



# Inteligência de Negócios (Business Intelligence – BI)

**DASHBOARD  
PYTHON STREAMLIT**



**PROF. Me UJEVERSON TAVARES**



<https://www.youtube.com/watch?v=8uJJeye-rOQ>



# Streamlit

O *Streamlit* é uma biblioteca de código aberto em Python que simplifica a criação de aplicações web interativas, especialmente voltadas para ciência de dados e aprendizado de máquina. Com uma API intuitiva, permite transformar scripts de dados em aplicativos funcionais com poucas linhas de código, eliminando a necessidade de conhecimentos avançados em desenvolvimento web



# Streamlit



## Principais vantagens do Streamlit:

**Facilidade de Uso:** A curva de aprendizado é suave, permitindo que desenvolvedores criem aplicações rapidamente sem se preocupar com detalhes complexos de front-end.

**Integração com Bibliotecas Populares:** Compatível com ferramentas como Pandas, Matplotlib e Keras, facilitando a incorporação de análises e visualizações de dados nos aplicativos.



# Streamlit



## Principais vantagens do Streamlit:

**Prototipagem Rápida:** Ideal para criar e compartilhar rapidamente modelos e análises, agilizando o processo de tomada de decisão baseada em dados.

**Interatividade:** Oferece widgets como sliders, botões e caixas de seleção, permitindo que os usuários interajam dinamicamente com os dados apresentados.

**Atualização em Tempo Real:** As aplicações refletem instantaneamente as mudanças no código, facilitando o desenvolvimento iterativo e a visualização imediata dos resultados.



# Streamlit



1. Acessar o Banco de Dados Estatísticos do Estado de Goiás (BDE-Goiás); <https://www.imb.go.gov.br/bde/imp.php>
2. Escolher dois tipos de variáveis
3. Escolher todas os municípios de Goiás;
4. Escolher o intervalo de tempo, 2004 - 2024;
5. Importar, explorar e manipular dados.
6. Realizar uma Análise Exploratória de Dados (EDA): Entender a estrutura do dataset, identificar e tratar dados faltantes.
7. Calcular KPIs para Business Intelligence (3 no mínimo)



<https://www.imb.go.gov.br/bde/imp.php>

O Banco de Dados Estatísticos do Estado de Goiás (BDE-Goiás) é um sistema de informações estatísticas relativas ao Estado de Goiás e a seus municípios. Contém séries históricas que, para algumas variáveis, cobrem desde o ano de 1980.

São informações das áreas física, econômica, social, financeira, política e administrativa, que podem ser pesquisadas por municípios, regiões geográficas do IBGE, regiões de planejamento do governo do Estado e total do Estado.

O BDE-Goiás é um banco de dados dinâmico. As consultas podem ser montadas conforme necessidade, interesse e critérios do usuário, sendo possível realizar tabulações e cruzamentos a partir das diversas séries históricas disponíveis. Para algumas consultas o BDE-Goiás possibilita, adicionalmente, a geração de mapas temáticos.

O sistema BDE-Goiás é resultado do sistema IMP desenvolvido pela Fundação Seade do Estado de São Paulo, cuja doação à Associação Nacional das Instituições de Planejamento, Pesquisa e Estatística (Anipes) permitiu seu uso pela Secretaria-Geral da Governadoria-GO/IMB, que promoveu as necessárias adequações. As melhorias realizadas no IMP pelo Instituto Paranaense de Desenvolvimento Econômico e Social (Ipardes) repassadas a esta instituição e o apoio técnico, enriqueceram o resultado final que, com muita satisfação, colocamos à disposição dos usuários.

**Seja bem-vindo ao Banco de Dados Estatísticos de Goiás!**

Conheça também o [Serviço de Informações de Dados do BDE](#).



▶ Iniciar Pesquisa

▶ Pesquisas Gravadas

? **Dúvidas**  
... veja como fazer ▶







## Preparação do Ambiente

Antes de fazer a implementação prática, precisamos instalar o *Streamlit* web framework. Basta executar o comando abaixo no prompt de comando.



```
pip install streamlit
```

Vamos verificar se o *streamlit* foi instalado com sucesso ou não. Basta digitar o comando, a seguir:

```
streamlit --version
```

```
C:\Users\Ujeverson Tavares>streamlit --version  
Streamlit, version 1.44.0
```





# Preparação do Ambiente

Vamos verificar se o streamlit foi instalado com sucesso ou não. Basta executar o comando abaixo, no prompt de comando.

`streamlit hello`

```
C:\Users\Ujeverson Tavares>streamlit hello

Welcome to Streamlit!

If you'd like to receive helpful onboarding emails, news, offers, promotions,
and the occasional swag, please enter your email address below. Otherwise,
leave this field blank.

Email:

You can find our privacy policy at https://streamlit.io/privacy-policy

Summary:
- This open source library collects usage statistics.
- We cannot see and do not store information contained inside Streamlit apps,
  such as text, charts, images, etc.
- Telemetry data is stored in servers in the United States.
- If you'd like to opt out, add the following to %userprofile%/.streamlit/config.toml,
  creating that file if necessary:

[browser]
gatherUsageStats = false

Welcome to Streamlit. Check out our demo in your browser.


Local URL: http://localhost:8501
Network URL: http://192.168.1.8:8501

Ready to create your own Python apps super quickly?
Head over to https://docs.streamlit.io


May you create awesome apps!
```





# Preparação do Ambiente

 Hello

 DataFrame demo

 Plotting demo

 Mapping demo

 Animation demo

## Welcome to Streamlit! 🖐️

Streamlit is an open-source app framework built specifically for machine learning and data science projects. 🖐️ **Select a demo from the sidebar** to see some examples of what Streamlit can do!

### Want to learn more?

- Check out [streamlit.io](https://streamlit.io)
- Jump into our [documentation](#)
- Ask a question in our [community forums](#)

### See more complex demos

- Use a neural net to [analyze the Udacity Self-driving Car Image Dataset](#)
- Explore a [New York City rideshare dataset](#)



## Estrutura inicial

Crie um arquivo chamado `dashboard_app.py` e insira:

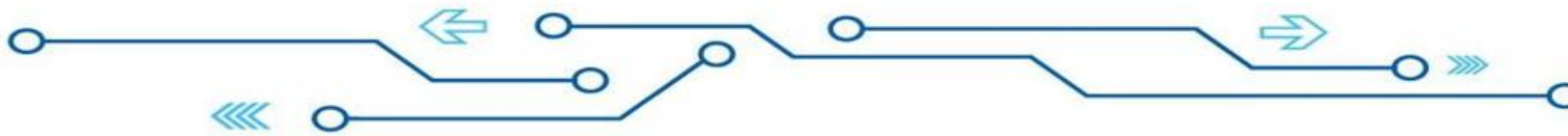
```
1 # Importação das bibliotecas necessárias
2 import streamlit as st          # Biblioteca para criação de dashboards
3 import pandas as pd             # Biblioteca para manipulação de dados
4 import matplotlib.pyplot as plt # Para gráficos
```



## Carregar os dados



```
1 # Carregando o arquivo de dados
2 df = pd.read_csv("finanAgroGoias2014-2024.csv" , encoding='utf-8', sep=';')
3 df.head()
```



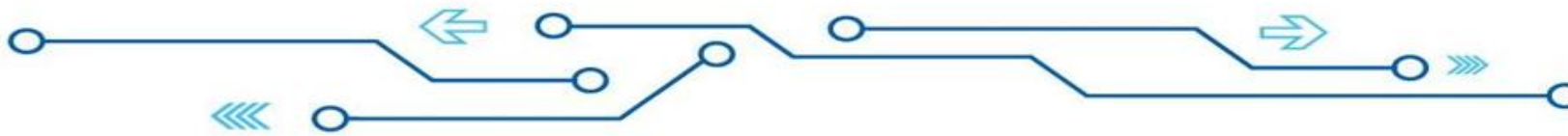
# Tratamento dos dados

|   | Localidade         | 2014          | 2015          | 2016          | 2017          | 2018       | 2019        | 2020        | 2021        | 2022        | 2023        | 2024        |
|---|--------------------|---------------|---------------|---------------|---------------|------------|-------------|-------------|-------------|-------------|-------------|-------------|
| 0 | Abadia de Goiás    | 1.682.176,83  | 359.230,36    | 843.952,23    | 524.909,59    | 1.102.677  | 944.144     | 2.116.228   | 4.646.637   | 3.565.464   | 2.370.000   | 2.819.375   |
| 1 | Abadiânia          | 14.843.437,29 | 15.326.296,30 | 19.138.436,47 | 24.447.300,06 | 17.306.762 | 20.822.652  | 21.229.967  | 35.531.783  | 40.681.067  | 37.894.251  | 43.155.083  |
| 2 | Acreúna            | 73.946.829,36 | 66.921.824,54 | 65.892.466,02 | 57.529.854,14 | 79.768.468 | 89.626.284  | 89.487.837  | 160.586.438 | 175.883.187 | 138.525.655 | 144.786.963 |
| 3 | Adelândia          | 47.155,75     | 53.000,00     | 96.800,00     | -             | 1.013.365  | 647.285     | 1.340.922   | 2.699.809   | 7.966.268   | 840.000     | 5.645.106   |
| 4 | Água Fria de Goiás | 97.449.438,98 | 97.130.008,25 | 79.667.062,28 | 81.113.135,54 | 97.395.702 | 122.200.333 | 167.341.415 | 177.472.154 | 111.771.836 | 105.053.474 | 255.453.433 |

Existem valores ausentes no data frame. Vamos trocar todos os “-” por zero “0”.



```
1 #troca todos os '-' por zero
2 df = df.replace('-', '0')
3 df.head(10)
```



# Tratamento dos dados

Verificar o tipo de dados das colunas.



```
1 #verificar o tipo de dado das colunas
2 df.dtypes
```

```
Localidade    object
2014          object
2015          object
2016          object
2017          object
2018          object
2019          object
2020          object
2021          object
2022          object
2023          object
2024          object
dtype: object
```



# Tratamento dos dados

Verificar o tipo de dados das colunas.



```
1 #verificando o tipo de dado das colunas
2 tipos = df['2014'].apply(type)
3 print(tipos)
```

```
0      <class 'str'>
1      <class 'str'>
2      <class 'str'>
3      <class 'str'>
4      <class 'str'>
...
241     <class 'str'>
242     <class 'str'>
243     <class 'str'>
244     <class 'str'>
245     <class 'str'>
Name: 2014, Length: 246, dtype: object
```





# Tratamento dos dados

Passando os valores das colunas para o tipo float.

```
1 # df.iloc[:, 1:] seleciona todas as colunas, exceto a primeira
2 # col.str.replace('.', '', regex=False) remove os pontos e
3 # col.str.replace(',', '.', regex=False) troca a vírgula por ponto
4 df.iloc[:, 1:] = df.iloc[:, 1:].apply(
5     lambda col: pd.to_numeric(
6         col.str.replace('.', '', regex=False).str.replace(',', '.', regex=False),
7         errors='coerce'
8     )
9 )
10 df.head(10)
```



# Tratamento dos dados

Passando os valores das colunas para o tipo float.



```
1 #Verificando o tipo de dado da coluna 2020
2 tipos = df['2020'].apply(type)
3 tipos.head(10)
```

```
0    <class 'float'>
1    <class 'float'>
2    <class 'float'>
3    <class 'float'>
4    <class 'float'>
5    <class 'float'>
6    <class 'float'>
7    <class 'float'>
8    <class 'float'>
9    <class 'float'>
Name: 2020, dtype: object
```



# Tratamento dos dados

Arredondando os valores das colunas, para 2 casas decimais.



```
1 # Seleciona as colunas numéricas (int e float)
2 numeric_cols = df.select_dtypes(include=['float', 'int']).columns
3
4 # Aplica o arredondamento de 2 casas decimais nas colunas numéricas
5 df[numeric_cols] = df[numeric_cols].round(2)
```



# Tratamento dos dados

Criando uma coluna com a média de investimentos para cada localidade.



```
1 # Calcula a média das colunas a partir da segunda (colunas de 2014 a 2024)
2 # para cada linha (localidade)
3 df["Md_Invest"] = df.iloc[:, 1:].mean(axis=1)
4
5 #arrendondar para duas casas decimais
6 df["Md_Invest"] = df["Md_Invest"].round(2)
```



# Tratamento dos dados

Inserir uma linha com a média de investimento para cada ano.

```
1 # Inserir uma linha com a média de investimento para cada ano
2
3 # 1. Calcula a média das colunas de investimento (excluindo a coluna 'Localidade')
4 # df.iloc[:, 1:] seleciona todas as colunas a partir da segunda (índices dos anos)
5 mean_values = df.iloc[:, 1:].mean()
6
7 # 2. Cria uma lista com o rótulo "Média" para a primeira coluna e os valores médios para as demais
8 nova_linha = ["Média"] + mean_values.tolist()
9
10 # 3. Converte a lista em um DataFrame com as mesmas colunas do df original
11 linha_media = pd.DataFrame([nova_linha], columns=df.columns)
12
13 # 4. Concatena a nova linha ao dataframe original (ignore_index=True para reindexar)
14 df = pd.concat([df, linha_media], ignore_index=True)
```



## Estrutura inicial do Dashboard

```
1 #iniciar o dashboard
2 # Configuração do layout da tabela na página
3 st.set_page_config(layout="wide")
4
5 # Configuração do título
6 st.title('Dashboard Financeiro - Agro Goiás - 2014-2024')
7 st.write('Dashboard Financeiro - Tabela')
8
9 # Calcula a média das colunas a partir da segunda (colunas de 2014 a 2024) para cada linha (localidade)
10 df["Md_Invest"] = df.iloc[:, 1:].mean(axis=1)
11 df["Md_Invest"] = df["Md_Invest"].round(2)
12 st.dataframe(df.style.format({'Md_Invest': '{:.2f}'}))
13 st.dataframe(df.head())
```



## Estrutura inicial

Abra o terminal e digite:

```
streamlit run dashboard_app.py
```





## Estrutura inicial

Cria um widget de seleção para escolher a localidade

```
1 # Cria um widget de seleção para escolher a localidade
2 # Removendo espaços antes e depois dos nomes das localidades (strip)
3 df["Localidade"] = df["Localidade"].astype(str).str.strip()
4 # Atualizar lista de localidades após limpeza
5 localidades = df.loc[df["Localidade"] != "Média", "Localidade"].unique().tolist()
6 # Opção Todos para seleção
7 opcoes_localidade = ["Todos"] + localidades
8 localidade_selecionada = st.sidebar.selectbox("Selecione a Localidade:", opcoes_localidade)
9 # Colunas com os anos (assumindo que primeira coluna é 'Localidade' e última 'Md_Invest')
10 colunas_anos = df.columns[1:-1]
```



```
1 # Filtro para localidade escolhida
2 if localidade_selecionada == "Todos":
3     # Filtrando excluindo a linha "Média"
4     df_plot = df[df["Localidade"] != "Média"]
5     # Calcula média por ano
6     investimentos_ano = df_plot[colunas_anos].mean()
7     st.write("Média dos investimentos por ano (Todas as localidades):")
8     st.bar_chart(investimentos_ano)
9 else:
10    # Certifique-se de que a seleção e os dados tenham exatamente o mesmo formato
11    df_local = df[df["Localidade"] == localidade_selecionada]
12    # Exibir dataframe para conferir se está correto (debug)
13    st.write(f"Dados filtrados para {localidade_selecionada}:")
14    st.dataframe(df_local)
15    if not df_local.empty:
16        investimentos_local = df_local.iloc[0][colunas_anos]
17        st.write(f"Evolução dos investimentos para {localidade_selecionada}:")
18        st.bar_chart(investimentos_local)
19    else:
20        st.error("Nenhum dado encontrado para essa localidade. Confira o nome exato.")
```



## Estrutura do Dashboard

Layout do Dashboard com gráficos distribuídos em duas linhas, cada linha com duas colunas.



```
1 #Definindo o layout do gráfico, primeira linha com 2 colunas
2 col1, col2 = st.columns(2)
3 #Segunda linha com 3 colunas
4 col3, col4 = st.columns(2)
```

```
1  with col1:
2      # Filtro para localidade escolhida
3      if localidade_selecionada == "Todos":
4          # Filtrando excluindo a linha "Média"
5          df_plot = df[df["Localidade"] != "Média"]
6          # Calcula média por ano
7          investimentos_ano = df_plot[colunas_anos].mean()
8          st.write("### Média dos investimentos por ano (Todas as localidades):")
9          st.bar_chart(investimentos_ano)
10     else:
11         # Certifique-se de que a seleção e os dados tenham exatamente o mesmo formato
12         df_local = df[df["Localidade"] == localidade_selecionada]
13         # Exibir dataframe para conferir se está correto (debug)
14         st.write(f"Dados filtrados para {localidade_selecionada}:")
15         st.dataframe(df_local)
16         if not df_local.empty:
17             investimentos_local = df_local.iloc[0][colunas_anos]
18             st.write(f"Evolução dos investimentos para {localidade_selecionada}:")
19             st.bar_chart(investimentos_local)
20         else:
21             st.error("Nenhum dado encontrado para essa localidade. Confira o nome exato.")
```





```
1 with col2:
2     st.write("### Gráfico de Linhas: Evolução dos Investimentos")
3     # Reaproveita os mesmos dados do gráfico de barras para a linha
4     if localidade_selecionada == "Todos":
5         investimentos_ano = df[df["Localidade"] != "Média"].loc[:, colunas_anos].mean()
6     else:
7         investimentos_ano = df_local.iloc[0][colunas_anos]
8     st.line_chart(investimentos_ano)
```

```
1 with col3:
2     st.write("### Comparação da Cidade com a Média Geral")
3     # Verifica se uma localidade específica foi selecionada
4     if localidade_selecionada != "Todos" and not df_local.empty:
5         # Pega os dados da localidade
6         investimentos_local = df_local.iloc[0][colunas_anos]
7         # Calcula a média geral das localidades (excluindo linha de média adicionada)
8         investimentos_media = df[df["Localidade"] != "Média"].loc[:, colunas_anos].mean()
9         # Cria DataFrame para visualização
10        df_comparacao = pd.DataFrame({
11            "Ano": colunas_anos,
12            localidade_selecionada: investimentos_local.values,
13            "Média Geral": investimentos_media.values
14        })
15        # Define o ano como índice para melhorar a visualização
16        df_comparacao.set_index("Ano", inplace=True)
17        # Exibe o gráfico
18        st.bar_chart(df_comparacao)
19    else:
20        st.info("Selecione uma localidade para visualizar a comparação.")
```



```
1  with col4:
2      # Passo 7: Exibir um quadro com o ano de maior investimento da cidade selecionada
3      st.write("### Ano com Maior Investimento")
4      # Verifica se foi selecionada uma localidade específica e se os dados filtrados não estão vazios
5      if localidade_selecionada != "Todos" and not df_local.empty:
6          # Seleciona os investimentos da localidade para as colunas dos anos
7          investimentos_local = df_local.iloc[0][colunas_anos]
8          # Encontra o ano com o maior investimento (o índice com o valor máximo)
9          ano_max = investimentos_local.idxmax()
10         valor_max = investimentos_local.max()
11         # Exibe a informação utilizando st.metric
12         st.metric(label="Ano com Maior Investimento", value=ano_max, delta=f"Valor: {valor_max:.2f}")
13     else:
14         st.info("Selecione uma localidade específica para visualizar o ano com maior investimento.")
```



