

- **Project Name:** Optimus

- **Problem Statement:**

Retraining classification and detection models takes a lot of time and effort. First, we need to prepare new datasets and merge them with older ones, which can be complex and time-consuming. Then, we have to evaluate the models by generating performance reports like graphs and confusion matrices. After that, it's important to store and organize these trained models properly so they can be used again when needed. On top of that, deploying these models for use adds another layer of difficulty. Managing all these steps together is a big challenge, and it slows down our overall progress.

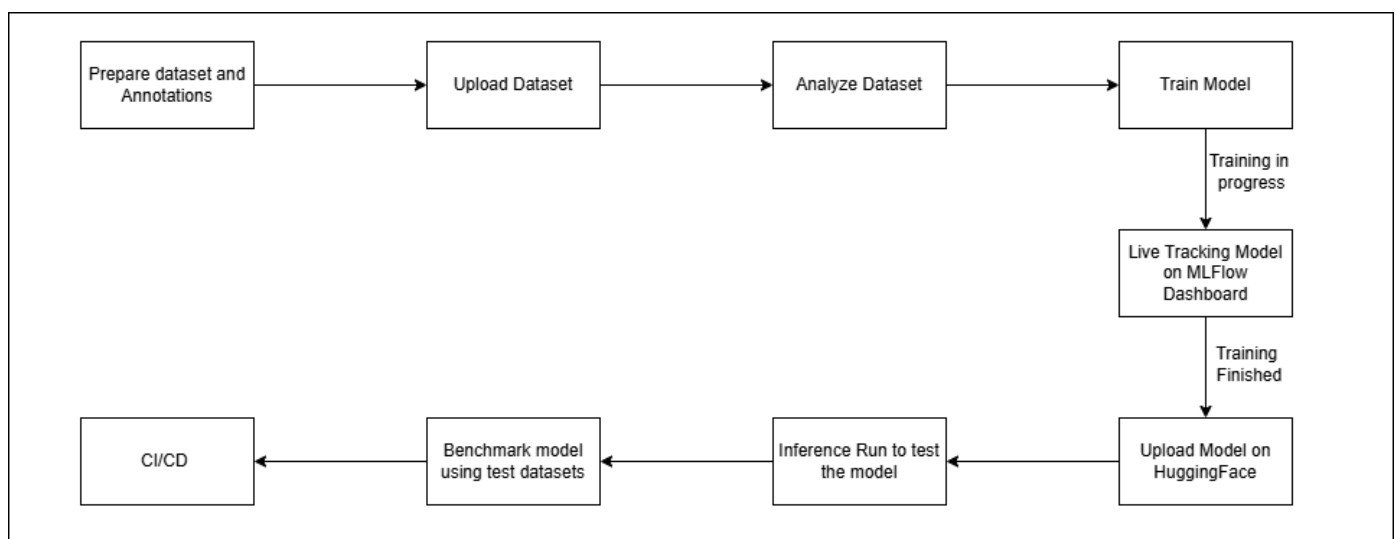
- **Project Objective:**

The goal of this project is to create an easy-to-use Streamlit application that simplifies the process of managing machine learning workflows using UI. This app will allow to:

1. Manage Datasets: Add, update, and organize datasets efficiently in one place.
2. Visualize Datasets: Explore and analyze datasets through interactive visualizations to better understand their content.
3. Train Models: Select multiple datasets for training machine learning models and run the training process directly from the app.
4. Track Performance: Automatically save and display important model performance metrics, such as graphs and confusion matrices, using an integrated MLflow dashboard.
5. Deploy Models: Seamlessly deploy trained models to Hugging Face, making them accessible for further use or sharing.

This app will streamline the end-to-end workflow, making it easier to manage datasets, train models, and monitor their performance.

- **Flow Diagram:**



- **Tech Stack:** Python, Streamlit, MLFlow, Docker, HuggingFace, Ultralytics, Pytorch, Label Studio

- **Workflow Overview:**

1. **Upload Dataset -**

- Upload a new dataset directly through the app by providing zip of dataset or select an existing dataset from a predefined list.
- The app ensures the datasets are stored in an organized structure for easy access.
- Basic validation is performed to check the dataset format, ensuring compatibility with the model training process.

Upload Dataset GO

Upload Dataset for Model Training

Upload Dataset for Benchmark Model

Select Dataset Type

classification

☐ Add New Model

Select Existing Model

ovd_classification

Dataset Version Name

Enter version name (e.g., version_1)

Upload a ZIP file

Drag and drop file here
Limit 200MB per file • ZIP

Browse files

Description

Optional: Add a description for this dataset version

Upload and Extract

Upload Dataset GO

Upload Dataset for Model Training

Upload Dataset for Benchmark Model

Select Dataset Type

classification

☐ Add New Model

Select Existing Model

ovd_classification_test

Dataset Version Name

Enter version name (e.g., version_1)

Upload a ZIP file

Drag and drop file here
Limit 200MB per file • ZIP

Browse files

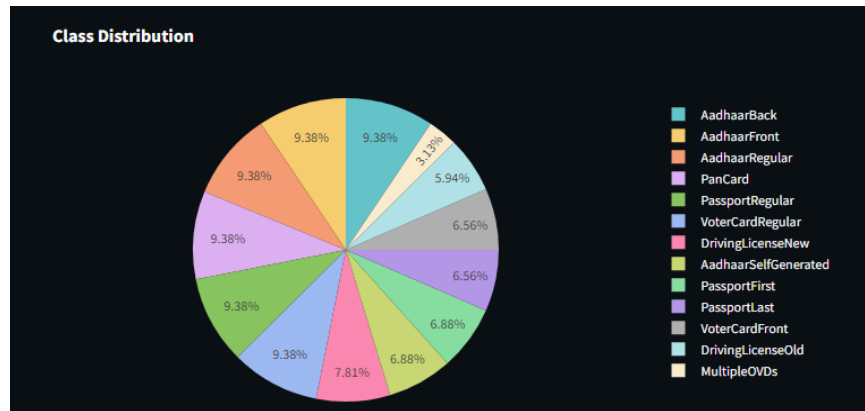
Description

Optional: Add a description for this dataset version

Upload and Extract

2. Data Visualization-

- The app provides interactive visualization tools to explore the dataset.
- Users can view data distributions, identify missing or inconsistent values, and understand key features.
- Common visualization techniques like bar graphs, pie charts, and scatter plots are available to analyze the dataset effectively

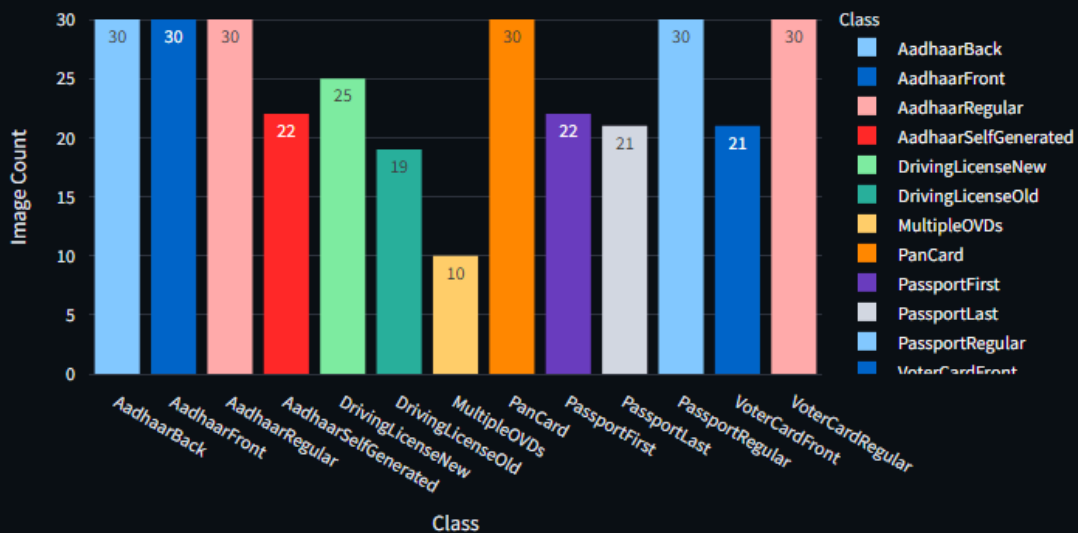


Test Dataset

Total Train Images: 320

Total Train Classes: 13

Testing Data Stats



3. Train the model:

- Select a dataset and initiate the training process by specifying the model configuration.
- Select multiple datasets and train the model on combined dataset.
- The app supports multiple training parameters (e.g., epochs, learning rate) and allows users to customize them.
- After training, the app saves logs and provides details of trained model in the MLflow dashboard.

Train Model

Select Model Type
YOLO

Select Dataset Type
classification

Select Model
ovd_classification

Select Dataset Versions
Choose an option

Training Parameters

Enter Model Run Name (Please enter proper version)
ovd_classification_20241230_v0

Enter Model Description

Select Model Size
n

Batch Size
16

Epochs
2

Learning Rate
0.00100

Start Training

Combining selected dataset versions...

[Click here to view MLFlow Dashboard](#)

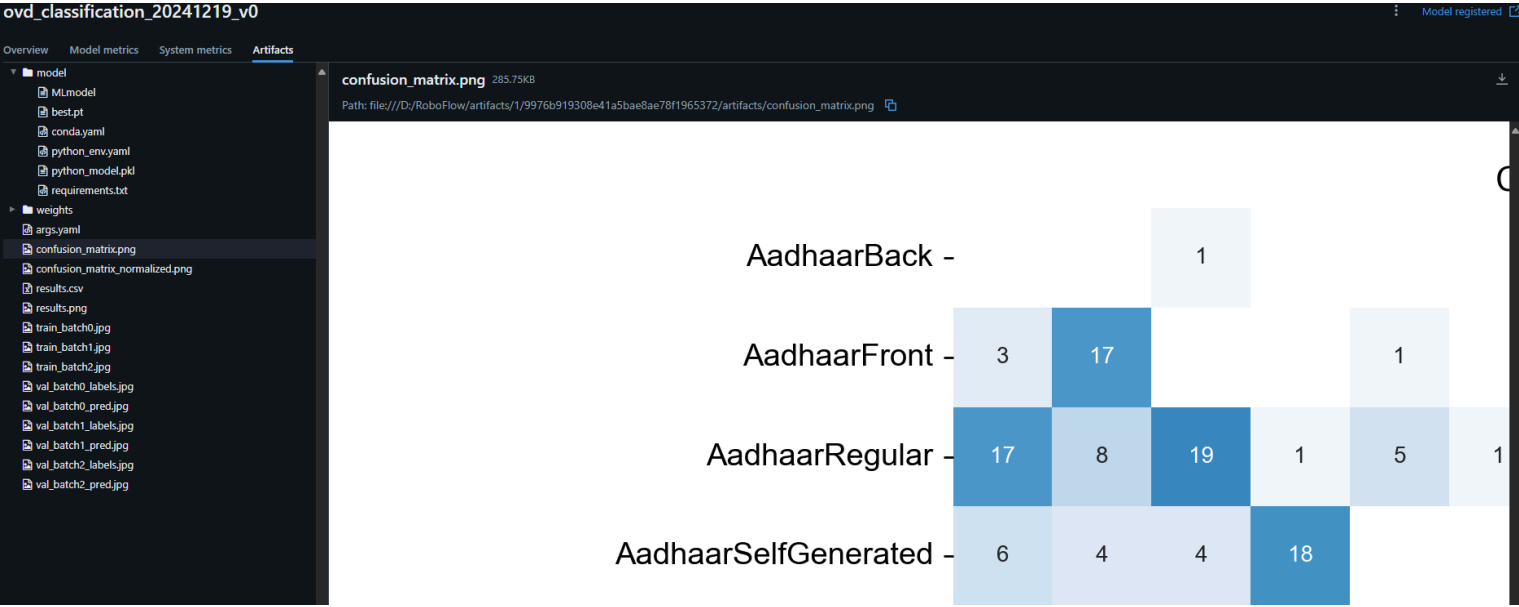
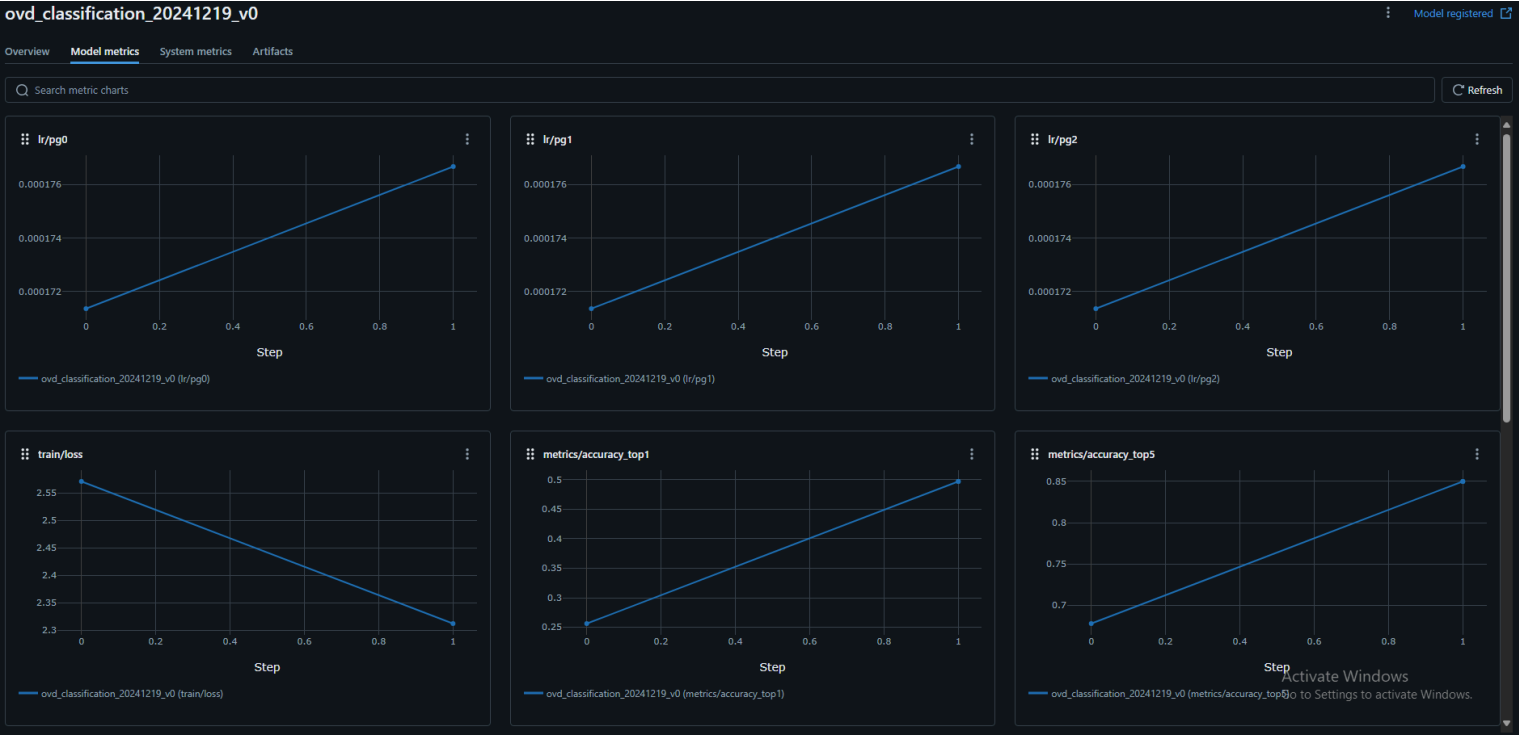
[MLFlow Dashboard](#)

4. Save Metrics in MLflow:

- Once the model is trained, the app automatically logs key performance metrics, such as accuracy, precision, recall, and confusion matrices, into MLflow.
- MLflow also stores the trained model's version, making it easy to retrieve or compare later.
- Users can access detailed performance reports through MLflow's dashboard.

The screenshot displays the MLflow 2.19.0 interface. At the top, there are tabs for 'Experiments' and 'Models'. Below the navigation bar, the breadcrumb 'ovd_classification >' is visible, followed by the experiment name 'ovd_classification_20241219_v0'. A sub-navigation bar includes 'Overview' (selected), 'Model metrics', 'System metrics', and 'Artifacts'. The 'Description' section shows the text 'Model to classify ovd's'. The 'Details' section contains a table with the following information:

Created at	2024-12-19 19:19:22
Created by	gaurav.chavan
Experiment ID	1
Status	Finished
Run ID	9976b919308e41a5bae8ae78f1965372
Duration	1.8min
Datasets used	—
Tags	datasets: ['v0', 'v1']
Source	Upload_Dataset.py d8187aa
Logged models	pyfunc
Registered models	ovd_classification v3



5. Deploy the Model to Hugging Face:

- After training, the app provides an option to deploy the model directly to Hugging Face.
- The app ensures the model is converted into the required format and uploads it along with relevant metadata.
- Once deployed, the model is accessible via Hugging Face's interface for further usage or sharing with others.

The screenshot shows the Hugging Face interface for the model 'mahmadamin08/ovd_classify'. The page has a dark theme. At the top, there's a navigation bar with the Hugging Face logo, a search bar, and links for Models, Datasets, Spaces, Posts, Docs, Enterprise, Pricing, Log In, and Sign Up. Below the navigation bar, the model name 'mahmadamin08/ovd_classify' is displayed with a like count of 0. There are tabs for 'Model card', 'Files and versions' (which is active), and 'Community'. The 'Files and versions' tab shows a commit history for the 'ovd_classify' branch. The commit history table has columns for the commit author, file name, size, commit message, and time. The files listed are '.gitattributes' (1.52 kB), 'README.md' (318 Bytes), and 'ovd_classify_20241219_v0.pt' (2.99 MB, marked as LFS). The commit 'ovd_classify_20241219_v0 model' is highlighted.

Commit	File	Size	Message	Time
mahmadamin08	.gitattributes	1.52 kB	initial commit	4 minutes ago
mahmadamin08	README.md	318 Bytes	Add model card	4 minutes ago
mahmadamin08	ovd_classify_20241219_v0.pt	2.99 MB	Upload ovd_classify_20241219_v0 model	4 minutes ago

The screenshot shows the file page for 'ovd_classify_20241219_v0 model' on Hugging Face. The page has a dark theme. At the top, the file name is displayed with a commit hash '3ce2ce0' and a 'VERIFIED' badge. Below the file name, there are links for 'download', 'Copy download link', 'history', 'blame', 'contribute', 'delete', and 'pickle'. The main content area contains a message: 'This file is stored with Git LFS . It is too big to display, but you can still [download](#) it.' Below this message, there is a section titled 'Git LFS Details' which contains the following information: 'SHA256: c7ad001b15fdce0f26351d71f6c4df66396e88dab6af1ebf6a3357f2cf2820e7', 'Pointer size: 132 Bytes', and 'Size of remote file: 2.99 MB'. There is a link for 'Raw pointer file'. At the bottom, there is a note: 'Git Large File Storage (LFS) replaces large files with text pointers inside Git, while storing the file contents on a remote server. [More info](#).'

This file is stored with [Git LFS](#) . It is too big to display, but you can still [download](#) it.

Git LFS Details

SHA256: c7ad001b15fdce0f26351d71f6c4df66396e88dab6af1ebf6a3357f2cf2820e7
Pointer size: 132 Bytes
Size of remote file: 2.99 MB

[Raw pointer file](#)

Git Large File Storage (LFS) replaces large files with text pointers inside Git, while storing the file contents on a remote server. [More info](#).

6. Inference Run:

- Users can choose from a list of deployed models through an intuitive interface, ensuring flexibility in testing different versions or architectures.
- The system allows users to upload images directly, providing a seamless method to test the model with custom inputs.
- Uploaded images are processed in real time, with predictions and associated metrics displayed instantly, enabling immediate evaluation of model performance.

Testing Model

▶ Test Model

▶ Download Model

Select Model Type to Test

classification

Select Model to Test

ovd_classification_20241220_v0

Upload img file

 Drag and drop file here
Limit 200MB per file • PNG, JPG, TIF, TIFF, JPEG, PDF

Browse files

Testing Model

▶ Test Model

▶ Download Model

Select Model Type

classification

Available Models

Model Name	Size (MB)	Download
ovd_classification_20241220_v0	2.85	Download

7. Benchmark Model:

- Provide an intuitive interface to choose any deployed model for benchmarking against a predefined test dataset or user-uploaded data.
- Show annotated examples of both correct and incorrect predictions, with corresponding confidence scores, to provide intuitive insights into model behavior.
- Generate and display an interactive confusion matrix, illustrating true positives, false positives, false negatives, and true negatives for each class, helping users identify specific areas of model strength and weakness.

Benchmark Model

Select Model Type
classification

benchmark\classification

Select Model to Test
ovd_classification_20241220_v0

Select Dataset to Test
ovd_classification_test

Select Dataset Versions
v0 x v1 x

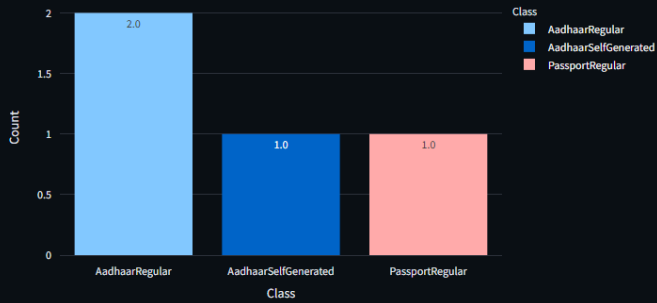
Processing datasets...



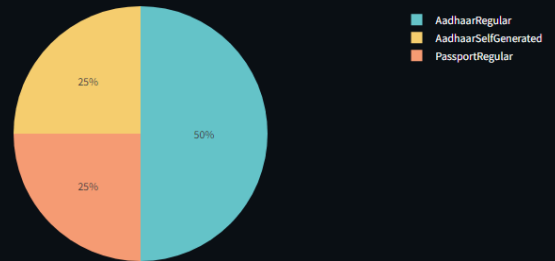
✓ Classified Data Summary

	Class	Count
0	AadhaarRegular	2
1	AadhaarSelfGenerated	1
2	PassportRegular	1

Misclassified Data Stats



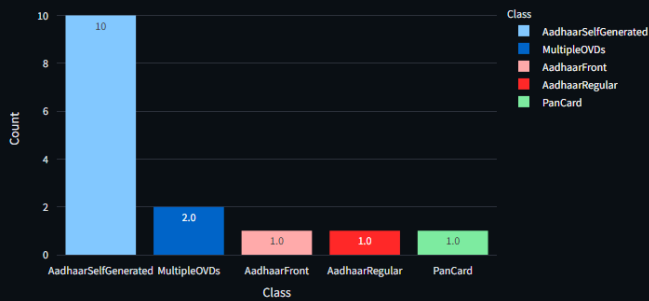
Class Distribution



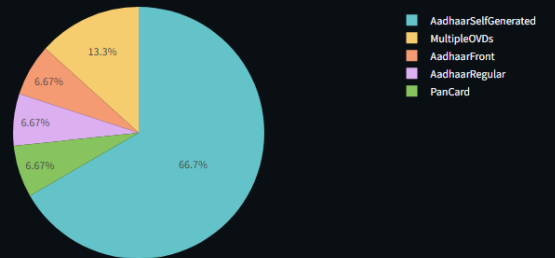
✗ Misclassified Data Summary ↻

	Class	Count
0	AadhaarSelfGenerated	10
1	MultipleOVDs	2
2	AadhaarFront	1
3	AadhaarRegular	1
4	PanCard	1

Misclassified Data Stats



Class Distribution



8. CI/CD using streamlit:

- Allow users to input the repository ID and model version, enabling automated setup for testing different versions of a model within the CI/CD pipeline.
- Once the repository ID and model version are provided, the system automatically generates a Docker image by pulling the relevant code and dependencies from the specified repository, ensuring a consistent and isolated environment for running the model.

CI/CD for Flask App

[▶ Build and Run Docker Image](#)[▶ View API URLs](#)

Hugging Face Repo ID

mahmadamin08/ovd_classification

Model Version

ovd_classification_20241226_v0

[Deploy Flask API](#)

Building the Docker image...

Test API

Select API endpoint

http://localhost:63696/predict

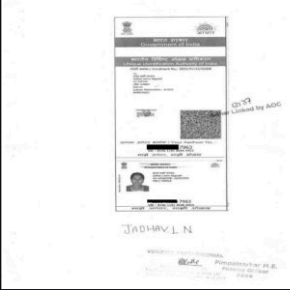
Upload an image

Drag and drop file here
Limit 200MB per file • PNG, JPG, JPEG

[Browse files](#)

download (1).png 34.6KB

Uploaded Image



Uploaded Image

API Response

```
{
  "confidence": [
    0 : 0.12
  ]
  "perdicted_class": "AadhaarSelfGenerated"
}
```

- **Conclusion:**

This project provides an integrated platform for dataset management, model training, performance tracking, and deployment. Users can easily upload or select datasets, visualize and analyze the data, and train models with customizable parameters. The app tracks model performance in real-time, logs key metrics into MLflow, and enables easy model deployment to Hugging Face. By automating key processes and providing intuitive tools for exploration and tracking, this project streamlines the machine learning workflow, making it more efficient and accessible for users to train, evaluate, and deploy models.