What is Streamlit?

Streamlit is an open-source Python library that makes it easy to create and share beautiful, custom web apps for machine learning and data science. With just a few lines of code, you can build interactive data applications without needing to know HTML, CSS, or JavaScript.

# 1. Creating the App Interface

```python

import streamlit as st

# Set app title

st.title("🏠 Rental Property Data Analysis - India")

```

- `st.title()` creates a main heading for your app

- Emojis (🏠) can be used to make the interface more visually appealing

# 2. File Upload

```python

uploaded\_file = st.file\_uploader("Upload your CSV file", type=['csv'])

if uploaded\_file:

df = pd.read\_csv(uploaded\_file)

st.success("File uploaded successfully!")

```

- `st.file\_uploader()` creates a widget for users to upload files

- We specify the allowed file types (CSV in this case)

- `st.success()` shows a success message when the file is uploaded

# 

# 3. Displaying Data

```python

st.write("Preview of dataset:", df.head())

```

- `st.write()` is a versatile function that can display dataframes, text, plots, and more

- It automatically formats the output appropriately

# 4. Creating Interactive Elements

```python

if st.checkbox("Show Missing Values"):

st.write(df.isnull().sum())

fill\_option = st.selectbox("Fill missing values with:", ["0", "-1", "Custom"])

```

- `st.checkbox()` creates a checkbox that shows/hides content

- `st.selectbox()` creates a dropdown menu for user selection

# 5. Creating Sections

```python

st.header("🔹 Data Cleaning")

st.header("🔹 Feature Engineering")

st.header("📊 Visualizations")

```

- `st.header()` creates section headers to organize your app

- You can also use `st.subheader()` for subsections

# 6. Displaying Visualizations

```python

plt.figure(figsize=(10, 6))

plt.hist(df['rent\_month'], bins=30, color='skyblue', edgecolor='black')

# ... more plotting code ...

st.pyplot(plt.gcf())

```

- Create plots using matplotlib/seaborn as usual

- `st.pyplot()` displays the plot in the Streamlit app

# 7. Adding Download Functionality

```python

csv = df.to\_csv(index=False).encode('utf-8')

st.download\_button(

label="Download Cleaned Data as CSV",

data=csv,

file\_name='cleaned\_rental\_data.csv',

mime='text/csv',

)

```

- `st.download\_button()` creates a button to download processed data

- We convert the DataFrame to CSV format for downloading

Key Streamlit Concepts

# 1. Widgets

Streamlit provides various interactive elements:

- `st.slider()` - for numeric range selection

- `st.text\_input()` - for text input

- `st.button()` - for clickable buttons

- `st.radio()` - for single selection from multiple options

- `st.multiselect()` - for multiple selections

# 2. Layout Control

You can organize your app layout:

```python

# Create columns

col1, col2 = st.columns(2)

with col1:

st.write("Content for column 1")

with col2:

st.write("Content for column 2")

```

# 3. Caching

To speed up your app, cache expensive computations:

```python

@st.cache\_data

def load\_data():

# Expensive data loading operation

return data

df = load\_data()

```

# 

# 4. Session State

Maintain state across reruns:

```python

# Initialize state

if 'counter' not in st.session\_state:

st.session\_state.counter = 0

# Update state

if st.button("Increment"):

st.session\_state.counter += 1

# Display state

st.write(f"Count: {st.session\_state.counter}")

```

## How to Run a Streamlit App

1. Save your code in a Python file (e.g., `app.py`)

2. Open your terminal/command prompt

3. Navigate to the directory containing your file

4. Run the command:

```

streamlit run app.py

```

5. Your app will open in a new browser tab

Best Practices for Beginners

1. Start Simple: Begin with basic elements and gradually add complexity

2. Organize Your Code: Use headers and sections to structure your app

3. Handle Errors: Use try-except blocks to gracefully handle errors

4. Add Explanations: Use `st.markdown()` or `st.write()` to explain what each section does

5. Test Frequently: Run your app often to check how changes look

6. Use Caching: Cache expensive operations to improve performance

7. Consider Responsiveness: Design your app to work on different screen sizes