Step-by-Step Guide: Jenkins Pipeline for Python Flask App (Windows)

# 1. Project Folder Structure

Create the following structure in your GitHub repository:  
  
Pipelining\_pythonApp/  
├── model.py  
├── app.py  
├── data.csv  
├── requirements.txt  
├── Jenkinsfile  
└── templates/  
 └── index.html

# 2. Sample File Contents

# data.csv – Training Dataset

A sample CSV file with more data points to help the neural network learn the addition pattern better:  
  
Feature1,Feature2,Output  
0,0,0  
1,1,2  
2,2,4  
3,3,6  
4,4,8  
5,5,10  
6,6,12  
7,7,14  
8,8,16  
9,9,18  
10,10,20  
1,2,3  
2,3,5  
3,4,7  
4,5,9  
5,6,11  
6,7,13  
7,8,15  
8,9,17  
9,10,19  
10,5,15  
2,6,8  
3,7,10

# model.py – Neural Network Training

This file loads the dataset and trains a neural network model using Keras:

import pandas as pd  
import numpy as np  
from tensorflow.keras.models import Sequential  
from tensorflow.keras.layers import Dense  
  
def train\_model():  
 df = pd.read\_csv('data.csv')  
 X = df[['Feature1', 'Feature2']].values  
 y = df[['Output']].values  
  
 model = Sequential()  
 model.add(Dense(32, input\_dim=2, activation='linear'))  
 model.add(Dense(16, activation='relu'))  
 model.add(Dense(1))  
  
 model.compile(optimizer='adam', loss='mean\_squared\_error')  
 model.fit(X, y, epochs=1000, verbose=0)  
  
 return model

# app.py – Flask Web Server

This file sets up the web application, loads the model, accepts user input, and displays prediction:

from flask import Flask, request, render\_template  
from model import train\_model  
import numpy as np  
  
app = Flask(\_\_name\_\_)  
model = train\_model()  
  
@app.route('/', methods=['GET', 'POST'])  
def home():  
 result = None  
 if request.method == 'POST':  
 try:  
 num1 = float(request.form['num1'])  
 num2 = float(request.form['num2'])  
 input\_data = np.array([[num1, num2]])  
 prediction = model.predict(input\_data, verbose=0)  
 result = f"Predicted Sum: {prediction[0][0]:.2f}"  
 except:  
 result = "Invalid input. Please enter valid numbers."  
 return render\_template('index.html', result=result)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run(debug=True)

# templates/index.html – Web Form

This HTML page contains a form for user input and displays the prediction result:

<!DOCTYPE html>  
<html>  
<head>  
 <title>Neural Network Web App</title>  
</head>  
<body>  
 <h2>Enter two numbers to add:</h2>  
 <form method="POST">  
 Number 1: <input type="text" name="num1"><br><br>  
 Number 2: <input type="text" name="num2"><br><br>  
 <input type="submit" value="Predict">  
 </form>  
  
 {% if result %}  
 <h3>{{ result }}</h3>  
 {% endif %}  
</body>  
</html>

# Jenkinsfile (Windows-compatible)

pipeline {  
 agent any  
  
 environment {  
 VENV = "venv"  
 }  
  
 stages {  
 stage('Clone GitHub Repo') {  
 steps {  
 git branch: 'main', credentialsId: 'github-https', url: 'https://github.com/your-username/Pipelining\_pythonApp.git'  
 }  
 }  
  
 stage('Set Up Python Virtual Environment') {  
 steps {  
 bat '"C:\\Users\\your-username\\AppData\\Local\\Programs\\Python\\Python310\\python.exe" -m venv venv'  
 bat '.\\venv\\Scripts\\python.exe -m pip install --upgrade pip'  
 bat '.\\venv\\Scripts\\pip install -r requirements.txt'  
 }  
 }  
  
 stage('Run Flask App') {  
 steps {  
 bat '.\\venv\\Scripts\\python app.py'  
 }  
 }  
 }  
}

# 5. Jenkins Configuration (Windows)

1. Install Jenkins and Git on your Windows machine.  
2. Add Python to your system PATH or use full path as in Jenkinsfile.  
3. Open Jenkins → Manage Jenkins → Credentials → Add Credentials.

**🛠 Generate GitHub PAT:**

1. Go to <https://github.com> and log into your account.
2. Click on your profile picture → **Settings**.
3. Scroll down → **Developer settings** → **Personal access tokens** → **Tokens (classic)**.
4. Click **Generate new token (classic)**.
5. Give it a name (e.g., "Jenkins Access Token").
6. Set expiration: (30 or 90 days is common).
7. Check scopes (permissions):
   1. ✅ repo (Full control of private repositories)
   2. ✅ workflow (Optional)
8. Click **Generate token**.
9. Copy the token — **this is the only time GitHub will show it.**

- Kind: Username with password  
 - Username: your GitHub username  
 - Password: GitHub Personal Access Token (PAT)  
 - ID: github-https

4. Create a new pipeline job.  
5. Choose "Pipeline script from SCM", use Git and paste your repo URL.  
6. Ensure the branch is set to \*/main.  
7. Build the job and monitor Console Output.  
8. Wait for pip installations (TensorFlow may take 3–10 mins).  
9. Flask app will start with respective url in console output and you have to manually end the task when done.