# Solução Lista 01

Nome: Luana de Sousa Silva
E-mail: sousa.luana@aluno.ufabc.edu.br

24 fevereiro, 2025

## Exercício 01

O aprendizado de máquina possui diversas aplicações práticas em diferentes áreas. A seguir, apresento exemplos para cada um dos problemas mencionados:

### a) Problema de classificação

*Exemplo: Detecção de spam em e-mails*

- **Descrição**: O objetivo é classificar e-mails como "spam" ou "não spam".
- **Vetores de características**: As características podem incluir a presença de certas palavras-chave, frequência de termos específicos, análise de remetentes, estrutura do e-mail, entre outros.
- **Rótulos**: Cada e-mail é rotulado como "spam" ou "não spam".

### b) Problema de regressão

*Exemplo: Previsão de retorno de portfólio e sugestão de investimentos*

- **Descrição**: O objetivo é prever o retorno futuro de um portfólio de investimentos ou de ativos individuais, auxiliando na tomada de decisão sobre onde investir.
- **Vetores de características**:
  - Indicadores financeiros: Retornos passados, volatilidade ($\sigma$), beta ($\beta$), correlação com outros ativos.
  - Indicadores macroeconômicos: Taxa de juros, inflação, PIB, desemprego.
  - Indicadores técnicos: Médias móveis, RSI (*Relative Strength Index*), bandas de Bollinger.
  - Comportamento do mercado: Volume de negociação, notícias financeiras (*Natural Language Processing* pode ser usado aqui).
  - Composição do portfólio: Pesos dos ativos e rebalanceamento.
- **Resposta (Variável alvo)**: O retorno esperado do portfólio ou de um ativo individual no futuro ($R_t$), que pode ser modelado como:

  $$R_t = f(X) + \epsilon$$

  Onde $X$ representa o vetor de características e $\epsilon$ é um termo de erro.
- **Possíveis modelos**:
  - **Regressão linear múltipla**: Modela a relação entre os fatores e o retorno esperado.
  - **Árvores de decisão e Random Forest**: Capturam relações não-lineares entre as variáveis.

  - **Redes neurais**: Podem modelar interações complexas entre variáveis financeiras.
  - **Modelos baseados em séries temporais** (ARIMA, LSTM): São úteis para prever retornos futuros com base em padrões históricos.

**c) Problema de agrupamento**

*Exemplo: Segmentação de clientes em marketing*

- **Descrição**: Agrupar clientes com comportamentos ou características semelhantes para direcionar campanhas de marketing específicas.
- **Vetores de características**: Podem incluir histórico de compras, frequência de compras, valor gasto, preferências de produtos, dados demográficos, entre outros.
- **Rótulos**: Neste caso, não há rótulos pré-definidos. Os grupos (*clusters*) são formados com base nas semelhanças identificadas nos dados.

Esses exemplos ilustram como o aprendizado de máquina pode ser aplicado para resolver diferentes tipos de problemas, utilizando vetores de características e rótulos ou respostas conforme apropriado.

## Exercício 02

A maldição da dimensionalidade foi introduzida por Richard E. Bellman, ao levar em consideração problemas em programação dinâmica. Refere-se ao problema que é causado pelo aumento exponencial no volume associado com a adição de dimensões extras a um espaço matemático.

Resumidamente, se dividirmos uma região do espaço em células regulares, o número de células cresce a medida de forma exponencial de acordo com a dimensão do espaço. Com isso, o número de amostrar deve crescer exponencialmente a fim de garantir que nenhuma célula fique vazia.

De forma prática, isso implica que dado um tamanho de amostras, existe um número máximo de características a partir do qual o desempenho do classificador irá degradar, ao invés de melhorar.

Uma solução seria reduzir a dimensão por meio da seleção de características ou métodos de redução

## Exercício 03

```python
import numpy as np
import pandas as pd
from collections import Counter

def knn(k, x, D):
    """Implementação do k-NN."""
    D_copy = D.copy()
    D_copy['dist'] = (D_copy['x_1'] - x[0])**2 + (D_copy['x_2'] - x[1])**2
    D_sorted = D_copy.nsmallest(k, 'dist')
    return Counter(D_sorted['y']).most_common(1)[0][0]

# Teste do k-NN
D = pd.DataFrame({
    'x_1': np.random.normal(1, 1, 100),
    'x_2': np.random.normal(-1, 2, 100),
    'y': np.random.choice(['one', 'two', 'three'], 100, replace=True)
})
```

```
x_test = [1, 2]
k = 10
print("Classe prevista pelo k-NN:", knn(k, x_test, D))
```

```
## Classe prevista pelo k-NN: one
```

### Exercício 04

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from collections import Counter

def knn(k, x, D):
    """Implementação do k-NN."""
    D_copy = D.copy()
    D_copy['dist'] = (D_copy['x_1'] - x[0])**2 + (D_copy['x_2'] - x[1])**2
    D_sorted = D_copy.nsmallest(k, 'dist')
    return Counter(D_sorted['y']).most_common(1)[0][0]

def test_knn(k):
    """Testa o k-NN no dataset Iris e calcula a acurácia."""
    iris = load_iris()
    iris_df = pd.DataFrame({
        'x_1': iris.data[:, 2],  # Petal Length
        'x_2': iris.data[:, 0],  # Sepal Length
        'y': iris.target
    })

    correct = sum(
        knn(k, (row['x_1'], row['x_2']), iris_df.drop(index)) == row['y']
        for index, row in iris_df.iterrows()
    )

    accuracy = correct / len(iris_df)
    print(f'Acurácia com k={k}: {accuracy:.2%}')

test_knn(1)
```

```
## Acurácia com k=1: 90.67%
```

```
test_knn(10)
```

```
## Acurácia com k=10: 94.67%
```

### Exercício 05

# Prova da Função Ótima

Queremos minimizar o risco esperado:

$$R(f) = \mathbb{E}_{X,Y}[|Y - f(X)|]$$

onde $\ell_1(y, y') = |y - y'|$ é a função de perda do erro absoluto. O risco condicional para um valor fixo de $X = x$ é dado por:

$$R(f|X = x) = \mathbb{E}[|Y - f(x)||X = x]$$

Agora, vamos calcular a derivada da esperança condicional da função de perda em relação a $f(x)$. A derivada de $|Y - z|$ em relação a $z$ é dada por:

$$\frac{\partial}{\partial z}|Y - z| = \mathrm{sgn}(z - Y)$$

onde $\mathrm{sgn}(z - Y)$ é a função sinal, que retorna $-1$ se $z < Y$, $1$ se $z > Y$, e $0$ se $z = Y$.

Portanto, a derivada da esperança condicional é:

$$\frac{\partial}{\partial z}\mathbb{E}[|Y - z||X = x] = \mathbb{E}[\mathrm{sgn}(z - Y)|X = x]$$

Para minimizar o risco, queremos que a derivada da esperança condicional seja igual a zero:

$$\mathbb{E}[\mathrm{sgn}(z - Y)|X = x] = 0$$

Isso implica que o número de valores de $Y$ maiores que $z$ deve ser igual ao número de valores de $Y$ menores que $z$, ou seja, $z$ deve ser a mediana condicional de $Y$ dado $X = x$.

Logo, a função $f(x)$ que minimiza o risco esperado com a função de perda do erro absoluto é dada por:

$$f(x) = \mathrm{Mediana}(Y|X = x)$$

Isso conclui a prova de que a função ótima $f(x)$ é a mediana condicional de $Y$ dado $X = x$.

## Exercício 06

```python
import numpy as np

def median_distance(m, d):
    """Calcula a mediana da distância do ponto mais próximo à origem em uma hiperesfera."""
    return (1 - 0.5**(1/m))**(1/d)

m = 100
d = 3
print(f'Mediana da distância para m={m}, d={d}: {median_distance(m, d):.4f}')
```

```
## Mediana da distância para m=100, d=3: 0.1904
```