Relatório 5 - Estatística p/ Aprendizado de Máquina (I)

Lucas Scheffer Hundsdorfer

Descrição da atividade

Aula 1 - Types of Data.

Existem vários tipos de dados, mas ele traz ênfase em 3 tipos, numérico, categórico e ordinal.

Numérico é o mais comum de todos os dados, basicamente é algo que você consegue quantificar, como altura de alguém, páginas de um livro. E têm dois tipos de dados numéricos, os dados discretos que são números inteiros que podem contar algum evento, como, quantas vezes choveu no ano, o outro tipo de dado numérico é o dado contínuo, é um dado que pode ter precisão infinita, como, quanta chuva caí em um determinado dia.

Categóricos são dados que não tem um significado numérico, como gênero, raça, endereço.

Ordinal é quase que uma mistura dos dados categóricos e numéricos, como, classificações de um filme por estrela.

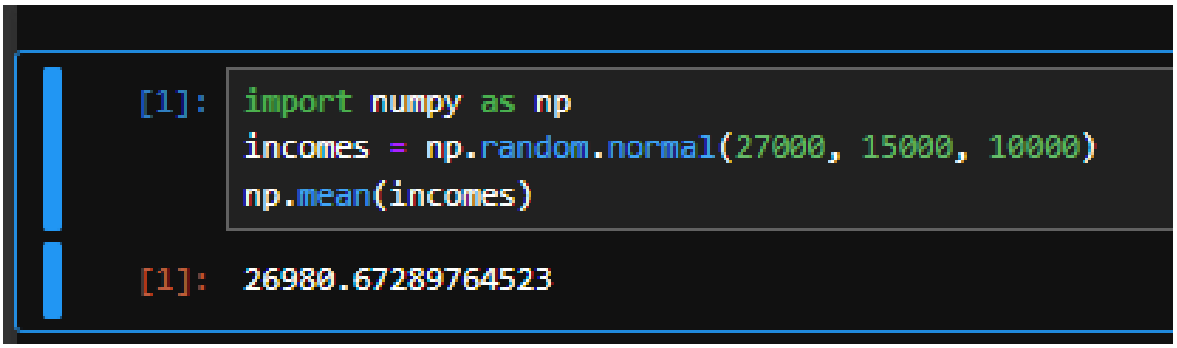
Aula 2 - Mean, Median, Mode

Média é a soma de todos os elementos e a divisão pela quantidade de elementos existentes.

Mediana é a ordenação de todos os elementos e pegar o elemento que fica no meio.(Se a quantidade de elementos for par, pega os 2 elementos do meio e aplica a média).

Moda é o valor que mais aparece em um conjunto de dados.

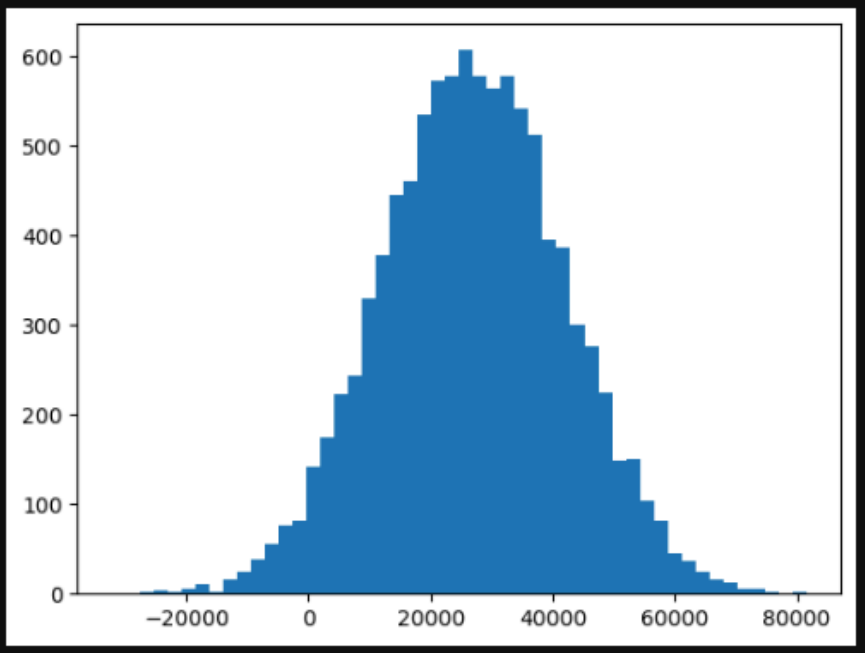
Aula 3 - Using mean, median, and mode in Python



```
[1]: import numpy as np
     incomes = np.random.normal(27000, 15000, 10000)
     np.mean(incomes)

[1]: 26980.67289764523
```

Aqui mostra a média de um exemplo da aula, onde ele passa os padrões como a média de 27000, o desvio padrão de 15000 e o número de exemplos como 10000.

```
[11]: %matplotlib inline
      import matplotlib.pyplot as plt
      plt.hist(incomes,50)
      plt.show()
```




```
[5]: np.median(incomes)

[5]: 26912.179093041384
```

Aqui ele importa a biblioteca matplotlib para poder exibir o gráfico, plt.hist serve para poder criar um histograma do nosso conjunto de dados e o 50 é para definir o número de intervalos dentro do histograma, e depois mostra a mediana.

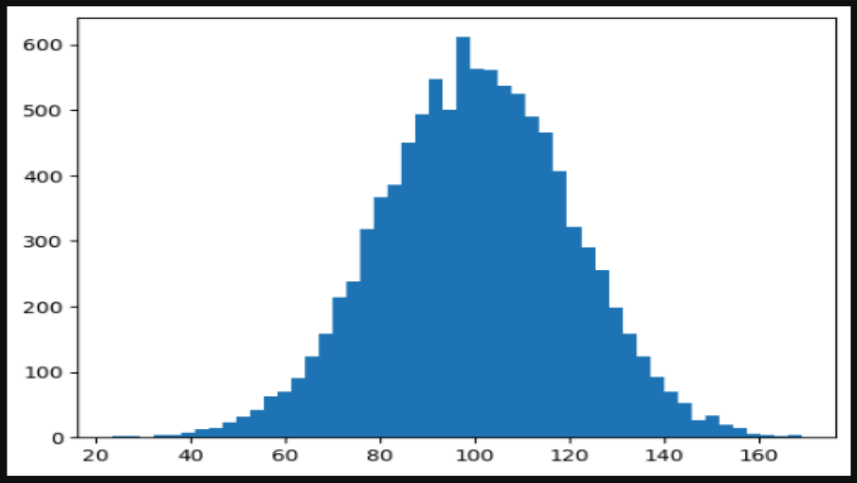
```
[5]: np.median(incomes)

[5]: 26912.179093041384

[13]: incomes = np.append(incomes,[1000000000])

[15]: np.median(incomes)

[15]: 26915.328005061045

[17]: np.mean(incomes)

[17]: 126967.97610003523
```

Aqui ele adiciona mais um elemento ao array dele com um valor de um bilhão, após isso ele faz o teste da média e da mediana, a mediana não muda quase nada pois só se acrescentou um valor, mas a média teve um aumento de 100000.

```
[19]: ages = np.random.randint(18, high=90, size=500)
      ages

[19]: array([27, 43, 31, 26, 32, 26, 70, 89, 43, 75, 68, 67, 63, 82, 65, 82, 57,
             47, 47, 36, 81, 45, 35, 25, 87, 24, 43, 39, 61, 79, 83, 27, 63, 36,
             29, 25, 83, 72, 56, 82, 35, 66, 42, 52, 53, 65, 64, 79, 77, 74, 32,
             87, 69, 57, 29, 21, 26, 35, 60, 46, 60, 18, 22, 21, 21, 55, 27, 71,
             43, 42, 33, 44, 88, 46, 61, 55, 32, 70, 70, 53, 39, 71, 56, 77, 26,
             30, 49, 27, 41, 70, 83, 77, 65, 51, 40, 24, 18, 37, 50, 29, 38, 31,
             51, 46, 41, 27, 35, 78, 50, 49, 47, 34, 65, 66, 62, 18, 71, 28, 80,
             67, 65, 40, 65, 71, 67, 46, 61, 38, 60, 43, 47, 40, 61, 27, 62, 57,
             58, 37, 73, 21, 30, 80, 51, 19, 50, 58, 35, 20, 77, 29, 76, 54, 72,
             56, 51, 81, 47, 62, 35, 54, 73, 36, 44, 79, 77, 70, 18, 34, 33, 50,
             60, 38, 20, 39, 70, 57, 49, 21, 62, 83, 52, 29, 57, 61, 42, 40, 23,
             32, 71, 25, 48, 40, 34, 18, 56, 30, 57, 27, 41, 76, 75, 75, 65, 30,
             45, 18, 55, 34, 68, 76, 43, 54, 80, 86, 72, 55, 34, 24, 66, 29, 64,
             37, 72, 87, 45, 87, 50, 84, 87, 57, 25, 20, 33, 39, 80, 35, 29, 49,
             45, 89, 58, 51, 49, 47, 65, 66, 23, 87, 88, 46, 89, 69, 42, 39, 59,
             83, 58, 34, 56, 45, 67, 22, 62, 60, 66, 54, 49, 28, 47, 60, 35, 49,
             51, 20, 67, 57, 49, 43, 20, 86, 62, 73, 82, 51, 27, 31, 57, 86, 38,
             42, 71, 89, 63, 81, 30, 30, 45, 21, 56, 43, 40, 56, 73, 24, 89, 54,
             26, 74, 80, 32, 77, 72, 22, 45, 37, 22, 24, 27, 40, 42, 28, 50, 82,
             25, 32, 66, 81, 74, 49, 24, 89, 89, 82, 77, 86, 26, 18, 36, 74, 74,
             50, 85, 49, 88, 82, 68, 53, 20, 47, 73, 74, 34, 57, 44, 79, 47, 75,
             60, 37, 63, 22, 86, 66, 60, 24, 83, 49, 28, 61, 70, 33, 38, 18, 18,
             83, 73, 28, 28, 65, 61, 60, 78, 48, 49, 65, 69, 63, 71, 37, 88, 81,
             64, 75, 55, 64, 28, 56, 79, 78, 33, 38, 34, 45, 58, 35, 88, 72, 29,
             66, 22, 42, 86, 69, 54, 26, 68, 41, 54, 63, 83, 58, 74, 47, 26, 45,
             57, 46, 60, 67, 75, 84, 24, 76, 54, 83, 46, 45, 72, 78, 68, 83, 55,
             66, 69, 30, 59, 46, 31, 25, 42, 20, 19, 81, 48, 50, 51, 47, 77, 36,
             22, 45, 23, 41, 62, 23, 35, 75, 71, 67, 48, 65, 51, 62, 82, 53, 39,
             74, 45, 32, 30, 66, 81, 80, 75, 56, 64, 85, 64, 37, 54, 54, 29, 59,
             52, 56, 50, 21, 66, 79, 72])

[23]: from scipy import stats
      stats.mode(ages)

[23]: ModeResult(mode=45, count=12)
```

Aqui ele cria um array com 500 idades variando de 18 a 90 randomicamente. E após isso importa uma biblioteca para poder descobrir a moda.

```
[28]: vetor = np.random.normal(100.00, 20.00, 10000)
      plt.hist(vetor, 50)
      plt.show()
```




```
[30]: np.mean(vetor)

[30]: 100.28550829945789

[32]: np.median(vetor)

[32]: 100.20325211509454
```

No final da aula ele pede para fazer um exercício, ele passa um conjunto de dados e pergunta qual a média e a mediana utilizando os códigos do python.

Novamente a média e a moda dão valores bem próximos, mas isso acontece porque os dados utilizados estão igualmente distribuídos, porém em um conjunto de dados reais, isso provavelmente não irá acontecer.

Aula 4 - Activity Variation and Standard Deviation

Variância é a média das diferenças quadradas da média:

σ^2 de (1, 5, 7, 3, 4):

média de (1,5,7,3,4) = 4, agora pegue a diferença dos elementos para a média.

(-3, 1, 3, -1, 0), agora eleve essa diferença ao quadrado = (9, 1, 9, 1, 0) e faça agora a média desse quadrado = (9,1,9,1,0) /5 = 4. Logo a variância de (1, 5, 7, 3, 4) é 4.

Desvio padrão é a raiz quadrada da variância.

Para descobrir o desvio médio de (1, 5, 7, 3, 4), basta pegar a raiz da sua variância.

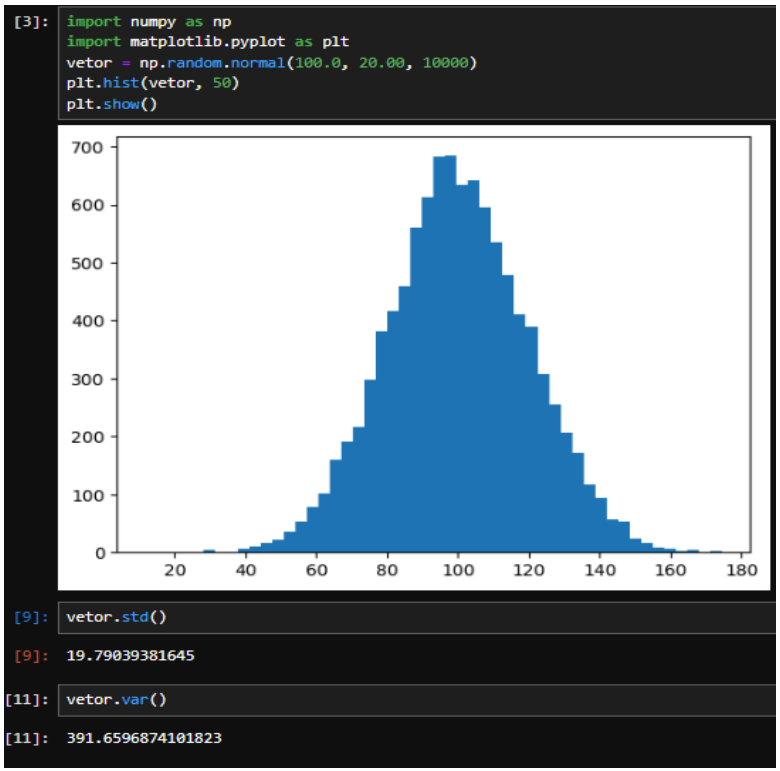
$\sqrt{4}$ = 2. Logo σ é 2.

Normalmente é usado para descobrir elementos atípicos do seu conjunto de dados. Pegamos a média do conjunto e fazemos +- o desvio padrão e vemos quais números estão fora desse intervalo, que seria (1,7).

Porém esse jeito de calcular serve apenas para quando você tem um conjunto de dados inteiro, se for apenas uma amostra do conjunto você divide a média dos quadrados por (N-1), exemplo:

σ^2 = (9,1,9,1,0) /5 = 4 (caso for um conjunto de dados inteiros).

s^2 = (9,1,9,1,0) /4 = 5 (caso for uma amostra do conjunto de dados).



Com o mesmo conjunto de dados utilizado na última aula, ele apresenta os métodos dentro do numpy para poder pegar o desvio padrão e a variância entre um vetor.

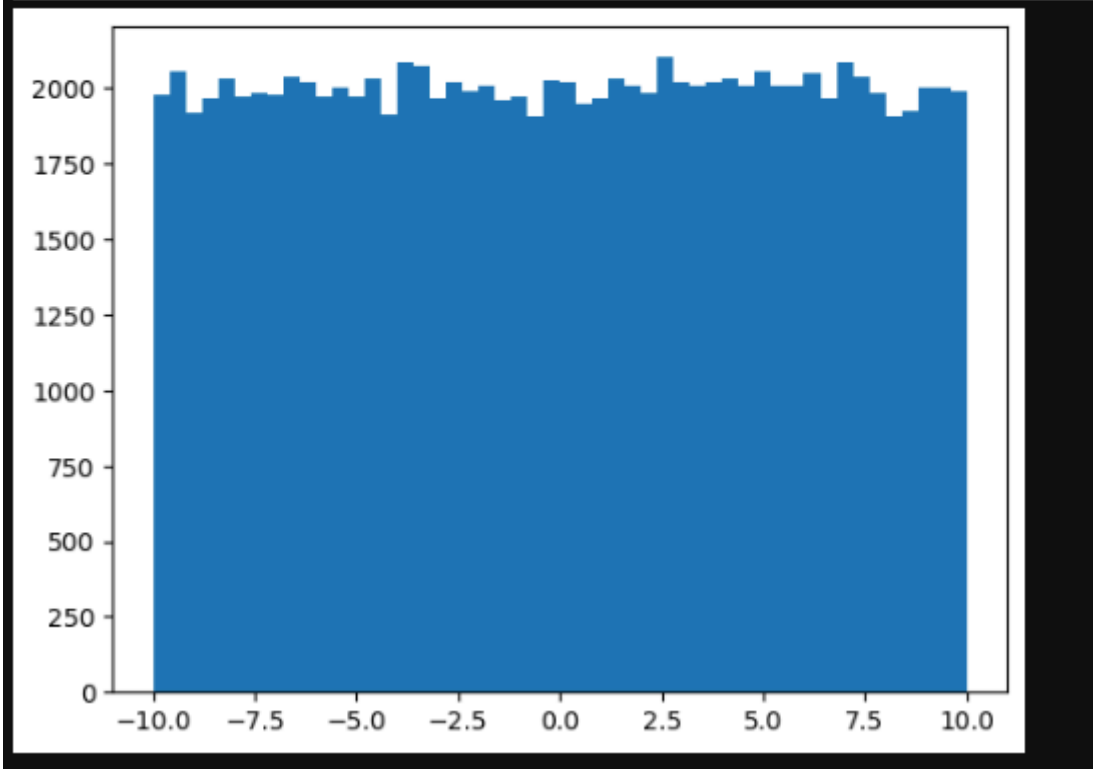
Aula 5 - Probability Density Function; Probability Mass Function:

Probability Density Function é usada para descrever como as probabilidades estão distribuídas ao longo de possíveis valores de uma variável contínua, não fornece diretamente a probabilidade de um valor específico, mas mostra a densidade de probabilidade em torno de diferentes valores.

Probability Mass Function é usada para variáveis discretas, e descreve a probabilidade de cada valor específico que a variável pode assumir.

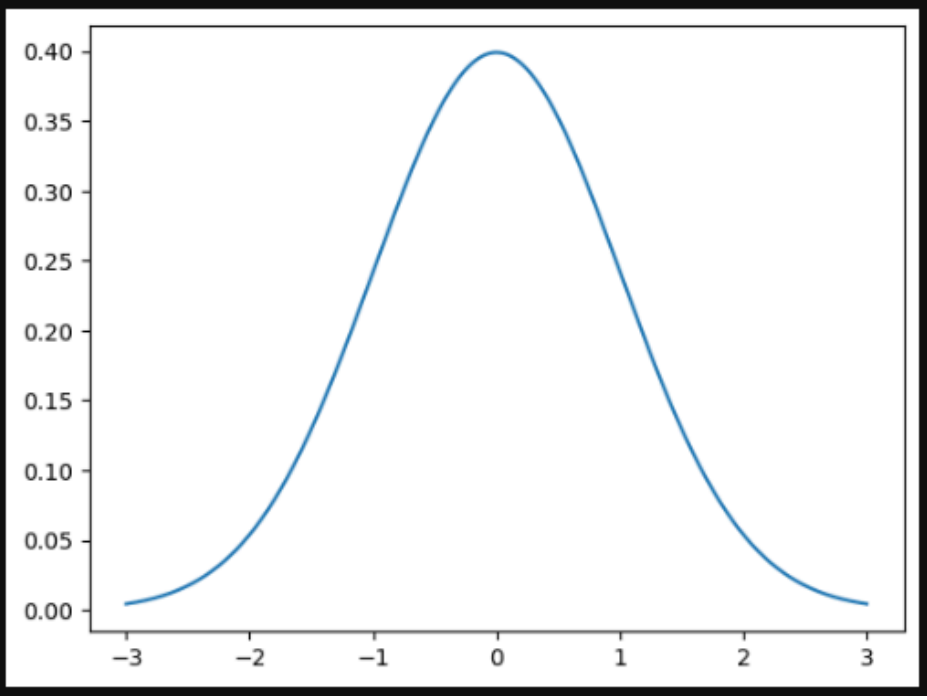
Aula 6 - Common Data Distributions (Normal, Binomial, Poisson, etc)

```python
import numpy as np
import matplotlib.pyplot as plt

valores = np.random.uniform(-10.0, 10.0, 100000)
plt.hist(valores,50)
plt.show()
```



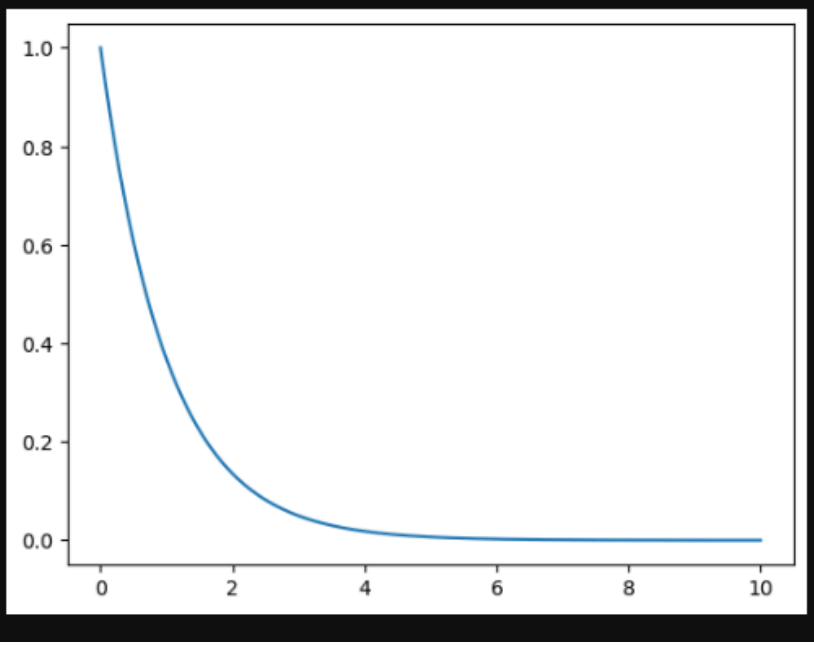
Distribuição uniforme significa que há uma probabilidade constante de um valor ocorrer dentro de um determinado intervalo. Por isso o histograma chega perto de ser uma linha plana, pois a probabilidade é quase constante.

```
[13]: from scipy.stats import norm

      x = np.arange(-3, 3, 0.001)
      plt.plot(x, norm.pdf(x))

[13]: [<matplotlib.lines.Line2D at 0x202fd5b4770>]
```



Distribuição Gaussiana é uma das distribuições mais utilizadas para modelar eventos naturais, porque ela é uma distribuição simétrica em torno de 0 e as pontas do gráfico são os desvios padrões.

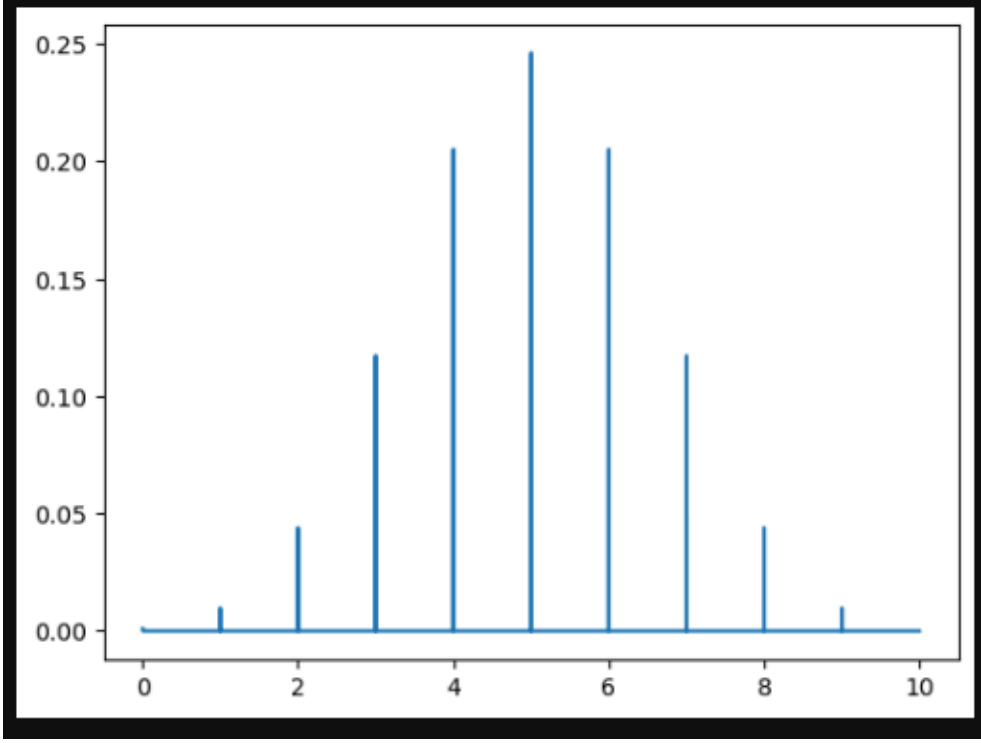
```
[19]: from scipy.stats import expon

      x = np.arange(0, 10, 0.001)
      plt.plot(x, expon.pdf(x))

[19]: [<matplotlib.lines.Line2D at 0x202800c86e0>]
```



Função de distribuição exponencial, se trata sobre dados que caem de uma forma exponencial. Basicamente é muito provável que algo aconteça perto de 0 e quanto mais aumenta o X menores as chances.

```
from scipy.stats import binom

n,p = 10, 0.5
x = np.arange(0, 10, 0.001)
plt.plot(x, binom.pmf(x, n, p))
```

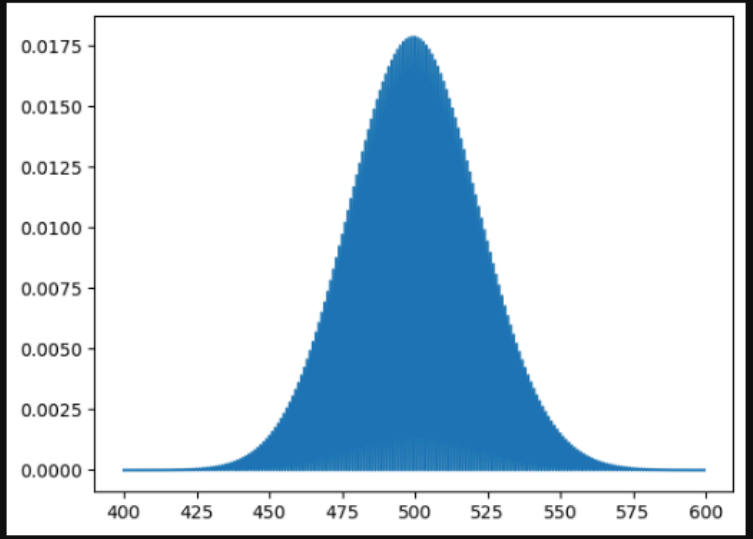
```
[<matplotlib.lines.Line2D at 0x202802550a0>]
```



Distribuição Binomial é uma probabilidade discreta usada para calcular a chance de ocorrerem x sucessos em n tentativas independentes, onde cada tentativa tem dois possíveis resultados. Um exemplo clássico é calcular a probabilidade de obter um certo número de "caras" ao jogar uma moeda várias vezes

```
[23]: from scipy.stats import poisson

      mu = 500
      x = np.arange(400, 600, 0.5)
      plt.plot(x, poisson.pmf(x, mu))
```

```
[23]: [<matplotlib.lines.Line2D at 0x202802b9130>]
```

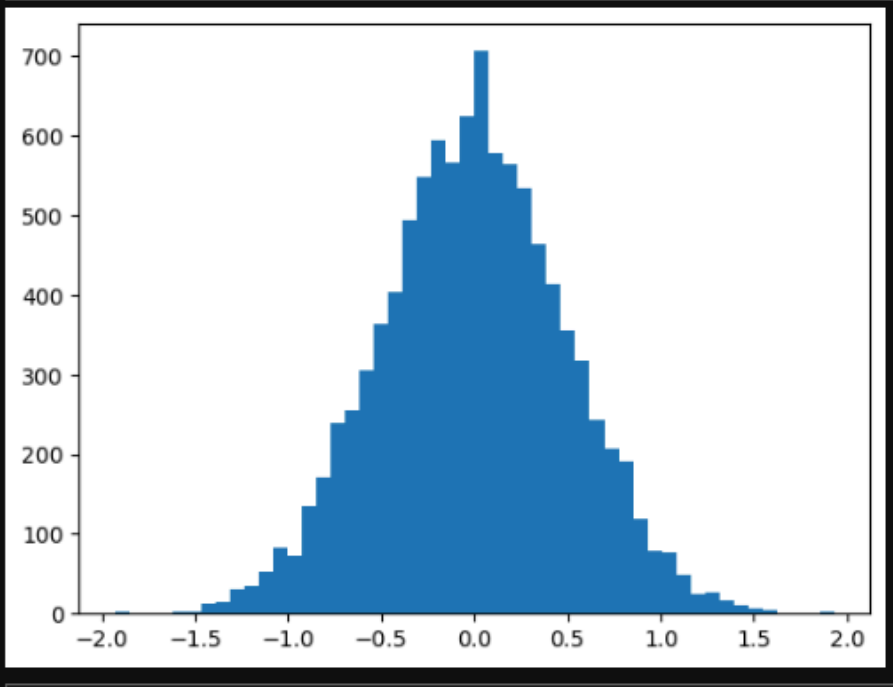


Distribuição de poisson, se parece muito com a distribuição normal mas, é usada para modelar a probabilidade do número de eventos ocorrerem em um intervalo fixo de tempo ou espaço, dado que os eventos ocorram de forma independente, tenham uma taxa média constante e que não ocorram simultaneamente.

Aula 7 - Porcentagem e Momentos

Porcentagem: em um conjunto de dados um determinado percentil é o ponto em que essa porcentagem dos dados é menor que o ponto em que você está.

```
[1]: import numpy as np
     import matplotlib.pyplot as plt
     vals = np.random.normal(0, 0.5, 10000)
     plt.hist(vals, 50)
     plt.show()
```



Iremos trabalhar em cima desse gráfico.

```
[3]: np.percentile(vals, 50)

[3]: -0.00017400222377105874

[5]: np.percentile(vals, 90)

[5]: 0.6337263997793653

[7]: np.percentile(vals, 20)

[7]: -0.4180303223844806
```

Esses são os valores abaixo das respectivas porcentagens de 50%, 90% e 20%.

Momentos são basicamente maneiras de medir a forma de uma distribuição. Existem 4 tipos de momentos:

1-Média, exato, a mesma média já aprendida antes.

2-Variância, a mesma variância já citada.

3-Inclinação, uma medida de quão desequilibrada é uma distribuição.

4-Curtose, uma medida do achatamento da curva, basicamente quão pontiaguda pode ser uma distribuição.

Os 2 primeiros momentos são simples e já foi ensinado a encontrá-los,

```
[9]: np.var(vals)

[9]: 0.24586357979008253

[11]: np.mean(vals)

[11]: -0.0041936553477059935
```

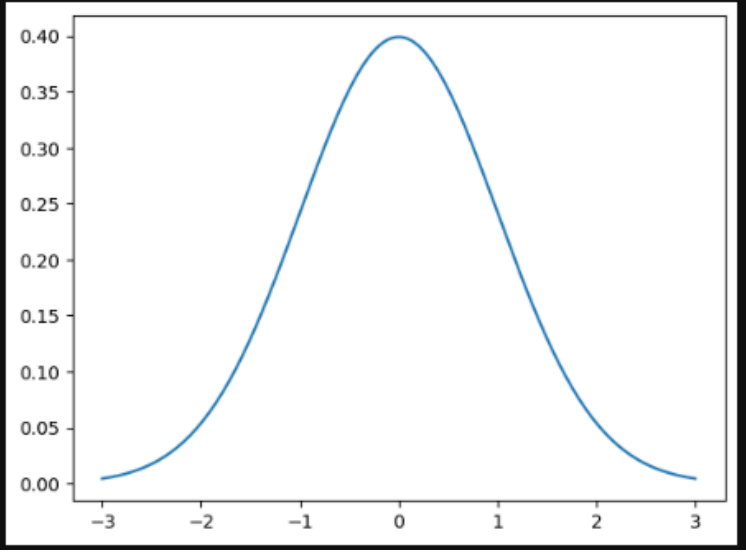
Já o 3 e o 4 é necessário importar a biblioteca Scipy, exemplo

```
[15]: import scipy.stats as sp
      sp.skew(vals)

[15]: -0.0011225003682918161

[17]: sp.kurtosis(vals)

[17]: -0.018584342320300706
```

Aula 8 - A crash course in Matplotlib

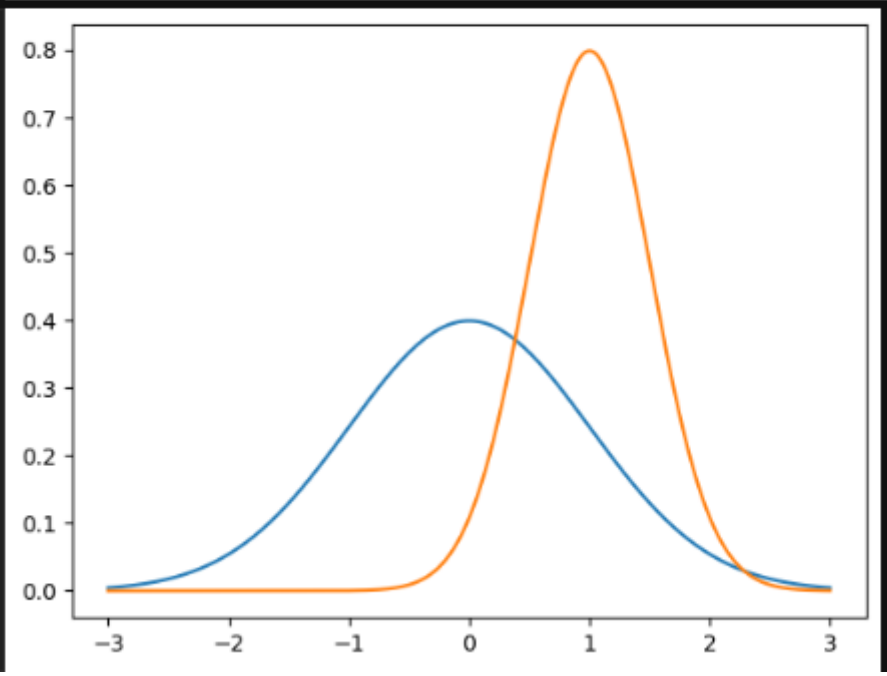
Com a biblioteca Matplotlib é possível comparar vários gráficos, exemplo:

```
[5]: import matplotlib.pyplot as plt
     import numpy as np
     from scipy.stats import norm

     x = np.arange(-3, 3, 0.001)

     plt.plot(x, norm.pdf(x))
     plt.show()
```



Esse gráfico sozinho fica muito vago dependendo do contexto, mas se você incrementar mais um fica mais objetivo.

```
[7]: plt.plot(x, norm.pdf(x))
     plt.plot(x, norm.pdf(x, 1.0, 0.5))
     plt.show()
```

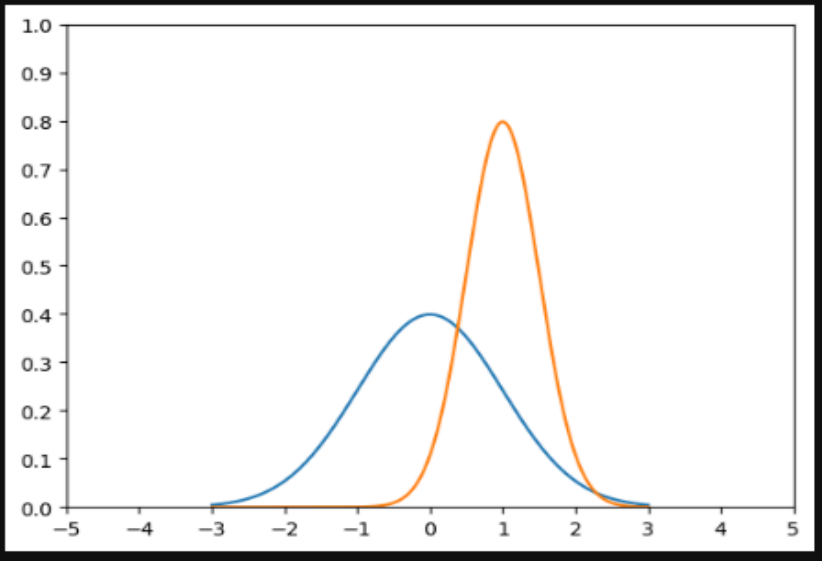


Com a biblioteca é possível salvar o gráfico que desejar na sua máquina, segue o exemplo:

```
[9]: plt.plot(x, norm.pdf(x))
     plt.plot(x, norm.pdf(x, 1.0, 0.5))
     plt.savefig('Exemplo.png', format='png')
```

A biblioteca permite dentro do gráfico você mudar os eixos x e y, para poder ter uma visualização melhor dentro dele, exemplo:

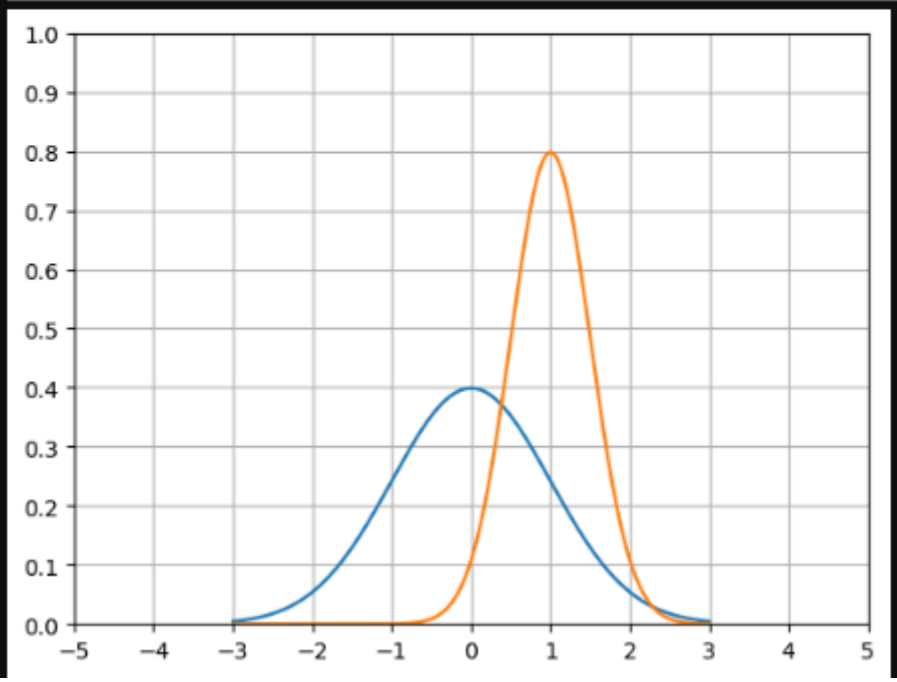
```
[11]: axes = plt.axes()
      axes.set_xlim([-5, 5])
      axes.set_ylim([0, 1.0])
      axes.set_xticks([-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5])
      axes.set_yticks([0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0])
      plt.plot(x, norm.pdf(x))
      plt.plot(x, norm.pdf(x, 1.0, 0.5))
      plt.show()
```



Foi incrementado o eixo x e y, para não ficar uma escala absoluta.

Para adicionar linhas de grade ao gráfico tem um método específico:

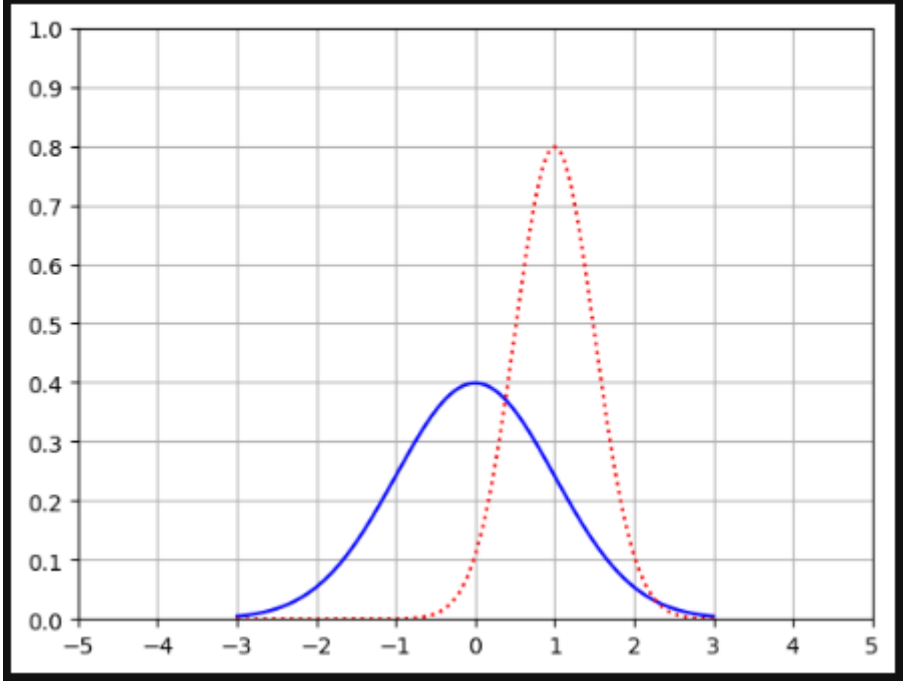
```
[13]: axes = plt.axes()
      axes.set_xlim([-5, 5])
      axes.set_ylim([0, 1.0])
      axes.set_xticks([-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5])
      axes.set_yticks([0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0])
      axes.grid()
      plt.plot(x, norm.pdf(x))
      plt.plot(x, norm.pdf(x, 1.0, 0.5))
      plt.show()
```



Aquele axes.grid() que torna possível isso.

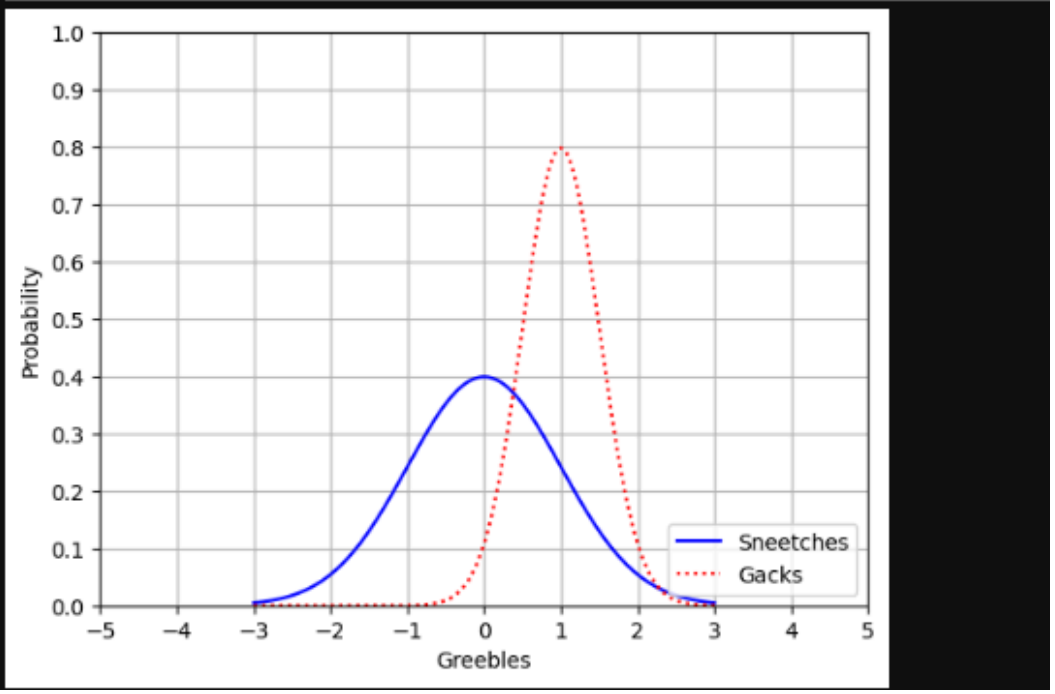
Conseguimos também alterar as cores e o jeito da linhas através de linhas de código, exemplo:

```
plt.plot(x, norm.pdf(x), 'b-')
plt.plot(x, norm.pdf(x, 1.0, 0.5), 'r:')
plt.show()
```



Para uma melhor visualização do gráfico a legenda é indispensável, exemplo:

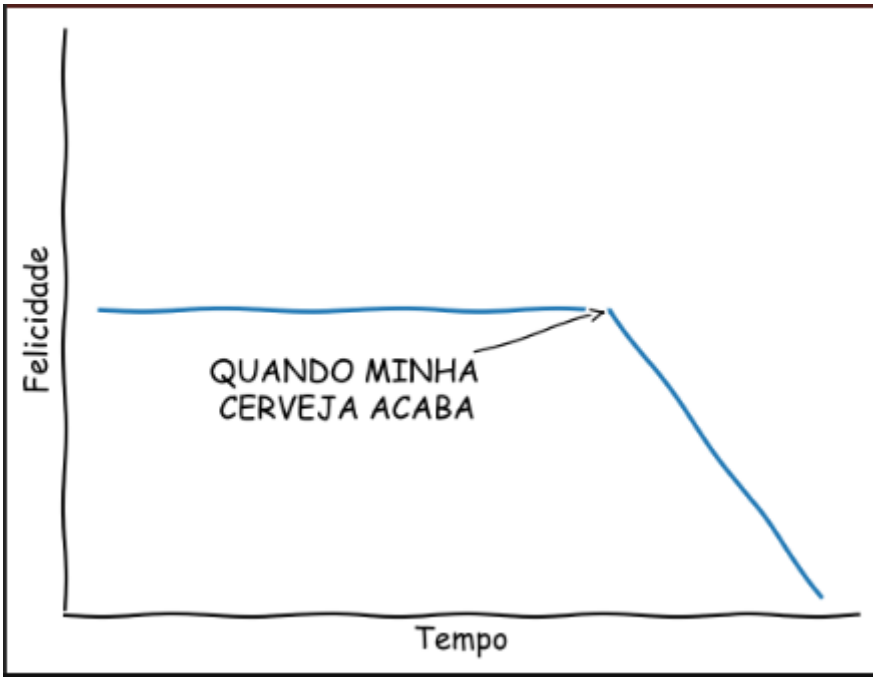
```
[17]: axes = plt.axes()
      axes.set_xlim([-5, 5])
      axes.set_ylim([0, 1.0])
      axes.set_xticks([-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5])
      axes.set_yticks([0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0])
      axes.grid()
      plt.xlabel('Greebles')
      plt.ylabel('Probability')
      plt.plot(x, norm.pdf(x), 'b-')
      plt.plot(x, norm.pdf(x, 1.0, 0.5), 'r:')
      plt.legend(['Sneetches', 'Gacks'], loc = 4)
      plt.show()
```



XKCD style é uma maneira de fazer humor utilizando gráficos, dentro da biblioteca tem um método que deixa os gráficos mais ‘humorizados’.

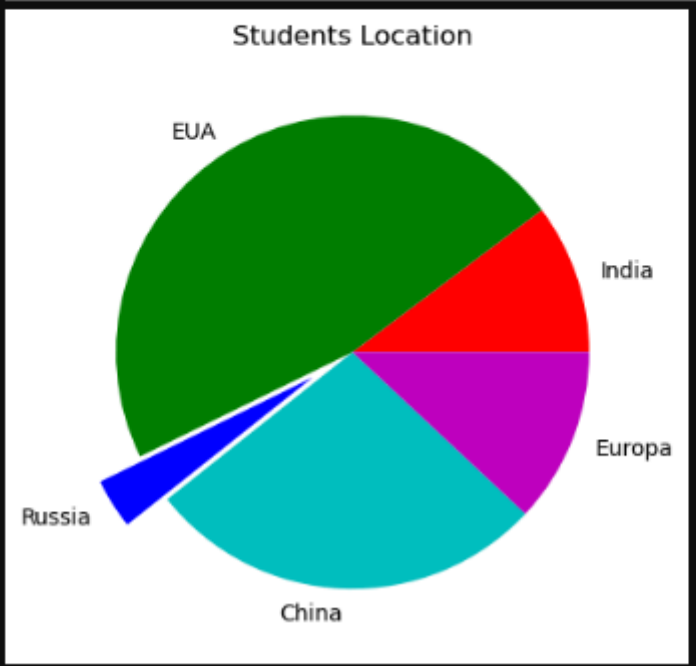
```
plt.xkcd()

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')
plt.xticks([])
plt.yticks([])
ax.set_ylim([-30, 30])

data = np.ones(100)
data[70:] -= np.arange(30)

plt.annotate(
    'QUANDO MINHA\n CERVEJA ACABA',
    xy=(70, 1), arrowprops=dict(arrowstyle='->'), xytext=(15, -10)
)
plt.plot(data)
plt.xlabel('Tempo')
plt.ylabel('Felicidade')
```

E esse é o resultado:



Gráficos em formato de pizza também é possível:

```
[31]: plt.rcdefaults() #retira o modo xkcd
      valores = [12,55,4,32,14]
      cores = ['r', 'g', 'b', 'c', 'm']
      explode = [0, 0, 0.2, 0, 0]
      labels = ['India', 'EUA', 'Russia', 'China', 'Europa']
      plt.pie(valores, colors=cores, labels=labels, explode=explode)
      plt.title('Students Location')
      plt.show()
```



Gráficos de barras:

```
[37]: valores = [12,55,4,32,14]
      cores = ['r', 'g', 'b', 'c', 'm']
      plt.bar(range(0,5), valores, color=cores)

[37]: <BarContainer object of 5 artists>
```

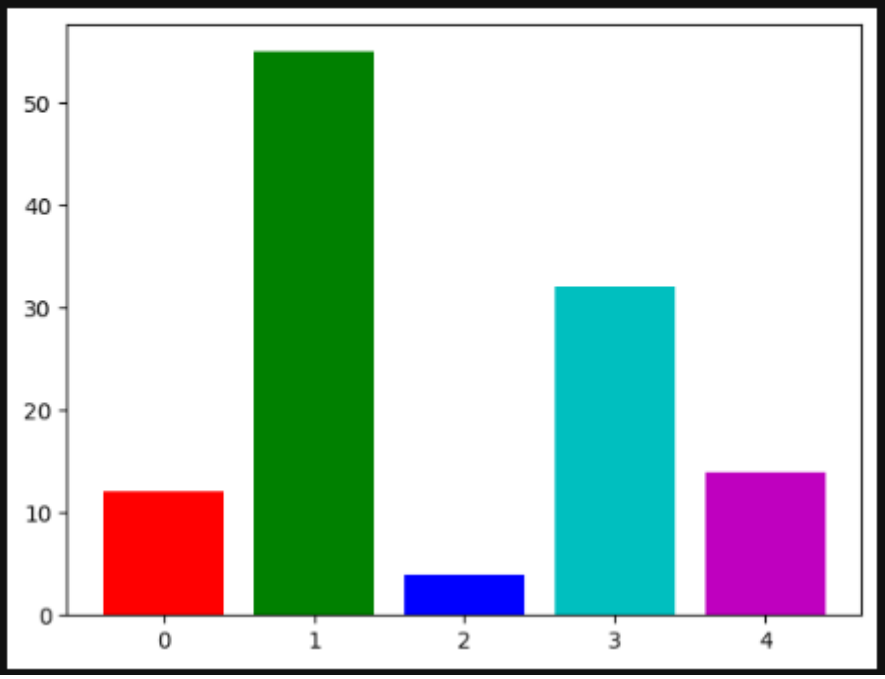
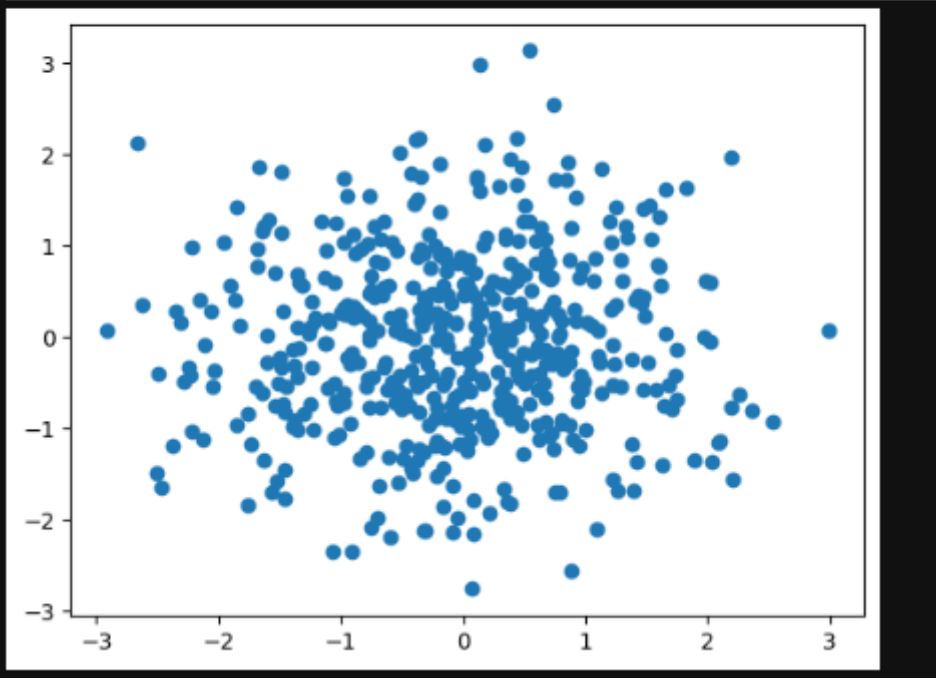


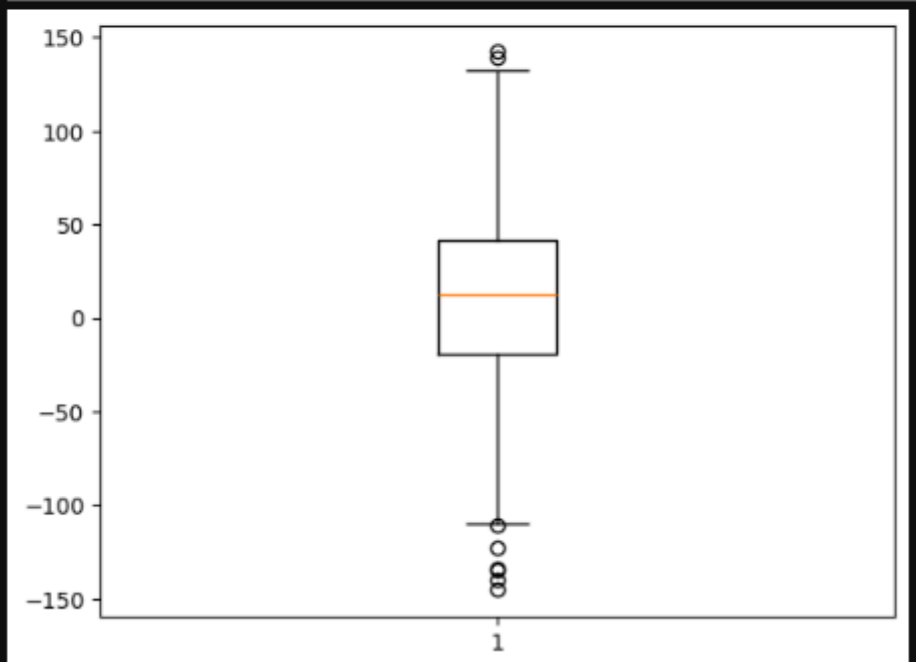
Gráfico de dispersão:

```
[ ]: x = np.random.randn(500)
     y = np.random.randn(500)
     plt.scatter(x,y)
     plt.show()
```



Box & Whisker plot é um tipo alternativo de histograma ele ajuda a visualizar localização, dispersão, assimetria, comprimento da cauda e outliers, exemplo:

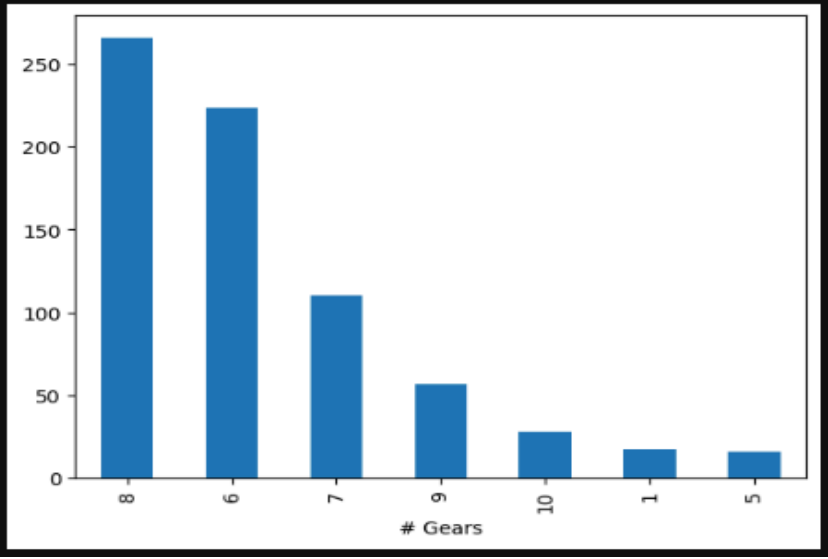
```
[43]: uniformSkewed = np.random.rand(100) * 100 - 40
      high_outliers = np.random.rand(10) * 50 + 100
      low_outliers = np.random.rand(10) * -50 - 100
      data = np.concatenate((uniformSkewed, high_outliers, low_outliers))
      plt.boxplot(data)
      plt.show()
```



Aula 9 - Advanced Visualization with Seaborn

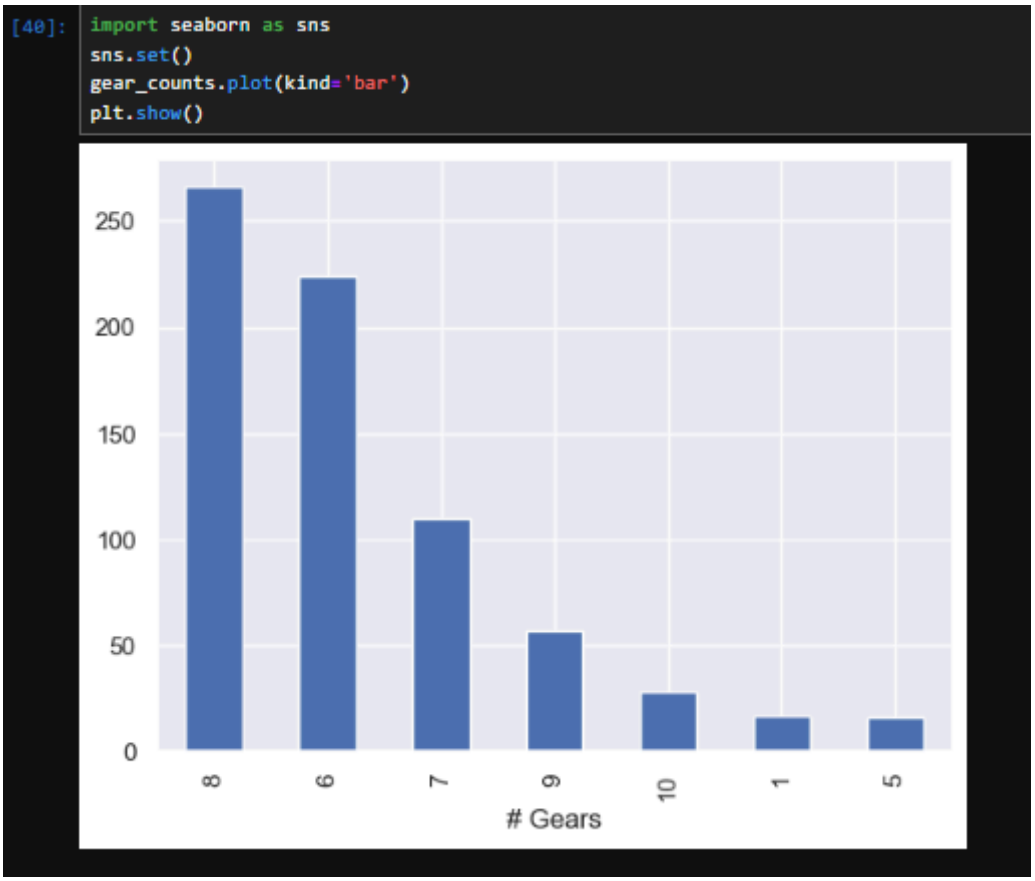
Seaborn é uma biblioteca de visualização, que deixa algumas tabelas do matplotlib mais bonitas mas também adiciona algumas tabelas e gráficos.

```
[30]: %matplotlib inline
      import matplotlib.pyplot as plt
      import pandas as pd

      df = pd.read_csv("http://media.sundog-soft.com/SelfDriving/FuelEfficiency.csv")

      gear_counts = df['# Gears'].value_counts()

      gear_counts.plot(kind='bar')
      plt.show()
```



Nesse código ele está lendo um arquivo csv disponibilizado na internet, transformando ele em um Data Frame, conta a frequência de cada valor na coluna ‘gears’ e retorna um gráfico de barras mostrando quantos carros têm aquele número de marchas.

O Seaborn vem para deixar mais agradável a visualização.



Seaborn que foi criado a partir do MatplotLib com o intuito de uma visualização melhor dos dados oferecendo uma interface mais agradável para a produção de gráficos e tabelas. Oferecendo uma enorme quantidade de ferramentas tanto para visualização quanto para a manipulação juntamente do pandas traz gráficos mais objetivos e mais amigáveis, como nos exemplos abaixo:

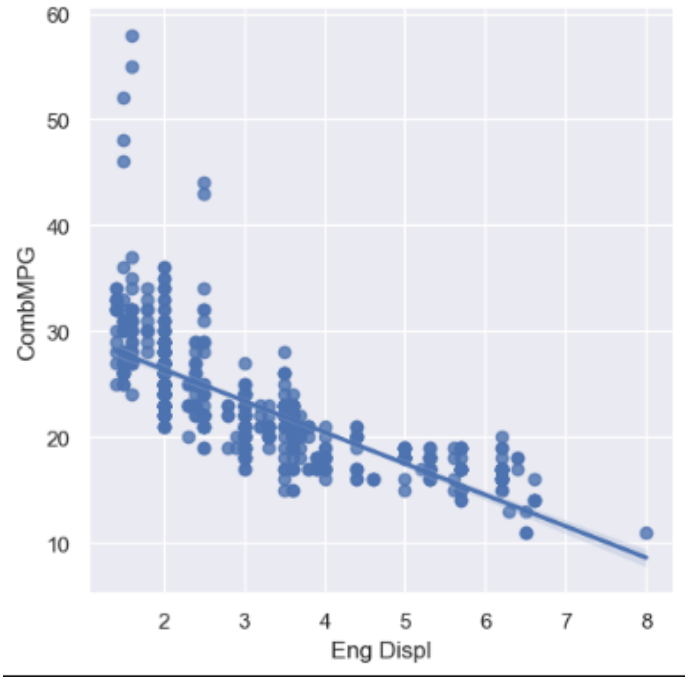
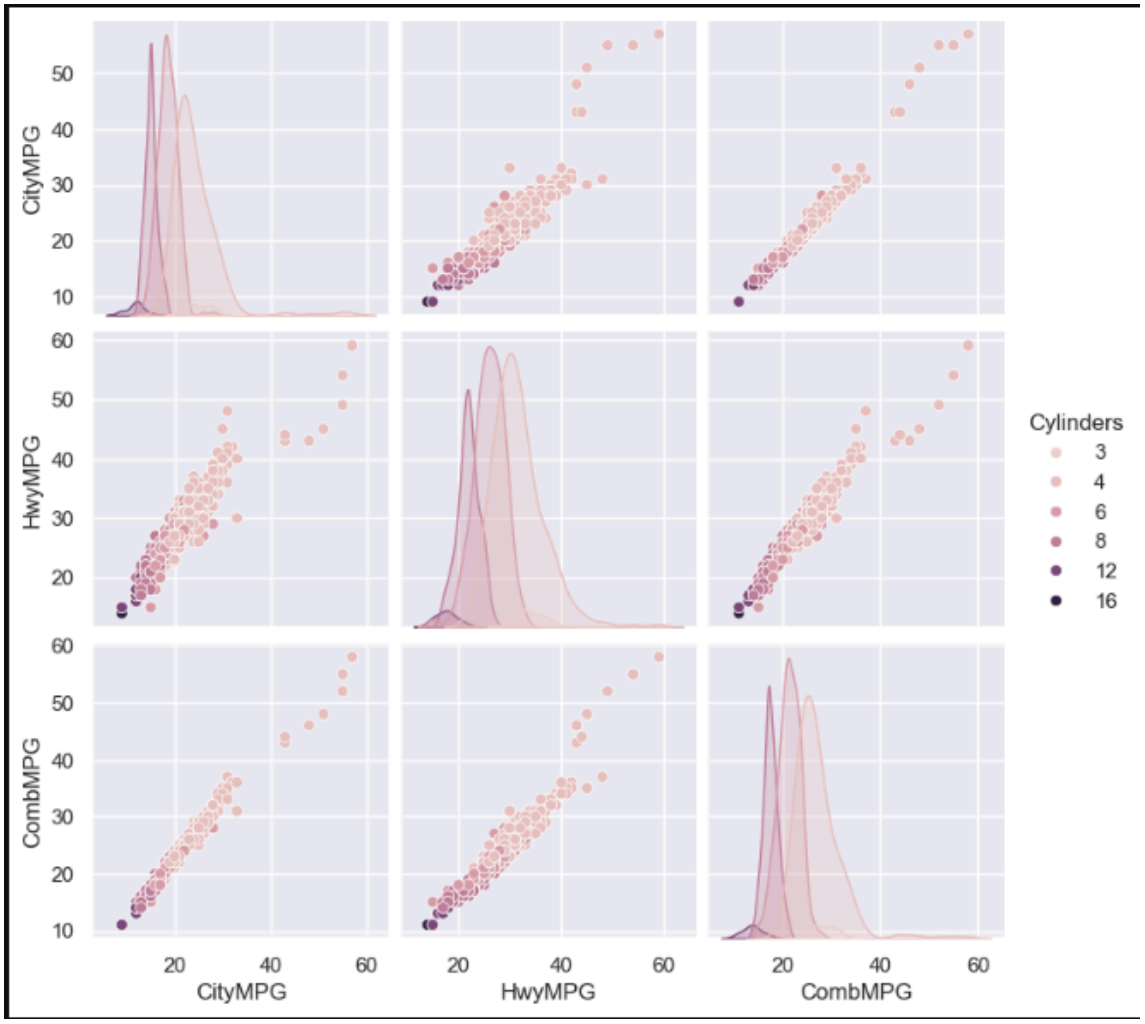
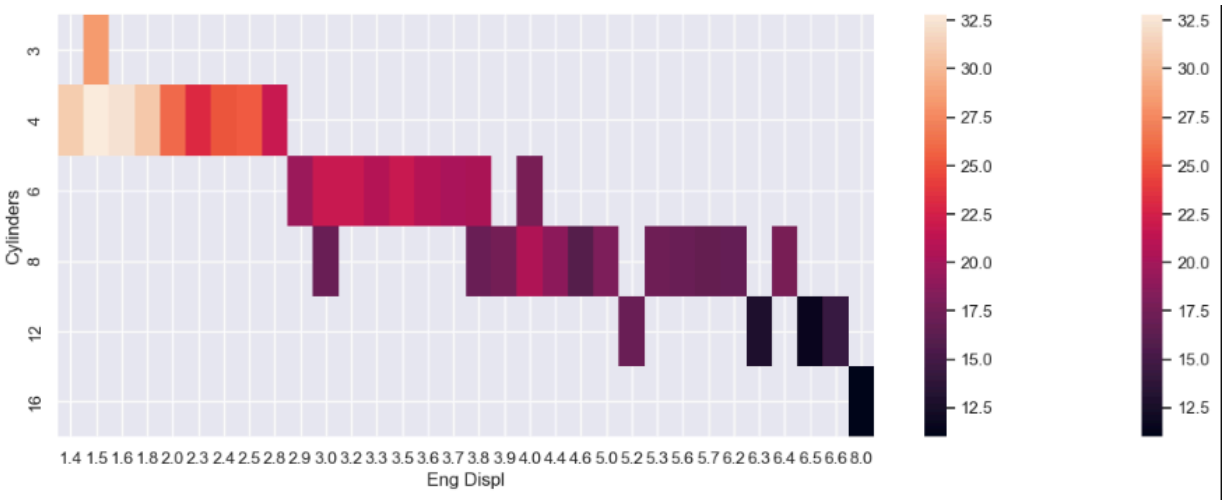


Gráfico de dispersão com uma linha de regressão.



Pair plot.



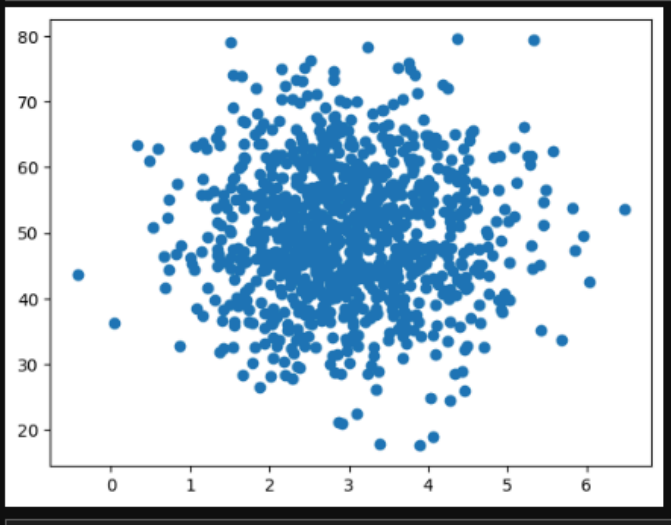
Mapa de calor 2d.

Aula 10 - Covariance and Correlation:

Covariância é a medida de quanto 2 variáveis se distanciam das médias.

Correlação é medida de -1:Correlação inversa perfeita, 0:Sem correlação alguma, 1:Correlação perfeita. Não é porque a correlação é alta que significa que um atributo é a causa do outro, apenas implica que eles estão relacionados de alguma maneira.

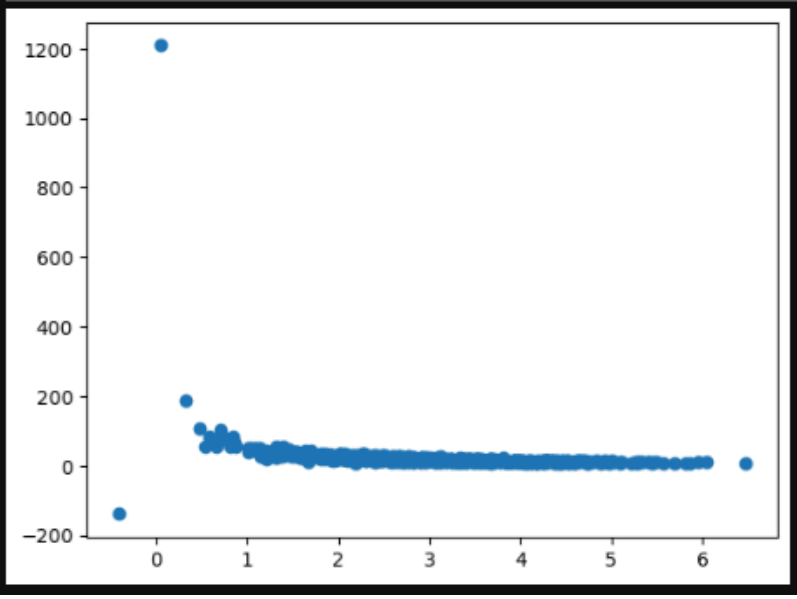
```
[7]: import matplotlib.pyplot as plt

     import numpy as np
     from pylab import *


     def de_mean(x):
         xmean = mean(x)
         return [xi - xmean for xi in x]


     def covariance(x, y):
         n = len(x)
         return dot(de_mean(x), de_mean(y)) / (n-1)

     pageSpeeds = np.random.normal(3.0, 1.0, 1000)
     purchaseAmount = np.random.normal(50.0, 10.0, 1000)

     scatter(pageSpeeds, purchaseAmount)

     covariance (pageSpeeds, purchaseAmount)

[7]: -0.04491651756461627

[9]: plt.show()
```



Aqui ele faz a análise entre 2 variáveis geradas aleatoriamente, e faz o cálculo da covariância. Logo com os dados são independentes o resultado da próximo de 0

Após isso ele altera os dados dividindo os valores de compra pelo tempo de carregamento, criando assim uma relação inversa entre as duas variáveis.

```
[11]: purchaseAmount = np.random.normal(50.0, 10.0, 1000) / pageSpeeds

      scatter(pageSpeeds, purchaseAmount)

      covariance (pageSpeeds, purchaseAmount)

[11]: -10.494825871292361

[13]: plt.show()
```



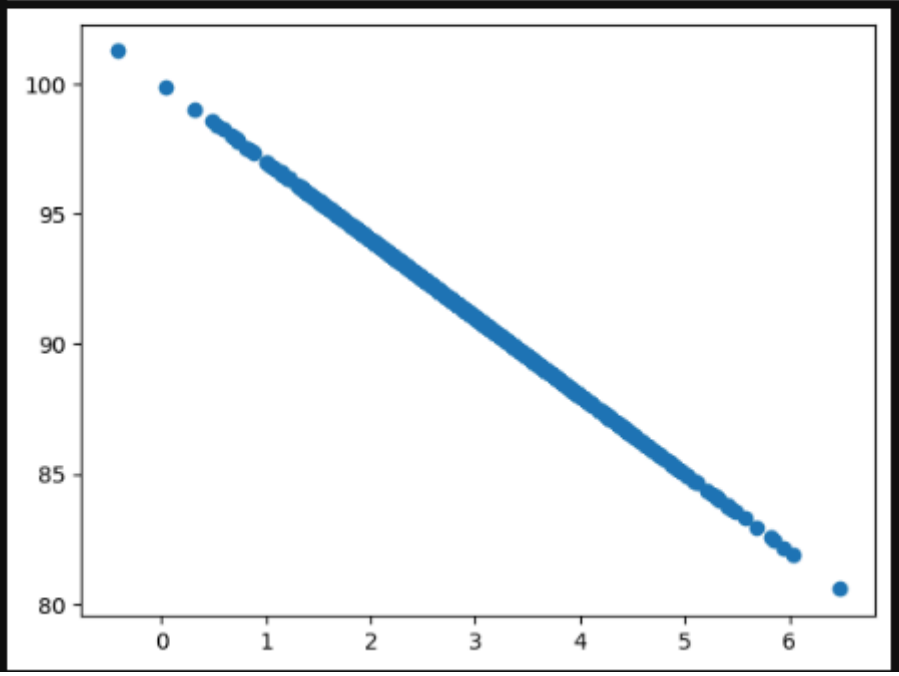
Após isso ele cria uma função para definir a correlação das duas variáveis e mostra como o numpy faz essa função:

```
[15]: def correlation(x, y):
          stddevx = x.std()
          stddevy = y.std()
          return covariance(x,y) / stddevx / stddevy

      correlation(pageSpeeds, purchaseAmount)

[15]: -0.27139059802250576

[17]: np.corrcoef(pageSpeeds, purchaseAmount)

[17]: array([[ 1.        , -0.27111921],
             [-0.27111921,  1.        ]])
```

O numpy relaciona todas as possíveis combinações, por isso resulta em 1 duas vezes, porém é apenas a correlação da variável com ela mesma.

Após isso ele altera os dados para poder mostrar uma correlação inversamente perfeita:

```
[19]: purchaseAmount = 100 - pageSpeeds * 3

      scatter(pageSpeeds, purchaseAmount)

      correlation (pageSpeeds, purchaseAmount)

[19]: -1.001001001001001

[21]: plt.show()
```



Aula 11 - Exercise Conditional Probability

A probabilidade condicional mede a probabilidade de um evento X acontecer dado outro evento Y.

```
[1]: from numpy import random
     random.seed(0)

     totals = {20:0, 30:0, 40:0, 50:0, 60:0, 70:0}
     purchases = {20:0, 30:0, 40:0, 50:0, 60:0, 70:0}
     totalPurchases = 0
     for _ in range(100000):
         ageDecade = random.choice([20, 30, 40, 50, 60, 70])
         purchaseProbability = float(ageDecade) / 100.0
         totals[ageDecade] += 1
         if (random.random() < purchaseProbability):
             totalPurchases += 1
             purchases[ageDecade] += 1
```

Aqui neste código ele passa 2 dicionários com as mesmas chaves de 20 a 70, porém com valor 0, abre um for para escolher randomicamente uma idade de 20 a 70, após a idade ser sorteada ele faz as chance de comprá ser baseada na idade, logo se a pessoa tiver 50 anos ela tem 50% de chances de comprar algo, e adiciona uma pessoa para o dicionário de totais para sua idade

correspondente, após isso entra em um if que sorteia um número de 0 a 1, que verifica se a probabilidade de comprá é maior que o número randômico, se for é adicionado 1 compra no total de compras e adicionado 1 no dicionário de compras.

```
[3]: totals

[3]: {20: 16576, 30: 16619, 40: 16632, 50: 16805, 60: 16664, 70: 16704}

[7]: purchases

[7]: {20: 3392, 30: 4974, 40: 6670, 50: 8319, 60: 9944, 70: 11713}
```

Como se pode ver o número de pessoas ficou bem parelho, porém o número de compras foi aumentando de acordo com a idade.

```
[27]: PEF = float(purchases[30]) / float(totals[30])
      print('P(purchase | 30s): ' + str(PEF))

      P(purchase | 30s): 0.29929598652145134

[29]: PF = float(totals[30]) / 100000.0
      print("P(30's): " +  str(PF))

      P(30's): 0.16619
```

Aqui ele faz um cálculo para ver qual a chance de alguém de 30 fazer alguma compra(restringindo a pessoas de 30 anos), e depois faz o cálculo para ver qual a chance de alguém ter 30 anos.

```
[31]: PE = float(totalPurchases) / 100000.0
      print("P(Purchase):" + str(PE))

      P(Purchase):0.45012
```

Aqui ele contabiliza a porcentagem de alguém fazer compras.

```
[33]: print("P(30's, Purchase)" + str(float(purchases[30]) / 100000.0))

      P(30's, Purchase)0.04974
```

Aqui a porcentagem de alguém ter 30 anos e comprar alguma coisa(comparando todas as idades).

Conclusões

A conclusão que eu chego após esse capítulo de estudos de probabilidade e estatística é que com um maior conhecimento dessa área podemos analisar e manipular dados de uma maneira que favorece a visualização e o entendimento deles. Fora que é uma área importantíssima para o machine learning que precisa de todo esse tratamento de dados.

Referências

https://www.escolaedti.com.br/o-que-e-um-box-plot/#:~:text=O%20Box%20Plot%20(tamb%C3%A9m%20chamado,e%20outliers%20(medidas%20discrepantes).

https://www.datacamp.com/pt/tutorial/seaborn-python-tutorial