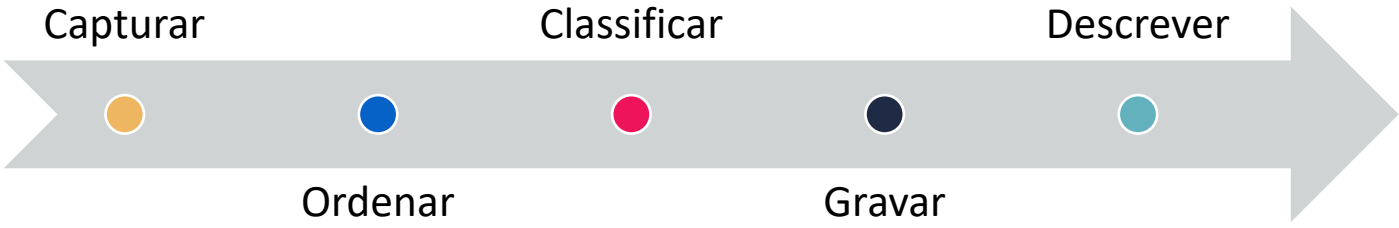
Drª. Marise Miranda
Msc. Eduardo Verri

Roadmap dos dados (por enquanto)



Até então coletamos os dados, utilizando a biblioteca psutil no python. Para podermos gravar no banco de dados, é necessário o ordenamento desses dados, que fizemos ao separar os dados durante o script de coleta, classificamos ao separar em conjunto conjuntos e então gravamos no banco local.

Agora necessitamos descrevermos esses dados para depois trabalharmos na modelagem desses dados.

Estatística Descritiva

Durante o processo de estudo dos dados a estatística se torna uma ferramenta importante para o entendimento desse estudo. Podemos englobar a estatística em duas etapas:

1. Estatística descritiva: coleta, organização e descrição dos dados
2. Estatística indutiva (ou inferencial): análise e interpretação dos dados.

Trabalharemos num primeiro momento apenas com a estatística descritiva, que consiste no simples levantamento da informação e apresentação em tabelas e/ou gráficos estatísticos de um dado fenômeno. É importante que a informação seja organizada para que possamos realizar uma posterior análise. Esta análise nos permite caracterizar como os elementos se distribuem, quais os valores de tendência central, qual a variabilidade, qual a relação entre as variáveis e a verificação das semelhanças e diferenças entre os elementos.

A estatística descritiva sumaria os dados e descreve as principais características de uma base de dados.

As principais métricas para trabalhar com estatística descritiva são:

1. Medidas de tendência central: Essas medidas ajudam a descrever o centro ou a média de uma base de dados
   1.1. Média: média aritmética de todos os valores da base
   1.2. Mediana: o valor do meio quando o conjunto de dados é organizado, menos sensível à *outliers* comparado à média
   1.3. Moda: o valor que aparece com maior frequência na base de dados
2. Medidas de dispersão: Essas medidas fornecem informações sobre a disseminação ou variabilidade dos dados

2.1. Alcance: a diferença entre o valor mínimo e máximo

2.2. Variância: a média das diferenças quadráticas entre cada ponto de dados e a média.
2.3. Desvio padrão: a raiz quadrada da variância; indica a distância típica entre os pontos de dados e a média

3. Percentis e Quartis: Eles dividem os dados em partes e ajudam a entender a distribuição.
   3.1. Percentis: o valor abaixo do qual cai uma determinada porcentagem de observações. Por exemplo, o percentil 25 é o valor abaixo do qual se encontram 25% dos dados.
   3.2. Quartis: percentis específicos que dividem os dados em quatro partes iguais.
4. Skewness e curtose:
   4.1. Skewness: Mede a assimetria da distribuição dos dados. A assimetria positiva indica uma cauda para a direita (cauda direita mais longa), enquanto a assimetria negativa indica uma cauda para a esquerda.
   4.2. Curtose: Mede a "cauda" da distribuição. A curtose alta indica caudas mais pesadas do que uma distribuição normal, enquanto a curtose baixa indica caudas mais leves.
5. Distribuições de frequência e histogramas:
   5.1. Distribuição de frequência: uma tabela que exibe a frequência (contagem) de cada valor no conjunto de dados.
   5.2. Histograma: Uma representação gráfica da distribuição de frequência, onde os dados são agrupados em intervalos (compartimentos) e plotados como barras.
6. Medidas de Associação:
   6.1. Covariância: Mede o grau em que duas variáveis mudam juntas.
   6.2. Correlação: Covariância normalizada, indicando a força e a direção de uma relação linear entre duas variáveis.
7. Intervalo Interquartil (IQR): O intervalo entre o primeiro quartil (Q1) e o terceiro quartil (Q3), que captura os 50% intermediários dos dados. É menos sensível a valores discrepantes do que toda a amostra.

Felizmente, várias funções já estão implementadas no python com bibliotecas para o uso direto de estatística como a **`statistics`** (statistics — Mathematical statistics functions — Python 3.11.5 documentation)

ID	Função	Descrição
1.1	`mean()`	Média aritmética ("média") dos dados.
1.2	`median()`	Mediana (valor médio) dos dados.
1.3	`mode()`	Modo único (valor mais comum) de dados discretos ou nominais.
2.1	`max()-min()`	Diferença entre o valor máximo e mínimo retorna o alcance
2.2	`pvariance()`	Variância populacional dos dados
	`variance()`	Variância amostral dos dados
2.3	`pstdev()`	Desvio padrão populacional dos dados
	`stdev()`	Desvio padrão amostral dos dados
3.1	`quantiles()`	Divida os dados em intervalos com igual probabilidade
3.2	`quantiles(n=4)`	

```
>>>import statistics as st
>>>set1 = [1, 3, 3, 4, 5, 7]
>>> print("a mediana da base é {}".format(st.median(set1)))
a mediana da base é 3.5

>>>set2 = [2, 3, 3, 4, 5, 5, 5, 5, 6, 6, 6, 7]
>>> set3 = [2.4, 1.3, 1.3, 1.3, 2.4, 4.6]
>>> set4 = ["red", "blue", "black", "blue", "black", "black", "brown"]
>>>print("A moda do data set 2 é % s" % (st.mode(set2)))
A moda do data set 2 é 5

>>> print("A moda do data set 3 é % s" % (st.mode(set3)))
A moda do data set 3 é 1.3

>>>print("A moda do data set 4 é % s" % (st.mode(set4)))
A moda do data set 4 é black

>>> arr = [1, 2, 3, 4, 5, -2, -4, -3, -1, -5, -6, 1, 2, 5, 4, 8, 9, 12]
>>>Maximum = max(arr)
>>>Minimum = min(arr)
>>>Range = Maximum – Minimum
>>> print("Maximum = {}, Minimum = {} e Range = {}".format(Maximum, Minimum, Range))
Maximum = 12, Minimum = -6 e Range = 18

>>> print("Variância de arr é {:.02f}".format(st.variance(arr)))
Variância de arr é 24.29

>>> print("O desvio padrão de arr é {:.02f}".format(st.stdev(arr)))
O desvio padrão de arr é 4.93

>>> st.quantiles(arr, n=4, method='inclusive')
Lembra da diferença entre quartil inclusivo e exclusivo?
[-1.75, 2.0, 4.75]

>>> st.quantiles(arr, n=10, method='inclusive')
[-4.3, -2.6, -0.8, 1.0, 2.0, 3.2, 4.0, 5.0, 8.3]
```

Quais as vantagens e desvantagens de aumentar a minha divisão de amostra?

Equações Lineares e não-lineares

Além de descrever é importante que tentemos criar alguma relação entre os dados, entender essa relação entre as variáveis pode nos fornecer ideias sobre as semelhanças e diferenças dos dados, nos permitindo assim criar modelos para futuramente trabalharmos com inferência estatística.

Equações Lineares

Equações lineares são fundamentais nesse processo, pois nos ajudam a modelar relações que apresentam uma mudança proporcional e constante. Numa equação linear, a maior potência de qualquer variável é 1. Isto significa que a relação entre as variáveis pode ser representada como uma linha reta num gráfico.

Exemplo 1: *Considere um cenário em que você analisa a relação entre o número de usuários (U) que acessam um site e o tempo de resposta do servidor (R). Se o tempo de resposta aumentar em um valor constante para cada usuário adicional, você terá um relacionamento linear. A equação pode parecer assim:*

$$R = m * U + b$$

Onde:

- ***m*** *é a inclinação, representando a mudança no tempo de resposta por usuário*
- ***b*** *é a interceptação y, indicando o tempo de resposta quando não há usuários.*

Em um gráfico, essa equação produziria uma linha reta.

Em suma podemos expressar uma equação linear com um componente angular, que determina qual é a angulação da reta, e um componente linear, que determina onde a reta cruza o eixo y.

A representação clássica de um sistema linear é:

$$y(x) = a * x + b$$

Uma outra forma geral de uma equação linear com uma variável x também pode ser expressa como:

$$a * x + b = 0$$

Podemos também ter mais de uma variável, x e y por exemplo:

$$a * x + b * y + c = 0$$

As equações lineares podem ter várias formas e envolver mais variáveis, mas todas seguem este princípio básico: as variáveis são elevadas à potência de 1 e combinadas usando adição, subtração e multiplicação por constantes.

Quando temos mais de uma variável, representamos na forma de um sistema, tendo uma quantidade de equação igual ao número de variáveis chamamos de um **sistema determinado**, e caso não tenhamos todas as equações chamamos de **sistema indeterminado**.

Podemos ir mais além, calculando o determinante da matriz dos coeficientes chegamos a três conjecturas sobre o sistema:

- Determinante diferente de zero, o sistema é possível e determinado **(SPD)**
- Determinante igual a zero e infinitas soluções, o sistema é possível e indeterminado **(SPI)**
- Determinante igual a zero e nenhuma solução, o sistema é impossível **(SI)**

SPD: ao ser resolvido encontraremos uma única solução, isto é, apenas um único valor para as incógnitas

$$\begin{cases} x + y = 5 \\ x - y = 3 \end{cases}$$

SPI: esse tipo de sistema possui infinitas soluções, os valores de x e y assumem inúmeros valores

$$\begin{cases} x + y = 4 \\ y + z = 3 \end{cases}$$

SI: ao ser resolvido, não encontramos soluções possíveis para as incógnitas, por isso esse sistema é classificado como impossível

$$\begin{cases} x + y = 9 \\ x + y = 5 \end{cases}$$

Em muitos casos, não sabemos as variáveis **m** e **b** como no exemplo, ou as variáveis **a** e **b** como na representação clássica. Sabemos a saída dada uma entrada, mas não conseguimos achar esses coeficientes.

Que bom que temos o python! E nesse caso mais específico a biblioteca sympy. SymPy é uma biblioteca Python para matemática simbólica. Seu objetivo é se tornar um sistema de álgebra computacional (CAS) completo, mantendo o código o mais simples possível para ser compreensível e facilmente extensível. SymPy é escrito inteiramente em Python. (SymPy, SymPy 1.12 documentation)

Com essa biblioteca, até os sistemas mais complexos podem ser resolvidos de uma forma simples

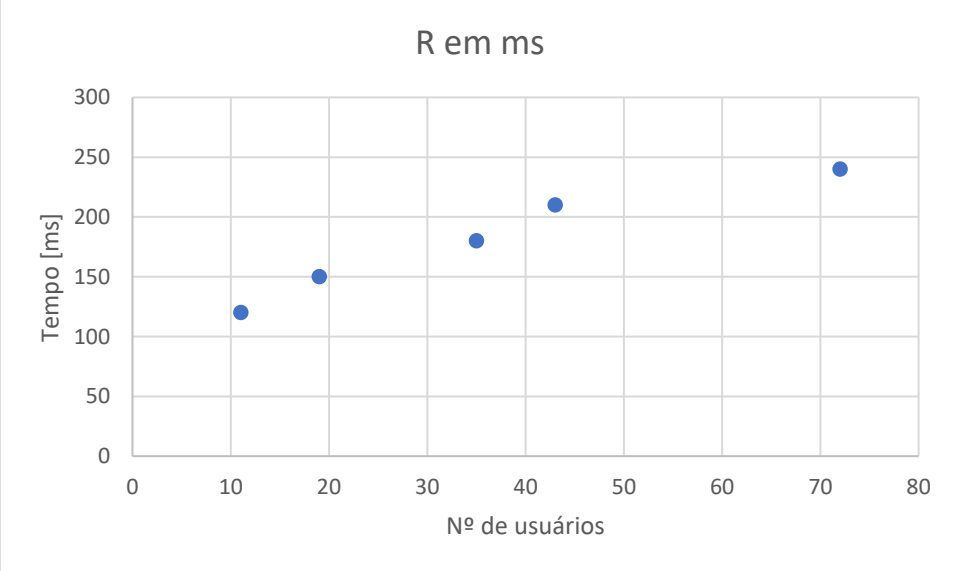
Vamos voltar ao exemplo 1 e resolver o sistema linear utilizando o sympy.

Tempo de resposta do servidor (R) em ms	Número de usuários (U)
120	11
150	19
180	35
210	43
240	72

Como podemos resolver esse sistema? Essa é a carinha dele:

$$R = m * U + b$$

$$\begin{cases} m * 11 + b = 120 \\ m * 19 + b = 150 \\ m * 35 + b = 180 \\ m * 43 + b = 210 \\ m * 72 + b = 240 \end{cases}$$



O que queremos é: Ou achar uma reta que passe por todos os pontos, ou achar uma reta que explique de maneira mais genérica os pontos. Pela representação gráfica (excel) percebemos que os pontos não estão numa linear perfeita.

Utilizaremos o sympy

C:\>pip install sympy

```
>>>import sympy as sp
>>>m, b = sp.symbols('m b')
>>>eq1 = sp.Eq(m*11 + b,120)
>>>eq2 = sp.Eq(m*19 + b,150)
>>>eq3 = sp.Eq(m*35 + b,180)
>>>eq4 = sp.Eq(m*43 + b,210)
>>>eq5 = sp.Eq(m*72 + b,240)
>>>ans = sp.solve((eq1,eq2,eq3,eq4,eq5),(m,b))
>>>ans
[ ]
```

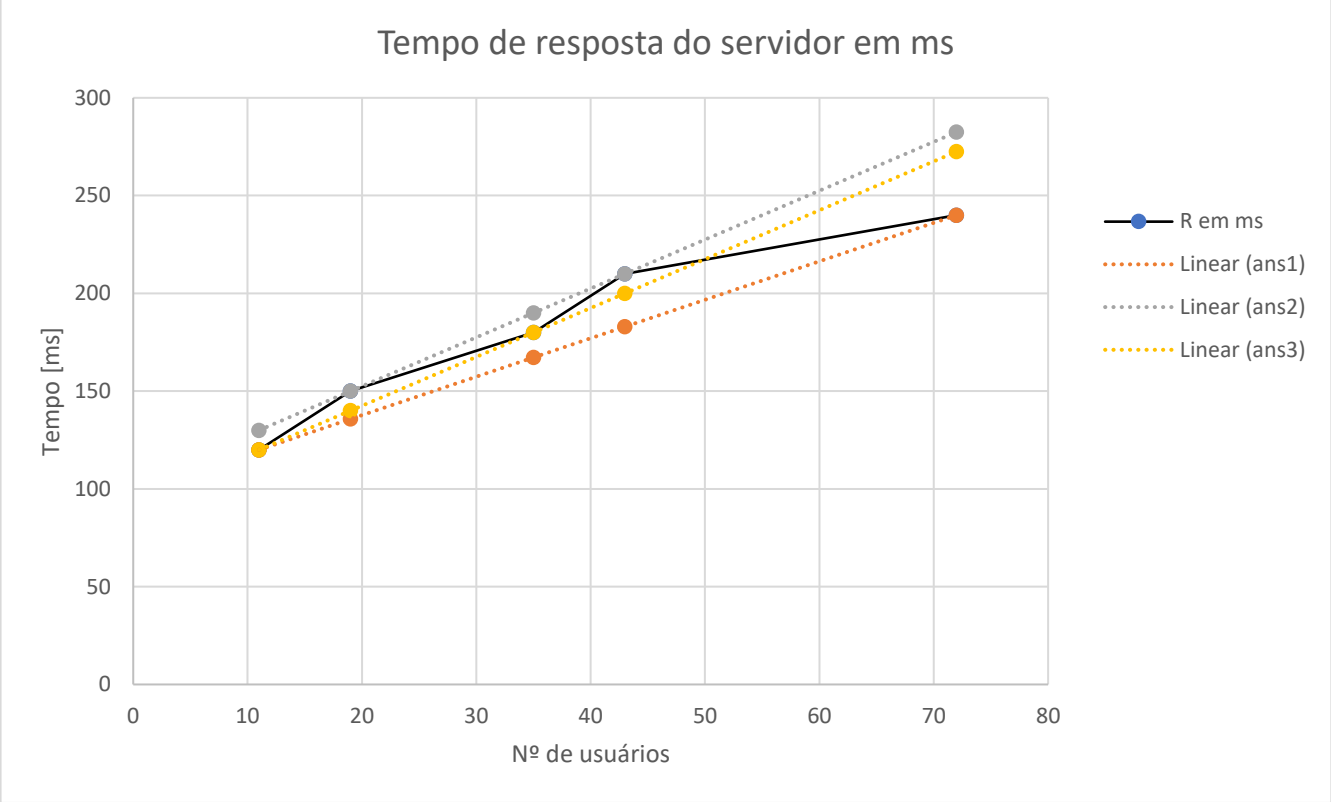
Deu um vetor vazio???? Por quê????

Resp.: Como apontado anteriormente, esses pontos não são uma reta perfeita, então temos que trabalhar com combinações de pontos para formar a reta, criando vários modelos, e testar cada modelo com possíveis erros

```
>>>ans1 = sp.solve((eq1,eq5),(m,b))
>>>ans2 = sp.solve((eq2,eq4),(m,b))
>>>ans3 = sp.solve((eq1,eq3),(m,b))
>>>ans1
{b: 6000/61, m: 120/61}
>>>ans2
{b: 205/2, m: 5/2}
>>>ans3
{b: 185/2, m: 5/2}
>>>float(ans2.get(b))
102.5
```

****Ou ainda****

```
>>>ans1 = list(sp.linsolve((eq1,eq5),(m,b)))
>>>ans2 = list(sp.linsolve((eq2,eq4),(m,b)))
>>>ans3 = list(sp.linsolve((eq1,eq3),(m,b)))
>>>ans1
[(120/61, 6000/61)]
>>>ans2
[(5/2, 205/2)]
>>>ans3
[(5/2, 185/2)]
>>>round(ans3[0][1],2)
92.50
```



Podemos observar que cada modelo pode responder um pouco do gráfico. Mas todos possuem algum erro associado. Mais para frente utilizaremos regressão linear para trabalhar com um melhor modelo que explique essa situação. Utilizaremos o R, ou então o scipy com numpy e matplotlib, uma outra biblioteca python para computação científica 😉 (R: The R Project for Statistical Computing (r-project.org, RStudio IDE - RStudio, SciPy, SciPy User Guide — SciPy v1.11.2 Manual, NumPy, NumPy user guide — NumPy v1.25 Manual, Matplotlib — Visualization with Python, Matplotlib documentation — Matplotlib 3.7.2 documentation).

Equações não lineares

Uma equação não linear é uma equação algébrica em que pelo menos um termo envolve uma variável elevada a uma potência diferente de 1 ou envolve o produto, quociente ou composição de variáveis. As equações não lineares representam relações que não podem ser representadas por linhas retas em um gráfico.

As equações não lineares podem assumir várias formas, como polinomiais, exponenciais, logarítmicas, trigonométricas e muito mais. Essas equações podem ter curvas, curvas e comportamentos complexos quando representadas graficamente.

$$a * x^2 + b * x + c = 0$$

Exercício: Prevendo o Desempenho do Sistema

Contexto:
Você é um administrador de sistema responsável pela manutenção de um servidor. Nos últimos meses, você coletou dados sobre o uso da CPU do servidor (em porcentagem) e o número de usuários ativos durante diferentes horários do dia. Você suspeita que o uso da CPU pode ser influenciado pelo número de usuários ativos. Você deseja analisar os dados para determinar se há uma relação entre o uso da CPU e os usuários ativos e criar um modelo preditivo para o uso da CPU com base no número de usuários ativos.

Dados:

Hora do dia	Nº ativo de usuários	Uso de CPU (%)
09:00	10	20,0
10:00	12	25,2
11:00	15	30,0
12:00	25	45,1
13:00	22	42,7
14:00	18	33,6
15:00	15	31,5
16:00	20	45,0
17:00	28	53,1
18:00	30	60,2

1. Estatísticas descritivas:
   a. Calcule a média, a mediana e o desvio padrão do número de usuários ativos e do uso da CPU.
2. Análise de equações lineares:
   a. Com base nos dados, levante a hipótese se pode haver uma relação linear entre o número de usuários ativos e o uso da CPU. Justifique sua hipótese.
   b. Se você acredita que existe uma relação linear, calcule a inclinação e a interceptação da linha de melhor ajuste que descreve a relação entre o uso da CPU e os usuários ativos.
3. Modelo de previsão:
   a. Supondo um relacionamento linear, crie uma equação linear para prever o uso da CPU (C) com base no número de usuários ativos (U). Você pode usar a inclinação e a interceptação da tarefa anterior.
   b. Preveja o uso da CPU se houver 20 usuários ativos às19:00.
4. Análise de equações não lineares:
   a. Considere a possibilidade de um relacionamento não linear entre o uso da CPU e os usuários ativos. Discuta as razões pelas quais isso pode acontecer.

Para curiosos

Considere essa nova fonte de dados

Nº ativo de usuários	Uso da CPU (%)
5	10
10	15
20	30
30	50
40	70

Este conjunto de dados representa o uso da CPU para diferentes números de usuários ativos. Como podemos ver, a relação não é linear e exploraremos a criação de uma equação não linear para modelar essa relação.

Uma equação não linear comum usada para modelar o crescimento é a equação exponencial:

$$C = a * e^{b*U}$$

Onde:

- **C** é a porcentagem de uso da CPU.
- **U** é o número de usuários ativos.
- **a** é uma constante de escala.
- **b** é uma constante de taxa.
- **e** é a base do logaritmo natural

Com esta equação, o uso da CPU cresce exponencialmente à medida que aumenta o número de usuários ativos. Você pode usar esta equação para ajustar os dados e estimar os valores de **a** e **b** que melhor correspondem ao uso de CPU observado para diferentes números de usuários ativos.

Usar o SymPy para ajuste de curva pode não ser tão simples quanto usar SciPy. SymPy é mais adequado para manipulação simbólica e análise algébrica, enquanto SciPy é projetado para cálculos numéricos e algoritmos científicos como ajuste de curvas. Para tarefas de ajuste de curvas, SciPy é a ferramenta recomendada

Vamos utilizar os dois!

```
>>>import sympy as sp
>>> a, b = sp.symbols('a b')
>>>ep1 = sp.Eq(a * (math.e**(b * 5)),10)
>>>ep2 = sp.Eq(a * (math.e**(b * 10)),15)
>>>ep3 = sp.Eq(a * (math.e**(b * 20)),30)
>>>ep4 = sp.Eq(a * (math.e**(b * 30)),50)
>>>ep2 = sp.Eq(a * (math.e**(b * 40)),70)
```

Necessitamos escolher pelo menos duas equações para achar os coeficientes e testar o modelo

```
>>>resposta = sp.solve((ep1,ep2),(a,b))
>>>resposta
[(7.57306542124553, 0.0555974328301517)]
```

Este modelo atende? É preciso? A curva faz sentido?

Teste esse código aqui. O que ele faz??

```
C:\>pip install numpy, matplotlib, scipy

>>>import numpy as np
>>>import matplotlib.pyplot as plt
>>>from scipy.optimize import curve_fit
>>>users = np.array([5, 10, 20, 30, 40])
>>>cpu_usage = np.array([10, 15, 30, 50, 70])
>>>def exponential_function(U, a, b):
        return a * np.exp(b * U)
>>>params, covariance = curve_fit(exponential_function, users, cpu_usage, maxfev = 5000)
>>>a_fit, b_fit = params
>>>predicted_cpu_usage = exponential_function(users, a_fit, b_fit)
>>>print("Fitted Parameters:")
Fitted Parameters:
>>>print("a:", a_fit)
a: 10.954099391531042
>>>print("b:", b_fit)
b: 0.047205787337791044
>>>plt.scatter(users, cpu_usage, label='Data')
<matplotlib.collections.PathCollection object at 0x000001FD9DFA7E50>
>>>plt.plot(users, predicted_cpu_usage, label='Fitted Curve', color='red')
[<matplotlib.lines.Line2D object at 0x000001FD9EFFB0D0>]
>>>plt.xlabel('Number of Active Users')
Text(0.5, 0, 'Number of Active Users')
>>>plt.ylabel('CPU Usage (%)')
Text(0, 0.5, 'CPU Usage (%)')
>>>plt.legend()
<matplotlib.legend.Legend object at 0x000001FDAD5158D0>
>>>plt.title('CPU Usage vs. Number of Active Users (Fitted Curve)')
Text(0.5, 1.0, 'CPU Usage vs. Number of Active Users (Fitted Curve)')
>>>plt.grid()
>>>plt.show()
```