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

Igor Carvalho Marchi

Descrição da atividade

Aula 1 – Tipos de Dados

- Numéricos: dados quantificáveis. Podem ser:
- Discretos: números inteiros que representam contagens (ex: número de dias de chuva por ano).
- Contínuos: valores com precisão infinita (ex: quantidade de chuva em milímetros).
- Categóricos: representam categorias sem valor numérico (ex: gênero, cor dos olhos).
- Ordinais: categorias com uma ordem definida (ex: classificação por estrelas de um filme).
-

Aula 2 – Média, Mediana e Moda

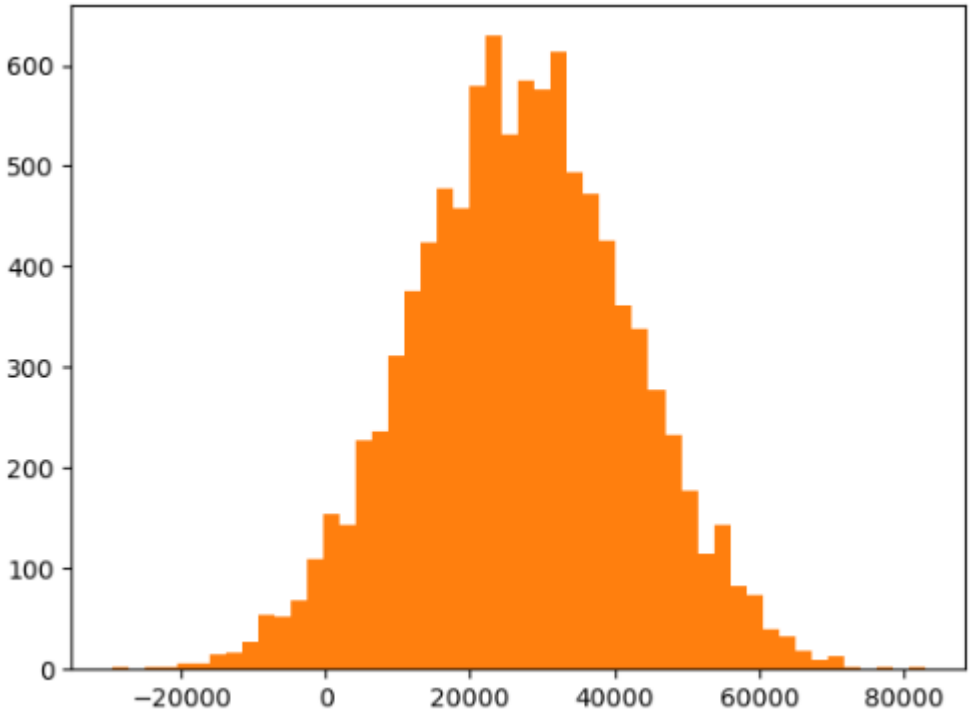
- Média: soma dos valores dividida pela quantidade total.
- Mediana: valor central em uma lista ordenada (ou a média dos dois centrais, se a quantidade for par).
- Moda: valor que mais se repete.

Aula 3 – Média, Mediana e Moda em Python

Utilizando a biblioteca matplotlib, é possível visualizar histogramas com distribuição de dados. Foi demonstrado como a média pode ser influenciada por valores extremos, ao contrário da mediana, que se mantém mais estável. Também foi mostrado como calcular a moda com um conjunto de idades geradas aleatoriamente.

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

[2]: 26993.095136098134
```

A imagem acima 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.




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

[5]: 27064.916213427605

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

[7]: np.median(incomes)

[7]: 27065.52025586698

[8]: np.mean(incomes)

[8]: 126980.39709638848
```

Na primeira imagem 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, depois 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.

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

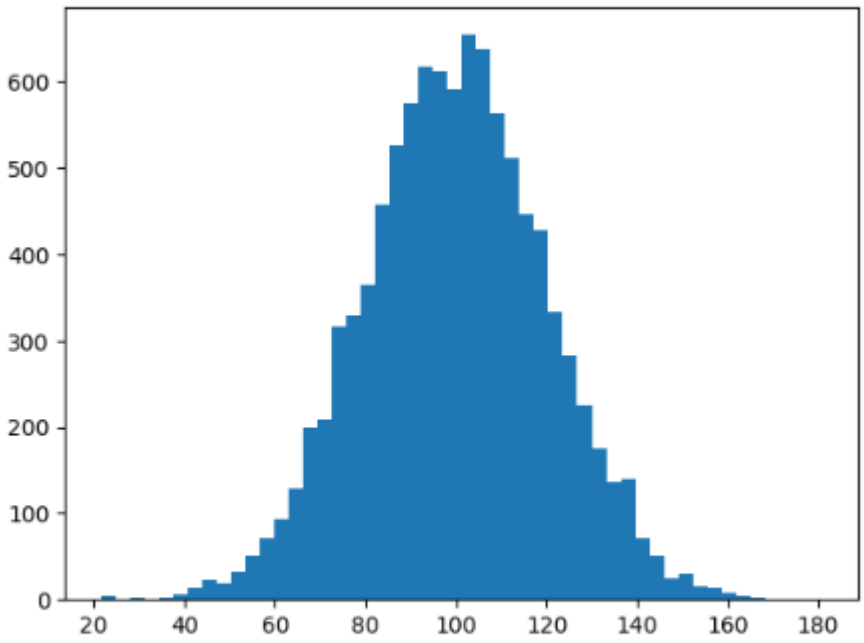
```
[12]: array([23, 85, 55, 80, 68, 57, 78, 19, 69, 55, 43, 74, 76, 26, 75, 77, 88,
             88, 27, 35, 53, 80, 44, 77, 57, 83, 45, 20, 61, 69, 26, 70, 39, 24,
             59, 28, 71, 84, 75, 81, 60, 66, 58, 87, 20, 31, 60, 44, 34, 26, 36,
             47, 18, 65, 70, 41, 71, 46, 43, 31, 72, 47,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

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.

Aula 4 – Variação e Desvio Padrão

- Variância: média dos quadrados das diferenças entre os valores e a média.
- Desvio padrão: raiz quadrada da variância.
  Essas métricas ajudam a identificar valores atípicos. Se estivermos lidando com uma amostra, usamos N−1 no cálculo da variância, em vez de N.
  O NumPy fornece métodos para calcular variância e desvio padrão diretamente.

```
[1]: import numpy as np
     import matplotlib.pyplot as plt
     incomes = np.random.normal(100.0, 20.0, 10000)
     plt.hist(incomes, 50)
     plt.show()
```




```
[4]: incomes.std() #desvio padrão

[4]: 19.991704057271765

[5]: incomes.var() #variancia

[5]: 399.6682311135363
```

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 – PDF e PMF

- PDF (Probability Density Function): descreve a densidade de probabilidade em torno dos valores de uma variável contínua.
- PMF (Probability Mass Function): aplica-se a variáveis discretas, fornecendo a probabilidade exata de cada valor.

Aula 6 – Distribuições de Dados

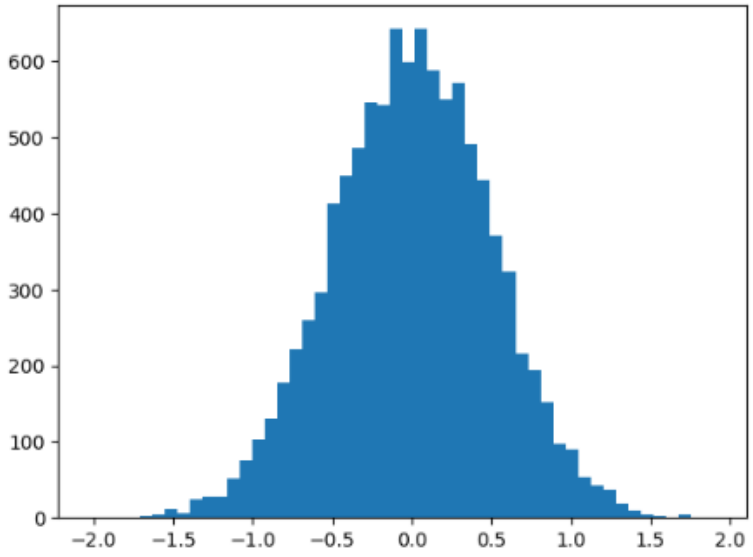
- Uniforme: probabilidade constante para todos os valores.
- Gaussiana (Normal): simétrica, com concentração de dados ao redor da média.
- Exponencial: alta probabilidade de eventos próximos a zero.
- Binomial: usada para contar sucessos em tentativas independentes (ex: cara ou coroa).

- Poisson: modela o número de eventos que ocorrem em um intervalo fixo, com taxa constante.

Aula 7 – Porcentagem e Momentos

- **Percentis:** indicam o ponto abaixo do qual está uma determinada porcentagem dos dados.
- **Momentos Estatísticos:**
    1. Média.
    2. Variância.
    3. Assimetria (skewness): indica o grau de inclinação da distribuição.
    4. Curtose: mede o achatamento (ou pontiagudez) da distribuição.
    Os dois últimos requerem a biblioteca scipy.

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

     plt.hist(vals, 50)
     plt.show()
```




```
[4]: np.percentile(vals, 50)
[4]: 0.010456501977953073

[5]: np.percentile(vals, 90)
[5]: 0.6327738023654492

[7]: np.percentile(vals, 20)
[7]: -0.4181191352588024

[8]: np.percentile(vals, 99)
[8]: 1.1555702046095522
```

Esses são os valores abaixo está usando o gráfico acima como base para as informações dos dados que de acordo com as seguintes respectivas porcentagens de 50%, 90%, 20% e 99% são mostradas.

Momentos são basicamente maneiras de medir a forma de uma distribuição.

Existem 4 tipos de momentos:

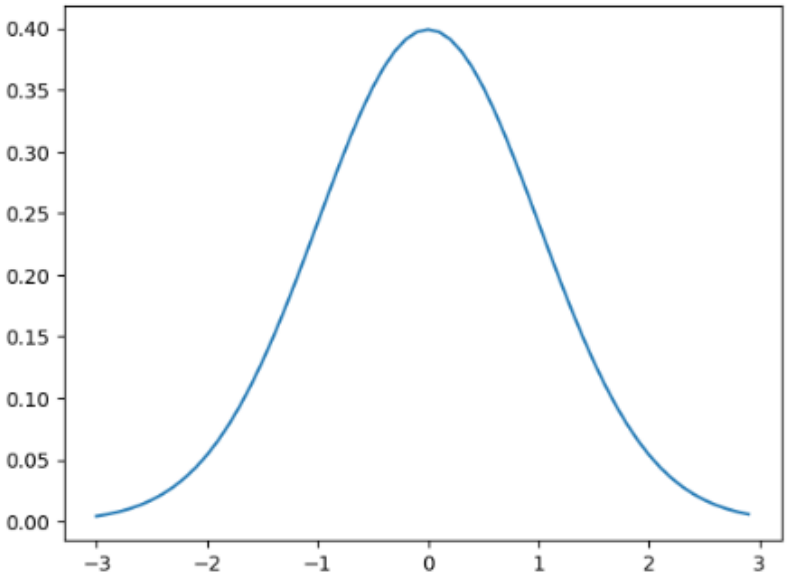
Aula 8 – Introdução ao Matplotlib

A biblioteca Matplotlib permite a criação de diversos tipos de gráficos, como:

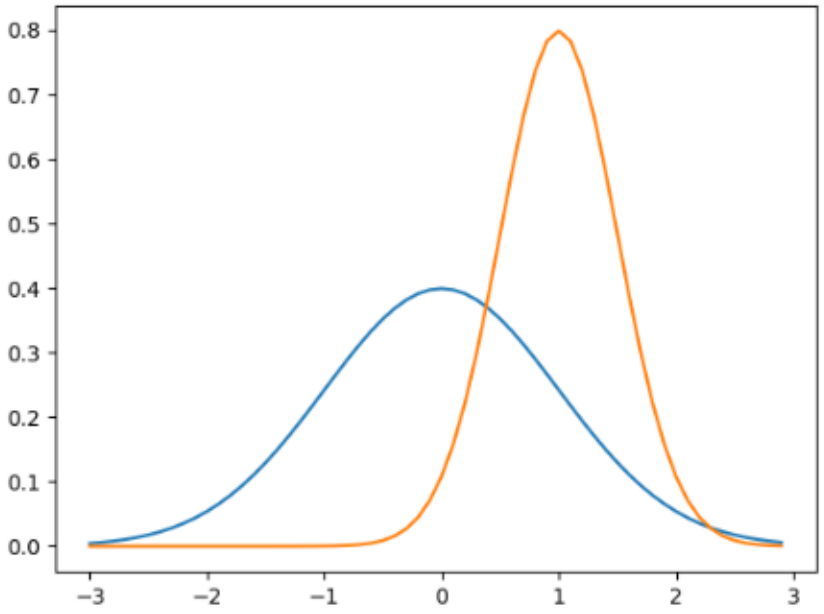
- Histogramas
- Gráficos de linha
- Gráficos de dispersão
- Gráficos de pizza
- Boxplots
- É possível personalizar cores, adicionar grades, alterar eixos, salvar gráficos e até usar o estilo “XKCD” para humor gráfico.

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

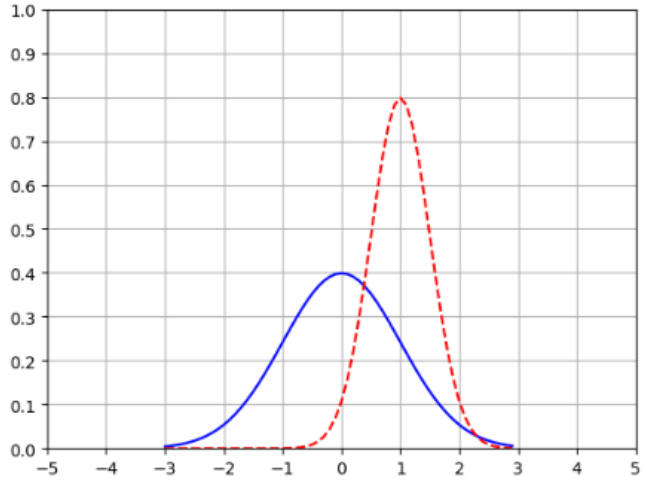
[3]:
```
from scipy.stats import norm

import matplotlib.pyplot as plt

x = np.arange(-3, 3, 00.1)
plt.plot(x, norm.pdf(x))
plt.show()
```



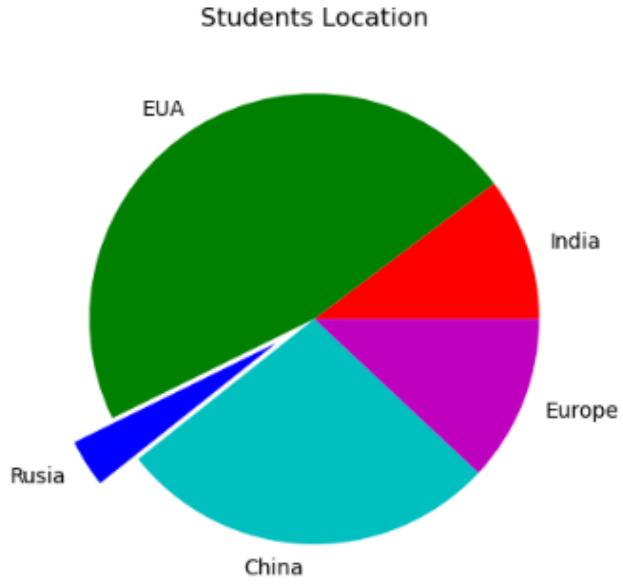
[5]:
```
plt.plot(x, norm.pdf(x))
plt.plot(x, norm.pdf(x, 1.0, 0.5))
plt.show()
```



[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])
axes.grid()
plt.plot(x, norm.pdf(x), 'b-')
plt.plot(x, norm.pdf(x, 1.0, 0.5), 'r--')
plt.show()
```



[19]:
```
plt.rcdefaults()

values = [12, 55, 4, 32, 14]
colors = ['r', 'g', 'b', 'c', 'm']
explode = [0, 0, 0.2, 0, 0]
labels = ['India', 'EUA', 'Rusia', 'China', 'Europe']
plt.pie(values, colors= colors, labels= labels, explode= explode)
plt.title('Students Location')
plt.show()
```

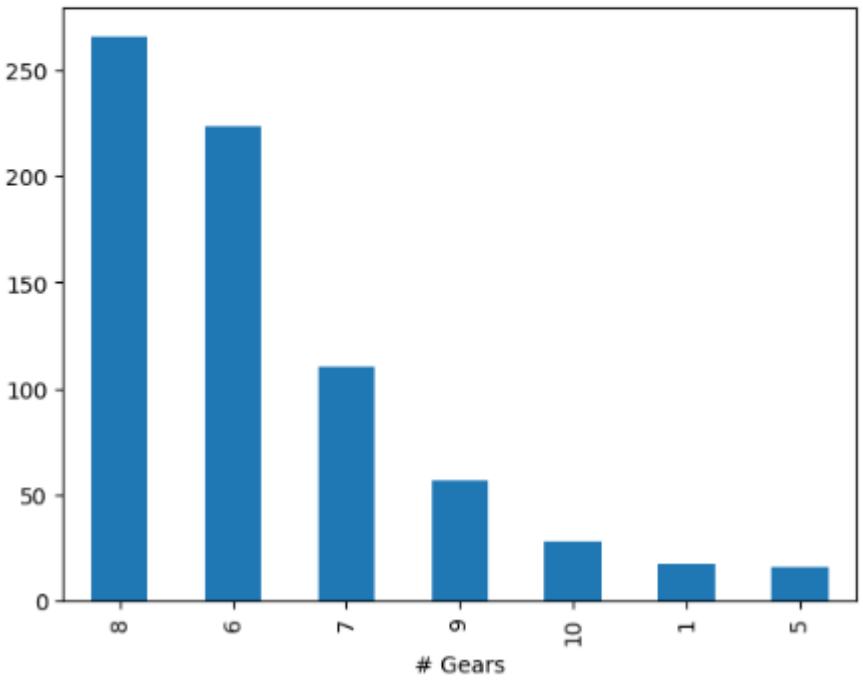


Aula 9 – Visualização Avançada com Seaborn

Seaborn é uma biblioteca baseada no Matplotlib, com foco em visualizações mais elegantes e intuitivas. Possui suporte nativo para:

- Gráficos de dispersão com linha de regressão
- Pair plots
- Mapas de calor
  Foi exemplificado com a leitura de um CSV, análise da coluna 'gears' e visualização com gráfico de barras.

```
%matplotlib inline
import pandas as pd
import matplotlib.pyplot as plt

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()
```

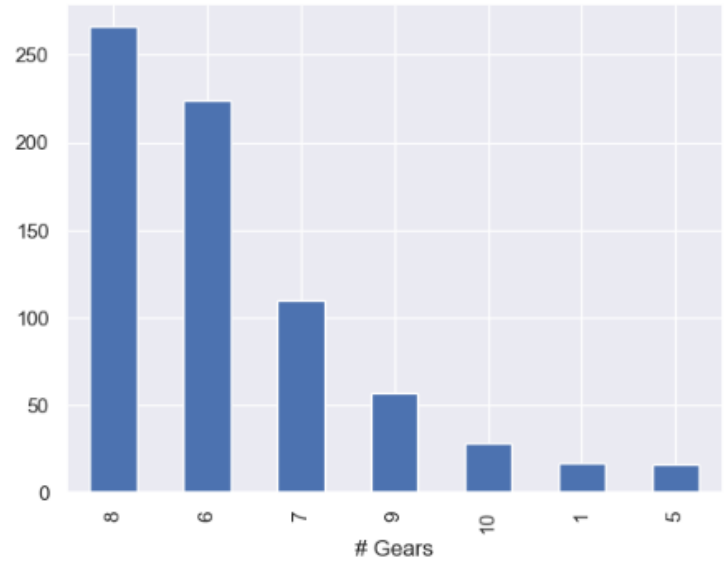


Nesse código ele está lendo um arquivo csv disponibilizado na internet, transformando-o em um Data Frame, conta a frequência de cada valor na coluna ‘gears’ e retorna um gráfico de barras mostrando quantos carros têm aquele número de marchas.

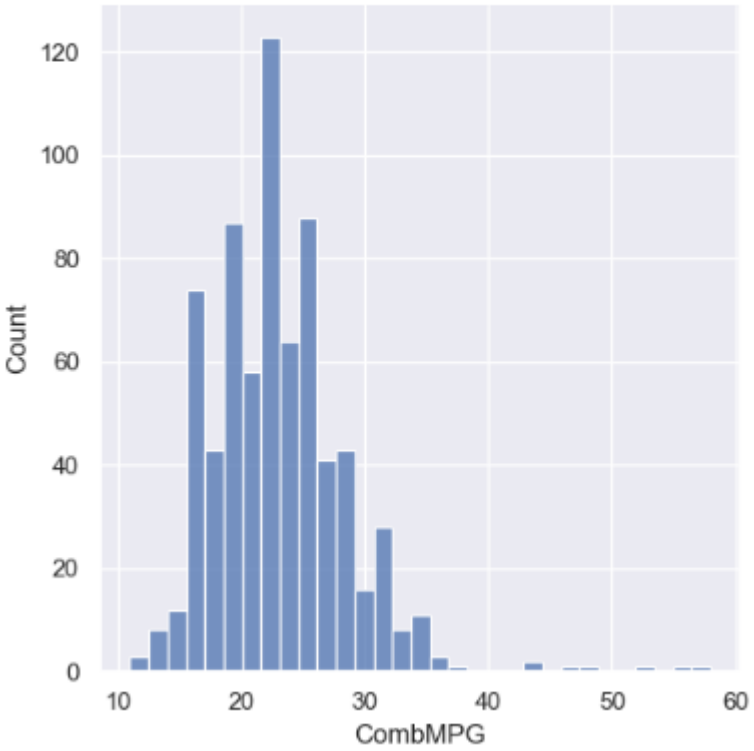
O Seaborn oferecendo uma enorme quantidade de ferramentas tanto para visualização quanto para a manipulação juntamente do pandas traz gráficos mais objetivos e mais amigáveis, como nos exemplos abaixo:

```
import seaborn as sns
sns.set()
```

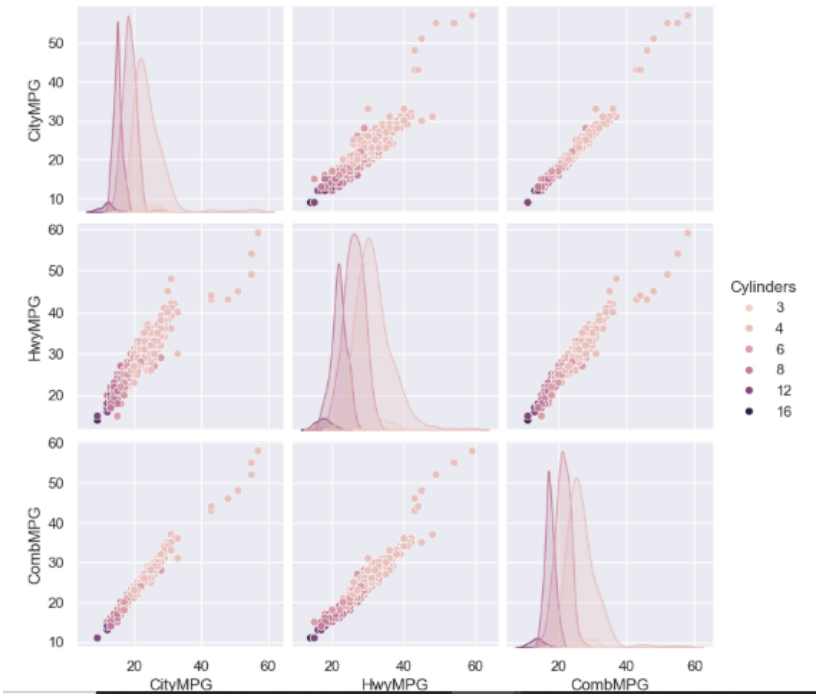
```
gear_counts.plot(kind='bar')
plt.show()
```



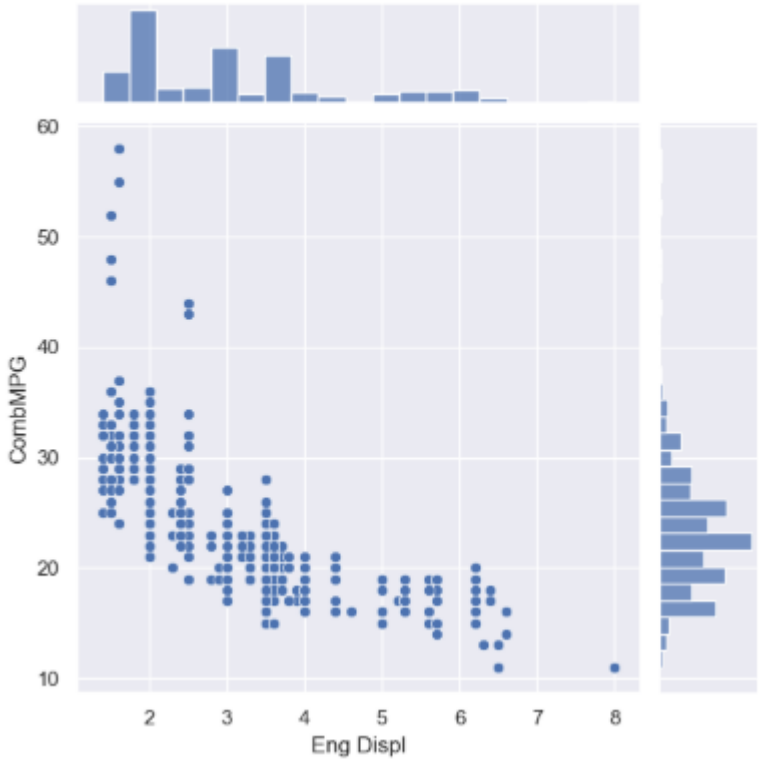

```
sns.displot(df['CombMPG'])
plt.show()
```



```
sns.pairplot(df2, hue='Cylinders', height=2.5);
plt.show()
```




```
sns.jointplot(x="Eng Displ", y="CombMPG", data=df)
plt.show()
```

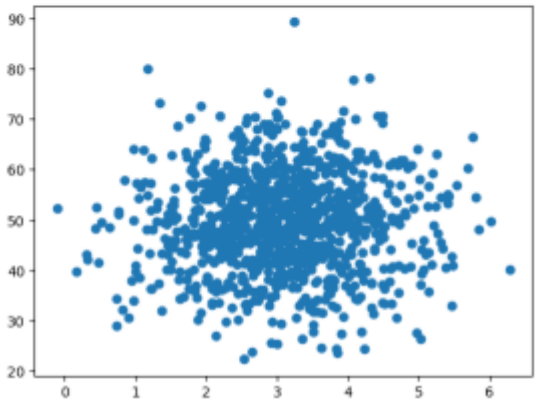


Aula 10 – Covariância e Correlação

- Covariância: mede a variação conjunta entre duas variáveis.

- Correlação: mede a força e direção da relação entre variáveis, com valores entre -1 e 1.
  Foi demonstrado como alterar variáveis para observar diferentes tipos de correlação com numpy.

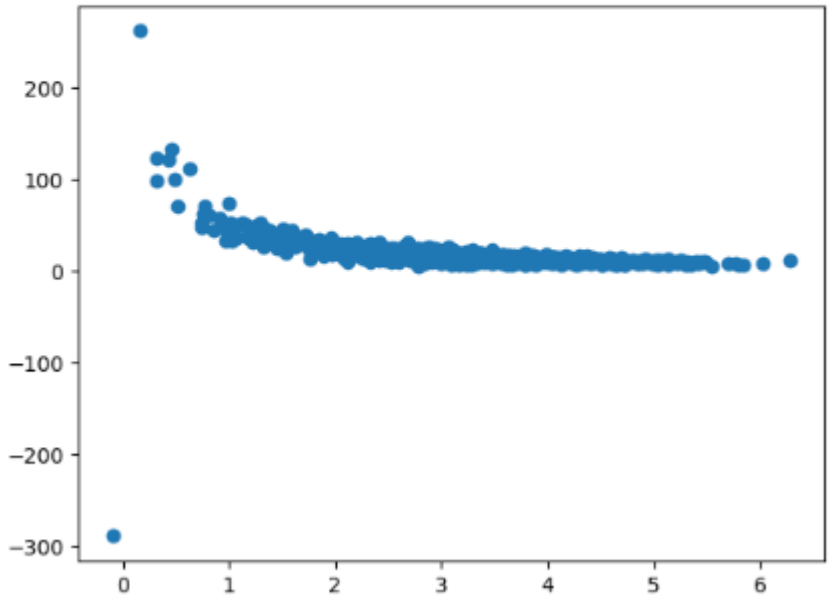
```
[1]: import matplotlib.pyplot as plt

     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)

[1]: 8.1434610884984059
```



Na imagem acima ele faz a análise entre 2 variáveis geradas aleatoriamente, e faz o cálculo da covariância. Logo com os dados são independentes o resultado do 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.

```
purchaseAmount = np.random.normal(50.0, 10.0, 1000) / pageSpeeds

scatter(pageSpeeds, purchaseAmount)
covariance(pageSpeeds, purchaseAmount)
```

```
-7.726126760575221
```



Após isso ele cria uma função para definir a correlação das duas variáveis:

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

correlation(pageSpeeds, purchaseAmount)
```

```
-0.4598249548345662
```

```
np.corrcoef(pageSpeeds, purchaseAmount)
```

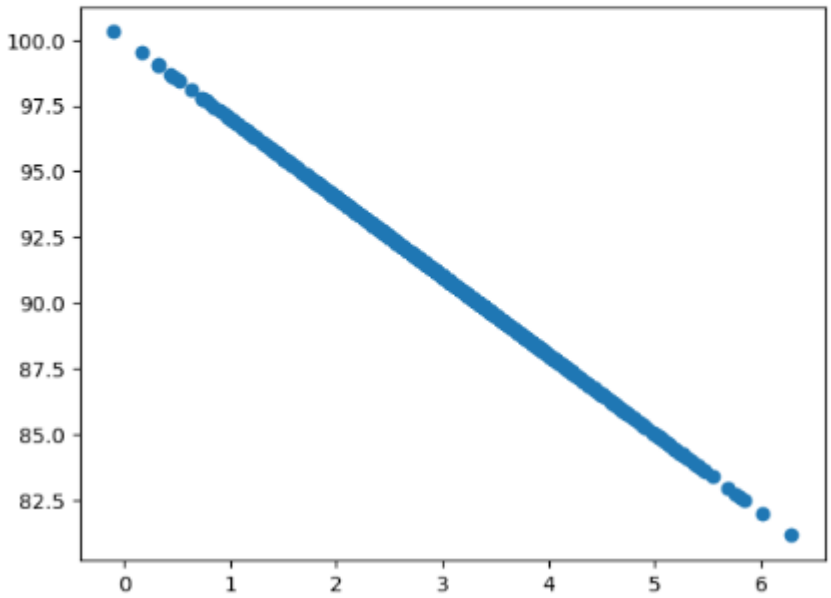
```
array([[ 1.        , -0.45936513],
       [-0.45936513,  1.        ]])
```

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

Depois disso ele altera os dados para poder demostrar uma correlação inversamente perfeita:

```
purchaseAmount = 100 - pageSpeeds * 3

scatter(pageSpeeds, purchaseAmount)

correlation (pageSpeeds, purchaseAmount)
```

```
-1.001001001001001
```



Aula 11 – Probabilidade Condicional

A probabilidade condicional calcula a chance de um evento ocorrer dado que outro já ocorreu.
Foi implementado um exemplo prático simulando compras por diferentes faixas etárias, relacionando idade e chance de compra. O código permitiu calcular probabilidades específicas, como:

- Probabilidade de alguém de 30 anos fazer uma compra.
- Probabilidade de uma pessoa ter 30 anos.
- Probabilidade conjunta de ser uma pessoa de 30 anos e fazer uma compra.

```
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
```

Neste trecho do código, são criados dois dicionários com as mesmas chaves (de 20 a 70), todas inicialmente com valor zero. Em seguida, é utilizado um laço de repetição para selecionar aleatoriamente uma idade entre 20 e 70. A chance de uma pessoa realizar uma compra é proporcional à sua idade.

Cada vez que uma idade é sorteada, o total de pessoas daquela faixa etária é incrementado no dicionário correspondente. Em seguida, é gerado um número aleatório entre 0 e 1. Se esse número for menor que a probabilidade baseada na idade, considera-se que a compra foi realizada, portanto, o total de compras para aquela idade é incrementado.

```
totals
```

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

```
purchases
```

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

O resultado da simulação mostra que a distribuição de pessoas por idade é bastante equilibrada. No entanto, o número de compras aumenta à medida que a idade cresce, refletindo a lógica probabilística aplicada.

Após isso, o código realiza os seguintes cálculos:

- A probabilidade de uma pessoa de 30 anos realizar uma compra (considerando apenas indivíduos dessa idade).
- A probabilidade de uma pessoa ter 30 anos, independentemente de comprar ou não.
- A porcentagem geral de pessoas que fizeram compras.
- A probabilidade conjunta de uma pessoa ter 30 anos e realizar uma compra (comparando com o total de pessoas simuladas).

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

```
P(purchase | 30s): 0.29929598652145134
```

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

```
P(30's): 0.16619
```

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

```
P(Purchase):0.45012
```

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

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

Conclusões

O estudo de estatística e probabilidade é essencial para a análise de dados em Machine Learning. Com esses conhecimentos, é possível entender melhor a distribuição dos dados, identificar padrões, lidar com outliers e aplicar técnicas adequadas para visualização e tratamento dos dados. As bibliotecas NumPy, Matplotlib, Seaborn e Scipy se mostraram ferramentas indispensáveis nesse processo.

Referencias

DataCamp – Seaborn Tutorial:
https://www.datacamp.com/pt/tutorial/seaborn-python-tutorial