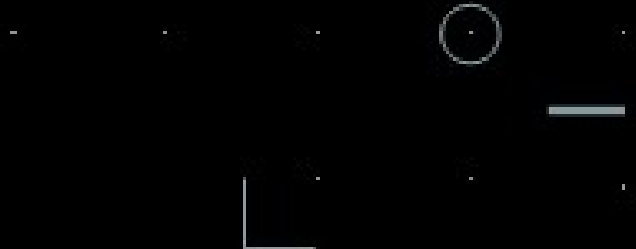


FIAP

## 3 Sprint Challenge

# Disruptive Architectures: IoT, IoB & Generative AI



## Membros do grupo:

Nicholas Santos – **RM: 551809**

Pedro Pacheco – **RM: 98043**

Vitor Kubica – **RM: 98903**

Eduardo Violante - **RM: 550364**

Beatriz Svestzka – **RM: 551534**



|    |    |    |    |    |    |
|----|----|----|----|----|----|
| 1  | 2  | 3  | 4  | 5  | 6  |
| 7  | 8  | 9  | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 | 17 | 18 |
| 19 | 20 | 21 | 22 | 23 | 24 |

## Sumário

|                          |   |
|--------------------------|---|
| Links .....              | 4 |
| Descrição da Ideia ..... | 5 |
| Arquitetura da IA.....   | 6 |

## Links

**link do repositório:** <https://github.com/nichol6s/analyzer-ai>

**link do vídeo:** [https://youtu.be/M\\_N6MQADyug](https://youtu.be/M_N6MQADyug)

## 1.Descrição da Ideia

A nossa solução consiste em um aplicativo de análise de campanhas de e-mail marketing voltados para a área de ecommerce usando Inteligência artificial. Essa IA vai ser treinada para classificar melhorias nas campanhas dos clientes usando o algoritmo KNN juntamente com outras bibliotecas bem famosas como Pandas, ScikitLearn, SNS, Matplotlib etc.

## 2. Arquitetura da IA

### 1. Teremos a Coleta e Pré-processamento de Dados

- **Coleta de Dados:** Os dados são carregados a partir de um arquivo CSV utilizando a biblioteca Pandas. Este arquivo contém métricas de desempenho das campanhas de e-mail, como cliques, aberturas e leads nos primeiros 7 e 30 dias.
- **Pré-processamento:**
  - As colunas são inspecionadas e o DataFrame é configurado para a análise.
  - Uma nova coluna, Melhoria, é criada para classificar se houve melhora significativa na campanha. A classificação é baseada em um aumento de mais de 10% em pelo menos duas das três métricas principais (cliques, aberturas, leads) após 30 dias comparados aos primeiros 7 dias.

### 2. Divisão de Dados

- Os dados são divididos em conjuntos de treino e teste usando `train_test_split` do Scikit-Learn. Isso permite avaliar a performance do modelo em dados não vistos durante o treinamento.

### 3. Normalização dos Dados

- O `StandardScaler` é utilizado para normalizar os dados. A normalização é crucial para algoritmos baseados em distância, como o KNN, pois garante que todos os recursos contribuam igualmente para a distância calculada.

#### **4. Modelo de Aprendizado de Máquina**

- Algoritmo K-Nearest Neighbors (KNN): Este algoritmo é escolhido devido à sua simplicidade e eficácia em problemas de classificação, especialmente quando o objetivo é identificar padrões de melhoria em dados de campanhas de marketing.
- Implementação: O modelo KNN é treinado com 5 vizinhos (`n_neighbors=5`). O modelo é ajustado com os dados normalizados de treino e, em seguida, testado com o conjunto de teste.

#### **5. Avaliação do Modelo**

- A acurácia do modelo é calculada e um relatório de classificação é gerado para avaliar a performance do KNN. Métricas como precisão, recall e f1-score são utilizadas para entender como o modelo está classificando as melhorias nas campanhas.

#### **Justificativa da Escolha da Arquitetura**

- O KNN é um algoritmo simples e fácil de interpretar e bem eficaz pra conjuntos de dados do tamanho do nosso, médio porte. Sem contar sua flexibilidade que é bem notável também, podemos realizar ajustes ou inclusão de outras métricas bem facilmente com ele.

#### **Implementação**

A implementação foi feita com Python, usando bibliotecas como Pandas para manipulação de dados, Scikit-Learn para modelagem e avaliação, e Matplotlib e Seaborn para visualização de dados. O código é estruturado de forma a carregar os dados, pré-processá-los, treinar o modelo e avaliar seu desempenho de maneira eficiente.

Essa abordagem modular facilita a manutenção e a expansão do projeto, permitindo adicionar novos algoritmos ou ajustar o pré-processamento conforme necessário.