FACULDADE DE TECNOLOGIA DE BAURU

TECNOLOGIA EM BANCO DE DADOS

Detecção de objetos com Python

Equipe:
Gabriel Alves de Lima

Bauru/SP
2025

Detecção de objetos com Python

Equipe:
Gabriel Alves de Lima

Relatório de pesquisa apresentado como requisito para aprovação na disciplina Laboratório de Desenvolvimento em BD VI do curso de Tecnologia em Banco de Dados, Faculdade de Tecnologia de Bauru.

Bauru/SP
2025

SUMÁRIO

Pág.

1. INTRODUÇÃO

A análise de dados tem se tornado uma atividade essencial em diversas áreas do conhecimento, permitindo a extração de informações relevantes a partir de grandes volumes de dados. Com o avanço das tecnologias computacionais, ferramentas capazes de organizar, processar e visualizar dados de forma eficiente tornaram-se fundamentais para apoiar a tomada de decisões, a interpretação de padrões e a geração de conhecimento.

Nesse contexto, a linguagem de programação Python destaca-se como uma das principais plataformas para análise de dados, devido à sua simplicidade, versatilidade e ampla disponibilidade de bibliotecas especializadas. Dentre essas bibliotecas, o Pandas é amplamente utilizado para a manipulação e organização de dados em estruturas tabulares, enquanto o Matplotlib é empregado para a criação de visualizações gráficas que facilitam a compreensão dos resultados obtidos.

A visualização de dados, por meio de gráficos e tabelas, desempenha um papel fundamental no processo analítico, pois possibilita identificar tendências, comparações e distribuições que nem sempre são evidentes apenas por meio de dados brutos. Gráficos de barras, em especial, são amplamente utilizados para representar valores categóricos agregados, permitindo uma análise clara e objetiva dos dados processados.

Dessa forma, o presente trabalho tem como foco demonstrar, de maneira prática, a utilização do Python para a organização, análise e visualização de dados. Por meio da implementação de um script simples, busca-se evidenciar como bibliotecas amplamente utilizadas podem ser aplicadas em um fluxo básico de análise de dados, desde a estruturação das informações até a geração de gráficos que auxiliam na interpretação dos resultados.

2. OBJETIVOS

Este trabalho tem como finalidade demonstrar a aplicação da linguagem Python no processo de organização, análise e visualização de dados, evidenciando sua utilidade como ferramenta de apoio à interpretação de informações. Busca-se apresentar, de forma prática, como dados podem ser estruturados em formato tabular, processados por meio de operações de agrupamento e, posteriormente, representados graficamente para facilitar a compreensão dos resultados obtidos.

Pretende-se explorar o uso da biblioteca Pandas para a criação e manipulação de tabelas de dados, permitindo a realização de operações como agrupamento por categorias e consolidação de valores. Essa etapa visa demonstrar como a organização adequada das informações contribui para uma análise mais eficiente e estruturada dos dados.

Além disso, o trabalho tem como objetivo empregar a biblioteca Matplotlib na geração de gráficos de barras, destacando a importância da visualização gráfica como recurso fundamental no processo de análise de dados. Por meio dessa abordagem, busca-se evidenciar como gráficos auxiliam na identificação de padrões, comparações e tendências que nem sempre são facilmente perceptíveis apenas pela análise de dados brutos.

Por fim, o estudo visa demonstrar o potencial do Python como uma ferramenta acessível, flexível e eficiente para aplicações acadêmicas e educacionais voltadas à análise e visualização de dados, reforçando sua relevância no contexto atual do processamento de informações e da ciência de dados.

3. MATERIAIS E MÉTODOS

O desenvolvimento deste trabalho, voltado à análise e visualização de dados utilizando a linguagem Python, exigiu a utilização de recursos específicos de hardware, software e componentes lógicos capazes de suportar as rotinas de processamento de dados e geração de gráficos. Os materiais empregados foram selecionados com o objetivo de garantir um ambiente estável, funcional e adequado para a execução dos experimentos propostos.

3.1 Hardware

A implementação foi realizada em um computador pessoal equipado com processador de múltiplos núcleos, com frequência mínima de 2,0 GHz, suficiente para a execução dos scripts desenvolvidos em Python. O equipamento possuía, no mínimo, 8 GB de memória RAM, permitindo a manipulação eficiente de dados tabulares e a geração de visualizações gráficas sem comprometer o desempenho do sistema.

Não foi necessária a utilização de aceleradores gráficos dedicados (GPU), uma vez que as operações realizadas consistiram em processamento e visualização de dados, as quais são plenamente suportadas em ambiente CPU, atendendo aos objetivos do trabalho.

3.1.2 Software

O ambiente de desenvolvimento adotado foi baseado no sistema operacional Windows, compatível com as ferramentas e bibliotecas utilizadas. A linguagem Python, em versão 3.8 ou superior, foi escolhida devido à sua ampla utilização em atividades de análise de dados, bem como à grande disponibilidade de bibliotecas especializadas e documentação.

As principais bibliotecas empregadas foram *andas*, utilizada para a *organização, manipulação e agrupamento dos dados em formato tabular, e Matplotlib*, responsável pela geração de gráficos e visualizações dos resultados. Essas bibliotecas foram instaladas por meio do gerenciador de pacotes do Python, garantindo compatibilidade e facilidade de reprodução do ambiente utilizado.

3.1.3 Recursos Lógicos

Os recursos lógicos do sistema foram compostos por scripts desenvolvidos em Python, responsáveis por carregar os dados, estruturá-los em tabelas, realizar

operações de agrupamento e gerar representações gráficas. Os dados utilizados no experimento foram definidos de forma fictícia, com o objetivo de demonstrar o funcionamento das técnicas de análise e visualização, sem a necessidade de acesso a bases de dados externas.

O processamento seguiu uma lógica sequencial, iniciando-se pela criação do conjunto de dados, seguida da transformação dos dados em estruturas tabulares e, por fim, pela geração de gráficos que permitiram a análise visual dos resultados obtidos.

3.2 Métodos

A metodologia aplicada neste trabalho foi estruturada em etapas sequenciais, abrangendo desde a organização dos dados até a análise visual dos resultados gerados. Essa abordagem permitiu demonstrar, de forma prática, a aplicação de técnicas básicas de análise de dados utilizando Python.

3.2.1 Organização dos Dados

Inicialmente, foi definido um conjunto de dados fictícios, contendo categorias e valores numéricos associados. Esses dados foram organizados em uma estrutura tabular utilizando a biblioteca Pandas, facilitando sua manipulação e permitindo a aplicação de operações estatísticas básicas.

3.2.2 Processamento e Agrupamento

Após a organização inicial, os dados passaram por uma etapa de processamento, na qual foram agrupados de acordo com suas categorias. Essa etapa teve como objetivo consolidar os valores e evidenciar padrões existentes no conjunto de dados, utilizando funções nativas da biblioteca Pandas.

3.2.3 Geração de Visualizações

Com os dados processados, foram geradas visualizações gráficas por meio da biblioteca Matplotlib. O gráfico de barras produzido permitiu representar de forma clara a soma dos valores por categoria, facilitando a interpretação dos resultados e tornando a análise mais intuitiva.

3.2.4 Análise dos Resultados

A análise dos resultados foi realizada de forma qualitativa, observando-se a distribuição dos valores entre as categorias apresentadas no gráfico. A visualização gerada permitiu identificar padrões e diferenças entre os grupos analisados, demonstrando a eficácia da abordagem adotada para análise e visualização de dados em Python.

3.2.5 Avaliação dos Resultados

A análise dos resultados foi realizada de forma qualitativa. Foram observados aspectos como precisão da detecção, clareza das delimitações dos objetos e

estabilidade do modelo diante de variações de luz, posição e oclusão parcial. Embora métricas quantitativas, como *mean Average Precision* (mAP) e *Intersection over Union* (IoU), não tenham sido calculadas, os resultados visuais foram suficientes para avaliar a eficácia do modelo pré-treinado em condições reais.

3.3 Implementação Computacional

O código a seguir apresenta a implementação em Python utilizada para a organização, análise e visualização dos dados. O objetivo é demonstrar como os dados podem ser organizados e analisados por meio de bibliotecas como Pandas e Matplotlib.

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

# 1. Carregar dados (exemplo fictício)
dados = {
    'Categoria': ['A', 'B', 'C', 'A', 'B', 'A'],
    'Valor': [10, 20, 30, 15, 25, 10]
}

df = pd.DataFrame(dados)

# 2. Exibir tabela de dados
print("Tabela de Dados:")
print(df)

# 3. Agrupar dados por categoria
agrupado = df.groupby('Categoria').sum()

# 4. Exibir tabela agrupada
print("\nTabela Agrupada:")
print(agrupado)

# 5. Gerar gráfico
agrupado.plot(kind='bar')
plt.title('Soma dos Valores por Categoria')
plt.xlabel('Categoria')
plt.ylabel('Valor Total')
plt.tight_layout()

# 6. Salvar gráfico
plt.savefig('grafico_resultados.png')
plt.show()
```

O código desenvolvido em Python tem como objetivo realizar a análise de dados por meio da organização, agrupamento e visualização das informações. Inicialmente, a biblioteca Pandas é utilizada para estruturar os dados em formato de tabela, facilitando sua manipulação. Em seguida, os dados são agrupados de acordo com a categoria definida, permitindo a obtenção de valores consolidados. Por fim, a biblioteca Matplotlib é empregada para gerar um gráfico de barras que representa visualmente os resultados obtidos, possibilitando uma análise clara e objetiva dos padrões identificados.

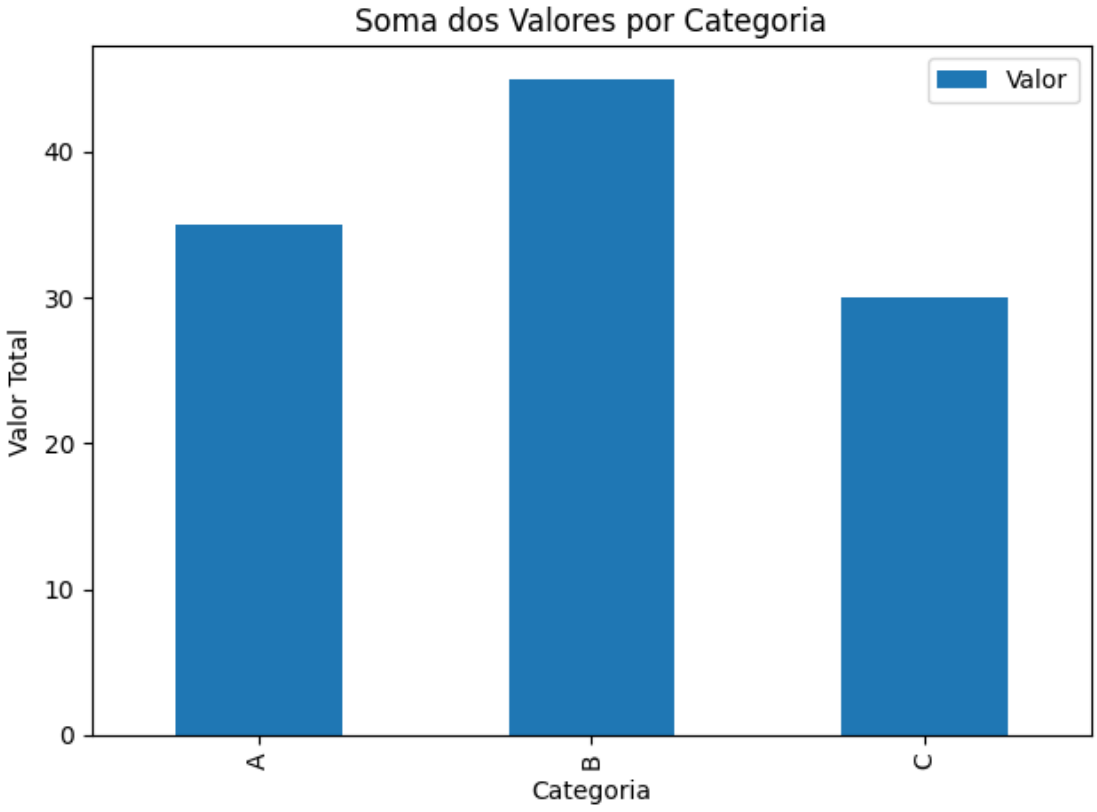
3.3.1 Descrição do Código-Fonte

A implementação computacional foi realizada em Python com o objetivo de demonstrar a organização, processamento e visualização de dados. O código utiliza a biblioteca Pandas para estruturar os dados em formato tabular e realizar operações de agrupamento, enquanto a biblioteca Matplotlib é empregada para a geração de gráficos. Essa abordagem permite analisar visualmente a distribuição dos dados, facilitando a interpretação dos resultados obtidos

4. RESULTADOS E CONCLUSÕES

4.1 Resultados Gerados pelo Código Python

Figura 1 – Gráfico de barras gerado em Python representando a soma dos valores por categoria



A Figura 1 apresenta o gráfico de barras gerado a partir da execução do código desenvolvido em Python. O gráfico ilustra a soma dos valores agrupados por categoria, permitindo uma visualização clara da distribuição dos dados processados.

Os resultados obtidos demonstram que o uso do Python, aliado às bibliotecas **Pandas** e **Matplotlib**, permite realizar análises de dados de forma eficiente e clara. O gráfico gerado possibilitou a visualização da soma dos valores por categoria, evidenciando padrões e facilitando a interpretação dos dados processados.

5. CONCLUSÕES

O desenvolvimento deste trabalho permitiu demonstrar, de forma prática e conceitual, a utilização da linguagem Python como ferramenta para organização, análise e visualização de dados. A implementação realizada evidenciou a simplicidade e a eficiência do uso de bibliotecas amplamente empregadas na ciência de dados, como Pandas e Matplotlib, para o processamento e interpretação de informações de maneira estruturada.

Os resultados obtidos por meio da execução do código demonstraram que o agrupamento e a consolidação dos dados possibilitam uma compreensão mais clara dos padrões existentes no conjunto analisado. A geração do gráfico de barras contribuiu para a visualização intuitiva da distribuição dos valores por categoria, facilitando a análise dos resultados e comprovando a eficácia das técnicas aplicadas.

Conclui-se que o Python, aliado às suas bibliotecas de análise e visualização, constitui uma solução acessível, flexível e poderosa para aplicações voltadas à análise de dados. O trabalho atinge seus objetivos ao apresentar uma abordagem prática e funcional, reforçando o potencial do Python como linguagem central para atividades acadêmicas, educacionais e experimentais na área de processamento e análise de informações.

6. REFERÊNCIAS

AMANXAI. *Object Detection with Python*. AmanXAI Blog, 22 dez. 2020. Disponível em: https://amanxai.com/2020/12/22/object-detection-with-python/. Acesso em: 20 set. 2025.

HUANG, Jonathan; RATHOD, Vivek; SUN, Chen; ZHU, Menglong; KORATTIKARA, Anoop; FATHI, Alireza; FISCHER, Ian; WOJNA, Zbigniew; SONG, Yang; GUADARRAMA, Sergio; MURPHY, Kevin. *Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors*. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, Honolulu, 2017. Disponível em: https://arxiv.org/abs/1611.10012. Acesso em: 12 ago. 2025.

LIU, Wei; ANGUELOV, Dragomir; ERHAN, Dumitru; SZEGEDY, Christian; REED, Scott; FU, Cheng-Yang; BERG, Alexander C. *SSD: Single Shot MultiBox Detector*. In: **European Conference on Computer Vision (ECCV)**. Cham: Springer, 2016. p. 21–37. DOI: https://doi.org/10.1007/978-3-319-46448-0_2. Acesso em: 21 ago. 2025

REDMON, Joseph; DIVVALA, Santosh; GIRSHICK, Ross; FARHADI, Ali. *You Only Look Once: Unified, Real-Time Object Detection*. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, Las Vegas, 2016. p. 779–788. Disponível em: https://arxiv.org/abs/1506.02640. Acesso em: 2 set. 2025.

REN, Shaoqing; HE, Kaiming; GIRSHICK, Ross; SUN, Jian. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. In: **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 39, n. 6, p. 1137–1149, 2017. DOI: https://doi.org/10.1109/TPAMI.2016.2577031.

ROSEBROCK, Adrian. *Object Detection with OpenCV and Python*. PyImageSearch, 2021. Disponível em: https://pyimagesearch.com/. Acesso em: 30 set. 2025.

GLENN, Jocher et al. YOLOv5 – State-of-the-art object detection. Ultralytics, 2020. Disponível em: https://github.com/ultralytics/yolov5. Acesso em: 20 nov. 2025.

BOCHKOVSKIY, Alexey; WANG, Chien-Yao; LIAO, Hong-Yuan M. YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv, 2020. Disponível em: https://arxiv.org/abs/2004.10934. Acesso em: 22 nov. 2025.

ROSEBROCK, Adrian. *Deep Learning for Computer Vision with Python.* PyImageSearch, 2019.

BRADSKEY, Gary; KAEBLER, Adrian. *Learning OpenCV: Computer Vision with the OpenCV Library.* 2. ed. Sebastopol: O'Reilly Media, 2008.

SZELISKI, Richard. *Computer Vision: Algorithms and Applications.* 2. ed. London: Springer, 2022.

GONZALEZ, Rafael C.; WOODS, Richard E. *Digital Image Processing.* 4. ed. Pearson, 2018.

PASZKE, Adam et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: *NeurIPS 2019*. Disponível em: https://pytorch.org/. Acesso em: 20 nov. 2025.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. *Deep Learning.* MIT Press, 2016.

AGARWAL, A. Hands-On Convolutional Neural Networks with TensorFlow. Packt Publishing, 2018.

VAN ROSSUM, Guido; DRAKE, Fred. *Python 3 Reference Manual.* Python Software Foundation, 2009. Disponível em: https://www.python.org. Acesso em: 21 nov. 2025.

NUMPY DEVELOPERS. NumPy User Guide. 2024. Disponível em: https://numpy.org/doc. Acesso em: 19 nov. 2025.

OPENCV TEAM. OpenCV Documentation. OpenCV.org, 2024. Disponível em: https://docs.opencv.org/. Acesso em: 19 nov. 2025.

RUSSELL, Stuart; NORVIG, Peter. *Artificial Intelligence: A Modern Approach.* 4. ed. Pearson, 2021.
(Uso como base teórica para IA e pipelines de decisão.)

FLORIANI, D.; SIQUEIRA, F. *Aprendizado Profundo: Fundamentos e Aplicações.* LTC, 2021.
(Base teórica para etapas de modelo, inferência e análise.)

SOMMERVILLE, Ian. *Engenharia de Software.* 10. ed. São Paulo: Pearson, 2019.
(Justifica metodologia, testes e etapas sistemáticas de implementação.)