Step 4: Install Required Python Libraries bash Copy Edit pip install pandas numpy scikit-learn matplotlib seaborn plotly streamlit joblib requests (You can also save this in a requirements.txt file.) GIVE STEP BY STEP GUIDE HOW TO DO ABOVE

**ChatGPT said:**

Here is a **step-by-step guide** on how to install the required Python libraries for your Post-Harvest Loss Predictor project.

**✅ Before You Begin**

Make sure you have completed:

1. **Python installed** (e.g., Python 3.13.5)
2. **Virtual environment created and activated**

If not yet done, let me know and I can walk you through those steps too.

**🔹 Step-by-Step: Install Required Python Libraries**

**📁 Step 1: Open Command Prompt**

1. Press Win + R, type cmd, and hit Enter.  
   → This opens the Command Prompt.

**🏃‍♀️ Step 2: Navigate to Your Project Folder**

In the Command Prompt, go to your project folder (e.g., post\_harvest\_predictor) like this:

bash

CopyEdit

cd C:\Users\YourName\Documents\post\_harvest\_predictor

Replace the path with your actual folder location.

**🌍 Step 3: Activate the Virtual Environment**

Activate the virtual environment you created earlier (venv):

bash

CopyEdit

venv\Scripts\activate

After activation, the prompt will change to show (venv) like this:

bash

CopyEdit

(venv) C:\Users\YourName\Documents\post\_harvest\_predictor>

**📦 Step 4: Install Required Libraries**

Now, run the following command to install all necessary libraries:

bash

CopyEdit

pip install pandas numpy scikit-learn matplotlib seaborn plotly streamlit joblib requests

✅ This will install the following libraries:

* pandas, numpy – for data handling
* scikit-learn – for machine learning
* matplotlib, seaborn, plotly – for data visualization
* streamlit – to create the web app
* joblib – to save ML models
* requests – to work with APIs (optional for real-time data)

**💾 Optional: Save Dependencies in a Requirements File**

You can save these installed libraries to a file for easy setup later:

bash

CopyEdit

pip freeze > requirements.txt

This creates a file called requirements.txt in your project folder.

**📥 Later, You (or Others) Can Install Using That File**

If someone else wants to set up the same environment:

bash

CopyEdit

pip install -r requirements.txt

**✅ Done!**

Now your environment is ready to run your Streamlit app and ML scripts.

Would you like me to prepare a ready-made requirements.txt for you?

**You said:**

if the above created in the desktop folder give tha answer

**ChatGPT said:**

Perfect! If your project is created inside a **folder on your Desktop**, here’s how you can follow **Step 4: Install Required Python Libraries**—with every step adjusted for that case.

**✅ Assume This Project Structure:**

You have a folder named **post\_harvest\_predictor** on your **Desktop**.

**🔹 Step-by-Step Guide (for Windows Desktop)**

**📁 Step 1: Open Command Prompt**

* Press Win + R, type cmd, and press **Enter**.

**📂 Step 2: Go to the Project Folder on Desktop**

Type this in the Command Prompt:

bash

CopyEdit

cd %USERPROFILE%\Desktop\post\_harvest\_predictor

✅ This command takes you directly to your project folder on the Desktop.

**🔄 Step 3: Activate the Virtual Environment**

If you previously created the virtual environment named venv, activate it like this:

bash

CopyEdit

venv\Scripts\activate

✅ Your prompt will change to show (venv) in front of the line, like:

scss

CopyEdit

(venv) C:\Users\YourName\Desktop\post\_harvest\_predictor>

**📦 Step 4: Install All Required Python Libraries**

Copy and paste this command to install all required packages:

bash

CopyEdit

pip install pandas numpy scikit-learn matplotlib seaborn plotly streamlit joblib requests

🟢 This will automatically download and install:

| **Library** | **Use** |
| --- | --- |
| pandas | Data handling |
| numpy | Numerical operations |
| scikit-learn | Machine learning |
| matplotlib | Basic charts and graphs |
| seaborn | Advanced statistical plots |
| plotly | Interactive charts |
| streamlit | Web dashboard |
| joblib | Save/load ML models |
| requests | For calling APIs (e.g., weather data) |

**💾 Optional Step: Save the List of Installed Packages**

If you want to save this setup for future use:

bash

CopyEdit

pip freeze > requirements.txt

📝 This creates a requirements.txt file inside your post\_harvest\_predictor folder on Desktop.

**🛠️ Optional: Reinstall Later Using That File**

Later (or on another machine), use:

bash

CopyEdit

pip install -r requirements.txt

**✅ You’re Done!**

Now your environment is fully set up.  
Would you like me to generate a ready-made requirements.txt file for your project?

**You said:**

Step 5: Prepare Your Dataset You can either: Create a simulated dataset using Python, or Use open datasets such as: FAO: https://www.fao.org/statistics/en/ Kaggle: https://www.kaggle.com/ (search for “fruit spoilage” or “supply chain”) Create a sample CSV like: csv Copy Edit temperature,humidity,transport\_time,distance,shelf\_life\_remaining 25,80,5,120,7 30,90,6,100,4 how to do the above step by step guidelines

**ChatGPT said:**

Here’s a **step-by-step guide** to help you complete **Step 5: Prepare Your Dataset** using either **manual creation** or **downloaded data** from online sources — all in a beginner-friendly way.

**✅ GOAL:**

Prepare a dataset in **CSV (Comma-Separated Values)** format for your spoilage prediction project. You have two options:

**🔹 Option 1: Manually Create a Sample CSV File**

Let’s start with a simple dataset like:

csv

CopyEdit

temperature,humidity,transport\_time,distance,shelf\_life\_remaining

25,80,5,120,7

30,90,6,100,4

28,85,4,150,5

**🪜 Step-by-Step Guide (Manual CSV Creation)**

**📁 Step 1: Open Notepad**

* Press Win + R, type notepad, and press Enter.

**✏️ Step 2: Paste the Sample Data**

Copy and paste this:

CopyEdit

temperature,humidity,transport\_time,distance,shelf\_life\_remaining

25,80,5,120,7

30,90,6,100,4

28,85,4,150,5

**💾 Step 3: Save as .csv File**

1. Click **File > Save As**
2. In the **File name** box, type:  
   spoilage\_data.csv
3. In **Save as type**, choose:  
   All Files (\*.\*)
4. In **Encoding**, select:  
   UTF-8
5. Choose your save location (e.g., Desktop or your post\_harvest\_predictor folder)
6. Click **Save**

✅ You now have a real dataset saved as spoilage\_data.csv

**✅ You Can Now Load This CSV in Python Using Pandas:**

python

CopyEdit

import pandas as pd

df = pd.read\_csv('spoilage\_data.csv')

print(df.head())

**🔹 Option 2: Download Open Datasets (Optional Advanced)**

If you prefer using real-world data:

**🌐 Step 1: Visit a Dataset Website**

1. Go to [**Kaggle**](https://www.kaggle.com/)  
   Search for terms like:
   * fruit spoilage
   * food shelf life
   * agriculture supply chain
2. Or go to [**FAO Stats**](https://www.fao.org/statistics/en/) for global agriculture data.

**📥 Step 2: Download a Dataset**

* On Kaggle: You’ll need to sign in, click "Download", and save the .csv file.

**📁 Step 3: Move It to Your Project Folder**

Move your downloaded .csv into the folder:

makefile

CopyEdit

C:\Users\YourName\Desktop\post\_harvest\_predictor

**📄 Step 4: Read It in Python**

Use:

python

CopyEdit

import pandas as pd

df = pd.read\_csv('your\_downloaded\_file.csv')

print(df.head())

**🔚 Final Tip: Add More Rows for Testing**

You can add more entries in Excel or Notepad as:

python-repl

CopyEdit

temperature,humidity,transport\_time,distance,shelf\_life\_remaining

26,82,6,130,6

29,88,7,110,4

...

Would you like me to generate a full 50-row realistic dataset for testing?

**You said:**

You Can Now Load This CSV in Python Using Pandas: python Copy Edit import pandas as pd df = pd.read\_csv('spoilage\_data.csv') print(df.head()) how to do the above step by step guide

**ChatGPT said:**

Here’s a complete **step-by-step guide** on how to load your spoilage\_data.csv file using **Python and Pandas** — perfect for your **Post-Harvest Loss Predictor** project.

**✅ GOAL:**

Read a CSV file (spoilage\_data.csv) into Python using the pandas library and display the data.

**🪜 Step-by-Step Instructions**

**📌 Prerequisites:**

Make sure you have:

* Created the spoilage\_data.csv file (from [Step 5](https://chat.openai.com/chat/)),
* Installed the pandas library,
* Activated your virtual environment.

If not done, let me know, I’ll guide you from scratch.

**🖥️ Step 1: Open Command Prompt or VS Code**

**Option 1: Command Prompt**

* Press Win + R, type cmd, and hit Enter.

**Option 2: VS Code**

* Open VS Code → Go to **File > Open Folder** → Select your post\_harvest\_predictor folder.
* Open a new Python file like test\_csv.py

**📂 Step 2: Navigate to Your Project Folder**

If using Command Prompt, type:

bash

CopyEdit

cd %USERPROFILE%\Desktop\post\_harvest\_predictor

**🔄 Step 3: Activate Virtual Environment**

bash

CopyEdit

venv\Scripts\activate

You should see:

scss

CopyEdit

(venv) C:\Users\YourName\Desktop\post\_harvest\_predictor>

**🐍 Step 4: Create a Python File**

If you're using Notepad:

1. Open Notepad
2. Paste the following code:

python

CopyEdit

import pandas as pd

# Load CSV file

df = pd.read\_csv('spoilage\_data.csv')

# Print first 5 rows

print(df.head())

1. Save the file as:  
   read\_csv\_test.py  
   (Save as type: All Files, Encoding: UTF-8)

**▶️ Step 5: Run the Python Script**

In your Command Prompt (in project folder):

bash

CopyEdit

python read\_csv\_test.py

**✅ Output Example:**

You should see something like:

nginx

CopyEdit

temperature humidity transport\_time distance shelf\_life\_remaining

0 25 80 5 120 7

1 30 90 6 100 4

2 28 85 4 150 5

**🧠 What This Does:**

* pandas.read\_csv() reads the dataset.
* df.head() shows the first 5 rows of your dataset.

Would you like me to help you create a read\_csv\_test.py file automatically, or show how to run this inside Streamlit?

**You said:**

Step 6: Build the Machine Learning Model (Random Forest) Create a file model\_train.py: python Copy Edit import pandas as pd from sklearn.ensemble import RandomForestRegressor from sklearn.model\_selection import train\_test\_split from joblib import dump # Load your dataset data = pd.read\_csv('data.csv') # change to your actual file X = data[['temperature', 'humidity', 'transport\_time', 'distance']] y = data['shelf\_life\_remaining'] X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2) model = RandomForestRegressor() model.fit(X\_train, y\_train) # Save the trained model dump(model, 'shelf\_life\_model.joblib') Then run: bash Copy Edit python model\_train.py give the step by step guide

**ChatGPT said:**

Absolutely! Here's a **step-by-step guide** to help you **build and run the Machine Learning model using Random Forest** in your post\_harvest\_predictor project folder.

**✅ GOAL:**

Train a **Random Forest model** to predict shelf life and save it to a file using joblib.

**🔧 Prerequisites (Ensure These Are Done):**

1. Python installed ✅
2. Virtual environment activated ✅
3. Libraries installed: pandas, scikit-learn, joblib ✅
4. CSV file (e.g., spoilage\_data.csv) is ready and in the project folder ✅

**🪜 Step-by-Step Instructions**

**📂 Step 1: Open Project Folder**

Make sure you are in your folder:

plaintext

CopyEdit

C:\Users\YourName\Desktop\post\_harvest\_predictor

**📝 Step 2: Create a Python File Named model\_train.py**

**Option A: Using Notepad**

1. Open Notepad (Win + R → type notepad → press Enter)
2. Copy and paste the following code:

python

CopyEdit

import pandas as pd

from sklearn.ensemble import RandomForestRegressor

from sklearn.model\_selection import train\_test\_split

from joblib import dump

# Load your dataset

data = pd.read\_csv('spoilage\_data.csv') # Use your actual filename

# Split features and target

X = data[['temperature', 'humidity', 'transport\_time', 'distance']]

y = data['shelf\_life\_remaining']

# Split into training and testing

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2)

# Create and train the model

model = RandomForestRegressor()

model.fit(X\_train, y\_train)

# Save the trained model to a file

dump(model, 'shelf\_life\_model.joblib')

print("✅ Model trained and saved as shelf\_life\_model.joblib")

1. Click **File > Save As**
2. Save the file as:  
   model\_train.py
   * Save as type: **All Files (\*.\*)**
   * Encoding: **UTF-8**
   * Location: post\_harvest\_predictor folder

**💡 Step 3: Activate Virtual Environment**

Open **Command Prompt**, and run:

bash

CopyEdit

cd %USERPROFILE%\Desktop\post\_harvest\_predictor

venv\Scripts\activate

You should see:

bash

CopyEdit

(venv) C:\Users\YourName\Desktop\post\_harvest\_predictor>

**▶️ Step 4: Run the Training Script**

Type this in the terminal:

bash

CopyEdit

python model\_train.py

**✅ Expected Output:**

If everything works, you'll see:

csharp

CopyEdit

✅ Model trained and saved as shelf\_life\_model.joblib

And a file named shelf\_life\_model.joblib will appear in your project folder — this is your **trained ML model**.

**🔚 Summary:**

| **Task** | **Description** |
| --- | --- |
| model\_train.py | Trains the Random Forest model |
| spoilage\_data.csv | Your dataset file |
| shelf\_life\_model.joblib | Saved model (used later in Streamlit dashboard) |

Would you like the next step to use this model in your **Streamlit dashboard**?

**You said:**

Step 8: Optional - Integrate APIs OpenWeatherMap (to fetch real-time weather) Sign up: https://openweathermap.org/api Use requests to call: python Copy Edit import requests API\_KEY = 'your\_api\_key' city = 'Colombo' url = f"http://api.openweathermap.org/data/2.5/weather?q={city}&appid={API\_KEY}&units=metric" res = requests.get(url).json() temp = res['main']['temp'] humidity = res['main']['humidity'] give step by step guide

**ChatGPT said:**

Here's a **complete step-by-step guide** to help you **integrate OpenWeatherMap API** into your Python project to fetch real-time **temperature** and **humidity** — useful for your Post-Harvest Loss Predictor.

**✅ GOAL:**

Use the requests library in Python to fetch live weather data (temperature and humidity) for a city like **Colombo**.

**🪜 Step-by-Step Guide**

**🌐 Step 1: Sign Up for OpenWeatherMap**

1. Visit: <https://openweathermap.org/api>
2. Click **Sign Up** (top right) and create an account.
3. Once logged in, go to **My API keys** → Copy your **default key** (e.g., b1a2c3d4e5f6...).

**📁 Step 2: Open Your Project Folder**

Navigate to your project folder:

bash

CopyEdit

cd %USERPROFILE%\Desktop\post\_harvest\_predictor

**📝 Step 3: Create a Python File to Fetch Weather Data**

You can name it get\_weather.py.

**Option A: Using Notepad**

1. Open Notepad
2. Paste the code below:

python

CopyEdit

import requests

# Replace with your real API key from OpenWeatherMap

API\_KEY = 'your\_api\_key' # Example: 'b1a2c3d4e5f6g7h8i9'

city = 'Colombo'

# Build the API request URL

url = f"http://api.openweathermap.org/data/2.5/weather?q={city}&appid={API\_KEY}&units=metric"

# Send request and parse JSON

res = requests.get(url).json()

# Extract temperature and humidity

temp = res['main']['temp']

humidity = res['main']['humidity']

# Display the results

print(f"✅ Current Temperature in {city}: {temp}°C")

print(f"✅ Current Humidity in {city}: {humidity}%")

1. Save as:  
   get\_weather.py
   * File type: **All Files**
   * Encoding: **UTF-8**
   * Location: in your project folder

**🔐 Step 4: Replace 'your\_api\_key'**

* In the code, replace 'your\_api\_key' with your actual key from OpenWeatherMap.

Example:

python

CopyEdit

API\_KEY = 'b1a2c3d4e5f6g7h8i9j0k1l2m3n4'

**✅ Step 5: Run the Python Script**

Make sure your virtual environment is activated:

bash

CopyEdit

venv\Scripts\activate

Then run:

bash

CopyEdit

python get\_weather.py

**📤 Output Example:**

bash

CopyEdit

✅ Current Temperature in Colombo: 28.5°C

✅ Current Humidity in Colombo: 82%

**🧠 What This Does:**

| **Part** | **Meaning** |
| --- | --- |
| requests.get(url).json() | Sends the request and parses the response as JSON |
| res['main']['temp'] | Gets temperature (in Celsius, using units=metric) |
| res['main']['humidity'] | Gets current humidity (%) |

**⚠️ Troubleshooting**

* If you see "401 Unauthorized" → Check your API key.
* If you see "city not found" → Double-check spelling (e.g., Colombo).

Would you like to integrate this live weather into your **Streamlit dashboard** or combine it with your model to make real-time predictions?

**You said:**

Step 4: Replace 'your\_api\_key' In the code, replace 'your\_api\_key' with your actual key from OpenWeatherMap. Example: python Copy Edit API\_KEY = 'b1a2c3d4e5f6g7h8i9j0k1l2m3n4' how to do the above give step by step guide

**ChatGPT said:**

Great! Here's a **clear step-by-step guide** to help you replace 'your\_api\_key' in your Python script with your actual API key from OpenWeatherMap.

**✅ GOAL:**

Update the placeholder 'your\_api\_key' in your get\_weather.py file with your **real API key** so the script can connect to OpenWeatherMap and fetch real-time weather data.

**🪜 Step-by-Step Instructions**

**🌐 Step 1: Get Your API Key from OpenWeatherMap**

1. Visit: https://home.openweathermap.org/api\_keys
2. Log in to your account.
3. Copy your API key (e.g., something like b1a2c3d4e5f6g7h8i9j0k1l2m3n4)

**📝 Step 2: Open Your get\_weather.py File**

**If using Notepad:**

1. Go to your Desktop → open the folder post\_harvest\_predictor
2. Right-click get\_weather.py → Select **Edit with Notepad**

**✏️ Step 3: Find the Line with 'your\_api\_key'**

You’ll see something like this:

python

CopyEdit

API\_KEY = 'your\_api\_key'

**🧪 Step 4: Replace with Your Actual API Key**

For example, change this:

python

CopyEdit

API\_KEY = 'your\_api\_key'

To this:

python

CopyEdit

API\_KEY = 'b1a2c3d4e5f6g7h8i9j0k1l2m3n4'

🔐 Make sure to **keep the single quotes ' '** around your API key. Don't add spaces or remove quotation marks.

**💾 Step 5: Save the File**

* In Notepad → Click **File > Save**

**✅ Step 6: Run the Script Again**

Go back to **Command Prompt**, activate your virtual environment (if not already active):

bash

CopyEdit

cd %USERPROFILE%\Desktop\post\_harvest\_predictor

venv\Scripts\activate

Run your script:

bash

CopyEdit

python get\_weather.py

**🎉 Expected Output:**

bash

CopyEdit

✅ Current Temperature in Colombo: 28.6°C

✅ Current Humidity in Colombo: 79%

**🛡️ Optional Tip: Hide Your API Key (for Safety)**

If you plan to share your code or use GitHub, it's best **not to hardcode** your API key. Use environment variables instead.

Let me know if you want a **secure version using .env files**.

Would you like to connect this real-time weather to your **Streamlit app prediction model**?

**You said:**

Step 9: Export Prediction Results You can allow exporting predictions as CSV: Add this to app.py: python Copy Edit import pandas as pd if st.button("Download Report"): report\_df = pd.DataFrame([{ "Temperature": temperature, "Humidity": humidity, "Transport Time": transport\_time, "Distance": distance, "Predicted Shelf Life": prediction }]) csv = report\_df.to\_csv(index=False).encode('utf-8') st.download\_button("Download as CSV", csv, "shelf\_life\_report.csv", "text/csv") give step by step guide to do this

**ChatGPT said:**

Here's a **complete step-by-step guide** to help you implement **Step 9: Export Prediction Results as CSV** in your app.py using **Streamlit**. This will allow users to **download a prediction report** after making a prediction.

**✅ GOAL:**

Add a “Download as CSV” button to your app.py that exports user inputs + predicted shelf life.

**🪜 Step-by-Step Instructions**

**📁 Step 1: Locate and Open Your app.py File**

If you don’t have one yet, create a new Python file:

1. Go to your post\_harvest\_predictor folder
2. Right-click → **New > Text Document** → Rename to app.py
3. Open it in **Notepad** or **VS Code**

**📋 Step 2: Paste/Modify Your Existing Streamlit Code**

If you already have prediction logic like this in app.py:

python

CopyEdit

import streamlit as st

import pandas as pd

import joblib

# Load model

model = joblib.load("shelf\_life\_model.joblib")

st.title("Shelf Life Predictor")

temperature = st.number\_input("Temperature (°C)")

humidity = st.number\_input("Humidity (%)")

transport\_time = st.number\_input("Transport Time (days)")

distance = st.number\_input("Distance (km)")

if st.button("Predict"):

input\_data = [[temperature, humidity, transport\_time, distance]]

prediction = model.predict(input\_data)[0]

st.success(f"Predicted Shelf Life: {prediction:.2f} days")

**✏️ Step 3: Add the CSV Export Code**

**Directly below the prediction code**, add this:

python

CopyEdit

# Step 9: Export prediction as CSV

if st.button("Download Report"):

report\_df = pd.DataFrame([{

"Temperature": temperature,

"Humidity": humidity,

"Transport Time": transport\_time,

"Distance": distance,

"Predicted Shelf Life": prediction

}])

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

st.download\_button("Download as CSV", csv, "shelf\_life\_report.csv", "text/csv")

✅ The full updated prediction section now looks like this:

python

CopyEdit

if st.button("Predict"):

input\_data = [[temperature, humidity, transport\_time, distance]]

prediction = model.predict(input\_data)[0]

st.success(f"Predicted Shelf Life: {prediction:.2f} days")

# Step 9: Export prediction as CSV

if st.button("Download Report"):

report\_df = pd.DataFrame([{

"Temperature": temperature,

"Humidity": humidity,

"Transport Time": transport\_time,

"Distance": distance,

"Predicted Shelf Life": prediction

}])

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

st.download\_button("Download as CSV", csv, "shelf\_life\_report.csv", "text/csv")

**💾 Step 4: Save the File**

Click **File > Save** in Notepad or VS Code.

**▶️ Step 5: Run the Streamlit App**

In your Command Prompt:

bash

CopyEdit

cd %USERPROFILE%\Desktop\post\_harvest\_predictor

venv\Scripts\activate

streamlit run app.py

**🧪 Step 6: Use the App**

1. Enter values for **Temperature, Humidity, Transport Time, Distance**
2. Click **Predict**
3. Click **Download Report**
4. A **CSV file will be downloaded** with your inputs and the predicted shelf life.

**✅ Output Example in shelf\_life\_report.csv:**

css

CopyEdit

Temperature,Humidity,Transport Time,Distance,Predicted Shelf Life

28,85,4,150,5.32

Would you like me to help combine this with live weather data from OpenWeatherMap next?

**You said:**

Step 7: Create the Streamlit Dashboard Create a new file called app.py: python Copy Edit import streamlit as st import numpy as np import pandas as pd from joblib import load # Load the model model = load('shelf\_life\_model.joblib') st.title("Post-Harvest Shelf Life Predictor") st.write("Enter current environmental and transport conditions:") temperature = st.slider("Temperature (°C)", 10, 40, 25) humidity = st.slider("Humidity (%)", 30, 100, 70) transport\_time = st.number\_input("Transport Time (hours)", min\_value=0.0, step=0.5) distance = st.number\_input("Distance (km)", min\_value=0.0, step=1.0) if st.button("Predict Shelf Life"): input\_data = np.array([[temperature, humidity, transport\_time, distance]]) prediction = model.predict(input\_data)[0] st.success(f"Estimated Shelf Life Remaining: {prediction:.2f} days") Run the dashboard: bash Copy Edit streamlit run app.py give step by step guide

**ChatGPT said:**

Here’s a **complete step-by-step guide** to help you **create and run a Streamlit dashboard** for your Post-Harvest Shelf Life Predictor using the code you shared.

**✅ GOAL:**

Create a Streamlit dashboard (app.py) that takes user input, runs your trained model (shelf\_life\_model.joblib), and shows the **predicted shelf life**.

**🪜 Step-by-Step Instructions**

**✅ Prerequisites**

Make sure you’ve already:

* Trained and saved the model as shelf\_life\_model.joblib
* Installed Streamlit and required packages
* Activated your virtual environment

If not done, let me know and I’ll guide you.

**🎯 Step 1: Open Your Project Folder**

Navigate to your project folder:

bash

CopyEdit

cd %USERPROFILE%\Desktop\post\_harvest\_predictor

**📝 Step 2: Create app.py Using Notepad or VS Code**

**🖊️ Option A: Using Notepad**

1. Press Win + R, type notepad, and hit Enter.
2. Paste this code:

python

CopyEdit

import streamlit as st

import numpy as np

import pandas as pd

from joblib import load

# Load the model

model = load('shelf\_life\_model.joblib')

st.title("🧪 Post-Harvest Shelf Life Predictor")

st.write("Enter current environmental and transport conditions:")

temperature = st.slider("🌡️ Temperature (°C)", 10, 40, 25)

humidity = st.slider("💧 Humidity (%)", 30, 100, 70)

transport\_time = st.number\_input("🚚 Transport Time (hours)", min\_value=0.0, step=0.5)

distance = st.number\_input("🛣️ Distance (km)", min\_value=0.0, step=1.0)

if st.button("Predict Shelf Life"):

input\_data = np.array([[temperature, humidity, transport\_time, distance]])

prediction = model.predict(input\_data)[0]

st.success(f"✅ Estimated Shelf Life Remaining: {prediction:.2f} days")

1. Save the file as:
   * File name: app.py
   * Save as type: **All Files**
   * Encoding: **UTF-8**
   * Location: inside your post\_harvest\_predictor folder

**🧪 Step 3: Activate Virtual Environment**

Open **Command Prompt**, and activate:

bash

CopyEdit

cd %USERPROFILE%\Desktop\post\_harvest\_predictor

venv\Scripts\activate

You should see:

bash

CopyEdit

(venv) C:\Users\YourName\Desktop\post\_harvest\_predictor>

**🚀 Step 4: Run the Streamlit App**

Type:

bash

CopyEdit

streamlit run app.py

✅ Streamlit will launch a browser window with your dashboard.

**🖥️ If it doesn't open automatically:**

Look for a line like:

bash

CopyEdit

Local URL: http://localhost:8501

Click the link or copy-paste it into your browser.

**🎉 Step 5: Use the Dashboard**

1. Adjust temperature, humidity, transport time, and distance
2. Click **Predict Shelf Life**
3. See your prediction in seconds!

**✅ You’ve Now Created:**

* A fully functional **interactive dashboard**
* With a **trained ML model**
* That estimates **remaining shelf life**

Would you like to now:

* Add **downloadable report** support?
* Integrate **live weather** API from OpenWeatherMap?

Let me know and I’ll guide you!

**You said:**

for the above code error generate like below "File "C:\Users\DELL\Desktop\post\_harvest\_predictor\venv\Lib\site-packages\streamlit\runtime\scriptrunner\exec\_code.py", line 128, in exec\_func\_with\_error\_handling result = func() File "C:\Users\DELL\Desktop\post\_harvest\_predictor\venv\Lib\site-packages\streamlit\runtime\scriptrunner\script\_runner.py", line 669, in code\_to\_exec exec(code, module.\_\_dict\_\_) # noqa: S102 ~~~~^^^^^^^^^^^^^^^^^^^^^^^ File "C:\Users\DELL\Desktop\post\_harvest\_predictor\app.py", line 29, in <module> "Predicted Shelf Life": prediction ^^^^^^^^^^ Copy Ask Google Ask ChatG"

**ChatGPT said:**

Thanks for sharing the error trace! The error you're seeing likely occurs because:

👉 **prediction is undefined** at the time when the Download Report button is clicked.

**🔍 Error Explanation:**

In your Streamlit app, when the user clicks **“Download Report”**, the app tries to use a variable called prediction — but prediction is only created **inside** the "Predict Shelf Life" button block.

So if you try to download before clicking "Predict", prediction doesn't exist yet → and Python throws an error.

**✅ Solution: Use a st.session\_state to store the prediction**

This way, the prediction stays available even outside the button block.

**🛠️ Fix: Updated app.py Code (with Export Button Working)**

python

CopyEdit

import streamlit as st

import numpy as np

import pandas as pd

from joblib import load

# Load the model

model = load('shelf\_life\_model.joblib')

st.title("🧪 Post-Harvest Shelf Life Predictor")

st.write("Enter current environmental and transport conditions:")

temperature = st.slider("🌡️ Temperature (°C)", 10, 40, 25)

humidity = st.slider("💧 Humidity (%)", 30, 100, 70)

transport\_time = st.number\_input("🚚 Transport Time (hours)", min\_value=0.0, step=0.5)

distance = st.number\_input("🛣️ Distance (km)", min\_value=0.0, step=1.0)

# Predict Button

if st.button("Predict Shelf Life"):

input\_data = np.array([[temperature, humidity, transport\_time, distance]])

prediction = model.predict(input\_data)[0]

st.success(f"✅ Estimated Shelf Life Remaining: {prediction:.2f} days")

# Store in session for download access

st.session\_state.prediction = prediction

st.session\_state.input\_data = {

"Temperature": temperature,

"Humidity": humidity,

"Transport Time": transport\_time,

"Distance": distance,

"Predicted Shelf Life": prediction

}

# Show Download Button only after prediction

if "prediction" in st.session\_state:

report\_df = pd.DataFrame([st.session\_state.input\_data])

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

st.download\_button("📥 Download Prediction Report as CSV", csv, "shelf\_life\_report.csv", "text/csv")

**✅ Now What’s Fixed:**

| **Feature** | **Status** |
| --- | --- |
| No crash on download click | ✅ Fixed |
| Prediction accessible anytime | ✅ Handled with st.session\_state |
| Clean download CSV button | ✅ Working |

Would you like me to help you add **live weather data integration** next, or embed this model into a more advanced dashboard layout (e.g., tabs, graphs, charts)?

**You said:**

Step 10: Final Documentation and Deliverables Include: Project Report – Background, problem, method, results. Presentation Slides – Overview, screenshots, conclusions. README.md – How to run your system. GitHub Repository (Optional but recommended): Create a repo at https://github.com/ Push your code there using Git. give step by step for the above

**ChatGPT said:**

Sure! Here’s a **step-by-step guide** for completing **Step 10: Final Documentation and Deliverables**, covering the **project report**, **presentation**, **README**, and **GitHub repository** setup.

**Step 10: Final Documentation and Deliverables**

**1. Project Report**

**What to include:**

* **Background:** Why the project matters (post-harvest losses, impact on Sri Lanka’s agriculture).
* **Problem Statement:** What problem you address (spoilage prediction, supply chain optimization).
* **Methodology:** How you solved it (data collection, model choice, Streamlit dashboard).
* **Results:** Model performance, dashboard features, sample predictions.
* **Conclusion & Future Work:** Summary and possible improvements.

**How to create:**

* Use **MS Word, Google Docs, or LaTeX**.
* Structure it clearly with headings.
* Add figures/screenshots from your Streamlit app or graphs.
* Export as **PDF** before submission.

**2. Presentation Slides**

**What to include:**

* Project overview
* Problem and objectives
* Methodology (model, data, API)
* Screenshots of your dashboard
* Key results & conclusions

**How to create:**

* Use **PowerPoint, Google Slides, or Canva**.
* Keep slides simple, visual, and not text-heavy.
* Save as **.pptx** or **PDF**.

**3. README.md File**

**Purpose:**

Help others understand how to run your project.

**What to include:**

markdown

CopyEdit

# Post-Harvest Shelf Life Predictor

## Description

A machine learning project to predict shelf life of fruits & vegetables to reduce post-harvest losses.

## Setup Instructions

1. Clone the repository

2. Create & activate a virtual environment

3. Install required libraries:

pip install -r requirements.txt

markdown

CopyEdit

4. Run the training script:

python model\_train.py

markdown

CopyEdit

5. Run the Streamlit dashboard:

streamlit run app.py

shell

CopyEdit

## Data

Use the included `spoilage\_data.csv` file or provide your own.

## API Integration

(Optional) Update `get\_weather.py` with your OpenWeatherMap API key to get live weather data.

## Author

Your Name

**4. GitHub Repository Setup (Optional but Recommended)**

**Step-by-step:**

**A. Create GitHub Repository**

1. Go to <https://github.com/>
2. Log in or sign up
3. Click **New repository**
4. Enter repo name (e.g., post-harvest-predictor)
5. Set visibility: Public or Private
6. Click **Create repository**

**B. Initialize Local Git and Push Code**

1. Open **Command Prompt** and navigate to your project folder:

bash

CopyEdit

cd %USERPROFILE%\Desktop\post\_harvest\_predictor

1. Initialize git (if not already):

bash

CopyEdit

git init

1. Add all files:

bash

CopyEdit

git add .

1. Commit changes:

bash

CopyEdit

git commit -m "Initial commit - post harvest loss predictor project"

1. Connect to remote repo (replace with your GitHub repo URL):

bash

CopyEdit

git remote add origin https://github.com/YourUsername/post-harvest-predictor.git

1. Push to GitHub:

bash

CopyEdit

git branch -M main

git push -u origin main

**Summary Table**

| **Deliverable** | **Tool/Platform** | **Steps to Complete** |
| --- | --- | --- |
| Project Report | Word/Google Docs/LaTeX | Write, add images, export as PDF |
| Presentation Slides | PowerPoint/Google Slides | Create slides, add screenshots, export |
| README.md | Text Editor/GitHub | Write instructions in markdown format |
| GitHub Repo | Git + GitHub | Create repo, push local code |

If you want, I can help you draft any of the documents or write the exact commands for Git in your project!

**You said:**

how to do the above in git bash software

**ChatGPT said:**

Got it! Here’s a **step-by-step guide** on how to do **Step 10** — especially **pushing your project to GitHub using Git Bash**.

**How to Push Your Project to GitHub Using Git Bash**

**🔹 Prerequisites**

* **Git Bash installed** (download from <https://git-scm.com/>)
* A **GitHub account** ready
* Your project folder (e.g., post\_harvest\_predictor) ready on your Desktop

**Step 1: Open Git Bash**

* On Windows, search **Git Bash** and open it.

**Step 2: Navigate to Your Project Folder**

Type this in Git Bash:

bash

CopyEdit

cd ~/Desktop/post\_harvest\_predictor

*Note:* ~ means your user home folder (like C:\Users\YourName).

**Step 3: Initialize Git Repository (if not done already)**

bash

CopyEdit

git init

This creates a hidden .git folder to track your files.

**Step 4: Add All Files to Staging Area**

bash

CopyEdit

git add .

This stages all files for commit.

**Step 5: Commit Your Changes**

bash

CopyEdit

git commit -m "Initial commit - Post Harvest Loss Predictor project"

**Step 6: Create a New Repository on GitHub**

1. Go to <https://github.com/>
2. Log in and click **New repository**
3. Name it (e.g., post-harvest-predictor)
4. Leave **README** unchecked (since you have local files)
5. Click **Create repository**

**Step 7: Link Local Repo to GitHub Remote**

GitHub will show you instructions, for example:

bash

CopyEdit

git remote add origin https://github.com/YourUsername/post-harvest-predictor.git

Paste this command into Git Bash and press Enter.

**Step 8: Rename Local Branch to main**

bash

CopyEdit

git branch -M main

**Step 9: Push Local Code to GitHub**

bash

CopyEdit

git push -u origin main

* It may ask your GitHub username and password (or token).
* Enter them to authenticate.

**Step 10: Confirm Your Code Is on GitHub**

* Refresh your GitHub repo page, and you should see all your files uploaded!

**Optional: Create a README.md**

1. In your project folder, create a file named README.md (you can use Notepad or VS Code).
2. Add your project description and instructions (see my previous message for an example).
3. Save it.
4. Add, commit, and push it via Git Bash:

bash

CopyEdit

git add README.md

git commit -m "Add README file"

git push

**Summary Commands for Git Bash:**

bash

CopyEdit

cd ~/Desktop/post\_harvest\_predictor

git init

git add .

git commit -m "Initial commit - Post Harvest Loss Predictor project"

git remote add origin https://github.com/YourUsername/post-harvest-predictor.git

git branch -M main

git push -u origin main

If you want, I can help you write a **README.md** or prepare your first **GitHub commit message**!

**You said:**

Step 3: Initialize Git Repository (if not done already) bash Copy Edit git init This creates a hidden .git folder to track your files. give step by step guite

**ChatGPT said:**

Sure! Here’s a **simple step-by-step guide** to initialize a Git repository in your project folder using **Git Bash**:

**Step 3: Initialize Git Repository (git init)**

**What it does:**

* git init creates a hidden folder called .git inside your project folder.
* This folder **tracks changes** you make to your project files.
* It turns your folder into a **Git repository**.

**Step-by-step instructions:**

**1. Open Git Bash**

* Click on the **Start menu** (Windows icon)
* Search for **Git Bash**
* Click to open it

**2. Navigate to Your Project Folder**

Type this command and press **Enter**:

bash

CopyEdit

cd ~/Desktop/post\_harvest\_predictor

* cd = “change directory” (go to folder)
* ~ means your home folder (e.g., C:\Users\YourName on Windows)

**3. Initialize Git Repository**

Now type:

bash

CopyEdit

git init

and press **Enter**.

**4. What Happens**

* You’ll see a message like:

bash

CopyEdit

Initialized empty Git repository in /c/Users/YourName/Desktop/post\_harvest\_predictor/.git/

* This means Git is now tracking your folder.

**5. Check that the .git folder exists (optional)**

To see hidden folders, you can run:

bash

CopyEdit

ls -a

You should see .git in the list — this means your Git repository is ready.

**What’s next?**

Now you can **add files** (git add .), **commit changes** (git commit -m "message"), and **push to GitHub** later.

Would you like me to guide you with the next step: **adding files to staging (git add)**?

**You said:**

yes

**ChatGPT said:**

Great! Let's move on to the **next step: adding files to Git's staging area**, which means telling Git **“these are the files I want to include in my next save (commit)”**.

**✅ Step 4: Add Files to Git (Staging)**

**📌 What this does:**

* Prepares your files to be **committed** (saved into Git history).
* You can add all files or select specific ones.

**🔧 Step-by-Step Using Git Bash:**

**1. Make sure you're still in your project folder:**

If you're not sure, type:

bash

CopyEdit

pwd

It should show something like:

bash

CopyEdit

/c/Users/YourName/Desktop/post\_harvest\_predictor

If not, go there again:

bash

CopyEdit

cd ~/Desktop/post\_harvest\_predictor

**2. See which files are unstaged:**

bash

CopyEdit

git status

You will see a list of files in red (these are untracked or changed).

**3. Add all files to staging area:**

bash

CopyEdit

git add .

* This adds **everything** in the folder to be tracked.
* You can also add individual files like:

bash

CopyEdit

git add app.py model\_train.py

**4. Confirm what’s staged:**

bash

CopyEdit

git status

* Now your files will appear in **green**.
* This means they’re staged and ready to be committed.

✅ Done! Your files are staged.

**🔜 Next Step: Commit your changes (Step 5)**

Would you like to continue and see how to **commit your code with a message**?

Top of Form

Tools

Bottom of Form

ChatGPT can make mistakes. Check important info