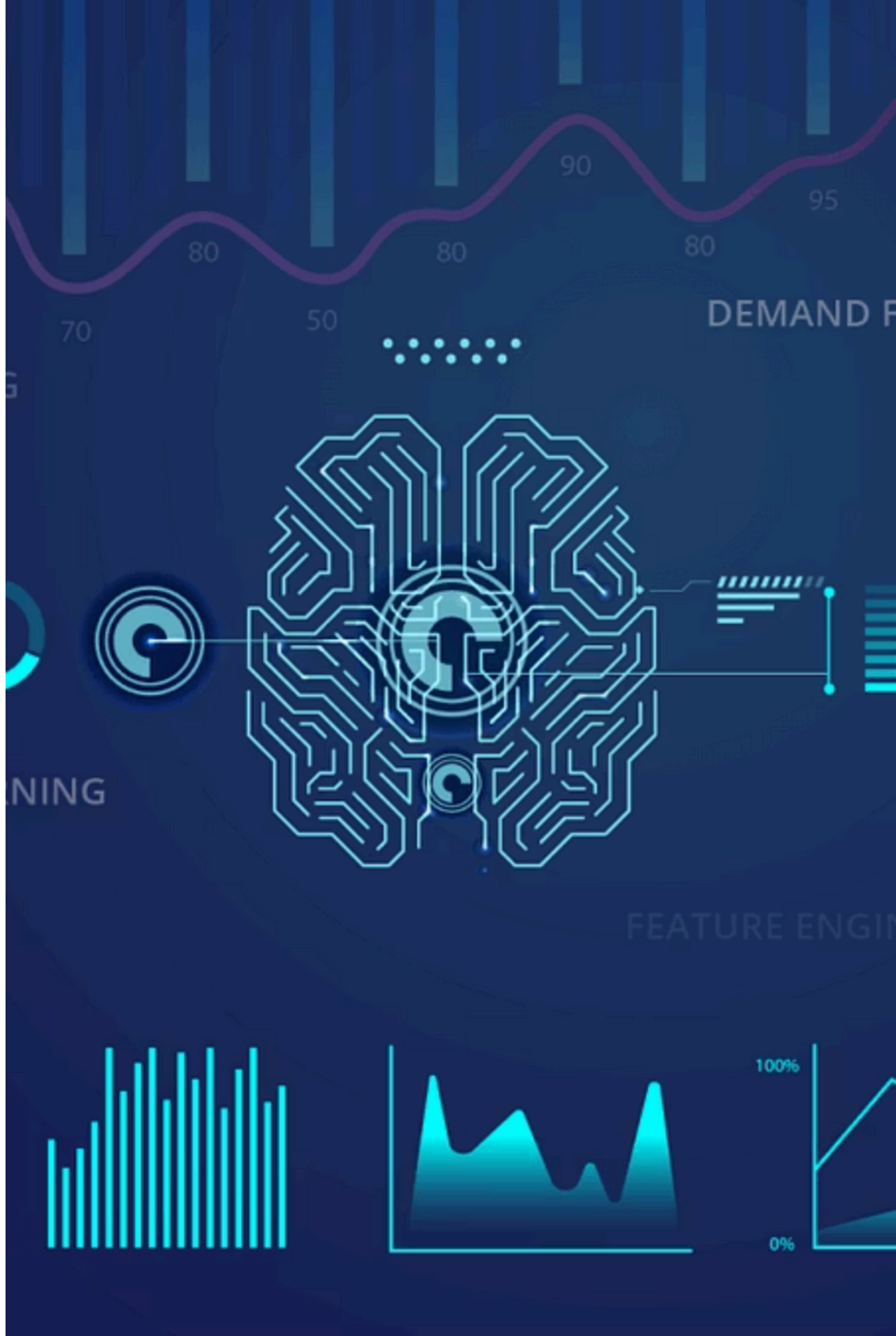
Building Interactive Data Apps with Streamlit

Streamlit

Presented By :
Ahmed Layouni

Overview

- Introduction to Streamlit
- Importance of Streamlit
- Streamlit use cases
- Benefits
- Streamlit Workflow
- Core Features
- Example Code
- Streamlit vs Other Tools
- Limitations



What is Streamlit?

Definition: Streamlit is a powerful and user-friendly open-source Python Framework that makes it easier to build interactive web applications for machine learning and data science.

Created for data scientists and ML engineers to:

- Visualize data interactively
- Share models and analysis without complex web dev

Why Streamlit is important?

- Fast prototyping of data dashboards and ML demos
- Pure Python (no HTML/CSS/JS required)
- Dynamic interactivity with sliders, buttons, inputs
- Auto-reload on code change
- Easy sharing with colleagues and stakeholders

Streamlit use cases

- Data exploration dashboards
- Interactive ML model deployment
- Sales & KPI monitoring dashboards
- Educational data demos
- Sharing notebooks as web apps

Benefits

Streamlit makes it easy to share data and machine learning models by turning Python scripts into interactive web apps.

It encourages users to explore data visually — for example, adjusting a slider to see how a model's predictions change.

You can deploy and share prototypes quickly, like a sales forecast dashboard or a simple recommendation system.
It's a "learn once, use often" tool that works well for ML pipelines, teaching concepts in classrooms, or building internal tools like data monitoring dashboards.

How It works?

- **Write a Python script using Streamlit functions**
- **Run using:**
  `streamlit run app.py`
- **Streamlit opens a web app in your browser**
- **Interact and iterate quickly**

Core Features

Display Elements:

- st.write(), st.dataframe(), st.metric(), st.image()

Interactive Widgets:

- st.slider(), st.text_input(), st.button(), st.selectbox()

Charts:

- Supports matplotlib, plotly, altair, native charts

Layouts:

- Sidebar, columns, expanders for structured UI

Example Code

Basic Example Code:

```python
import streamlit as st
import pandas as pd

st.title("My First Streamlit App")

name = st.text_input("Enter your name:")
st.write(f"Hello, {name}!")

x = st.slider("Pick a number", 0, 100, 50)
st.write(f"You selected: {x}")
```

Example Code

Data Exploration Example:

```
uploaded_file = st.file_uploader("Upload a CSV")
df = pd.read_csv(uploaded_file)
st.write(df.head())
column = st.selectbox("Select column", df.columns)
st.line_chart(df[column])
```

Example Code

ML Model Deployment Example:

Use Streamlit to deploy ML models interactively:

- Load trained model
- Get user input via sliders/text
- Predict and display results live

```
import joblib
model = joblib.load("model.pkl")
input = st.slider("Input feature", 0, 10, 5)
prediction = model.predict([[input]])
st.write(f"Prediction: {prediction}")
```

Streamlit Vs Other Tools

Feature	Streamlit	Dash	Gradio
Language	Python	Python	Python
Ease of Use	Very Easy	Moderate	Very Easy
UI Control	Limited	Flexible	Limited
Best For	Dashboards, ML apps	Complex dashboards	ML model demos

Limitations

- Limited advanced UI customization
- Re-runs entire script on interaction (unless using session state)
- Not ideal for multi-page complex apps without additional structuring
- Primarily designed for prototyping and lightweight dashboards