**USING STREAMLIT TO DEVELOP REGRESSION MODEL DATA WEB APP**

**Table of Contents**

* Introduction
* What streamlit is about
* Why streamlit
* Prerequisite
* Setup and Installation
* Use your environment
* Create your user interface for model
* Prepare to load data and model
* **Create a Streamlit App Script**
* Import libraries
* **Setup the App Title and Description**
* **Create Input Widgets**
* **Load the Trained Regression Model**
* **Make Predictions Using User Inputs**
* **Display Predictions and Analysis**
* **Add Visualizations (Optional)**
* **Customize the Layout and Style**
* **Provide User Guidance**
* **Testing and Debugging**
* **Deployment** (Optional)
* **Documentation**
* Conclusion

**1.0 Introduction**

Data science has indeed been a rapidly growing field in recent years. It combines statistics, programming, and domain knowledge expertise to extract valuable insights and predictions from large and complex datasets. The data science community is known for its vibrant and collaborative nature, which has led to the development of a wide range of frameworks, libraries, and tools aimed at making the work of data scientists and researchers more efficient and productive. These tools help streamline various aspects of the data science workflow, from data preprocessing to model deployment. One such library is Streamlit. It is a valuable addition to the data science ecosystem and has gained popularity for its simplicity and speed in building interactive web applications with Python. It seamlessly integrates with other python libraries like NumPy, Pandas, Matplotlib, and many more.

**What is Streamlit**

**Streamlit** is a Python library that simplifies the process of building interactive web applications for data science. It allows developers to create intuitive user interfaces with minimal effort, enabling seamless exploration and presentation of data and models. Streamlit provides a wide range of interactive widgets and visualization tools, allowing users to customize and manipulate data on the fly. With its simplicity and ease of use, Streamlit enhances the iterative and collaborative nature of data science projects, enabling rapid prototyping, sharing, and deployment of data-driven applications.

**Why Streamlit?**

Currently, real-world applications are in high demand and developers are developing new libraries and frameworks to make on-the-go dashboards easier to build and deploy. Streamlit is a library that reduces your dashboard development time from days to hours. Following are some reasons to choose the Streamlit:

1. It is a free and open-source library.
2. Installing Streamlit is as simple as installing any other python package
3. It is easy to learn because you won’t need any web development experience, only a basic understanding of Python is enough to build a data application.
4. It is compatible with almost all machine learning frameworks, including Tensorflow and Pytorch, Scikit-learn, and visualization libraries such as Seaborn, Altair, Plotly, and many others.

**Required applications and packages**

We will need the following applications and packages to work with streamlit.

1. **Python** — We need at least the [python 3.7](https://www.python.org/downloads/) version or greater.
2. **pip** — We can install [pip](https://phoenixnap.com/kb/install-pip-windows) with the help of the terminal or using the code editor.
3. **Streamlit —** We have to install the Streamlit library before launching any Streamlit application. Run the following command in the terminal to install streamlit.
4. **virtualenv -** It allows you to create isolated Python environments for your projects, which can help avoid conflicts between different project dependencies.

**Setup and Installation**

Create a new environment with Streamlit

1. Navigate to your project folder:

cd myproject

1. Create a new virtual environment in that folder and activate that environment:

python -m venv .venv

When you run the command above, a directory called .venv will appear in myproject/. This directory is where your virtual environment and its dependencies are installed.

1. Install Streamlit in your environment:

pip install streamlit

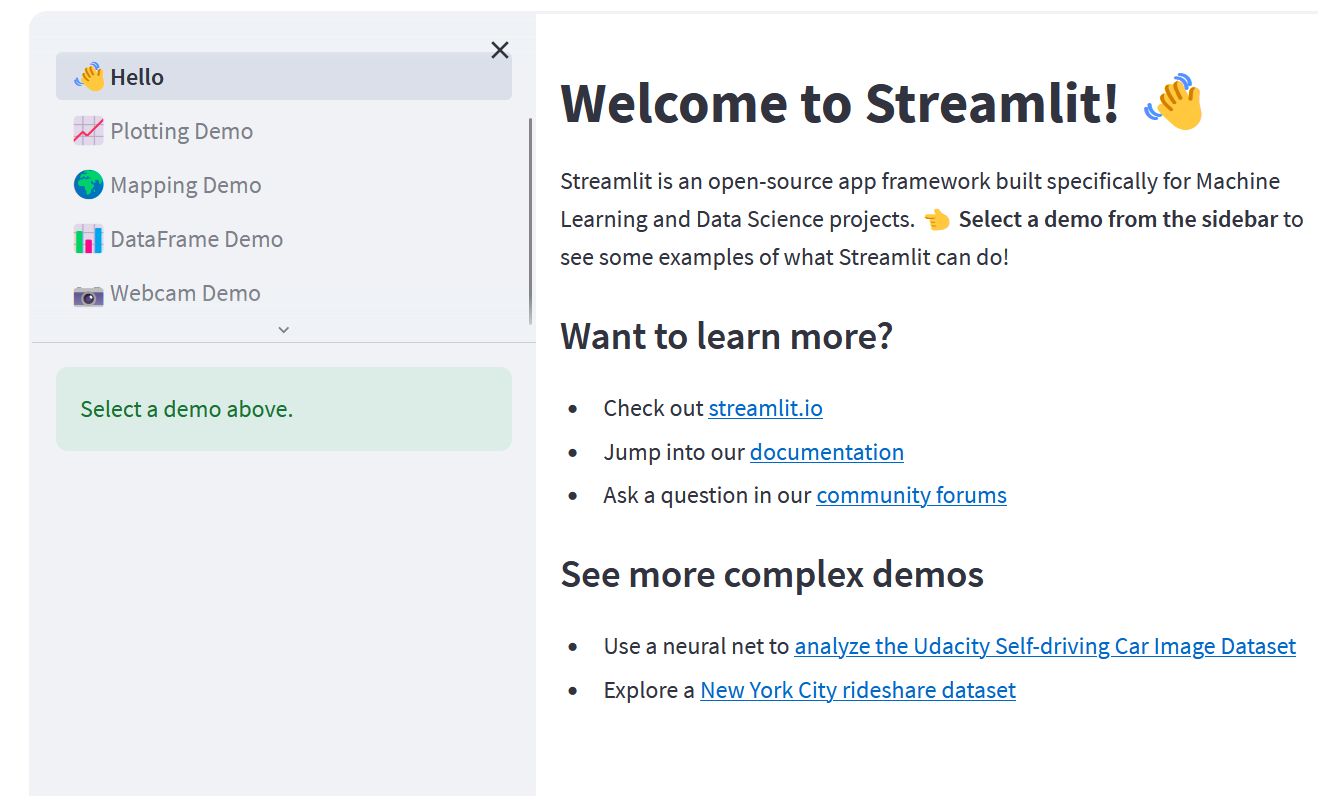
1. To Update your pip

python.exe -m pip install --upgrade pip

1. Test that the installation worked:

streamlit hello

Streamlit's Hello app should appear in a new tab in your web browser!



1. Create a requirement.txt file by typing the following command in the terminal

pip freeze > requirements.txt

1. Create a new python file app.py in the same directory.

**Use your new environment**

Any time you want to use the new environment, you first need to go to your project folder (where the .venv directory lives) and run:

source .venv/bin/activate

Now you can use Python and Streamlit as usual:

streamlit run app.py

Another way of running Streamlit is to run it as a Python module. This can be useful when configuring an IDE like PyCharm to work with Streamlit:

# Running

python -m streamlit run app.py

# is equivalent to:

streamlit run app.py

You can also pass a URL to streamlit run! This is great when combined with GitHub Gists. For example:

streamlit run https://raw.githubusercontent.com/streamlit/demo-uber-nyc-pickups/master/streamlit\_app.py

For further information on installation navigate to <https://docs.streamlit.io/library/get-started/installation>

**Create a User Interface for Your Regression Model**

**Prepare Your Data and Model**:

Before building the user interface, make sure you have your data ready and have already trained a regression model using tools like Scikit-Learn or any other machine learning library. Your model should be saved in a format that can be loaded and used in your Streamlit app.

**Create a Streamlit App Script**:

Create a new Python script (e.g., app.py) where you will build your Streamlit app. This script will contain the code for your user interface.

You can start with a basic structure like this:

1. **Import Required Libraries**: Import the necessary libraries at the beginning of your Streamlit script, including Streamlit itself and any other libraries you plan to use for data manipulation, visualization, and model

# app.py

# Import important libraries

import streamlit as st

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import pickle

from datetime import datetime

1. **Setup the App Title and Description**: Use Streamlit functions to set up the title and a brief description of your app:

st.title("Store Sales Forecasting ")

st.write("Welcome to the Store Sales Forecasting Analysis app.")

1. **Create Input Widgets**: Design input widgets that allow users to provide the necessary information for sales forecasting. For example, you might create input fields for the date, store ID, and any other relevant features:

# Date input

input\_date = st.date\_input("Select a date", value=pd.to\_datetime('today'))

# Store ID input

input\_store\_id = st.number\_input("Enter Store ID", min\_value=1, max\_value=1000)

1. **Load the Trained Regression Model**: If you have saved your regression model to a file (e.g., 'your\_model\_file.pkl'), load it using joblib (or another library you used for model serialization) as mentioned in a previous response.

# Load the regression model

loaded\_model = joblib.load('your\_model\_file.pkl')

Using the ‘**pickle’** library to load a trained regression model

# Load the regression model using pickle

with open('your\_model\_file.pkl', 'rb') as model\_file:

loaded\_model = pickle.load(model\_file)

1. **Process User Inputs and Make Predictions:** Create functions for data preprocessing and model prediction. In this example, we assume the existence of functions ‘**process\_input’** and **‘make\_prediction’**.

# Function for data preprocessing

def process\_input(date, store\_id):

# Perform any necessary data preprocessing

# This is just a placeholder, replace with your actual preprocessing logic

processed\_data = pd.DataFrame({'Date': [date], 'Store\_ID': [store\_id]})

return processed\_data

# Function for making predictions

def make\_prediction(processed\_data, model):

# Use the loaded model to make predictions

prediction = model.predict(processed\_data)

return prediction

# Prepare input data for prediction (assuming a DataFrame structure)

input\_data = pd.DataFrame({'Date': [input\_date], 'Store\_ID': [input\_store\_id]})

# Make predictions

predicted\_sales = loaded\_model.predict(input\_data)

1. **Display the Prediction Button and Result:** Display the prediction button and handle user interactions. When the user clicks the "Predict" button, process the user inputs and make predictions. Display the results using Streamlit components.

# When the user clicks a button

if st.button("Predict"):

# Process the user inputs

processed\_data = process\_input(input\_date, input\_store\_id)

# Make predictions

prediction = make\_prediction(processed\_data, loaded\_model)

# Display the prediction

st.subheader("Sales Forecast:")

st.write(f"The predicted sales for Store ID {input\_store\_id} on {input\_date} are: {prediction[0]:.2f}")

1. **Add Visualizations (Optional)**: If appropriate, you can include data visualizations or charts to illustrate the analysis results:

# Example: Display a bar chart of predicted sales by store

st.subheader("Predicted Sales by Store:")

st.bar\_chart(predicted\_sales\_by\_store)

1. **Customize the Layout and Style** (Optional): You can use Streamlit's layout and styling options to customize the appearance of your app, including fonts, colors, and layout structure. This can enhance the user experience:

# Example: Set the page width and background color

st.set\_page\_config(layout="wide")

st.markdown(

"""

<style>

.stApp {

background-color: #f3f3f3;

}

</style>

""",

unsafe\_allow\_html=True,

)

1. **Provide User Guidance**: Add explanatory text, instructions, or tooltips to guide users on how to use the app effectively.
2. **Testing and Debugging**: Thoroughly test your app by running it locally and interacting with the interface. Check for any errors or unexpected behavior and debug as needed to ensure a smooth user experience.
3. **Deployment** (Optional): If you want to share your app with others, you can deploy it on a web server or cloud platform like Streamlit Sharing, Heroku, or AWS. Follow the deployment instructions for the platform of your choice.
4. **Documentation**: Consider providing documentation or user guides to explain how to use your app, its features, and its limitations.

**Conclusion**

This documentation can guide you to successfully create a Store Sales Forecasting Analysis app using Streamlit. Users can input data, and the app predicts sales based on your trained regression model. You can further enhance the app by adding more features and improving the user interface.