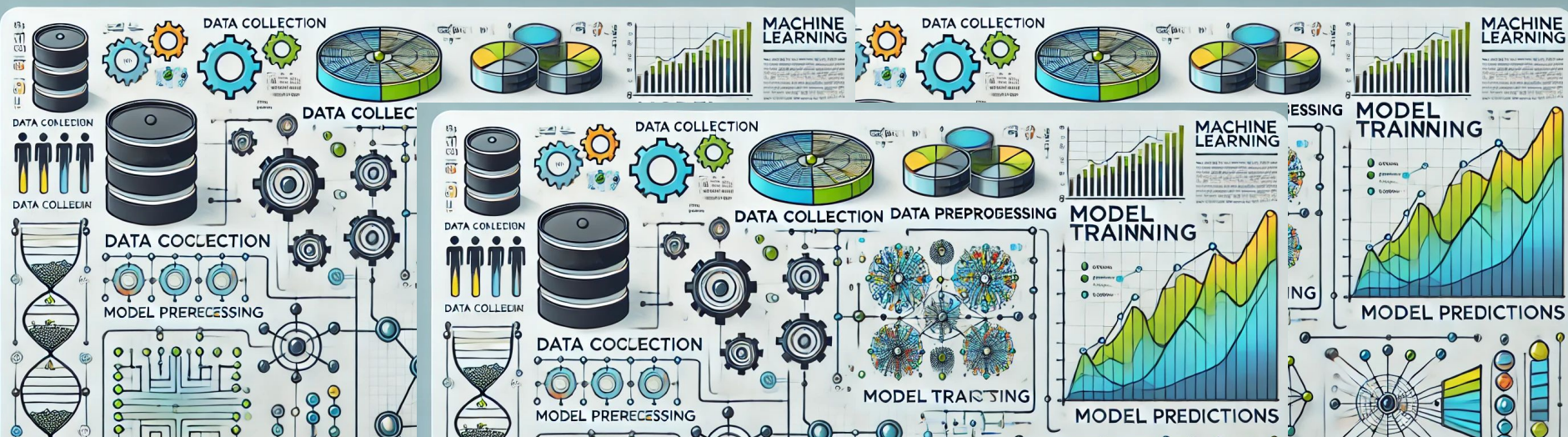


## Tarefa 1 - Aprendizado de máquina

### Designing Machine Learning Systems - cap 1-5

Emelyn Clementino Freire



O'REILLY®

# Designing Machine Learning Systems

An Iterative Process  
for Production-Ready  
Applications



Chip Huyen

# Capítulo 1: Overview of Machine Learning Systems

1. Overview of Machine Learning Systems.....	1
When to Use Machine Learning	3
Machine Learning Use Cases	9
Understanding Machine Learning Systems	12
Machine Learning in Research Versus in Production	12
Machine Learning Systems Versus Traditional Software	22
Summary	23

## Objetivo:

Explicar o que são sistemas de aprendizado de máquina (ML) e como eles são usados.

## Principais pontos:

- **Quando usar ML:**
  - Para identificar padrões complexos nos dados.
  - Para fazer previsões, como recomendar filmes ou prever o clima.
  - Para lidar com grandes volumes de informações.
- **Como funciona um sistema de ML:**
  - É mais do que só o algoritmo: inclui dados, infraestrutura e como tudo isso se conecta para resolver problemas reais.
- **Desafios:**
  - Um sistema de ML precisa ser rápido, confiável e atender às necessidades de quem vai usá-lo.

```

# Capítulo 1: Overview of Machine Learning Systems
print("### Capítulo 1: Overview of Machine Learning Systems ###")
# Exemplo de quando usar ML: Classificação binária simples
X, y = make_classification(n_samples=500, n_features=5, random_state=42)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Treinamento simples de modelo
model = LogisticRegression()
model.fit(X_train, y_train)

# Previsões e avaliação
predictions = model.predict(X_test)
print("\nRelatório de Classificação:\n")
print(classification_report(y_test, predictions))

# Visualização simples
plt.scatter(X_test[:, 0], X_test[:, 1], c=predictions, cmap='viridis', alpha=0.5)
plt.title("Classificação: Previsões de Teste")
plt.show()

```

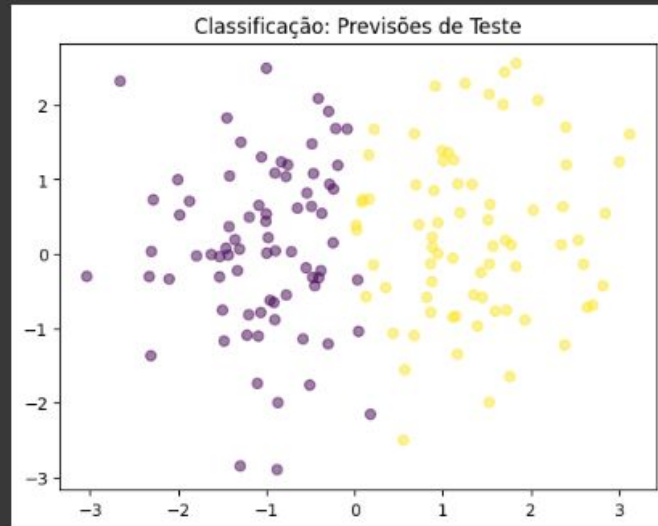
```

### Capítulo 1: Overview of Machine Learning Systems ###

```

Relatório de Classificação:

	precision	recall	f1-score	support
0	0.89	0.91	0.90	75
1	0.91	0.89	0.90	75
accuracy			0.90	150
macro avg	0.90	0.90	0.90	150
weighted avg	0.90	0.90	0.90	150



# Capítulo 2: Introduction to Machine Learning Systems Design

## Objetivo:

Apresentar como planejar sistemas de ML que sejam confiáveis, eficientes e fáceis de ajustar.

## Principais pontos:

- **Começando pelo objetivo:**
  - Entender o que o negócio precisa e alinhar isso com o que o sistema de ML deve fazer.
  - Definir métricas claras para medir o sucesso.
- **O que faz um bom sistema:**
  - **Confiabilidade:** Funciona bem sem falhas.
  - **Escalabilidade:** Aguenta crescer junto com o aumento de dados ou usuários.
  - **Manutenção:** Fácil de atualizar e monitorar.
- **Como fazer:**
  - Seguir um processo de tentativa e erro: definir problemas, testar soluções, avaliar e melhorar.

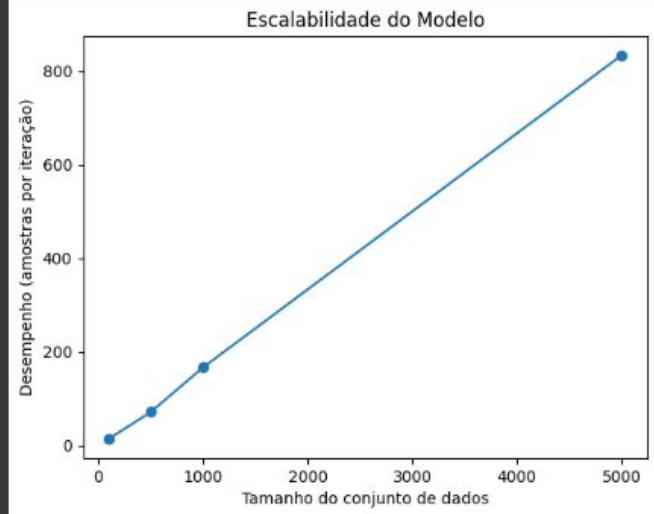
2. Introduction to Machine Learning Systems Design.....	25
Business and ML Objectives	26
Requirements for ML Systems	29
Reliability	29
Scalability	30
Maintainability	31
Adaptability	31
Iterative Process	32
Framing ML Problems	35
Types of ML Tasks	36
Objective Functions	40
Mind Versus Data	43
Summary	46

```
# Capítulo 2: Introduction to Machine Learning Systems Design
print("### Capítulo 2: Introduction to Machine Learning Systems Design ###")
# Demonstração de confiabilidade e escalabilidade
print("Simulação de análise de desempenho em diferentes tamanhos de amostras")
sizes = [100, 500, 1000, 5000]
latencies = []

for size in sizes:
    X_sample, y_sample = make_classification(n_samples=size, n_features=5, random_state=42)
    model = LogisticRegression()
    %time model.fit(X_sample, y_sample)
    latencies.append(size / model.n_iter_[-1])

plt.plot(sizes, latencies, marker='o')
plt.xlabel("Tamanho do conjunto de dados")
plt.ylabel("Desempenho (amostras por iteração)")
plt.title("Escalabilidade do Modelo")
plt.show()
```

```
### Capítulo 2: Introduction to Machine Learning Systems Design ###
Simulação de análise de desempenho em diferentes tamanhos de amostras
CPU times: user 6.03 ms, sys: 0 ns, total: 6.03 ms
Wall time: 6.05 ms
CPU times: user 4.44 ms, sys: 0 ns, total: 4.44 ms
Wall time: 4.46 ms
CPU times: user 4.16 ms, sys: 0 ns, total: 4.16 ms
Wall time: 4.22 ms
CPU times: user 8.11 ms, sys: 7.92 ms, total: 16 ms
Wall time: 12.3 ms
```





# Capítulo 3: Data Engineering Fundamentals

## Objetivo:

Mostrar como organizar e processar dados para que o sistema de ML funcione bem.

## Principais pontos:

- **De onde vêm os dados:**
  - Podem ser organizados em tabelas (dados estruturados) ou mais livres, como texto e imagens (não estruturados).
- **Como organizar os dados:**
  - Usar bancos de dados relacionais (como tabelas) ou NoSQL para casos mais flexíveis.
- **Transformando dados em algo útil:**
  - Usar processos como ETL (Extrair, Transformar, Carregar) para preparar os dados.
- **Desafios:**
  - Garantir que os dados sejam confiáveis, atualizados e possam ser usados em larga escala.

3. Data Engineering Fundamentals.....	49
Data Sources	50
Data Formats	53
JSON	54

iii

Row-Major Versus Column-Major Format	54
Text Versus Binary Format	57
Data Models	58
Relational Model	59
NoSQL	63
Structured Versus Unstructured Data	66
Data Storage Engines and Processing	67
Transactional and Analytical Processing	67
ETL: Extract, Transform, and Load	70
Modes of Dataflow	72
Data Passing Through Databases	72
Data Passing Through Services	73
Data Passing Through Real-Time Transport	74
Batch Processing Versus Stream Processing	78
Summary	79

```
# Capítulo 3: Data Engineering Fundamentals
print("### Capítulo 3: Data Engineering Fundamentals ###")
# Exemplo de preparação de dados: Limpeza e transformação
raw_data = {'idade': [23, 35, np.nan, 29], 'salario': [50000, 70000, 80000, np.nan]}
data = pd.DataFrame(raw_data)
print("Dados Brutos:\n", data)
data.fillna(data.mean(), inplace=True)
data['salario_escalonado'] = StandardScaler().fit_transform(data[['salario']])
print("Dados Pós-Limpeza:\n", data)
```

```
### Capítulo 3: Data Engineering Fundamentals ###
Dados Brutos:
   idade  salario
0   23.0   50000.0
1   35.0   70000.0
2    NaN   80000.0
3   29.0     NaN

Dados Pós-Limpeza:
   idade  salario  salario_escalonado
0   23.0  50000.000000          -1.543033
1   35.0  70000.000000           0.308607
2   29.0  80000.000000           1.234427
3   29.0  66666.666667           0.000000
```



# Capítulo 4: Training Data

## Objetivo:

Ensinar como escolher, preparar e melhorar os dados usados para

"treinar" o sistema de ML.

## Principais pontos:

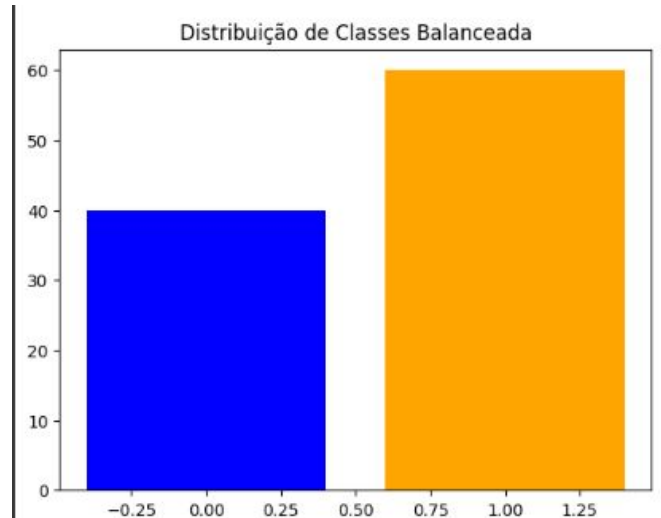
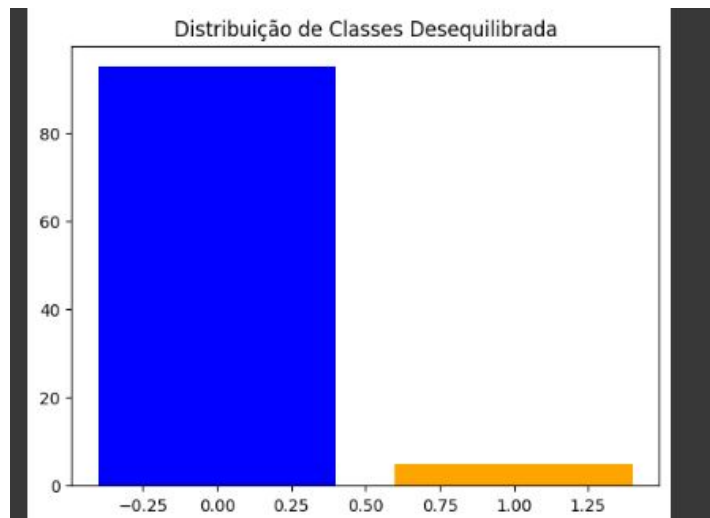
- **Escolher bem os dados:**
  - Garantir que eles sejam representativos do problema que você quer resolver.
- **Rotular os dados:**
  - Identificar o que cada dado representa, como classificar imagens em “gato” ou “cachorro”.
- **Lidando com problemas comuns:**
  - Resolver casos de desbalanceamento, como quando temos muitas imagens de um tipo e poucas de outro.
- **Aumentando os dados:**
  - Criar variações dos dados, como ajustar cores de uma imagem ou girá-la.

4. Training Data.....	81
Sampling	82
Nonprobability Sampling	83
Simple Random Sampling	84
Stratified Sampling	84
Weighted Sampling	85
Reservoir Sampling	86
Importance Sampling	87
Labeling	88
Hand Labels	88
Natural Labels	91
Handling the Lack of Labels	94
Class Imbalance	102
Challenges of Class Imbalance	103
Handling Class Imbalance	105
Data Augmentation	113
Simple Label-Preserving Transformations	114
Perturbation	114
Data Synthesis	116
Summary	118



```
# Capítulo 4: Training Data
print("### Capítulo 4: Training Data ###")
# Exemplo de balanceamento de classes
imbalanced_y = [0] * 95 + [1] * 5
plt.bar([0, 1], [imbalanced_y.count(0), imbalanced_y.count(1)], color=['blue', 'orange'])
plt.title("Distribuição de Classes Desequilibrada")
plt.show()

balanced_y = np.random.choice([0, 1], size=100, p=[0.5, 0.5])
plt.bar([0, 1], [list(balanced_y).count(0), list(balanced_y).count(1)], color=['blue', 'orange'])
plt.title("Distribuição de Classes Balanceada")
plt.show()
```



# Capítulo 5: Feature Engineering

**Objetivo:**

Ensinar como extrair informações importantes dos dados para que o sistema funcione melhor.

**Principais pontos:**

- **O que são features:**
  - São características dos dados que ajudam o modelo a aprender, como o tamanho ou a cor de um objeto em uma imagem.
- **Como melhorar os dados:**
  - Preencher valores faltantes, ajustar os dados para ficarem na mesma escala e transformar textos em números.
- **Evitar erros:**
  - Garantir que o modelo só use informações que ele realmente terá quando estiver funcionando.
- **Generalização:**
  - Criar features que funcionem bem com novos dados, não apenas com os dados usados no treinamento.

5. Feature Engineering.....	119
Learned Features Versus Engineered Features	120
Common Feature Engineering Operations	123
Handling Missing Values	123
Scaling	126

Discretization	128
Encoding Categorical Features	129
Feature Crossing	132
Discrete and Continuous Positional Embeddings	133
Data Leakage	135
Common Causes for Data Leakage	137
Detecting Data Leakage	140
Engineering Good Features	141
Feature Importance	142
Feature Generalization	144
Summary	146

```
# Capítulo 5: Feature Engineering
print("### Capítulo 5: Feature Engineering ###")
# Exemplo de criação de features
raw_features = pd.DataFrame({
    'idade': [23, 45, 31],
    'salario': [50000, 80000, 60000]
})
raw_features['idade_quadrado'] = raw_features['idade'] ** 2
raw_features['salario_log'] = np.log(raw_features['salario'])
print("Dados com Novas Features:\n", raw_features)
```

```
### Capítulo 5: Feature Engineering ###
Dados com Novas Features:
   idade  salario  idade_quadrado  salario_log
0     23   50000           529     10.819778
1     45   80000          2025     11.289782
2     31   60000           961     11.002100
```