Steps to Create and Run a Machine Learning Application in VS Code

This guide explains how to set up, develop, and run a machine learning application using Visual Studio Code (VS Code). It includes Windows and macOS specific commands, as well as explanations for why each step is necessary. Screenshot placeholders are provided for you to embed your own captures.

# Step 1. Install Python

Download and install the latest stable version of Python from https://www.python.org/downloads/. Python is the programming language used to build and run the machine learning application. Make sure to check the option to 'Add Python to PATH' during installation (on Windows).

[Insert Screenshot: Python.org download page (Windows & macOS)]

# Step 2. Install Visual Studio Code (VS Code)

Download VS Code from https://code.visualstudio.com/. VS Code is the Integrated Development Environment (IDE) where you will write, edit, and run your machine learning code.

[Insert Screenshot: VS Code download page]

# Step 3. Create a Project Folder

Create a folder on your computer to hold all project files. A typical structure is:  
 - data/ → raw or processed datasets  
 - preprocessing.py → script to clean and transform data  
 - model/ → folder for trained model files  
 - app.py → main Flask or FastAPI application file  
 - requirements.txt → list of dependencies  
 - test\_payload.json → sample input data for testing the API  
 - README.md → project description and usage notes

[Insert Screenshot: VS Code Explorer showing folder structure]

# Step 4. Open the Folder in VS Code

Launch VS Code, then go to File → Open Folder and select your project folder. This ensures VS Code can access all files and manage terminals relative to this folder.

# Step 5. Create a Virtual Environment

Open the integrated terminal in VS Code and create a virtual environment to isolate project dependencies. A virtual environment ensures that the libraries used for this project do not conflict with libraries used in other projects.

Windows (PowerShell): python -m venv venv\_income\_api

macOS/Linux (bash): python3 -m venv venv\_income\_api

[Insert Screenshot: VS Code terminal creating virtual environment (Windows & macOS)]

# Step 6. Activate the Virtual Environment

Windows (PowerShell): venv\_income\_api\Scripts\activate

macOS/Linux (bash): source venv\_income\_api/bin/activate

After activation, your terminal prompt should show (venv\_income\_api), meaning packages will be installed into this isolated environment.

[Insert Screenshot: Activated terminal prompt showing (venv\_income\_api)]

# Step 7. Install Required Packages

With the environment active, install dependencies from requirements.txt:  
pip install -r requirements.txt  
  
This ensures you have all the libraries needed to run the preprocessing, model training, and API application.

[Insert Screenshot: Terminal running pip install -r requirements.txt]

# Step 8. Start the Application

Run the application with:  
python app.py  
  
If using Flask, you should see output similar to:  
\* Running on http://127.0.0.1:5000  
  
This means the API server is running locally and ready to accept requests.

[Insert Screenshot: Flask app running message in terminal]

# Step 9. Test the API with curl

curl is a command-line tool used to send HTTP requests to servers. It allows you to test your API endpoints without building a frontend.  
  
Windows (PowerShell):  
curl.exe -X POST "http://127.0.0.1:5000/predict" ^  
 -H "Content-Type: application/json" ^  
 --data-binary "@test\_payload.json"  
  
macOS/Linux (bash):  
curl -X POST http://127.0.0.1:5000/predict \  
 -H "Content-Type: application/json" \  
 --data @test\_payload.json  
  
This sends the contents of test\_payload.json to the /predict endpoint and returns the model prediction.

[Insert Screenshot: curl command and JSON response in terminal (Windows & macOS)]