**TEMPLATE: LIVRO-TEXTO**

**Unidade I**. Análises e cálculos para avaliar desempenhos

Ana Luiza Cerchiari de Andrade

### Objetivos da unidade

* Aprender a fazer cortes em sensibilidade e especificidade;
* Ter introdução a NumPy;
* Aprender a fazer cálculos de probabilidades;
* Conhecer estratégias de mineração de textos e TDIDF;
* Compreender desvio padrão e valor esperado.

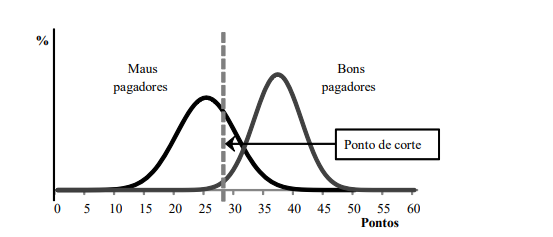
### Tópicos de estudo

* Visualização do desempenho do modelo;
  + Cortes em curva Roc;
  + Armazenamento de dados em NumPy para aplicar análises;
* Evidências e probabilidades;
  + Localizando evidências na prática;
* Representação e mineração de textos;
  + IDF e TDIDF em *bag of words*;
  + Cálculos de IDF e TDIDF para ponderação de TF;
* Engenharia analítica;
  + Desvio padrão;
  + Criação de um robô captando interesse através de dados.

## Visualização do desempenho do modelo

O objetivo, nessa etapa, é definir um ponto de corte para análises de resultados. O Gráfico 1 é usado como exemplo, e representa um ponto de corte no qual existe um limite mediano entre bons pagadores e maus pagadores. Para isso, teremos aplicação prática em SPSS para criar esse limite.

**Gráfico 1. Ponto de corte**



Fonte: BORGES, et al. 2017, p. 273. (Adaptado)

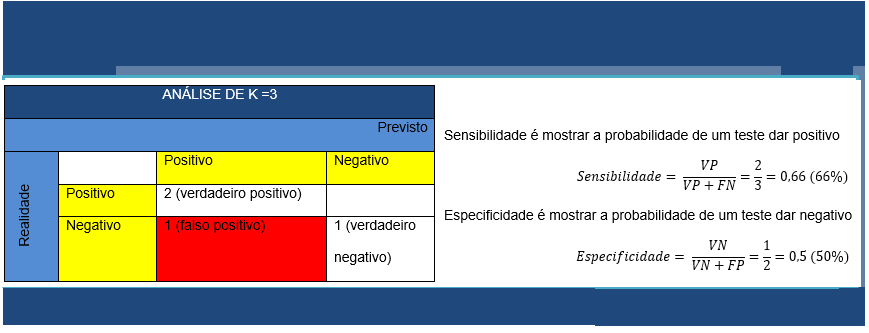
Após isso, teremos mais conceitos de programação em Pyhton com NumPy, a fim de trazer mais contato com a linguagem.

### Cortes em curva Roc

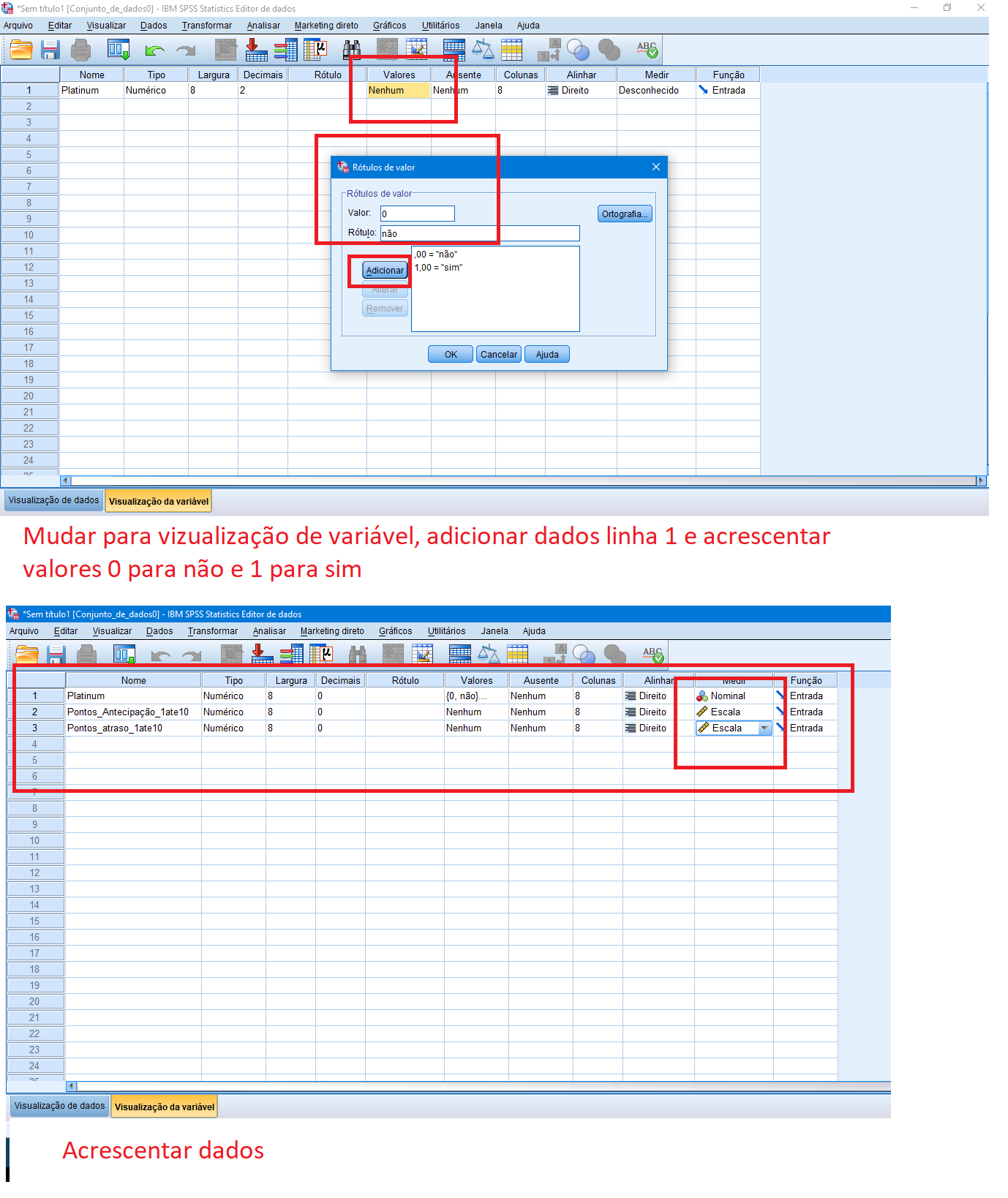
A curva Roc (receptor de operação) foi criada em torno de 1950 e era usada para sinais em radares, mas evoluiu para aplicações em medicina, para lidar com diagnósticos. A proposta de entender a curva Roc e o corte para, consequentemente, obter mais certezas, será realizada no SPSS.

Para fazer a curva Roc é útil lembrar os conceitos de matriz de confusão e análise de sensibilidade. Após realizar uma análise com treinamento de vizinhos e aplicar em um teste, pode-se fazer a matriz de confusão para descobrir a sensibilidade e a especificidade, conforme Diagrama 1.

**Diagrama 1. Relembrando matriz de confusão**



Agora, vamos fazer um exercício voltado para compreensão de corte em SPSS. A Fig. 1 mostra a criação de variáveis no SPSS. Comece clicando no canto inferior da esquerda, em **visualização de variáveis**.

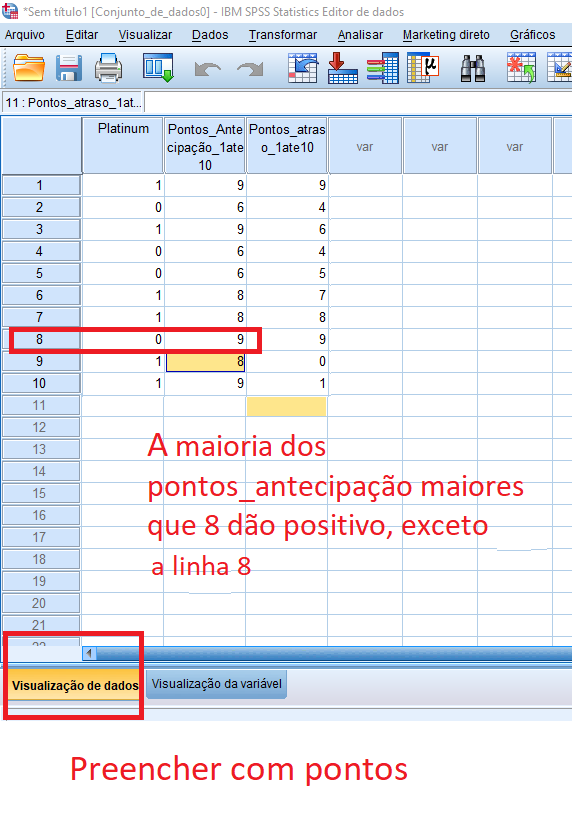


**Figura 1**. Inserindo variáveis.

[interno: trocar “visualização” para “visualização” e

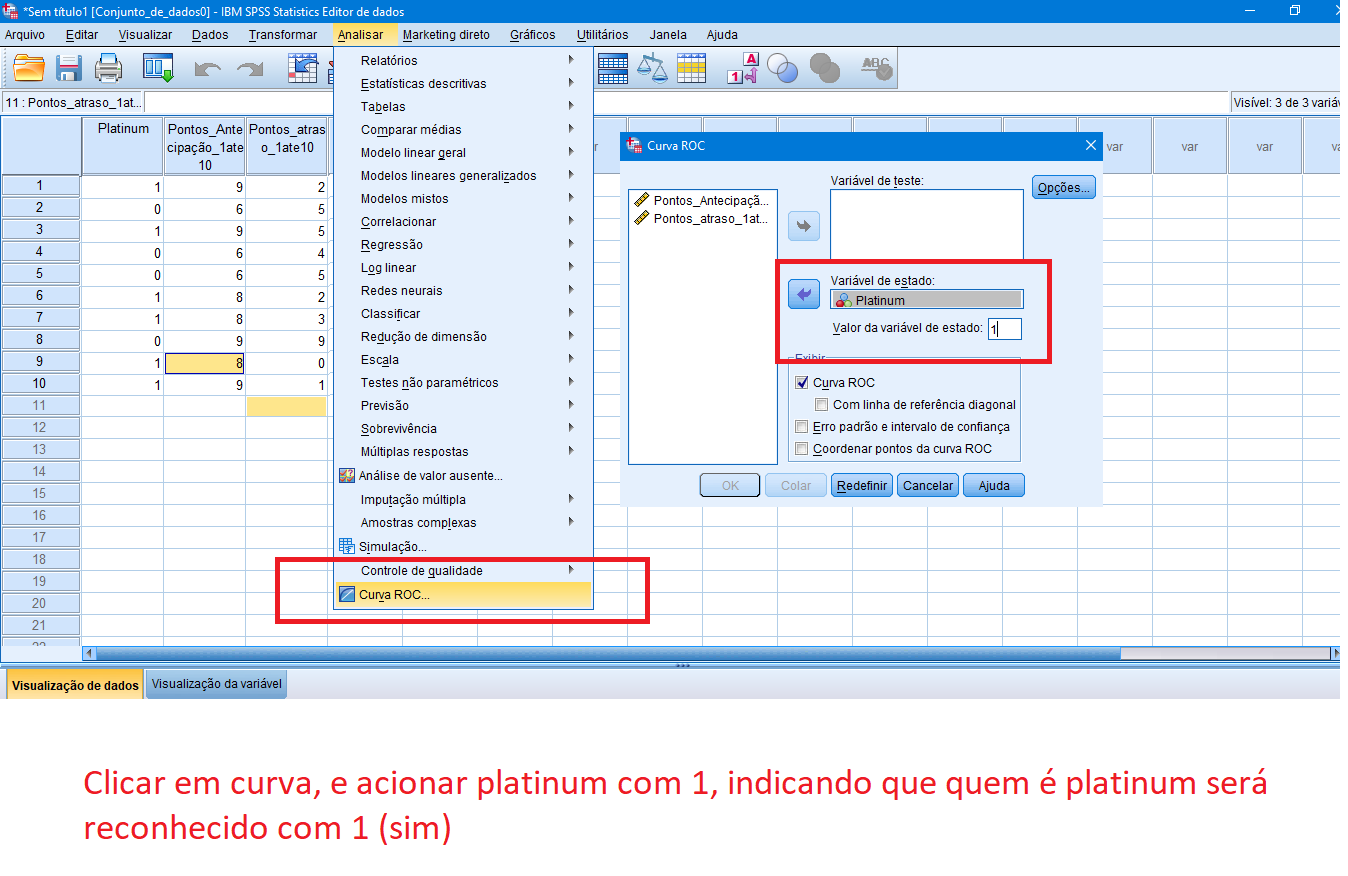
“dados linha 1” para “dados na linha 1”]

Conforme a Fig. 2, insira as variáveis, sendo Platinum a saída, pontos\_antecipação tendo uma relação de quanto maior, melhor, e pontos\_atraso, quanto menor, melhor, ou mais chance de ser Platinum.



**Figura 2**. Inserindo variáveis.

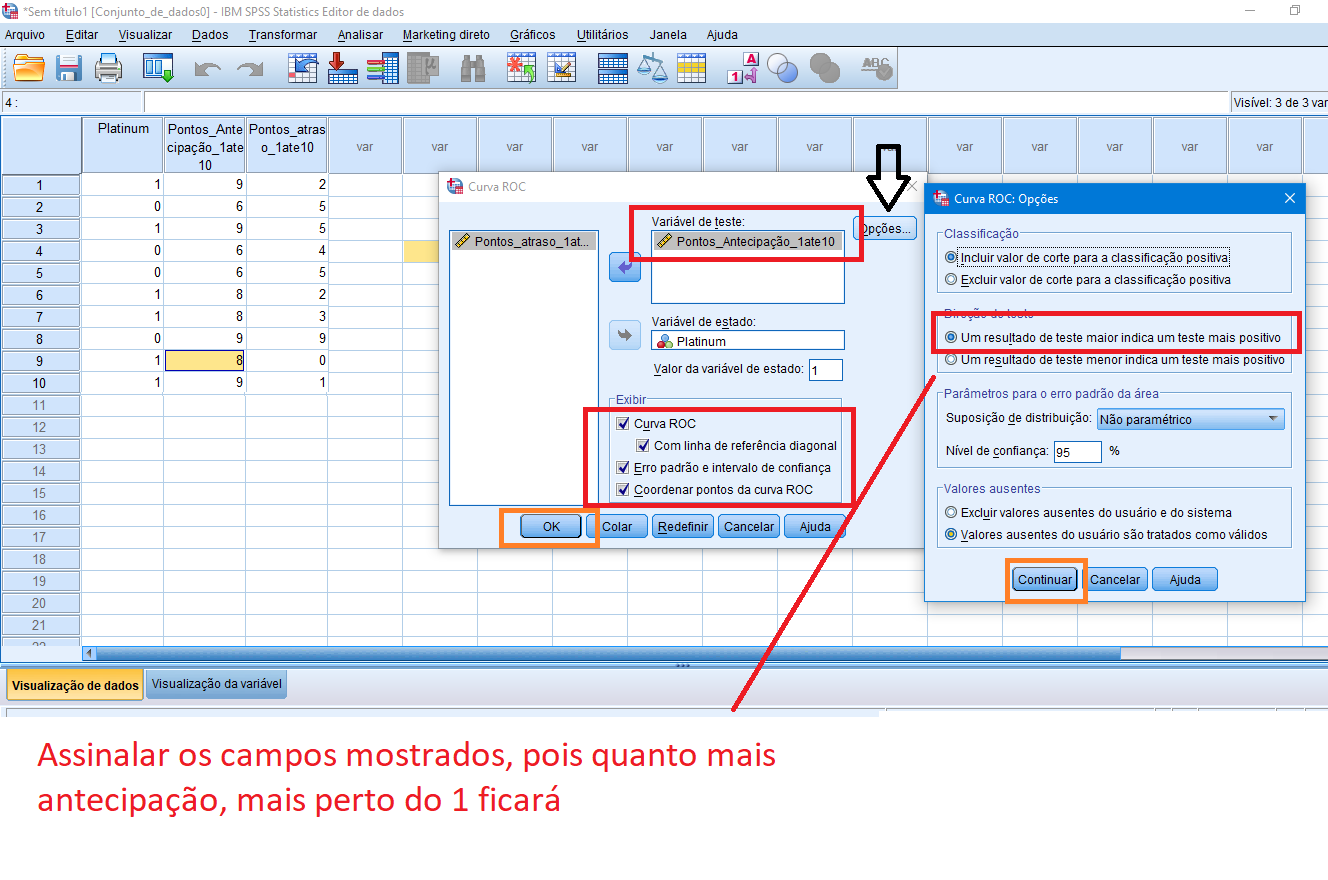
A seguir, criaremos um gráfico Roc e destinaremos Platinum como saída, sendo 1 igual a sim.



**Figura 3**. Destinação de Platinum.

[interno: mudar “platinum” para “Platinum”.]

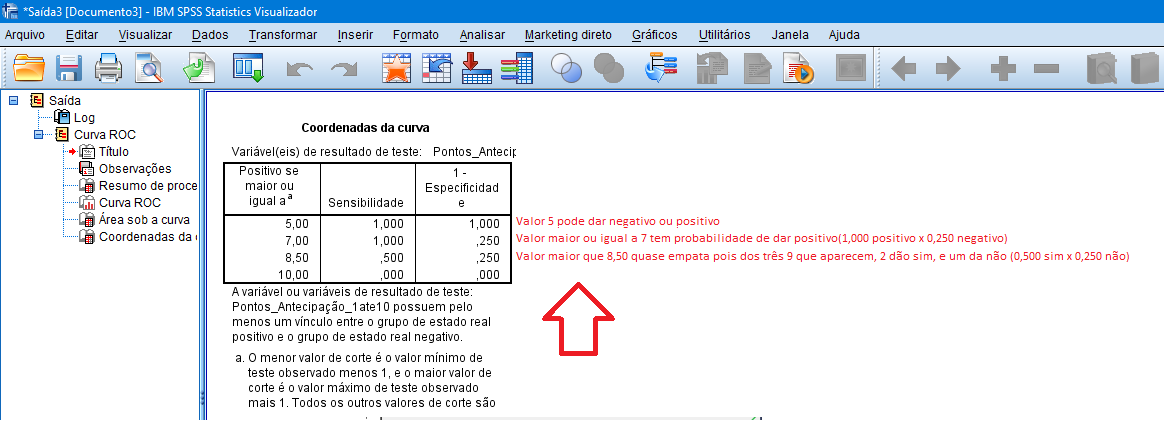
Pontos\_antecipação segue a lógica de quanto maior, menor, e irá se tornar uma entrada na Fig. 4.



**Figura 4**. Colocando pontos\_antecipação como entrada.

O próprio programa já faz uma estimativa de sensibilidade e especificidade, conforme os dados da linha estejam ou não fazendo sentido. Por exemplo, a linha 9 fugia da lógica, pois possuía 9 pontos\_antecipação e não tornou o cliente Platinum. Isso gerou uma confusão, e por isso a figura mostra que valores maiores que 8,5 tem sensibilidade (probabilidade de dar positivo) 50% (0,500) e especificidade 9 (probabilidade de dar negativo) de 25% (0,250).

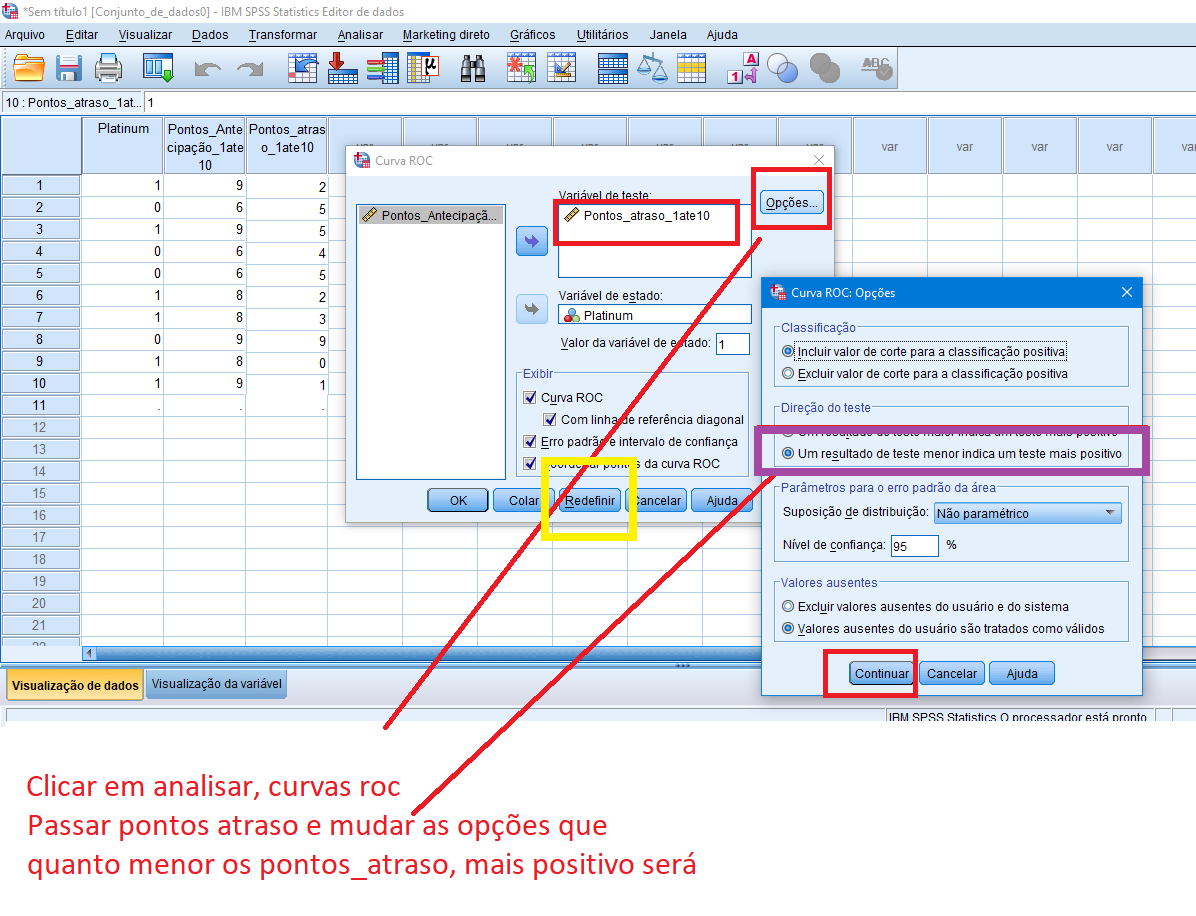
A seguir, será gerado um relatório com tabelas que mostrarão dados de sensibilidade e especificidade. A explicação está na própria imagem.



**Figura 5.** Sensibilidade e especificidade de pontos\_antecipação.

[interno: “positivo(1,000 positivo)” para “positivo (1,000 positivo)” e “quase empata pois” para “quase empata, pois”]

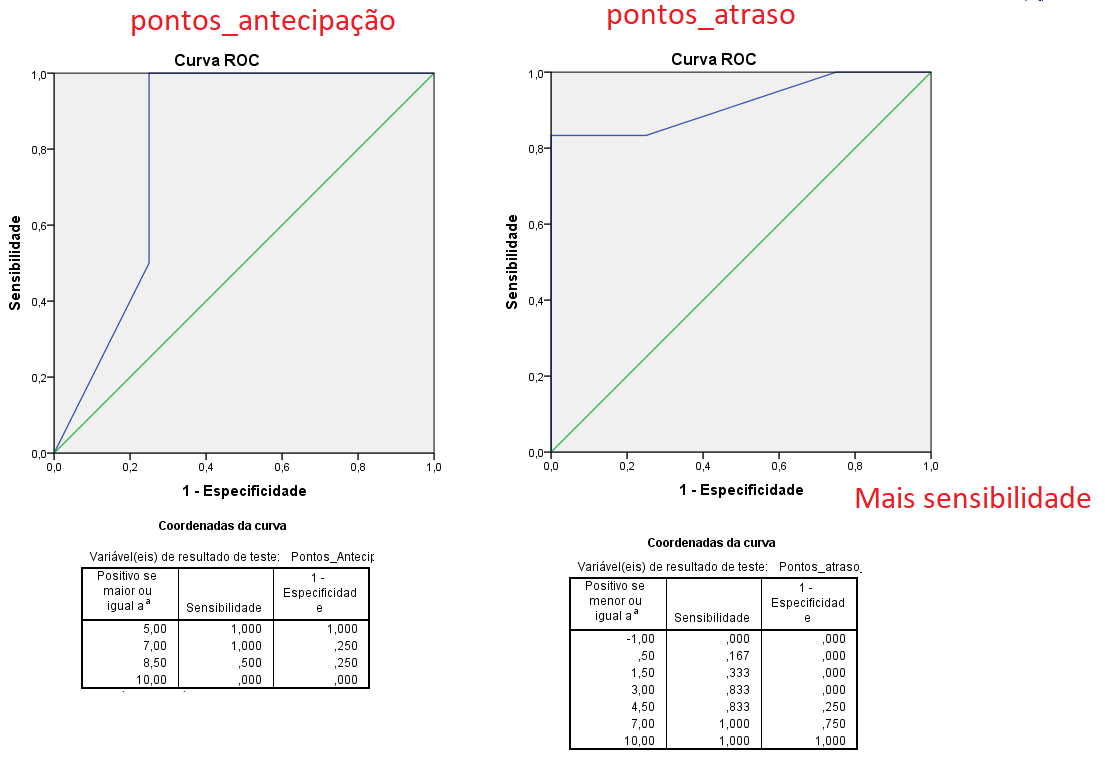
Agora iremos fazer o mesmo procedimento com pontos\_atraso, conforme Fig. 6. Porém, observe a notação em laranja de quanto menor, mais probabilidade de dar positivo, pois são pontos em atraso. Clique em redefinir, conforme a linha amarela.



**Figura 6**. Gerando relatório de pontos\_atraso.

[interno: “curvas roc” para “curvas Roc”, “pontos atraso” para “pontos\_atraso” e “opções que quanto maior” para “opções, pois quanto maior”]

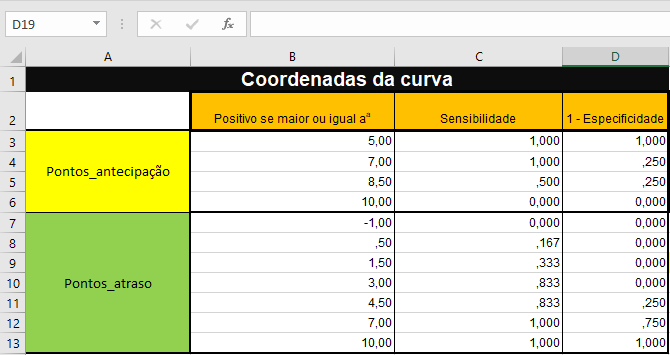
Será gerado um relatório primeiro com dados de pontos\_antecipação e depois com pontos\_atraso, conforme Fig. 7, que demonstra uma comparação, e nas tabelas abaixo do gráfico, mostra a probabilidade de dar positivo (sensibilidade) ou negativo (especificidade) por faixas de valor para cada linha.



**Figura 7**. Comparação de resultados.

Agora veremos como fazer um corte. Há um ponto de intersecção em que a probabilidade de dar sim é alta. E para isso, faremos alguns cálculos, que traçam um ponto seguro de dar sim nas duas colunas, ao mesmo tempo (pontos\_atraso e pontos\_antecipação).

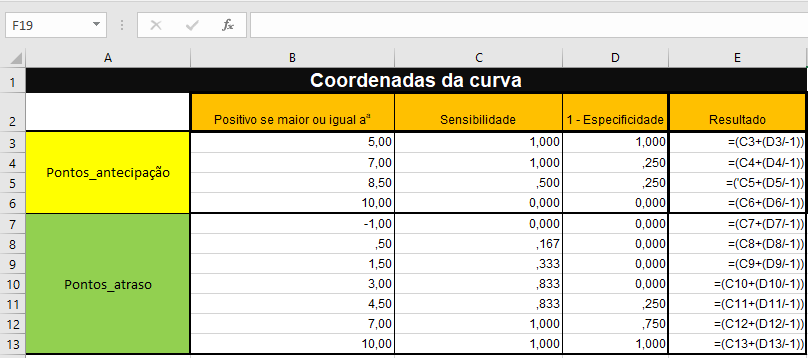
A primeira coisa a fazer é passar as tabelas da Fig. 7 para o Excel, conforme Fig. 8.



**Figura 8.** Passando coordenadas da curva.

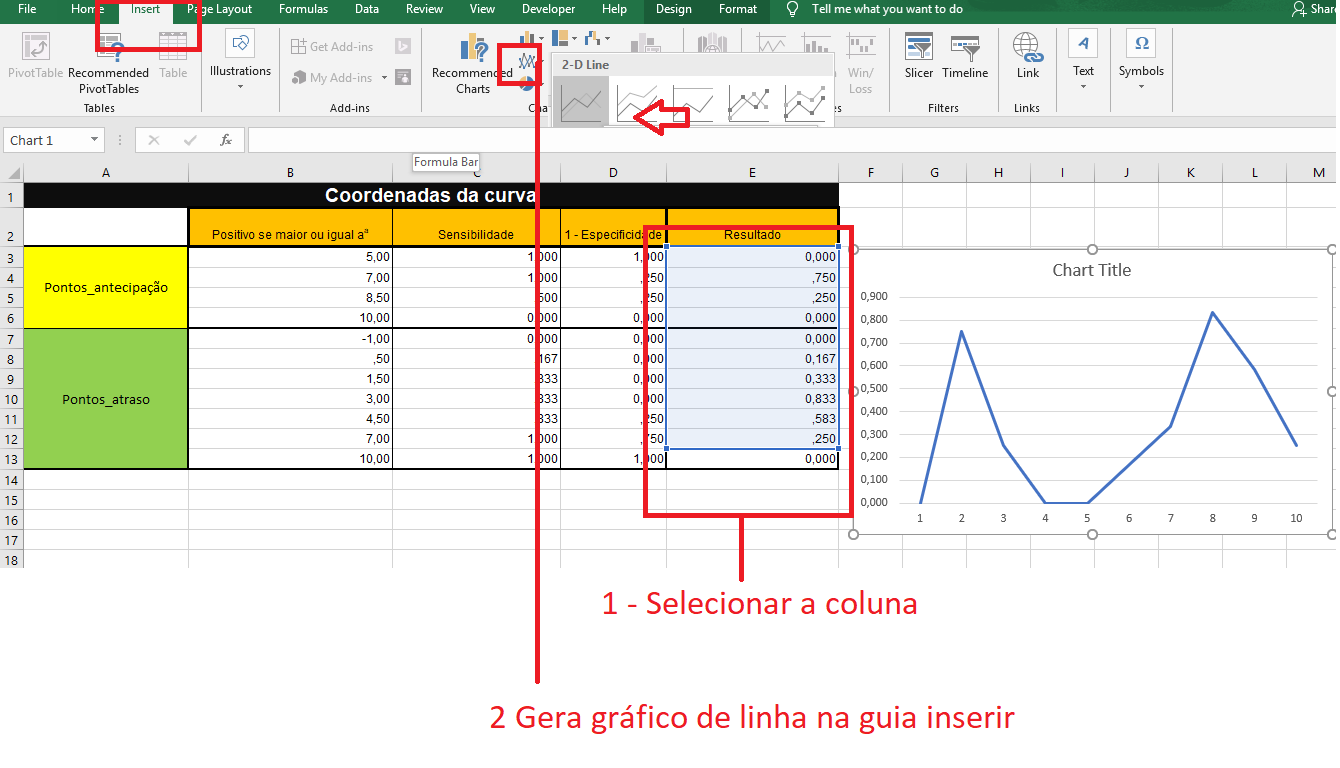
Após ter feito isso, iremos fazer um cálculo para descobrir ponto de corte, cuja fórmula é: )

Observe, na Fig. 9, que ao invés de usar o número, usou-se C3, C4, C5, D3, D4 etc. Esses são endereços no Excel para agilizar o cálculo.



**Figura 9.** Executando cálculos para gerar resultados que levam a linha de corte.

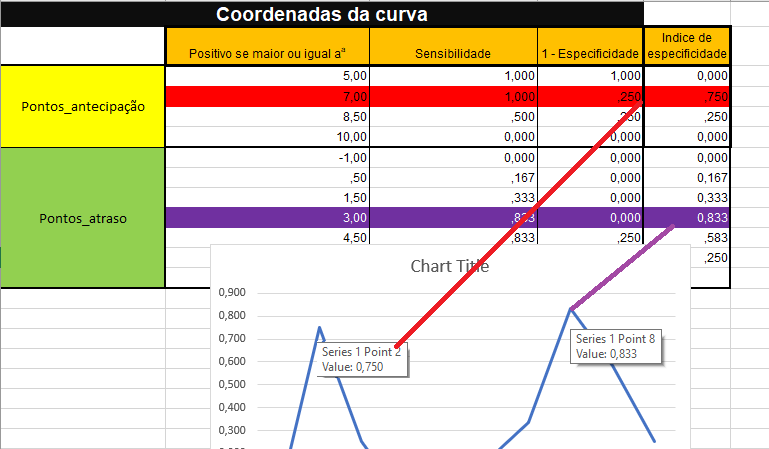
O próximo passo é gerar um gráfico com base nos resultados, conforme Fig. 10.



**Figura 10**. Gerar gráfico.

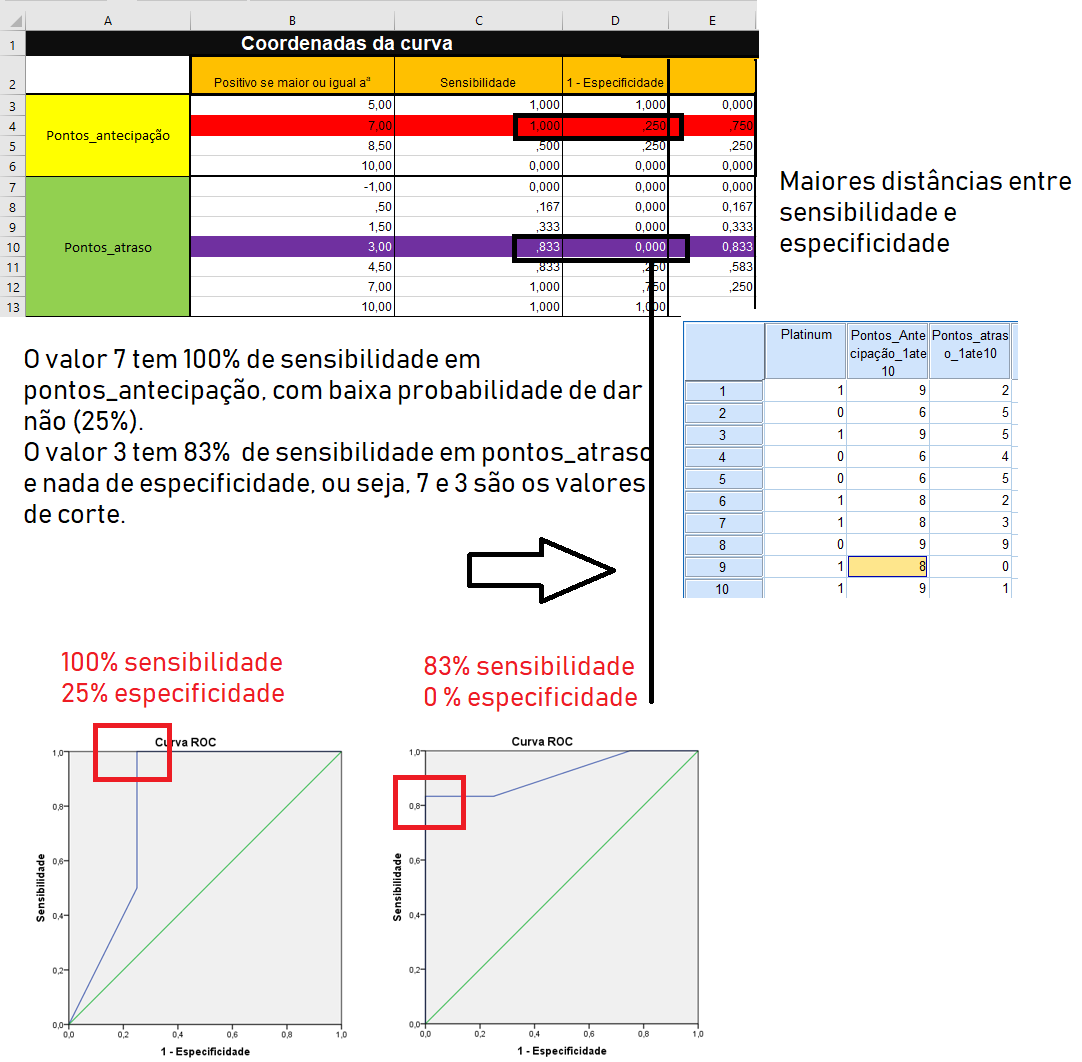
[interno: “2 Gera gráfico” para “ 2 - Gerar gráfico”]

Agora, ao pausar o mouse sobre os dois pontos mais altos, você terá os valores contidos na tabela de coordenadas de curvas, conforme Fig. 11.



**Figura 11**. Pontos mais altos do gráfico.

Só resta, então, analisar na tabela de entradas qual é o ponto de corte, gerando sim ou não, conforme Fig. 12.

**Figura 12**. Pontos de corte.

### Armazenamento de dados em NumPy para aplicar análises

Além de saber realizar análises estatísticas, é necessário saber armazenar dados. Para fazer os exercícios aqui propostos, realize três ações:

* Digite **pip install numpy** no cmd para instalar NumPy;
* Digite **jupyter-notebook** no cmd para abrir o jupyter;
* Crie um novo arquivo python 3.

As matrizes são tabelas multidimensionais e podem ser criadas a partir de vetor. Vetores e matrizes podem ter números, *strings* (textos entre aspas) e valores booleanos (True ou False).

Caso o programador ou *data scientist* queira abrir uma nova página, é necessário digitar **importnumpyasnp**.

O exemplo a seguir demonstra a criação de quatro vetores e, então, a criação de uma matriz com 4 linhas e 4 colunas.

In[13]:

gabriel\_gostos=['vídeo game','cinema','tecnologia','livros']

marcio\_gostos=['futebol','natação','golfe','hipismo']

juliana\_gostos=['gastronomia','viagem','cinema','teatro']

daniela\_gostos=['caminhada','livros','séries','desenho digital']

**mostrar\_gostos=np.array([gabriel\_gostos,marcio\_gostos,juliana\_gostos,daniela\_gostos])**

**mostrar\_gostos**

Out[13]:

array([['vídeo game', 'cinema', 'tecnologia', 'livros'],

['futebol', 'natação', 'golfe', 'hipismo'],

['gastronomia', 'viagem', 'cinema', 'teatro'],

['caminhada', 'livros', 'séries', 'desenho digital']], dtype='<U15')

Para demonstrar apenas a linha 1, basta citar o número 0, conforme exemplo a seguir, que mostrará os gostos do Gabriel.

In[14]: mostrar\_gostos[0]

Out[14]:array(['vídeo game', 'cinema', 'tecnologia', 'livros'], dtype='<U15')

O exemplo a seguir conta 5 itens da esquerda para a direita, começando pelo 0 (zero).

In[15]: mostrar\_gostos.item(5)

Out[15]:'natação'

Para demonstrar a segunda linha (1) e a primeira coluna (0), basta digitar o código a seguir:

In[16]: mostrar\_gostos[1][0]

Out[16]:'futebol'

É possível, contudo, criar listas e exibir como matriz, sem utilizar palavra reservada como ocorreu em In[14]:, onde foi criada a variável mostrar\_gostos. A seguir, vamos ter dois exemplos, o primeiro com o comando **np.array** e o segundo, só com as **listas**. Ambos têm o mesmo resultado.

In[17]:

marketing\_janeiro=['5 cartazes','3 banners','3 malas diretas']marketing\_fevereiro=['2 promoções','3 spots','4 banners']

marketing\_marco=['2 jingles','3 promoções','3 banners']

**np.array**([marketing\_janeiro,marketing\_fevereiro,marketing\_marco])

Out[17]:

array([['5 cartazes', '3 banners', '3 malas diretas'],

['2 promoções', '3 spots', '4 banners'],

['2 jingles', '3 promoções', '3 banners']], dtype='<U15')

A seguir, é criada a matriz sem np.array.

In[18]:

marketing\_janeiro=['5 cartazes','3 banners','3 malas diretas']marketing\_fevereiro=['2 promoções','3 spots','4 banners']

marketing\_marco=['2 jingles','3 promoções','3 banners']

**[marketing\_janeiro,marketing\_fevereiro,marketing\_marco])**

Out[18]:

array([['5 cartazes', '3 banners', '3 malas diretas'],

['2 promoções', '3 spots', '4 banners'],

['2 jingles', '3 promoções', '3 banners']], dtype='<U15')

Será demonstrado, então, como **criar uma faixa** (*range*) com o comando **np.range** e depois, será demonstrado como **dividir faixas** em 4 linhas e 2 colunas com o comando **np.reshape**. O exemplo a seguir mostra a *range* de 0 até 8.

In[19]: criar\_range=np.arange(0,8)

print(criar\_range)

Out[19]: [0 1 2 3 4 5 6 7]

Segue, portanto, a divisão com o np.reshape.

In[20]: dividir = np. Reshape(criar\_range, (2,4))

print(dividir)

Out[20]:

[[0 1 2 3]

[4 5 6 7]]

Outra ação que o NumPy faz em array é remover e deletar linhas ou colunas. A

sintaxe **np.delete(nome\_da\_matriz, 0, 1)** por exemplo, deletaria a primeira coluna. Já a sintaxe **np.delete(nome\_da\_matriz, 2, 0)** deletaria a segunda linha, conforme o exemplo a seguir.

In[21]

gabriel\_gostos = ['vídeo game', 'cinema','tecnologia','livros']

marcio\_gostos = ['futebol', 'natação','golfe','hipismo']

juliana\_gostos = ['gastronomia', 'viagem','cinema','teatro']

daniela\_gostos = ['caminhada', 'livros','séries','desenho digital']

**mostrar\_gostos = np.array([gabriel\_gostos, marcio\_gostos, juliana\_gostos, daniela\_gostos])**

**mostrar\_gostos**

Out[21]:

array([['vídeo game', 'cinema', 'tecnologia', 'livros'],

['futebol', 'natação', 'golfe', 'hipismo'],

['gastronomia', 'viagem', 'cinema', 'teatro'],

['caminhada', 'livros', 'séries', 'desenho digital']], dtype='<U15')

Note agora que a linha 2, referente à gastronomia etc., será deletada:

In[22]:np.delete(mostrar\_gostos,2,0)

Out[22]:

array([['vídeo game', 'cinema', 'tecnologia', 'livros'],

['futebol', 'natação', 'golfe', 'hipismo'],

['caminhada', 'livros', 'séries', 'desenho digital']], dtype='<U15')

Para deletar a linha 1 e a linha 3, o programa percorrerá do 0 (será :) até o 2 (que será a terceira linha).

In[23]:np.delete(mostrar\_gostos,np.s\_[::2],0)

Out[23]:

array([['futebol', 'natação', 'golfe', 'hípismo'],

['caminhada', 'livros', 'séries', 'desenho digital']], dtype='<U15')

Antes de abordar a inserção de dados, é necessário explicar a diferença entre **axis = 0 e axis = 1**. Enquanto **axis 0 é linha**, **axis 1 é coluna**. Dessa forma, a sintaxe np.insert(mostrar\_gostos,1,'vazio',axis=0), está dizendo que o eixo (axis=0) a ser utilizado será de linhas e não colunas. Sendo assim, resta explicar os outros dados**.** O valor 1 representa a segunda linha (pois a primeira é o zero), e a palavra 'vazio' representa a substituição na linha 1.

In[24]:np.insert(mostrar\_gostos,1,'vazio',axis=0)

Out[24]:

Array([['video game', 'cinema', 'tecnologia', 'livros'],

**['vazio', 'vazio', 'vazio', 'vazio'],**

['futebol', 'natação', 'golfe', 'hipismo'],

['gastronomia', 'viagem', 'cinema', 'teatro'],

['caminhada', 'livros', 'séries', 'desenho digital']], dtype='<U15')

## Evidências e probabilidades

Neste momento, nos concentraremos no estudo de probabilidades e evidências. Para isso, usaremos técnicas de classificação de Naive Bayes, que foram criadas com base no Teorema de Bayes. Essa forma de classificação é útil parta discernir textos e é muito usada em *Machine Learn*. Por exemplo:

* É spam ou não é spam, por exemplo, o Gmail, que contém caixa para entrada de promoções, social etc;
* Separação de emoções;
* Separação de documentos por tema.

### 2.1 Localizando evidências na prática

A equação usada foi baseada no Teorema de Bayes.

A Tabela 1 é uma tabela de compras, mas poderia ser de curtidas no Facebook, por exemplo: quem curtiu pertence a que classe? É jovem ou adulto? Bebe ou não?

**Tabela 1. Decisão de compra e atributo**

|  |  |  |  |
| --- | --- | --- | --- |
| **CLASSE SOCIAL** | **ESTADO CIVIL** | **FILHOS** | **COMPRAR CARRO?** |
| MÉDIA | SOLTEIRO | SIM | SIM |
| BAIXA | CASADO | NÃO | NÃO |
| ALTA | DIVORCIADO | NÃO | SIM |
| MÉDIA | SOLTEIRO | NÃO | SIM |
| MÉDIA | CASADO | SIM | NÃO |
| ALTA | SOLTEIRO | NÃO | NÃO |
| BAIXA | CASADO | SIM | SIM |
| BAIXA | CASADO | SIM | NÃO |

A Tabela 2 isola o item classe social e compras, pois o ideal é analisar classe por classe.

**Tabela 2. Análise de saída por classe social**

|  |  |
| --- | --- |
| **CLASSE SOCIAL** | **COMPRAR CARRO?** |
| MÉDIA | SIM |
| BAIXA | NÃO |
| ALTA | SIM |
| MÉDIA | SIM |
| MÉDIA | NÃO |
| ALTA | SIM |
| BAIXA | SIM |
| BAIXA | NÃO |

A Tabela 3 contém cores para facilitar o entendimento do cálculo, uma vez que com os dados em mãos, basta aplicar a teoria aqui contida nos dados da empresa e terá a probabilidade, classe por classe.

**Tabela 3. Cálculos de decisões por classe**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Não** | **Sim** |  |  |
| BAIXA | 2 | 1 | =3/8 | 0,375 |
| MÉDIA | 1 | 2 | =3/8 | 0,375 |
| ALTA | 1 | 1 | =2/8 | 0,25 |
| Total | 4 | 4 |  |  |
|  | 4/8 | 4/8 |  |  |
|  | 0,5 | 0,5 |  |  |

Quem é de classe alta compra carro (sim) com qual probabilidade?

Quem é classe média compra carro (sim) com qual probabilidade?

Quem é classe baixa compra carro (sim) com qual probabilidade?

Após fazer esses cálculos, deve-se refazer com as colunas “estado civil” e “filhos”, tendo, assim, uma análise geral de probabilidade.

As vantagens das técnicas de Naive Bayes é que tudo é feito de maneira rápida e testa várias classes. A principal desvantagem é o fato de que se analisam probabilidades momentâneas e separadas, sendo necessário complementar com outras técnicas preditivas. Cabe citar que, na área de filtro de textos, ele é bem eficaz. Ao analisar textos, ele consegue detectar bem os gostos principais dos usuários, sendo assim, é útil em sistemas de aprendizado de máquina e spam.

## Representação e mineração de textos

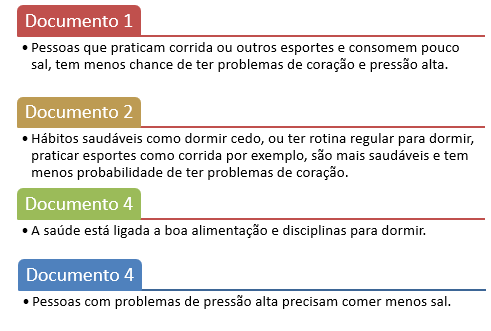
Palavras, em mineração de texto, são chamadas por *token*. Elas refletem emoções e trazem verdades. Em mineração de texto, documentos representam tanto artigos, e-mails, opiniões e comentários, quanto avaliações de cinco estrelas. Para avaliar palavras importantes, deve-se excluir palavras “*stop word*”, que são palavras sem significado, como “e”, “ou” e etc. Palavras raras geralmente são dispensadas, e todas precisam estar em formatação igual, ou seja, todas em letras minúsculas. Uma boa análise coloca limites máximos e mínimos ao fazer um filtro de palavras.

Segundo Marzagão (s. d., p. 11), deve-se:

* Identificar automaticamente o autor de um documento (classificação);
* Agrupar um conjunto de documentos por autores ou tópicos (clusterização).

### 3.1 IDF e TDIF em *bag of words*

*Bag of words* são multiconjuntos de palavras. Nesses conjuntos, para entender Frequência de Termos (TF), pode-se analisar essa afirmação: “O peso de um termo que ocorre em um documento é diretamente proporcional à sua frequência.” (NOLASCO; OLIVEIRA, 2016, p. 17), mas há palavras que se repetem mais em um texto e menos em outros. A seguir, temos uma análise de quatro documentos:



Agora, a Tabela 4 a seguir faz a contagem dessas repetições, ou seja, dos *tokens* nos quatro documentos:

**Tabela 4.** **Repetições de palavras**



Se temos oito documentos e em dois deles, um *token* se repete por três ou quatro vezes, ele é importante nesses dois documentos dentre os oito.Ou seja, sendo frequente uma palavra em um texto e não aparecendo nos outros, ela se torna mais importante para o documento que a tem e menos significante para a coleção. Para isso utiliza-se o IDF (frequência inversa do documento).

Para resumir, enquanto:

Pode-se calcular o TFIDF e aplicar distância Euclidiana para descobrir vizinhos mais próximos. Sendo que:

### Cálculos de IDF e TF IDF para ponderação de TF

Para facilitar, vamos usar um exemplo de análises em documentos, supondo que temos uma coleção de documentos. Em um documento dentro dessa coleção, por exemplo, em um artigo, aparecem as palavras abaixo com as seguintes frequências:

* Em um documento analisado: Corrida (3 vezes), Esportes (2 vezes), Academia (1 vez);
* A frequência dos termos normalizada, que objetiva remover diferenças sutis, é dita pela fórmula:

Na coleção, existem dez documentos. Ao analisar a frequência dessas palavras na coleção inteira, tem-se:

* Na coleção: **Corrida** (ocorre em 5 documentos), **Esportes** (ocorre em 7 documentos), **Academia** (ocorre em 10 documentos).

Agora será feito o cálculo TFIDF:

Análise de **Corrida**:

* Corrida:
* TF= 3 vezes /3 (frequência máxima) = 1
* IDF= log2 (10 documentos /5 repetições de corrida)
* IDF= log2 (2)
* IDF = 1
* TFIDF = TF x IDF = 1 x 1= 1

A importância da palavra **corrida** nesse documento para o conjunto é de 1. Leia os demais tópicos para entender.

Análise de **Esportes**:

* Esportes:
* TF= 2 vezes /3 (frequência máxima) = 0,66
* IDF= log2 (10 documentos / 7 repetições de esportes)
* IDF= log2 (1,42)
* IDF = 0,5
* TFIDF = TF x IDF = 0,66 x 0,5 = 0,33

A importância da palavra **esportes** nesse documento, em comparação com o conjunto, é de 0,33. Continue lendo para entender.

Análise de **Academia**:

* Academia
* TF= 1 vez /3 (frequência máxima) = 0,33
* IDF= log2 (10 documentos /10 repetições de academia)
* IDF= log2 (1)
* IDF = 0
* TFIDF = TF x IDF = 0,33 x 1= 0,33

A importância da palavra **academia** nesse documento, em comparação com o conjunto, é de 0,33 (igual a esportes), pois ainda que tivesse menos em um documento, havia mais na coleção. Assim, ao invés de ser menos importante, ela é igualmente importante à palavra **esportes**.

**Box: Dica**

Para fazer os cálculos a seguir, sugere-se digitar no Google: “Log Base 2 Calculator”. Assim, você terá opções de calculadoras on-line para facilitar o processo.

Em outras palavras, TF IDF é destinado a normalização. Com isso, supondo que você tem um site e deseja criar conteúdo mais exclusivo para uma matéria numa subpágina, você irá, por exemplo, analisar os comentários, e fará o estudo proporcional normalizado dessa subpágina, comparando as palavras-chaves com o site inteiro. O que der valor maior naquela subpágina, mesmo normalizando com o Log do IDF, será, de fato, o conteúdo mais adequado a ser postado, bem como as palavras-chaves. Isso porque foi comparada a frequência daquele item naquele documento em relação ao site inteiro, também foi comparado com os demais, e nesse ranking de saber qual *token* é melhor para a subpágina, um item foi o vencedor.

## Engenharia analítica

Nesse momento, vamos abordar três conceitos: valor esperado (esperança matemática), variância e desvio padrão. Para concluir, será passado um exemplo de *Machine Learn* aplicado em valor esperado.

### 4.1 Desvio padrão

Dado um empate entre dois atributos, o desvio padrão analisa e identifica a probabilidade de erro de cada atributo para ver qual é mais confiável, seguindo a fórmula a seguir:

Supondo que uma loja quer decidir qual produto precisa de mais propaganda, ela analisa os seguintes dados:

**Tabela 5.** **Comparação de tipos de roupas**

|  |  |  |
| --- | --- | --- |
| **Produto** | **Blusas (esportivas)** | **Blusas (sociais)** |
| Mês 1 | 1.810,00 | 1.400,00 |
| Mês 2 | 1.800,00 | 1950 |
| Mês 3 | 1.900,00 | 2100 |
| Média | 1.836,67 | 1.816,67 |

Como pode ser visto, as médias são bem parecidas, então existe uma fórmula a ser aplicada para descobrir a variância e o desvio padrão.

Sabendo que o desvio para mais ou para menos das blusas sociais é muito maior, faz-se mais necessário criar planos de fidelidade de compra e promoções voltadas às blusas sociais. De um lado, varia muito, e de outro, demonstra alto potencial por ter compras mais altas.

Uma forma de avaliar comportamentos em web e interações é através de *Machine Learn*. A proposta de aplicação seria implantar o *Machine Learn* e, com a interação dos usuários, fazer uma tabela com interações para avaliar as proporções.

### Valor esperado

Em um restaurante, um prato traz um determinado lucro. Duas pessoas trazem um lucro de R$ 100,00, três pessoas trazem um lucro de R$ 150,00 e quatro pessoas trazem um lucro de R$ 175,00.

Em média, analisando um mês, são ocupadas 100 mesas por dia no restaurante, onde, em média, 50 mesas são ocupadas por 2 pessoas, 15 mesas por 3 pessoas e 35 mesas por 4 pessoas. O valor esperado será o lucro esperado nesse exemplo, e é o produto do lucro pela probabilidade. Isso pode ser identificado com a seguinte fórmula:

**Tabela 6.** **Valor esperado (lucro esperado)**

|  |  |  |  |
| --- | --- | --- | --- |
| Clientes | Lucro da compra de serviços | Probabilidade | Valor Esperado |
| 2 | 100 | 0,50 | 50 |
| 3 | 150 | 0,15 | 22,5 |
| 4 | 175 | 0,35 | 61,25 |
| Soma (lucro esperado por dia) | | | 133,75 |

### 4.3 Criação de um robô captando interesse através de dados

Para concluir, pode-se fazer um robô que faça a captação de interesses e tabular para analisar valor esperado, ou seja, que palavra gera mais efeito.

Páginas comerciais, como as do Facebook, podem conter ferramentas robotizadas, a fim de melhorar o relacionamento com o público ou captar interesses.

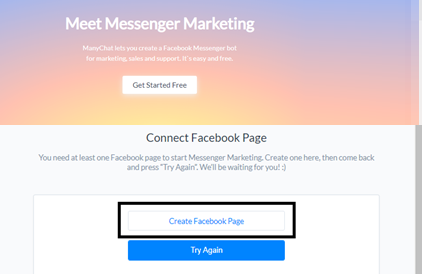
Temos, a seguir, um exemplo de como conhecer o público-alvo através de respostas dos usuários em comentários de posts dessa página comercial do Facebook. Não é necessário ter uma página comercial, basta ter uma página pessoal.

Um ponto importante é que as plataformas de criação de robôs em chats, ou *chatbot*, são diversas, inclusive por *Python* e suas bibliotecas. Assim, é aconselhável praticar essas plataformas e saber usar várias delas, uma vez que as necessidades mudam de empresa para empresa e de sistemas para sistemas. Todavia, faz-se necessário praticar o conteúdo de *chatbot* e não apenas ler, assim, absorve-se melhor os conteúdos.

**Box**: **Dica**

Além das aplicações aprendidas neste curso, existem diversos cursos on-line sobre *chatbot*, e muitos deles são gratuitos.

A Fig. 13 demonstra a tela inicial da plataforma escolhida como exemplo. Depois de logar com o usuário do Facebook, o *data scientist* criaria uma página comercial de forma simples.

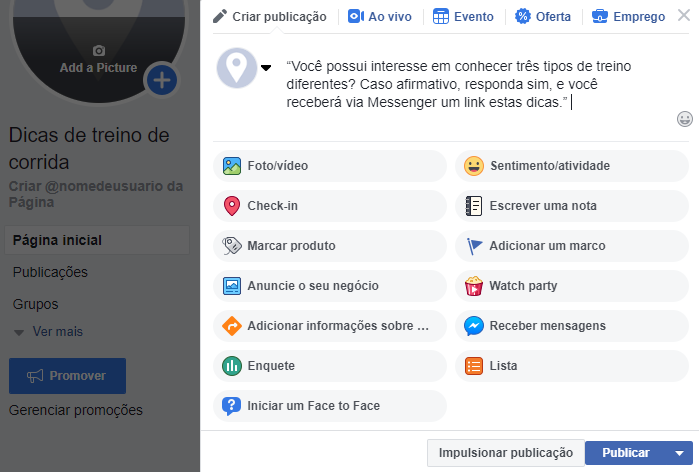


**Figura 13**. Criação de Robôs para Data Science no Meet Messenger.

Uma vez logado, ao clicar em **criar**, a plataforma solicitaria o nome da empresa, o endereço e o ramo, e após digitar esses dados, a página comercial é criada.

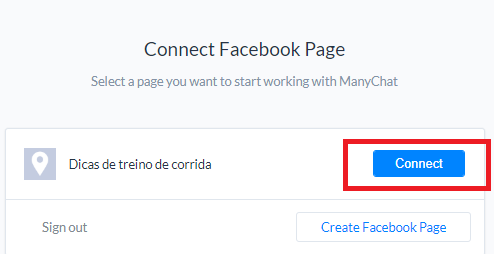
No exemplo apresentado, seria criada a página com o nome “Dicas de treino de corrida”, cujo objetivo era postar dicas esportivas e, então, além de divulgar os planos e dicas, também descobrir quem é o público-alvo de maior interesse.

O primeiro passo seria entrar na página comercial e fazer um *post* com uma pergunta, tal como: “Você possui interesse em conhecer três tipos de treino diferentes? Em caso afirmativo, responda sim, e você receberá via Messenger um link com dicas.”. Caso a pessoa responda “SIM”, “sim”, “Sim”, “quero”, ou “QUERO”, ou seja, com as palavras certas, automaticamente é enviado um texto em mensagem (*chatbot*) com dicas no Messenger da pessoa que comentou, e embaixo do texto, um link para o site da empresa. A Fig. 14 exemplifica o *post*.



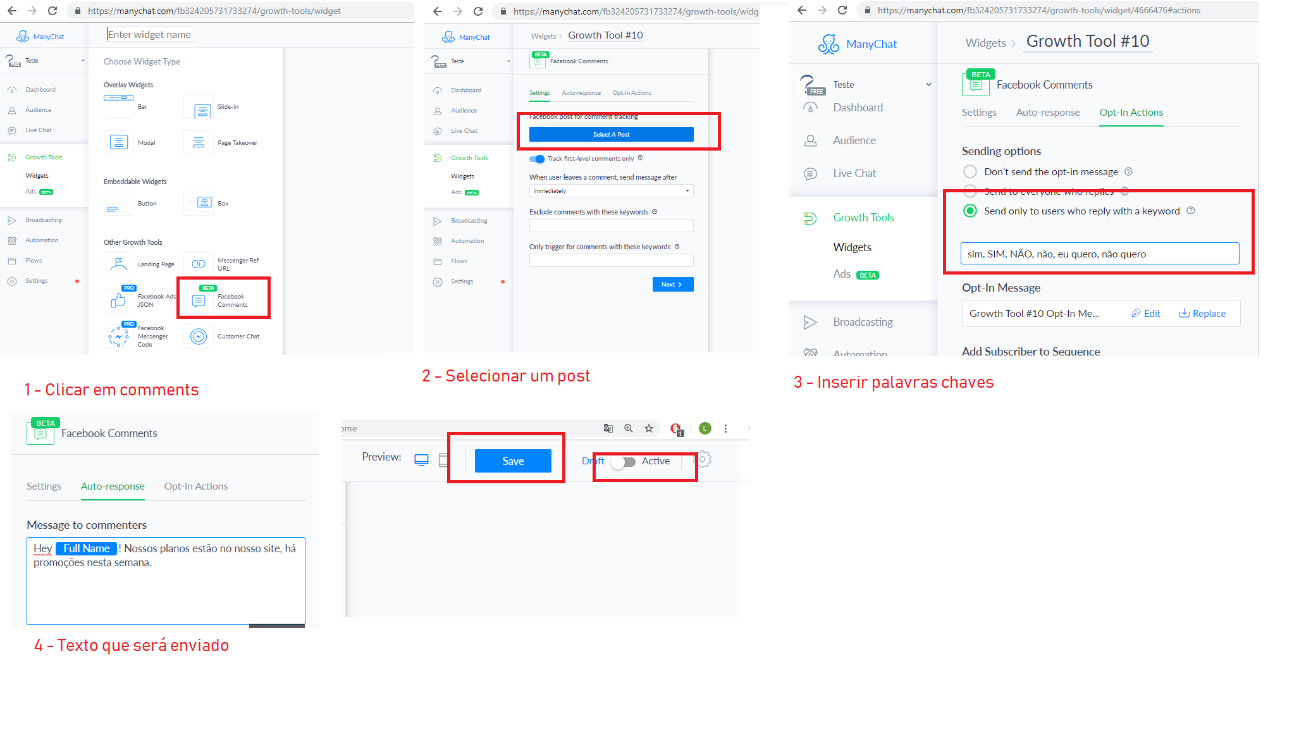
**Figura 14**. Criando um post na página comercial do Facebook.

Após postar na página do Facebook, bastaria clicar em *connect*, conforme a Fig. 15.



**Figura 15**. Entrar na página comercial pelo Many Chat.

Estando o *Meet Messenger* conectado com a página, o *data scientist* precisaria apenas criar uma regra clicando em *Growth Tools*, e depois em + *Nem Growth Tool*, e assim, ao responder com as palavras certas, o robô já manda a dica instantaneamente ao usuário, trazendo o resultado esperado. Conforme Fig. 16, pode-se ver o processo final de criação de regra.



**Figura 16**. Criar Growth Tools.

Após isso, pode-se criar várias perguntas sequenciais para descobrir mais dados quantitativos e qualitativos.

O *data scientist* deve armazenar os usuários em um banco de dados, e é recomendado criar um *dataset* ou gráficos com essas informações. Caso o trabalho seja a prioridade do setor, ao analisar o *dataset* ou gráfico pode-se rapidamente fazer cálculos de análises probabilísticas.

# Sintetizando

Essa unidade começou retomando o conceito de matriz de confusão, que analisa atributos e, após testar, faz uma comparação das saídas supostas no treino com as saídas reais. Ela analisa os falsos positivos, verdadeiros positivos, falsos negativos e verdadeiros negativos, e após retomar, já fez duas análises: de acordo com nosso exemplo, houve uma para o corte de pontos\_antecipação e a outra de pontos\_atraso, para delimitar a partir de quais valores o resultado é mais confiável. Um pouco de NumPy foi passado, mas não completamente, pois isso será abordado em outro momento. Foi passado, ainda, o conceito de evidência, que analisa probabilidade de vários atributos com cálculos, cuja finalidade é pontuar cada atributo. O TF, ou frequência de termos, foi abordado em mineração de texto, bem como os passos iniciais antes de contar as frequências de cada palavra em um *bag of words*, e foi dito que é necessário eliminar palavras que se repetem em demasia e não são úteis naquele contexto, para, assim, identificar os *tokens*. Uma vez com esses *tokens* em mãos, pode-se fazer primeiro o cálculo de TF, depois de IDF, isto é, quanto mais uma palavra se repete em um texto, e menos em um conjunto, mais importante ela é para aquele texto. Sendo assim, ao realizar uma mineração, é necessário analisar tanto um texto, quanto o conjunto, e o nome dessa comparação dupla é TFIDF, a qual faz a ponderação. O conceito de desvio padrão demonstrou como um item pode variar para cima e para baixo. O conceito de valor esperado também foi passado, demonstrando que ele é nada mais que a média das quantidades multiplicadas pela probabilidade. Para concluir, foi passado um exercício de *machine learn* para sustentar o aluno com uma prática *front end*, isto é, em relacionamento com cliente.

# Palavras-chave

1. Curva Roc
2. Evidências
3. IDF
4. TF
5. TFIDF
6. Mineração
7. *Bag of words*
8. Desvio padrão
9. Valor esperado
10. Probabilidade
11. Bayes
12. Sensibilidade
13. Especificidade
14. Ponderação
15. Array
16. Cliques
17. Multiconjunto
18. Preditivo
19. Documento
20. Coleção

# Referências bibliográficas

BORGES, R. et al. **Gerenciamento do Ponto de Corte na Concessão do Crédito Direto ao Consumidor**. Rio de Janeiro: ANPAD, 2017, p. 273.

FOSTER, P.; FAWCETT, P. ***Data Science* para negócios**: O que você precisa saber sobre mineração de dados e pensamento analítico de dados? Rio de Janeiro: Alta books, 2016.

MARZAGÃO, T. **Mineração de textos**. São Paulo: UNB. [s.d.]. Disponível em: <http://thiagomarzagao.com/assets/teaching/mineracao/slides10.pdf>. Acesso em: 03 fev. 2019.

NOLASCO, D.; OLIVEIRA, J. **Modelagem de tópicos e criação de rótulos:** identificando temas em dados semiestruturados e não-estruturados. Rio de Janeiro: URFJ, 2016. p. 17.