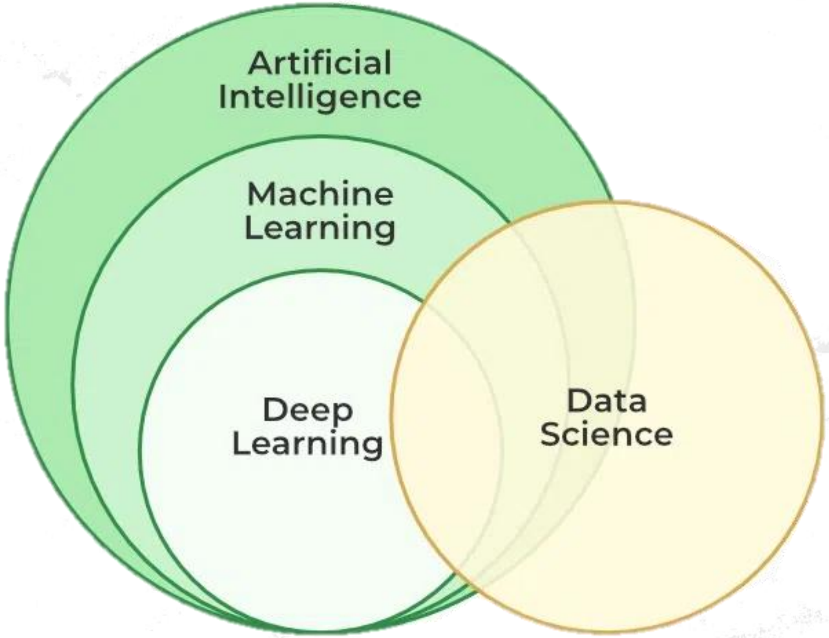
futureskills prime

A MeitY - NASSCOM Digital Skilling Initiative

Machine Learning and Deep Learning Concepts



NIELIT Chandigarh/Ropar

"Big Data is at the foundation of all the megatrends that are happening today."

Introduction

- **Topic**: Machine Learning & Deep Learning Overview
- **Objective**: Understand the key algorithms and concepts.
- **Topics Covered**:
  - ML Algorithms
  - Neural Networks & Gradient Descent
  - Deep Learning (ANN, CNN, RNN)
  - OpenAI & Case Studies

Machine Learning Algorithms (1/2)

1. Linear Regression

- Predicts continuous outcomes.
- Equation:
- Application: Price prediction.

2. Logistic Regression

- Predicts categorical outcomes (binary).
- Uses the sigmoid function for probabilities.
- Application: Fraud detection.

3. Decision Tree

- Tree-like model with nodes and leaves.
- Application: Credit scoring.

Machine Learning Algorithms (2/2)

4. K-Means Clustering

- Groups data into clusters.
- Application: Customer segmentation.

5. Association Rule Mining

- Finds relationships in datasets.
- Algorithm: Apriori.
- Application: Market basket analysis

Introduction to Neural Networks

Key Components:

- **Input Layer**: Features of the dataset.
- **Hidden Layers**: Perform computations.
- **Output Layer**: Predictions.
- **Core Concepts:**
- Weights and biases.
- Activation functions (e.g., ReLU, Sigmoid).

Gradient Descent

- **Purpose**: Minimize the loss function.
- **Types**:
  - Batch Gradient Descent.
  - Stochastic Gradient Descent (SGD).
  - Mini-Batch Gradient Descent.
- Application: Training neural networks.

Activation Functions

- **Sigmoid**: Values between 0 and 1.
- **ReLU**: Outputs if , else 0.
- **Tanh**: Values between -1 and 1.
- **Softmax**: Converts logits to probabilities.

Deep Learning Algorithms (1/2)

- **Artificial Neural Network (ANN)**
- Fully connected layers.
- Applications: Regression, classification.
- **Convolutional Neural Network (CNN)**
- Specialized for images.
- Components: Convolution layers, pooling, fully connected layers.
- Applications: Image classification, object detection.

Deep Learning Algorithms (2/2)

Recurrent Neural Network (RNN)

- Designed for sequential data.
- Incorporates feedback loops and memory.
- Variants: LSTM, GRU.
- Applications: Language modeling, speech recognition.

OpenAI Overview

- **Company**: AI research and deployment.
- **Key Products**:
  - GPT: Text generation.
  - Codex: AI for coding.
- **Applications**:
  - Creative writing, customer support.
  - Code suggestions.

Case Study: **Brain Tumor Prediction**

Objective:

- Detect and classify brain tumors using medical imaging.

Dataset:

- MRI images with labeled tumor.

Approach:

- Preprocessing: Image normalization and augmentation.
- Algorithms: CNN for feature extraction and classification.
- Evaluation Metrics: Accuracy, precision, recall, F1-score.

Outcome:

- High accuracy in tumor detection.
- Support for early diagnosis and treatment planning.

Conclusion

- Reviewed:
  - ML & DL algorithms.
  - Neural networks and optimization techniques.
  - OpenAI and its applications.
  - Real-world case study.
- Ready to apply these concepts in projects!
- Let’s Create a Live Project.

Capstone Project on Big Data & Data Science

Brain Tumor Detection Using Scalable Database MongoDB and CNN

NIELIT Chandigarh/Ropar

"Big Data is at the foundation of all the megatrends that are happening today."

Introduction

- **Topic**: Brain Tumor Detection Using CNN and MongoDB
- **Objective**: Understand how to implement a Flask-based application for detecting brain tumors using a pre-trained CNN model and MongoDB for scalable storage.
- **Topics Covered**:
  - Image preprocessing and CNN model prediction
  - Flask web framework for user interaction
  - MongoDB for storing predictions

Required Libraries

- **Python Libraries Used**
- **Flask**: Lightweight web framework for building applications.
- **OpenCV (cv2)**: Image processing library.
- **imutils**: Helper functions for image manipulation.
- **NumPy**: Array and mathematical operations.
- **TensorFlow/Keras**: Loading the pre-trained brain tumor detection model.
- **MongoDB**: Scalable NoSQL database for storing user inputs and predictions.
- **Werkzeug**: Securely handling file uploads.
- **Datetime**: Timestamp for predictions.

Pre-trained CNN Model

- **Model Details**
- **Architecture**: Convolutional Neural Network (CNN)
- **Purpose**: Predict the presence of a brain tumor from MRI images.
- **File**: braintumor.h5 (pre-trained model file)
- **Input Shape**: 128x128x3 (image size and color channels)
- **Features**
- High accuracy in tumor detection.
- Trained on diverse MRI datasets.

Application Workflow

- **Steps Involved**

1. **User Input**: Upload an MRI image and provide details.
2. **File Validation**: Ensure valid image formats (e.g., PNG, JPG).
3. **Image Preprocessing**: Resize and crop images for model input.
4. **Prediction**: Use CNN model to detect the presence of a tumor.
5. **Database Storage**: Save results and user details to MongoDB.
6. **Result Display**: Show prediction to the user.

Flask Application

- **Configuration**
- **Disable Caching**: Ensures updated images load.
- **Secret Key**: For session and message management.
- **Routes**:
  - /: Main upload page.
  - /resultbt: Predict brain tumor.
  - /dbresults: View all stored predictions.
- **Code Snippet**
- app = Flask(__name__)
- app.config['SEND_FILE_MAX_AGE_DEFAULT'] = 0
- app.secret_key = "your_secret_key"

MongoDB Integration

- **Connection Setup**
- **Database**: brain_tumor_detection
- **Collection**: predictions
- **Code Snippet**
- from pymongo import MongoClient
- client = MongoClient("mongodb+srv://<username>:<password>@cluster.mongodb.net")
- db = client['brain_tumor_detection']
- collection = db['predictions’]
- **Data Stored**
- User details (e.g., name, timestamp)
- Prediction results (e.g., tumor status, confidence score)

Image Preprocessing

- **Cropping Region of Interest (ROI)**
- **Objective**: Focus on the brain region for better accuracy.
- **Steps**:
  - Convert to grayscale.
  - Apply Gaussian blur.
  - Detect contours to extract ROI.
- **Resizing Image**
- **Purpose**: Match the input size required by the CNN model.
- **Method**: Use OpenCV's cv2.resize() function.
- **Code Snippet**

```
def preprocess_imgs(set_name, img_size):
set_new = []
for img in set_name:
img = cv2.resize(img, dsize=img_size, interpolation=cv2.INTER_CUBIC)
set_new.append(img)
return np.array(set_new)
```

Prediction Logic

- **Workflow**

1. Load the pre-trained CNN model.
2. Process the input image (crop and resize).
3. Predict the tumor presence using the model.
4. Classify result as "Tumor Detected" or "No Tumor Detected" based on confidence score.

- **Code Snippet**
- pred = braintumor_model.predict(img)
- prediction = 'Tumor Detected' if pred[0][0] >= 0.5 else 'No Tumor Detected'
- confidence_score = float(pred[0][0])

Storing Predictions in MongoDB

- **Data Format**
- **Firstname**: User-provided name.
- **Prediction**: Tumor status.
- **Confidence Score**: Probability value from the model.
- **Timestamp**: Date and time of prediction.
- result = {
- "firstname": firstname,
- "prediction": prediction,
- "confidence_score": confidence_score,
- "timestamp": datetime.utcnow()
- }
- collection.insert_one(result)

Viewing Results

- **Dashboard**
- Predict Brain Tumor by uploading MRI Image.
- Fetch all predictions stored in MongoDB.
- Display total patients and tumors detected.
- Sort results by timestamp.
- **Code Snippet**

```
all_results = collection.find().sort("timestamp", -1)

tumor_count = sum(1 for r in all_results if r['prediction'] == 'Tumor Detected')

total_patients = collection.count_documents({})
```

Summary

- Implemented a Flask application to predict brain tumors.
- Preprocessed images for accurate CNN predictions.
- Used MongoDB for scalable and efficient data storage.
- Designed routes for user interaction and admin review.

Live Project Deployment on Hugging Face – Brain Tumor Detection Using Scalable Database MongoDB and CNN

Hugging Face

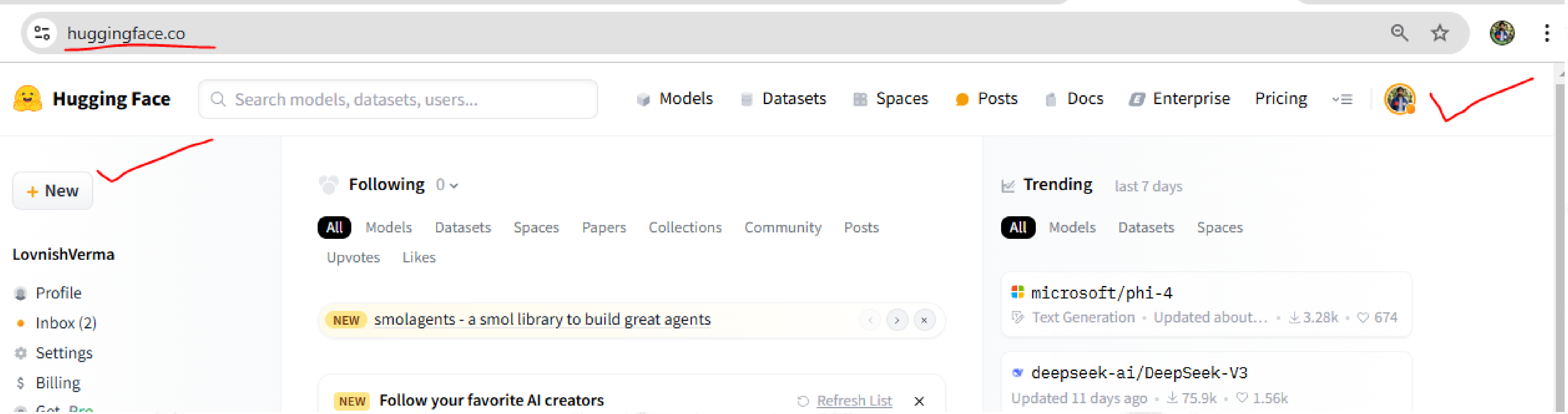
NIELIT Chandigarh/Ropar

"Big Data is at the foundation of all the megatrends that are happening today."

Deployment on Hugging Face

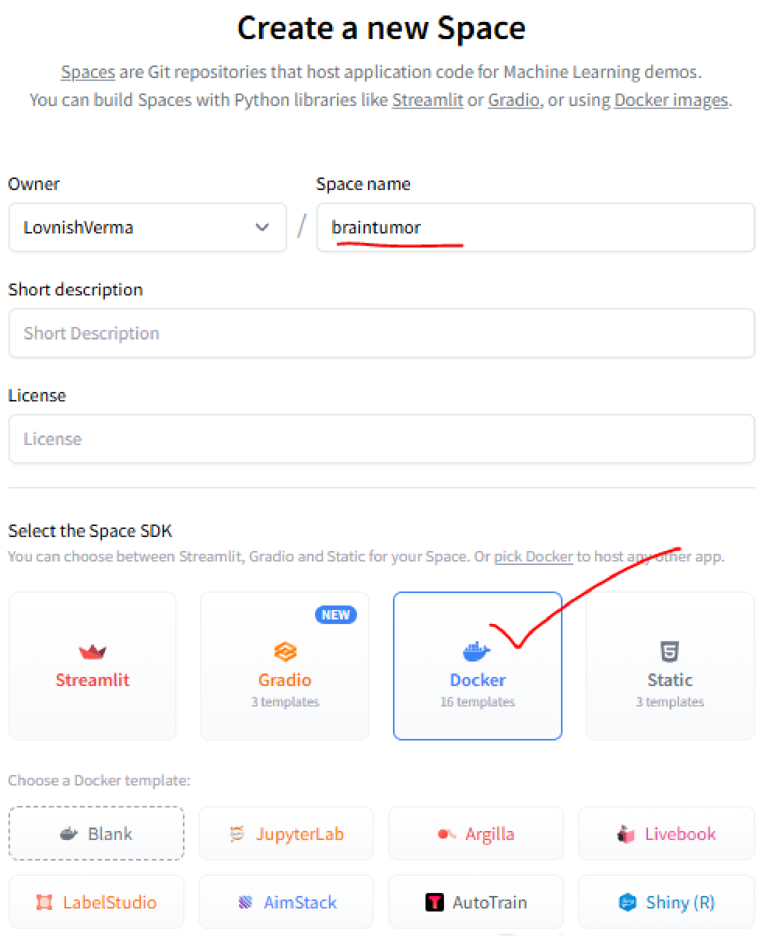
1. **Create Account**: Sign up on Hugging Face.
   https://huggingface.co/

2. **Create a New Space** by clicking on **New** then **Space**.



Deployment on Hugging Face

- Set Space name.
- Select **Docker**.
- Set Space Visibility Public.
- Click on **Create Space**.



Create a new Space

Spaces are Git repositories that host application code for Machine Learning demos.
You can build Spaces with Python libraries like Streamlit or Gradio, or using Docker images.

Owner: LovnishVerma

Space name: braintumor

Short description: Short Description

License: License

Select the Space SDK

You can choose between Streamlit, Gradio and Static for your Space. Or pick Docker to host any other app.

Deployment on Hugging Face

1. Set Up Repository:

1. Clone the repository locally.
2. Use this command to clone project repository locally.

# Make sure you have git-lfs installed (https://git-lfs.com)

```
git lfs install
git clone https://huggingface.co/spaces/LovnishVerma/braintumor
```

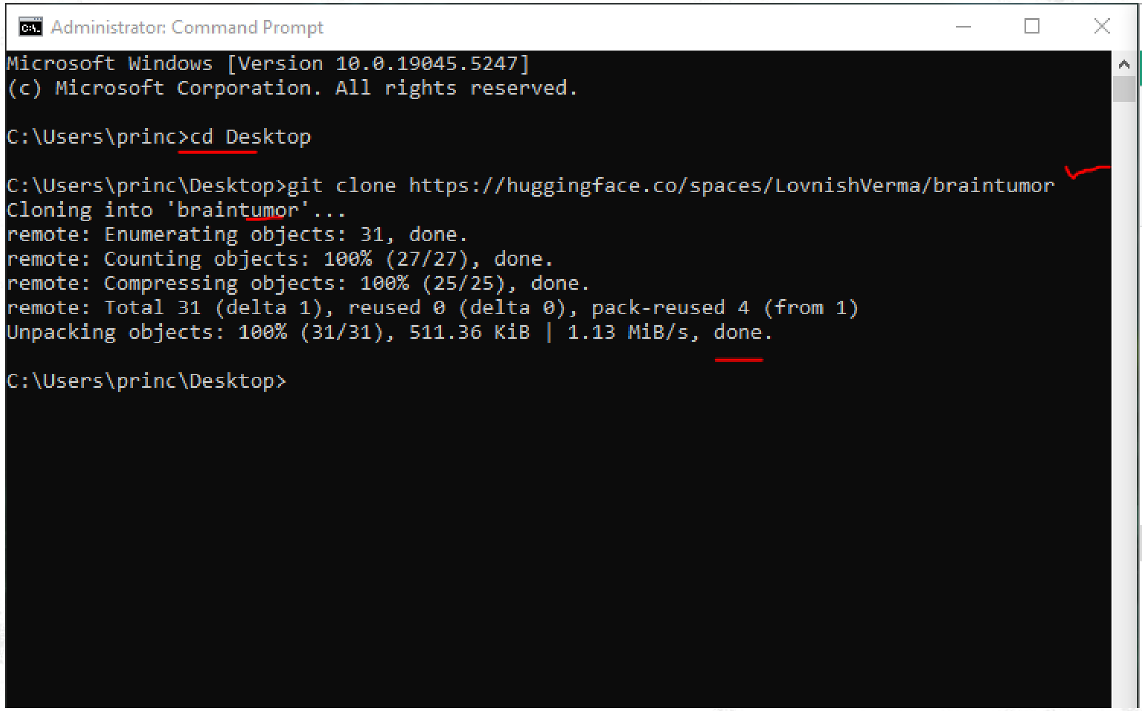
# If you want to clone without large files - just their pointers

```
GIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/spaces/LovnishVerma/braintumor
```

```
# Make sure you have git-lfs installed (https://git-lfs.com) git lfs install
git clone https://huggingface.co/spaces/LovnishVerma/braintuor
# If you want to clone without large files - just their pointersGIT_LFS_SKIP_SMUDGE=1 git clone https://huggingface.co/spaces/LovnishVerma/braintumor
```

Deployment on Hugging Face

- Make sure your PC has git installed.
- Open CMD then enter git clone command
- CD to folder you want to clone repo
- git clone https://huggingface.co/spaces/LovnishVerma/braintumor

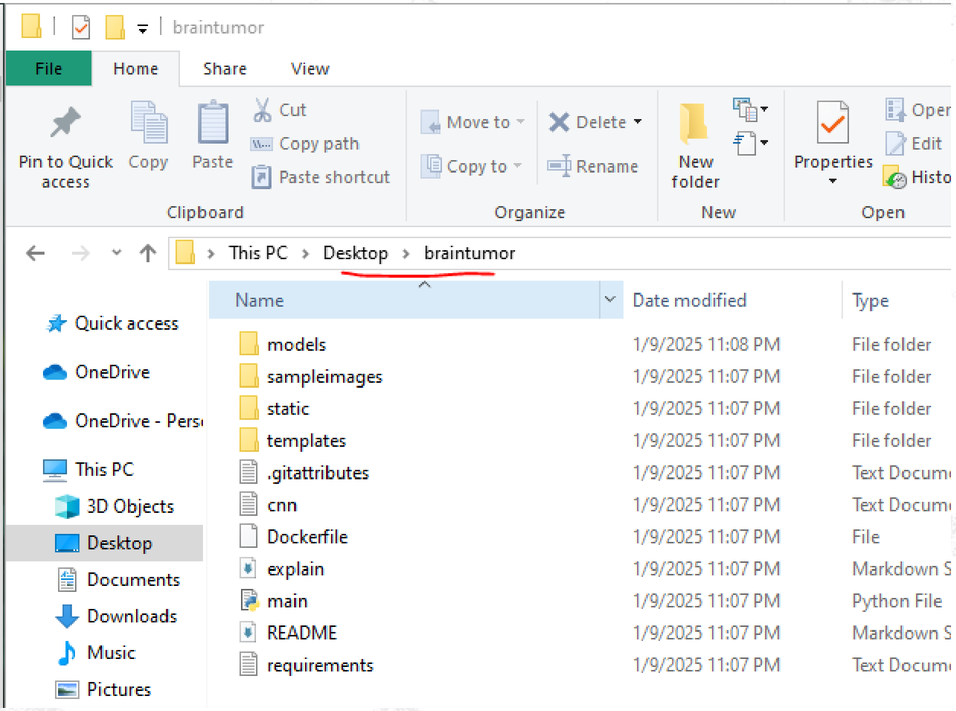


```
Microsoft Windows [Version 10.0.19045.5247]
(c) Microsoft Corporation. All rights reserved.

C:\Users\princ>cd Desktop

C:\Users\princ\Desktop>git clone https://huggingface.co/spaces/LovnishVerma/braintumor
Cloning into 'braintumor'...
remote: Enumerating objects: 31, done.
remote: Counting objects: 100% (27/27), done.
remote: Compressing objects: 100% (25/25), done.
remote: Total 31 (delta 1), reused 0 (delta 0), pack-reused 4 (from 1)
Unpacking objects: 100% (31/31), 511.36 KiB | 1.13 MiB/s, done.

C:\Users\princ\Desktop>
```

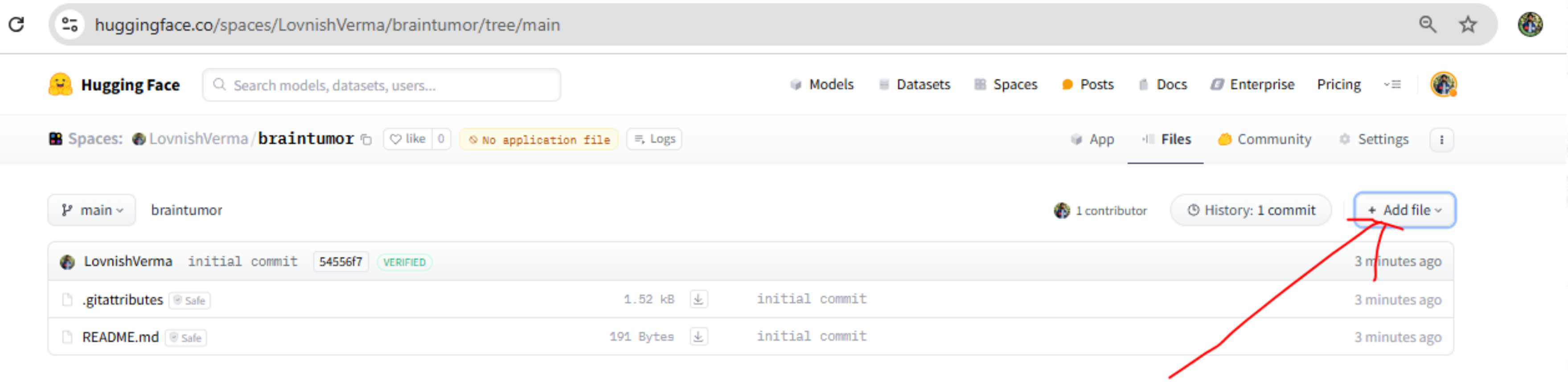


Name	Date modified	Type
models	1/9/2025 11:08 PM	File folder
sampleimages	1/9/2025 11:07 PM	File folder
static	1/9/2025 11:07 PM	File folder
templates	1/9/2025 11:07 PM	File folder
.gitattributes	1/9/2025 11:07 PM	Text Docum
cnn	1/9/2025 11:07 PM	Text Docum
Dockerfile	1/9/2025 11:07 PM	File
explain	1/9/2025 11:07 PM	Markdown S
main	1/9/2025 11:07 PM	Python File
README	1/9/2025 11:07 PM	Markdown S
requirements	1/9/2025 11:07 PM	Text Docum

Deployment on Hugging Face

1. Add Files:

1. Upload Python files, model weights (braintumor.h5), and requirements.txt to your space from your local pc by clicking **add files** and **drag and drop all the files and folde**r to hugging face space from your local pc.



huggingface.co/spaces/LovnishVerma/braintumor/tree/main

Spaces: LovnishVerma/braintumor

main braintumor

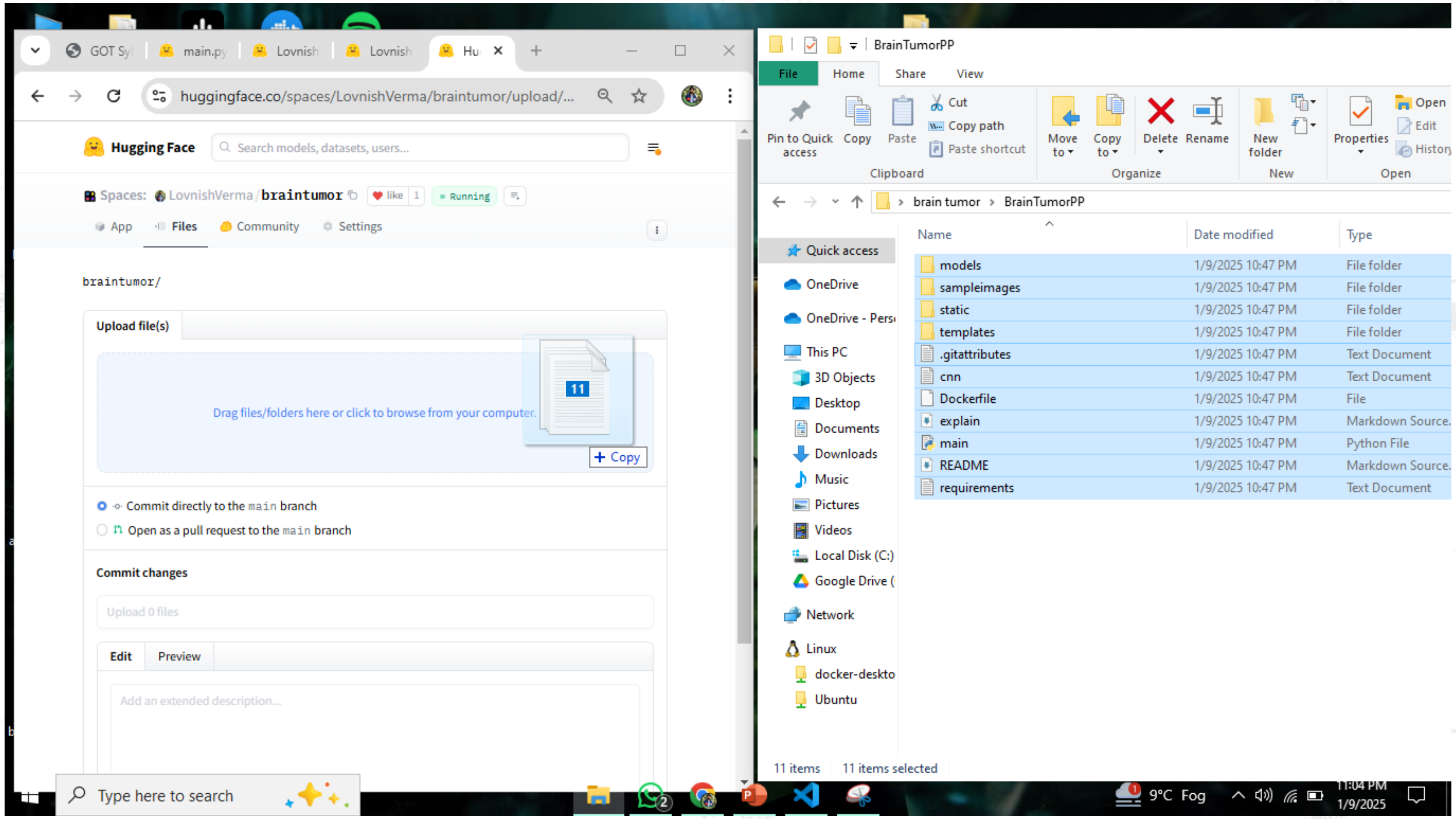
LovnishVerma initial commit 54556f7 VERIFIED 3 minutes ago

.gitattributes Safe 1.52 kB initial commit 3 minutes ago

README.md Safe 191 Bytes initial commit 3 minutes ago

Deployment on Hugging Face

- Drag and Drop all the file wait for letting them upload then click on **Commit Changes**.



braintumor/

Upload file(s)

Drag files/folders here or click to browse from your computer.

● Commit directly to the `main` branch

○ Open as a pull request to the `main` branch

Commit changes

Upload 0 files

Edit | Preview

Add an extended description...

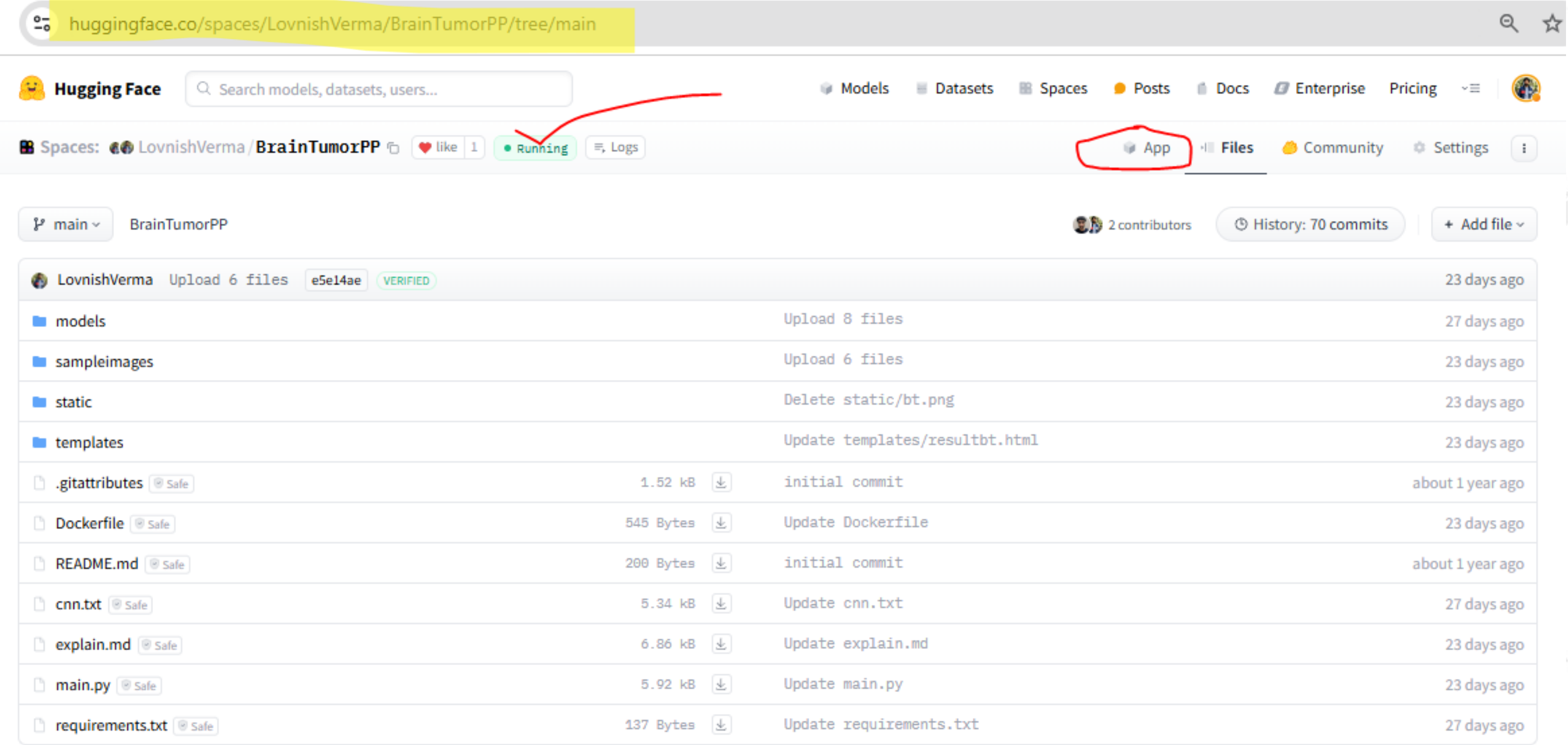
Name	Date modified	Type
models	1/9/2025 10:47 PM	File folder
sampleimages	1/9/2025 10:47 PM	File folder
static	1/9/2025 10:47 PM	File folder
templates	1/9/2025 10:47 PM	File folder
.gitattributes	1/9/2025 10:47 PM	Text Document
cnn	1/9/2025 10:47 PM	Text Document
Dockerfile	1/9/2025 10:47 PM	File
explain	1/9/2025 10:47 PM	Markdown Source
main	1/9/2025 10:47 PM	Python File
README	1/9/2025 10:47 PM	Markdown Source
requirements	1/9/2025 10:47 PM	Text Document

11 items   11 items selected

Deployment on Hugging Face

Wait few Minutes Untill your app status is running

Launch App: Access the app on your Hugging Face URL.

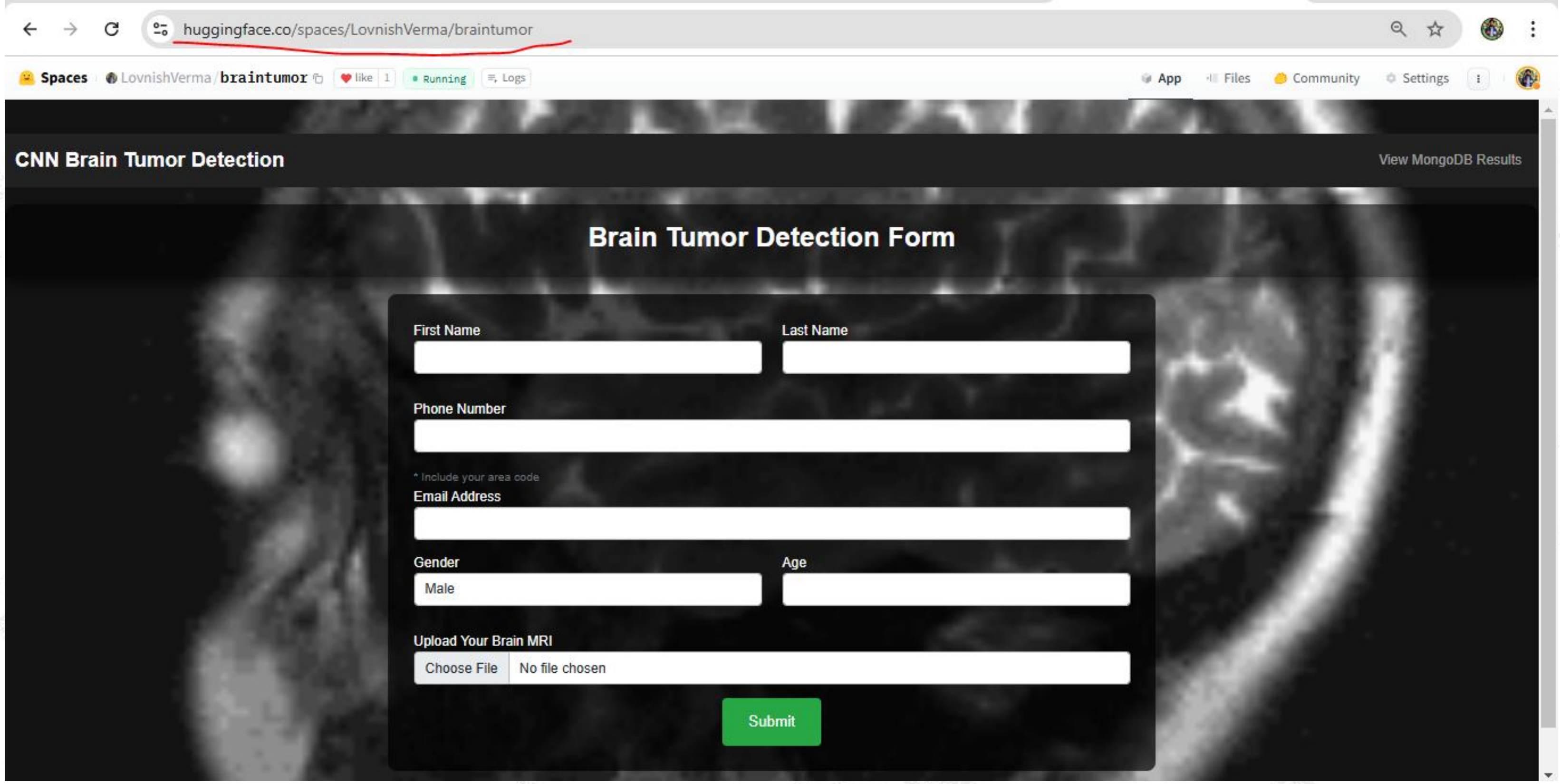


LovnishVerma Upload 6 files e5e14ae VERIFIED 23 days ago

models		Upload 8 files	27 days ago
sampleimages		Upload 6 files	23 days ago
static		Delete static/bt.png	23 days ago
templates		Update templates/resultbt.html	23 days ago
.gitattributes Safe	1.52 kB	initial commit	about 1 year ago
Dockerfile Safe	545 Bytes	Update Dockerfile	23 days ago
README.md Safe	200 Bytes	initial commit	about 1 year ago
cnn.txt Safe	5.34 kB	Update cnn.txt	27 days ago
explain.md Safe	6.86 kB	Update explain.md	23 days ago
main.py Safe	5.92 kB	Update main.py	23 days ago
requirements.txt Safe	137 Bytes	Update requirements.txt	27 days ago

Deployment on Hugging Face

- Your App is ready and Live on Hugging Face
- Access the app on your Hugging Face URL.



huggingface.co/spaces/LovnishVerma/braintumor

CNN Brain Tumor Detection

View MongoDB Results

Brain Tumor Detection Form

First Name

Last Name

Phone Number

* Include your area code

Email Address

Gender: Male

Age

Upload Your Brain MRI

Choose File No file chosen

Submit

Summary

- Implemented a Flask application to predict brain tumors.
- Preprocessed images for accurate CNN predictions.
- Used MongoDB for scalable and efficient data storage.
- Deployed the application on Hugging Face Spaces for accessibility.

https://huggingface.co/spaces/LovnishVerma/braintumor