Handout: Streamlit

1. Introduction to Streamlit

Streamlit is an open-source Python library used to create beautiful, interactive web applications with minimal code. It is designed specifically for data scientists, engineers, and analysts to turn data scripts into shareable web apps in just a few minutes. Unlike other frameworks, Streamlit requires no front-end web development skills.

Key Features:

- **Simple Syntax**: Streamlit apps are written in pure Python. No need for HTML, CSS, or JavaScript.
- **Interactive Widgets**: Easily add interactive widgets like sliders, text input, buttons, and more.
- **Real-time updates**: Streamlit automatically reruns the code whenever you interact with a widget, keeping the UI and data up to date.
- **Data Visualization**: Streamlit supports integration with popular libraries like Matplotlib, Plotly, Altair, and more to create interactive data visualizations.

Installation

You can install Streamlit using pip:

```
pip install streamlit
```

After installation, you can run a Streamlit app with:

```
streamlit run app.py
```

2. Example Code for different Widgets

Below are examples of how to use different Streamlit widgets like Labels, Textboxes, Buttons, ComboBoxes, etc.

a. Label (Text)

A label in Streamlit is created using `st.write()` or `st.text()`. You can use it to display static text in your application.

```
import streamlit as st

# Label
st.write("Welcome to my Streamlit Application!")
```

b. Textbox (Text Input)

The `st.text_input()` function can be used to create a textbox where users can input text.

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

c. Button

The `st.button()` function creates a button that triggers an action when clicked.

```
# Button
if st.button('Click Me'):
    st.write("Button clicked!")
```

d. ComboBox (Select Box)

You can use `st.selectbox()` to create a combobox or dropdown menu that lets users select one option from a list.

```
# ComboBox (Select Box)
option = st.selectbox('Choose a programming language:', ['Python', 'Java', 'C++'])
st.write(f'You selected: {option}')
```

e. Image

The `st.image()` function is used to display an image in your app.

```
from PIL import Image

# Load and display an image
image = Image.open('path_to_image.jpg')
st.image(image, caption='Sample Image', use_column_width=True)
```

f. Sidebar

Streamlit also provides a sidebar using `st.sidebar` where you can add widgets to the side panel.

```
# Sidebar with a select box
sidebar_selection = st.sidebar.selectbox('Choose a fruit:', ['Apple', 'Banana',
'Orange'])
st.sidebar.write(f'You selected: {sidebar_selection}')
```

g. Toggle (Checkbox)

A checkbox can be used to toggle between True and False states. You can create one using `st.checkbox()`.

```
# Toggle (Checkbox)
toggle = st.checkbox('Show/Hide')
if toggle:
    st.write("Checkbox is checked!")
else:
    st.write("Checkbox is unchecked.")
```

h. Radio Buttons

You can create radio buttons with `st.radio()` to allow users to choose from a set of options.

```
# Radio buttons
radio_selection = st.radio("Choose a color:", ['Red', 'Green', 'Blue'])
st.write(f'You selected: {radio_selection}')
```

i. SelectBox (Dropdown)

Similar to ComboBox, a SelectBox allows users to choose from a dropdown list.

```
# SelectBox
selectbox_option = st.selectbox('Choose a car brand:', ['Toyota', 'Ford', 'BMW'])
st.write(f'You selected: {selectbox_option}')
```

3. Example Code for DataFrames and Matplotlib Plots

Streamlit allows you to easily display Pandas DataFrames and Matplotlib plots.

a. Displaying a DataFrame

You can use the `st.dataframe()` function to display a Pandas DataFrame interactively.

```
import pandas as pd

# Sample DataFrame
data = {'Name': ['John', 'Anna', 'Peter', 'Linda'],
        'Age': [28, 24, 35, 32],
        'City': ['New York', 'Paris', 'Berlin', 'London']}

df = pd.DataFrame(data)
```

```
# Display the DataFrame
st.dataframe(df)
```

b. Matplotlib Plot

Streamlit supports Matplotlib for creating interactive charts. You can use the `st.pyplot()` function to display the plot.

```
import matplotlib.pyplot as plt
import numpy as np

# Sample Matplotlib plot
x = np.linspace(0, 10, 100)
y = np.sin(x)

plt.plot(x, y)
plt.title("Sine Wave")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")

# Display the plot
st.pyplot(plt)
```

4. Example of Streamlit application

we'll build a Streamlit app that trains a **RandomForestClassifier** on the **Digits Dataset** from `sklearn`. The app will include a sidebar that allows the user to select the number of estimators (trees) used in the Random Forest model. After clicking a button to start training, the app will display the classification score and confusion matrix.

Key Features:

- Sidebar for selecting the number of estimators (trees) for the Random Forest model.
- Button to trigger the training process.
- Display of model accuracy and a confusion matrix after training.

Complete Code

```
import streamlit as st
from sklearn.datasets import load_digits
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Set title for the app
st.title("Digit Dataset Classification using Random Forest")
```

```
# Load the digit dataset from sklearn
digits = load_digits()
X, y = digits.data, digits.target

# Display dataset information
st.write("The Digits dataset contains images of hand-written digits (0-9).")
st.write(f"Total samples: {X.shape[0]}, Number of features per sample:
{X.shape[1]}")

# Sidebar to select the number of estimators
n_estimators = st.sidebar.slider("Number of Trees in the Random Forest",
min_value=10, max_value=200, value=100, step=10)

# Sidebar button to start training
if st.sidebar.button("Start Training"):

    # Split the dataset into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

    # Initialize the RandomForestClassifier
    model = RandomForestClassifier(n_estimators=n_estimators, random_state=42)

    # Train the model
    model.fit(X_train, y_train)

    # Predict on the test set
    y_pred = model.predict(X_test)

    # Calculate accuracy
    accuracy = accuracy_score(y_test, y_pred)
    st.write(f"Accuracy of the model: {accuracy * 100:.2f}%")

    # Confusion Matrix
    cm = confusion_matrix(y_test, y_pred)

    # Display the confusion matrix using Seaborn's heatmap
    st.write("Confusion Matrix:")
    fig, ax = plt.subplots(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", ax=ax)
    st.pyplot(fig)
```

Running the App

To run this Streamlit app, save the code to a Python file (e.g., `app.py`) and run it using the following command:

```
streamlit run app.py
```

Once the app is running, you will see an interactive web page where you can adjust the number of estimators, click the button to train the model, and view the accuracy and confusion matrix.