de fraude para tomada de decisões [Bertsch 2015]. Esses algoritmos, como o algoritmo de *Naive*

Bayes, nada mais são que uma técnica de classificação que se baseia no teorema de *Bayes* para supor a independência entre a previsão de algo.

2. O Teorema de Bayes

A princípio, será comentado o conceito de probabilidade. A probabilidade informa as chances que um evento tem de ocorrer, e ela é expressa com um número. Por exemplo, ao lançar uma moeda para cima, a probabilidade de ser cara é a mesma de ser coroa, que neste caso é de cinquenta por cento, uma vez que a moeda possui dois lados e só um resultado é permitido. Se um evento ocorre de **h** formas diferentes em um total de **n** formas possíveis, todas sendo igualmente prováveis, então a probabilidade do evento é **h/n**. Neste caso pode-se expressar na forma: **P(A**), onde se lê a probabilidade do evento **A** ocorrer [Spigel et al. 2012]. Para escrever sobre o algoritmo de ***Naive Bayes*** é necessário primeiro explicar o teorema de *Bayes* que é de onde ele deriva. O princípio matemático do teorema de ***Naive Bayes*** tem sua base na teoria da probabilidade e estatística.

O teorema descreve a probabilidade condicional de um evento, dado a ocorrência de outros eventos. Essa abordagem matemática é essencial para a compreensão e aplicação do teorema na classificação de dados [Faceli et al. 2011]. Pelo teorema de *Bayes* temos que a probabilidade de um evento **A** ocorrer, que pode ser uma classe (por exemplo, uma pessoa ter câncer), dado um evento B, que pode ser o conjunto de valores dos atributos de entrada (por exemplo, ter um resultado positivo em um exame de raios-x), não depende apenas da relação entre **A** e **B**, mas também da probabilidade de observar **A** independentemente de observar **B**. Abaixo temos a fórmula de Bayes:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \quad (2.1)$$

Onde P(B) pode ser expressado como P(B|A)P(A)+P(B|-A)P(-A), reescrevendo a fórmula temos:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B|A)P(A)+P(B|-A)P(-A)} \quad (2.2)$$

- **P(A|B)** é a probabilidade de **A** ocorrer dado que **B** ocorreu.
- **P(B|A)** representa a probabilidade de um evento **B** ocorrer dado que um evento **A** já aconteceu. Em muitos casos, o evento **A** é a variável que já possuímos conhecimento prévio.
- **P(A)** é a probabilidade de **A** ocorrer.
- **P(B)** é a probabilidade de **B** ocorrer.

Naive Bayes tem como sua característica uma suposição "ingênua" de independência condicional entre as características, dada a classe e de que os dados possuem uma distribuição normal ou gaussiana. O algoritmo pode ser generalizado nas seguintes etapas abaixo:

- Aprende as probabilidades ligando as caraterísticas a cada classe possível.
- Multiplica todas as probabilidades relacionadas a cada classe resultante.

- Normaliza as probabilidades, dividindo cada uma delas por sua soma total.
- Pega como resposta a classe que representa a probabilidade mais alta.

[Muller, J. et al 2019].

3. O Algoritmo de *Naive Bayes*

3.1. Como o algoritmo funciona

Neste estudo de caso será utilizado o modelo gaussiano de ***Naive Bayes***. Esse método assume que cada classe segue uma distribuição normal. A primeira coisa que o algoritmo faz é calcular de uma amostra o desvio padrão e a média de cada classe. Como por exemplo, em dois grupos de pacientes onde um grupo tem câncer e o outro é saudável. Se forem calculados a média e o desvio padrão do PSA(*Prostate-Specific Antigens* ou Antígeno Prostático Específico) de cada grupo podemos determinar os pesos(o quanto cada ponto infere no resultado probabilístico) e montar as duas curvas gaussianas, que são expressadas por essa função:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (3.1)$$

onde **x** corresponde a cada valor PSA, "μ" corresponde a média e "σ" o desvio padrão. Após pegar a média e o desvio padrão de cada classe, podemos determinar o valor de corte que indica a P($x_1|B$). No nosso exemplo seria a P(Saudavel | PSA) e P(Cancer | PSA) [Larson, R. et al. 2023]. A figura 1 foi retirada de um vídeo[1] e modificada para fins de demonstração.

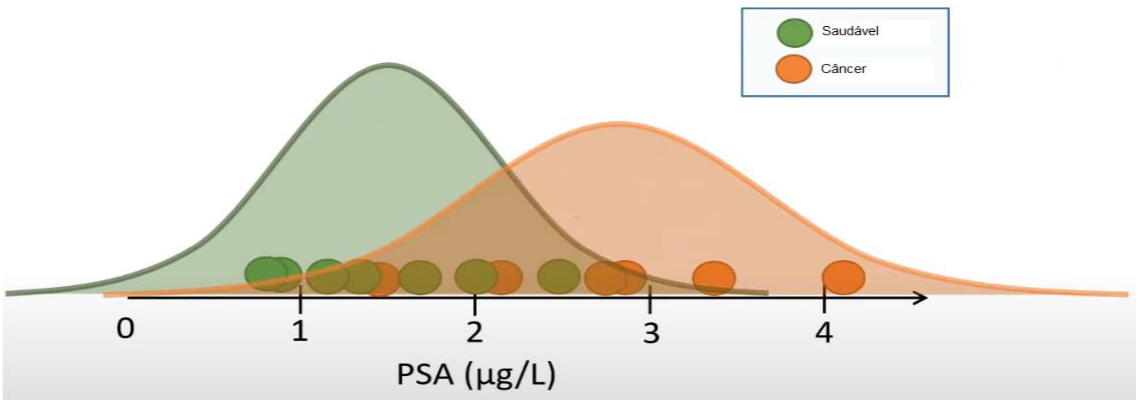

Figura 1. Um exemplo de duas curvas de distribuição normal, onde quanto mais a direita maior o PSA [Inspirado em TileStats, 2022]

Note que na figura 1, o valor de corte seria o ponto de interseção mais alto entre as duas curvas gaussianas, "saudável" e "câncer". É importante entender que, a imprecisão desse método está na interseção das áreas das curvas, no nosso exemplo, quanto mais perto o PSA estiver do valor de corte mais impreciso vai ser a previsão.

[1] https://www.youtube.com/watch?v=yRzIyWVEaCQ

O processo se repete se adicionarmos mais variáveis à expressão, bastando somente multiplicar os resultados da expressão Bayesiana aplicada a essa nova variável com os resultados das outras. Neste exemplo, ao acrescentar a variável "idade" tudo o que é preciso fazer é repetir a fórmula Bayesiana para idade como foi feito para o PSA, multiplicando o resultados dos eventos: P(câncer | PSA), P(câncer | idade) e os comparando com a multiplicação dos eventos: P(saudável | PSA), P(saudável | idade). O evento com a maior probabilidade vai ser o escolhido.

3.2. O problema da falta de dados

Na fórmula bayesiana, existe algo que precisa ser esclarecido. Quando se deseja inferir para o qual não há dados numa tabela amostra de teste, como no exemplo anterior, se houvesse um atributo binário chamado "instável" (onde ele só pode ser "sim" ou "não") e todos os pacientes estivessem instáveis, isso significa que, quando a fórmula for aplicada para pacientes que não são instáveis, os resultados terão o valor zero. No exemplo, em algum momento a situação seria que P(Saudável | não) = 0 , e, por se tratar de multiplicações, toda a equação igualaria a zero.

Para resolver este problema será utilizado o método de Laplace. Aplicando diretamente ao exemplo, consiste em somar o valor 1 ao numerador na fórmula bayesiana, e 2 ao denominador, pois dois é o número de possibilidades (sim e não) em "instável". É necessário fazer essa soma toda vez que se calcula o "sim" e o "não" para que não haja um viés para "não" [Harrison. 2019].

4. Implementação e uso do algoritmo

Nesta implementação, será utilizada uma base de dados pequena para fins de demonstração. Por meio do algoritmo ***Naive Bayes*** junto a uma base de dados que contêm simples dados de motoristas, será determinado se um motorista tem um risco alto, baixo ou moderado de se acidentar. O princípio pode ser aplicado a qualquer base de dados devidamente filtrada e preparada para tal propósito. Vale notar que a presença de dados nulos não será abordada porque a correção laplaciana, que em sua parte envolve registros nulos, é algo que a biblioteca ***sklearn*** lida de forma automática [Harrison. 2019].

```
import pandas as pd
from sklearn.naive_bayes import GaussianNB
```

Figura 2. Bibliotecas utilizadas neste código

Como pode ser visto na figura 2, neste código é utilizado as bibliotecas ***pandas*** e ***GaussianNB*** importado do ***sklearn***, que é um aglomerado de ferramentas para análise de aprendizado de máquina. Os dados que serão usados para treinamento e seus significados estão aqui listados como:

- **"historico"**: representando o histórico de direção, ou seja, quantas vezes o motorista esteve em um acidente, que é separado por três tipos, **bom**, **ruim** e **desconhecido**.

- **"tdirigindo"**: representando o tempo que o motorista possui sua habilitação, esse atributo somente possui dois tipos, "**pouco**" e "**muito**".

- **“vistoria”**: que representa um outro atributo binário indicando se ele passou por uma vistoria indicado por, “**nenhuma**” e “**adequada**”.

- **“kmedio”**: representando a média em quilômetros diários e que é representado por três tipos, **5km_40Km**, **40km_100km** e **acima_100km**.

- **“risco”**: que é o classificador que queremos prever que pode ser denominado por “**alto**”, “**moderado**” e “**baixo**”, onde “**alto**” representa uma boa chance do motorista se acidentar.

4.1. Preparação do Conjunto de Dados

A preparação dos dados é de muita importância para os fins desta tarefa, será usado um arquivo do tipo *csv* (um arquivo de dados separados por vírgula) onde nele estará somente os dados necessários para a implementação da tarefa, os dados ordenados em questão são os atributos previsores e a classe que queremos prever. Esses atributos estão escritos para o entendimento do programa e para execução da tarefa.

Os dados podem ser apresentados de uma forma mais descritiva, porém o propósito é a preparação deles para a tarefa de treinamento no código. Neste caso estão ordenados respectivamente como: “**historico**”; “**tdirigindo**”; “**vistoria**”; “**kmedio**” e “**risco**”. Este arquivo *csv* possui quatorze linhas no total onde cada linha representa todos os dados de um motorista. A seguir na tabela 1, pode-se observar uma representação dos dados que estão no arquivo *csv*.

Base de Dados				
Coluna 0	Coluna 1	Coluna 2	Coluna 3	Coluna 4
historico	tdirigindo	vistoria	kmedio	risco
ruim	pouco	nenhuma	acima_100km	**alto**
desconhecido	pouco	nenhuma	40km_100km	**alto**
desconhecido	muito	nenhuma	40km_100km	**moderado**
desconhecido	muito	nenhuma	5km_40Km	**alto**
desconhecido	muito	nenhuma	5km_40Km	**baixo**
desconhecido	muito	nenhuma	5km_40Km	**baixo**
ruim	muito	nenhuma	acima_100km	**alto**
ruim	muito	adequada	5km_40Km	**moderad**o
bom	muito	nenhuma	5km_40Km	**baixo**
bom	pouco	adequada	5km_40Km	**baixo**
bom	pouco	nenhuma	acima_100km	**alto**
bom	pouco	nenhuma	40km_100km	**moderado**
bom	pouco	nenhuma	5km_40Km	**baixo**
ruim	pouco	nenhuma	40km_100km	**alto**

Tabela 1. Representando cada motorista com seus dados, a tabela que será usada para o treinamento.

4.2. Separação e transformação dos dados

No conjunto de dados apresentados é necessário separar os atributos e as classes, neste caso seria separar tudo que engloba o "risco" com os demais dados. É realizada uma atribuição dos dados da coluna 0 a coluna 3 a uma variável, designada como "X", (tudo é passado a esta variável exceto os nomes das colunas), fazendo o mesmo para uma variável denominada "Y", porém somente a aos dados que estão na coluna 4.

```
X_risco_acidente = base_risco_acidente.iloc[:, 0:4].values

y_risco_acidente = base_risco_acidente.iloc[:, 4].values
```

Figura 3. Código que separa os atributos das classes

‘ Com os dados separados em vetores, “X” e “Y”, é necessário transformar os nomes, que estão categoricamente organizados nas colunas, em valores numéricos. Caso contrário a biblioteca e o algoritmo retornaram erros, pois elas foram preparadas para trabalhar com números. Para isso, será importado do ***sklearn*** o ***Label Encoder*** que permite essa transformação. Em seguida, cada coluna, “historico”, “tdirigindo”, “vistoria” e “kmedio” deve ser instanciada com a atribuição do objeto do tipo ***Label Encoder***.

```
#transformo os atributos categoricos em numericos
from sklearn.preprocessing import LabelEncoder
label_encoder_historico = LabelEncoder()
label_encoder_tdirigindo = LabelEncoder()
label_encoder_vistoria = LabelEncoder()
label_encoder_kmedio = LabelEncoder()

#atribuindo as transformações na matriz
X_risco_acidente[:, 0] = label_encoder_historico.fit_transform(X_risco_acidente[:, 0])
X_risco_acidente[:, 1] = label_encoder_tdirigindo.fit_transform(X_risco_acidente[:, 1])
X_risco_acidente[:, 2] = label_encoder_vistoria.fit_transform(X_risco_acidente[:, 2])
X_risco_acidente[:, 3] = label_encoder_kmedio.fit_transform(X_risco_acidente[:, 3])
```

Figura 4. Código que faz a transformação, e o código que atribui a matriz “X”

Como pode ser visto na figura 4 acima, usando a função ***fit transform*** para cada uma das colunas do vetor “X” é possível transformar cada um dos nomes das colunas em valores numéricos. Agora que o vetor é somente numérico, fica mais difícil a compreensão e visualização dele. Isso fica evidente na figura 5 abaixo, é por esse motivo a necessidade da tabela original para a identificação dos valores utilizando a posição deles através do índice. Para elucidar esta transformação, o valor correspondente ao número 2 na figura 5 abaixo corresponde a palavra “ruim” na tabela referente a coluna de “historico” pois está na mesma posição, isso significa que todos os valores com número 2 nesta coluna representam a palavra “ruim”.

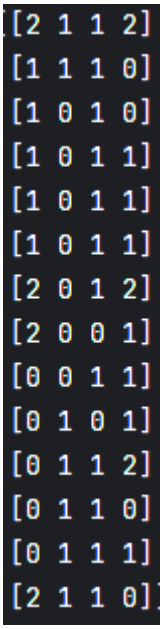

Figura 5. A variável "X" representando os atributos em forma numérica após a transformação

4.3. Treinamento e previsões

Antes de fazer o treinamento, será realizada uma gravação em um arquivo tipo ***pkl*** usando a biblioteca ***pickle*** para que não haja a necessidade de instanciar todos os objetos em "X" e "Y". Isso pode ser feito utilizando a função ***pickle.dump*** no arquivo que será nomeado "risco_acidente.pkl". Para o treinamento, é criada uma variável chamada "naive_risco_acidente" e logo em seguida é atribuído o objeto ***GaussianNB***, que foi importado da biblioteca ***sklearn***, a variável em questão. O algoritmo já faz a distribuição normal ou Gaussiana para os valores atribuídos a variável e é de fato o algoritmo ***Naive Bayes***. Em seguida é necessário usar a função ***fit*** para inserir os valores em "naive_risco_credito", de tal forma que, ao fazer isso o algoritmo criará a tabela de probabilidade internamente e estará essencialmente treinado para tentar prever futuras adições, que neste caso seriam novos motoristas com novos dados.

Para fazer previsões basta usar a função do ***sklearn*** chamada ***predict*** com o objeto "naive_risco_credito" e inserir os parâmetros desejados em forma numérica. Como exemplo, serão utilizados os seguintes parâmetros: "historico desconhecido"; "tdirigindo muito"; "vistoria adequada"; "kmedio 40km_100km". É importante lembrar que, para utilizar esses parâmetros é necessário por eles em sua forma numérica, que ao observar a tabela original e a comparar com a matriz numérica e seus índices, pode-se concluir que neste caso os parâmetros mencionados em forma numérica seriam "1,1,0,0". Para fins de demonstração, também será inserido mais um parâmetro de previsão "2,0,1,0" como pode-se observar na figura 6.

```
naive_risco_acidente.predict([[1,1,0,0],[2,0,1,0]])
```

Figura 6. Função *predict* da biblioteca *sklearn*

Ao executar essa função será retornado "moderado", que é a previsão de "risco" para o primeiro parâmetro, e "alto" que é a previsão de risco para o segundo parâmetro.

5. Conclusão

A Partir deste teste é possível concluir duas coisas: (i) o tempo da previsão é menor que o tempo de treinamento, que é o esperado pois exige menos recursos computacionais, (ii) e o G***aussianNB*** fez a distribuição correta pois dessa forma identificamos as probabilidades de riscos de acidentes em cada suposição. A partir desse modelo, um motorista com um histórico desconhecido, que dirige a muito tempo, tem uma vistoria em dia (aqui no caso representada por "adequada") e que tem uma média de 40 a 100 quilômetros diários tem uma boa chance de apresentar um "risco moderado" de se acidentar no trânsito ao conduzir seu veículo. Neste artigo, também consegue concluir que é possível fazer previsões com poucos dados utilizando *Naive Bayes*.

5.1. Trabalhos Futuros

Existem vários casos onde o *Naive Bayes* é um dos melhores algoritmos para solucionar o problema. Uma das aplicações que usa o algoritmo discutido, é a filtragem de emails classificando e separando com rótulos diferentes como: spam, anúncio, importante, entre outros. É possível determinar isso pelas palavras contidas no email, algumas palavras chave podem ser usadas como classificadores para determinar o tipo do email. Outra aplicação é na área da saúde, a prevenção de doenças como o câncer de mama pode ser possível usando *Naive Bayes*. Se usarmos os dados disponíveis como o *Breast Cancer Wisconsin (Diagnostic) Data Set* que é um conjunto de dados de pacientes com câncer de mama, é possível prever a doença de pacientes que não adoeceram ainda e tomar as medidas possíveis antes que seja tarde demais para o paciente. Seria interessante a utilização deste algoritmo para o desenvolvimento de um software de análise de quedas de aeronaves ou de barcos perdidos em alto mar, a fim de prever a localização dos tripulantes e do veículo. Por não ser uma ocorrência comum, às vezes poucos dados de acidentes estão disponíveis para serem utilizados como inferência para a previsões, por esse motivo que *Naive Bayes* talvez seja a solução para esse problema.

A eficácia de um algoritmo que consegue inferir resultados dentro da margem de erro varia da tarefa e do problema a ser solucionado. *Naive Bayes* não é a única técnica de aprendizado supervisionado e certamente não é a melhor em todas as ocasiões. Técnicas como *K-Nearest Neighbors* (K vizinhos mais próximos), não requer a distribuição normal dos dados como *Naive Bayes,* mas é mais lenta em grandes conjuntos de dados. A SVM (máquinas de vetores de suporte) é muito boa para problemas que demandam mais precisão, também é mais eficiente para trabalhar com dados de alta dimensão (dados densos, que dependem de muitos outros dados). Vale mencionar a técnica de árvores de decisão, que se torna eficiente ao capturar as interações entre os atributos, um modelo de decisão baseado em características. A comparação entre estas técnicas e *Naive Bayes* é uma boa proposta para resolução dos problemas apresentados, entre outros problemas demandados em todas as outras áreas da ciência.

Referências

Bertsch, M. S. (2015). **A teoria que não morreria: como a lei Bayes decifrou o código enigma, perseguiu submarinos russos e emergiu triunfante de dois séculos de controvérsias**, Perspectiva S.A., pag 479

Faceli, K., Lorena, A. C., Gama, J. e Carvalho, C. P. L. F. A. (2011). **Inteligência Artificial: Uma Abordagem de Aprendizado de Máquina, Página** 73-82.

Harrison, M. (2019) Machine Learning – **Guia de referência Rápida: Trabalhando com dados estruturados em Python**, página 107-109.

Larson, R. e Farber, B. (2023). **Estatística Aplicada: Retratando O Mundo** - Pearson Bookman, páginas 233-259.

Muller, J.P., Massaron, L. e Tortello, J.E.N. (2019) **Aprendizado de Máquina Para Leigos**, página 19-211.

Paulino, C. D. Murteira, M. A. A. T. B. e Silva, G. (2018). **Estatística Bayesiana - Fundação Calouste Gulbenkian**, 216-221.

Spigel, Murray R., Schiller, J.J., Srinivasan, R.A. e Viali, L. (2012). **Probabilidade e Estatística** - Coleção Schaum, página 5.

TileStats (2022) ***Gaussian naive Bayes - explained with a simple example***. Disponível em <https://www.youtube.com/watch?v=yRzIyWVEaCQ>, Acesso em: 21 de maio. 2024.