ML Tools

Streamlit

Building Interactive ML Applications

Kostya Numan
AI & AGI Researcher

Intro

Core Idea:

Streamlit turns Python scripts into shareable interactive web applications without requiring frontend expertise.

Why Use It:

Rapidly prototype and deploy ML models with user-friendly interfaces in minutes, not weeks.

Works With:

Any Python library - PyTorch, TensorFlow, scikit-learn, Hugging Face, NLTK, spaCy, and more.

Target Users:

Data scientists and ML engineers who want to share models without DevOps or frontend skills.

Kostya Numan

AI & AGI Researcher

Core Components

App Structure:

Single Python script that handles both computation and UI rendering - no separate frontend/backend.

State Management:

Automatic caching system prevents redundant computations when users interact with the app.

Reactive Flow:

Scripts rerun from top to bottom when user input changes, creating a reactive experience.

Deployment:

Run locally with `streamlit run` or deploy to Streamlit Cloud, AWS, GCP, or Azure.

Versioning:

Built-in Git integration for tracking changes and collaboration.

Kostya Numan

AI & AGI Researcher

Key UI Elements

Input Widgets:

- Sliders, buttons, and text inputs
- File uploaders for data ingestion
- Date and time selectors
- Multi-select and dropdown menus
- Color pickers and camera input

Output Elements:

- Dynamic charts and plots
- Interactive maps
- Progress bars and status indicators
- Expandable sections
- Formatted text with Markdown

Flexibility:

Mix and match elements to create anything from simple demos to complex dashboards.

Kostya Numan

AI & AGI Researcher

Data Visualization

Native Charts:

Built-in functions for line charts, bar charts, scatter plots, and histograms.

Integration:

Seamless support for Matplotlib, Plotly, Altair, Bokeh, and other visualization libraries.

Interactivity:

Interactive plots with zoom, pan, and hover-for-details functionality.

Maps:

Geographic data visualization with `st.map()` for location-based insights.

Auto-update:

Charts instantly refresh when underlying data changes.

Kostya Numan

AI & AGI Researcher

ML Integration

Model Inference:

Run predictions in real-time as users adjust parameters through the UI.

Caching:

Use `@st.cache_data` and `@st.cache_resource` to optimize heavy computations and model loading.

Image Pipeline:

Process images with OpenCV and display results instantly.

NLP Demos:

Create text analysis apps with real-time tokenization and classification visualization.

A/B Testing:

Compare models side-by-side with metrics and visualizations.

Kostya Numan

AI & AGI Researcher

Performance Tips

Lazy Loading:

Use conditional statements to only run expensive operations when needed.

Proper Caching:

Understand the difference between `cache_data` (for dataframes) and `cache_resource` (for models).

Session State:

Store computation results in `st.session_state` to maintain context between reruns.

Callbacks:

Use button callbacks to trigger specific functions rather than rerunning everything.

Dependency Management:

Create tight requirements.txt files to ensure quick deployments.

Kostya Numan

AI & AGI Researcher

Layout Options

Containers:

Group related elements with `st.container()` for logical organization.

Columns:

Create multi-column layouts with `st.columns()` for responsive designs.

Tabs:

Organize content with `st.tabs()` to create separate views within one app.

Expanders:

Hide complex details in collapsible sections with `st.expander()`.

Sidebar:

Place controls in a persistent sidebar with `st.sidebar` for cleaner interfaces.

Kostya Numan

AI & AGI Researcher

Customization

Theming:

Customize colors, fonts, and styles with `.streamlit/config.toml` files.

Custom CSS:

Inject custom styling with `st.markdown()` and HTML/CSS snippets.

Component Ecosystem:

Extend functionality with community-built components from streamlit.io/components.

Branding:

Add logos and custom favicons to match your organization's identity.

Authentication:

Add user login systems via Streamlit Cloud or custom middleware.

Kostya Numan

AI & AGI Researcher

Common Use Cases

ML Applications:

- Interactive model demos
- Exploratory data analysis tools
- Parameter tuning interfaces
- Dataset annotation systems
- Real-time inference dashboards

Business Tools:

- Data exploration for stakeholders
- Predictive analytics dashboards
- Custom reporting solutions
- Internal tool prototypes
- Client-facing model showcases

Time-to-Value:

Most applications can be built in hours or days instead of weeks.

Kostya Numan

AI & AGI Researcher

Getting Started

Installation:

Simple setup with `pip install streamlit` and a single Python file.

First App:

Create hello.py with `import streamlit as st` and `st.write('Hello World')`.

Launch:

Run with `streamlit run hello.py` to start development server.

Resources:

Extensive documentation, gallery, and community forums at streamlit.io.

Templates:

Jumpstart with pre-built templates for common ML applications.

Kostya Numan

AI & AGI Researcher

Advanced Features

WebRTC:

Stream real-time video with computer vision processing using Streamlit WebRTC.

Database Connection:

Connect to SQL databases using st.connection() for persistence and querying.

Forms:

Group inputs with st.form() to batch process multiple inputs at once.

Multiplexing:

Create multi-page applications using `streamlit.pages` directory structure.

API Integration:

Connect to external APIs and services for extended functionality.

Kostya Numan

AI & AGI Researcher

Subscribe

to Numan Substack
for more insights

Kostya Numan
AI & AGI Researcher