ARCH TECHNOLOGIES

NAME	MISBAH ULLAH
PHONE NUMBER	03113902053
INTERN ID	ARCH-2505-0237
SUBMITTED TO	ARCH TECHNOLOGIES.

Review of the Project

This project refers to the detection and distribution of brain tumors with the help of Yolov11.
I used:

- Yolov11 determines the object (finding the tumor in the figure).
- Sam2 (its own demo -ZIS of the tumor area).
- Sam2 and then used the tumors more precisely.

Study Material and References

Below is the list of videos, tutorials, and articles I used to understand and complete the Brain Tumor Segmentation Project using YOLOv11 and SAM2:

◈ 1. Yolov11 and SAM2 for Custom Instance Segmentation

YouTube Video:
Yolov11 and SAM2 for Custom Instance Segmentation - Code With Arohi
🔗 https://www.youtube.com/watch?v=a7fHmqkUu5Q

Used for:

- Learning how to train YOLOv11
- Understanding how to use SAM2 for segmentation
- Connecting YOLO results with SAM2

Learning outcomes from video

Following are handwritten notes of YouTube video:

Topic Name: Brain Tumor detection
Video By : Code with Arohi
Tools : Yollo 11, SAM2

STEPS

→ Install Libraries

- → Install ultralytics
- → Install Segment Anything Model
- → use pip install for dependencies

→ Prepare dataset:

- → use Roboflow or custom dataset.
- → Export data in Yollo Format.
- → Creat folder: train, val, test

→ Train Yollo 110

- → Training data in runs/detect/train
- → Best weights saved as "best.pt"

→ Test the Modell.

- → use your model to detect brain tumor from predict folder.

→ Integrate Yollo with SAM:

→ Import SAM
→ Get bounding boxes from yollo result.
→ Pass them to SAM for segmentation.

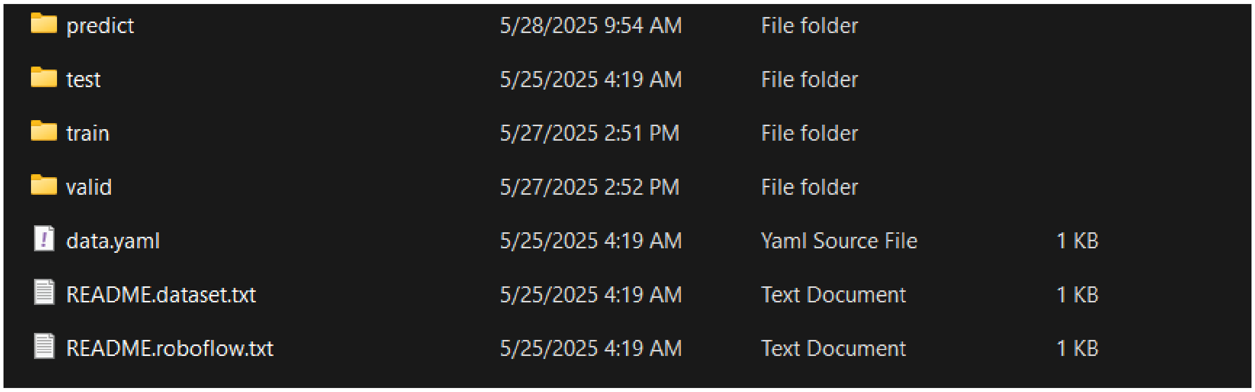
→ Run Segmentations:

→ ~~Re~~ Run Segmentation and save output in runs/detect/predict

Images and Diagrams.

These are images of brain tumors before and after training.

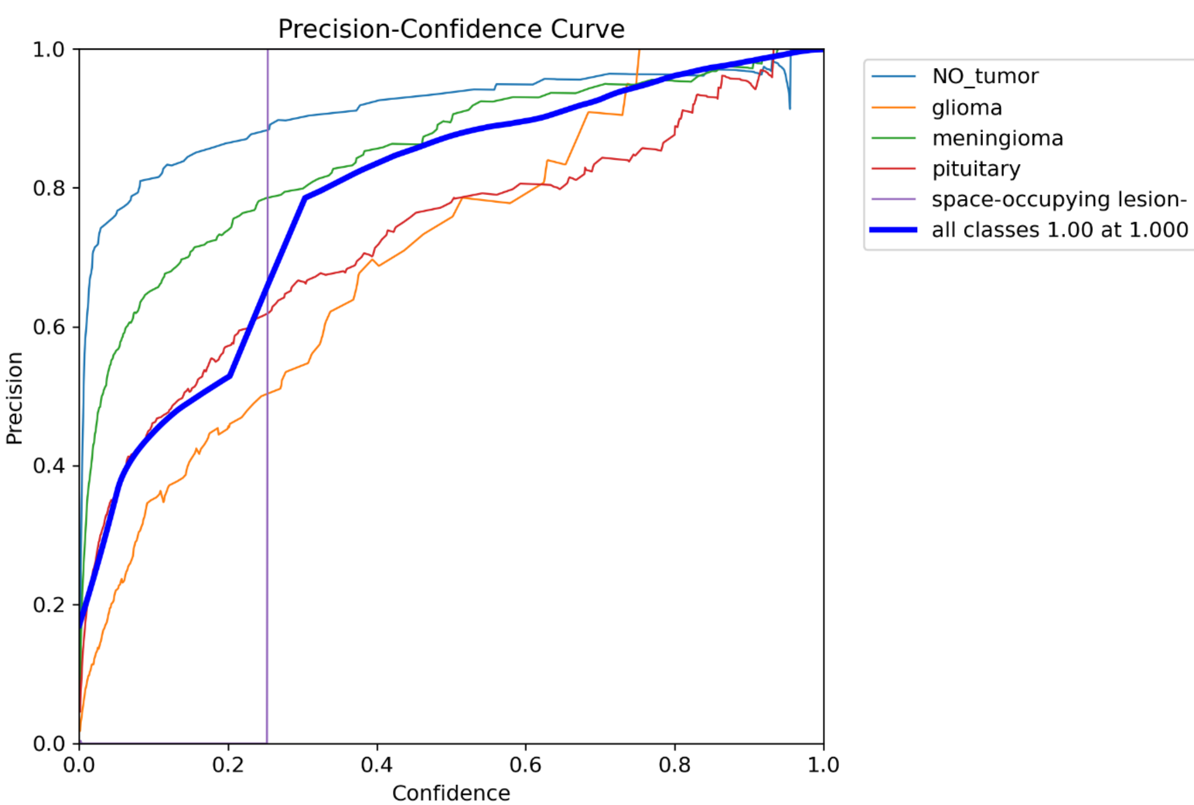
Dataset Structure Diagram

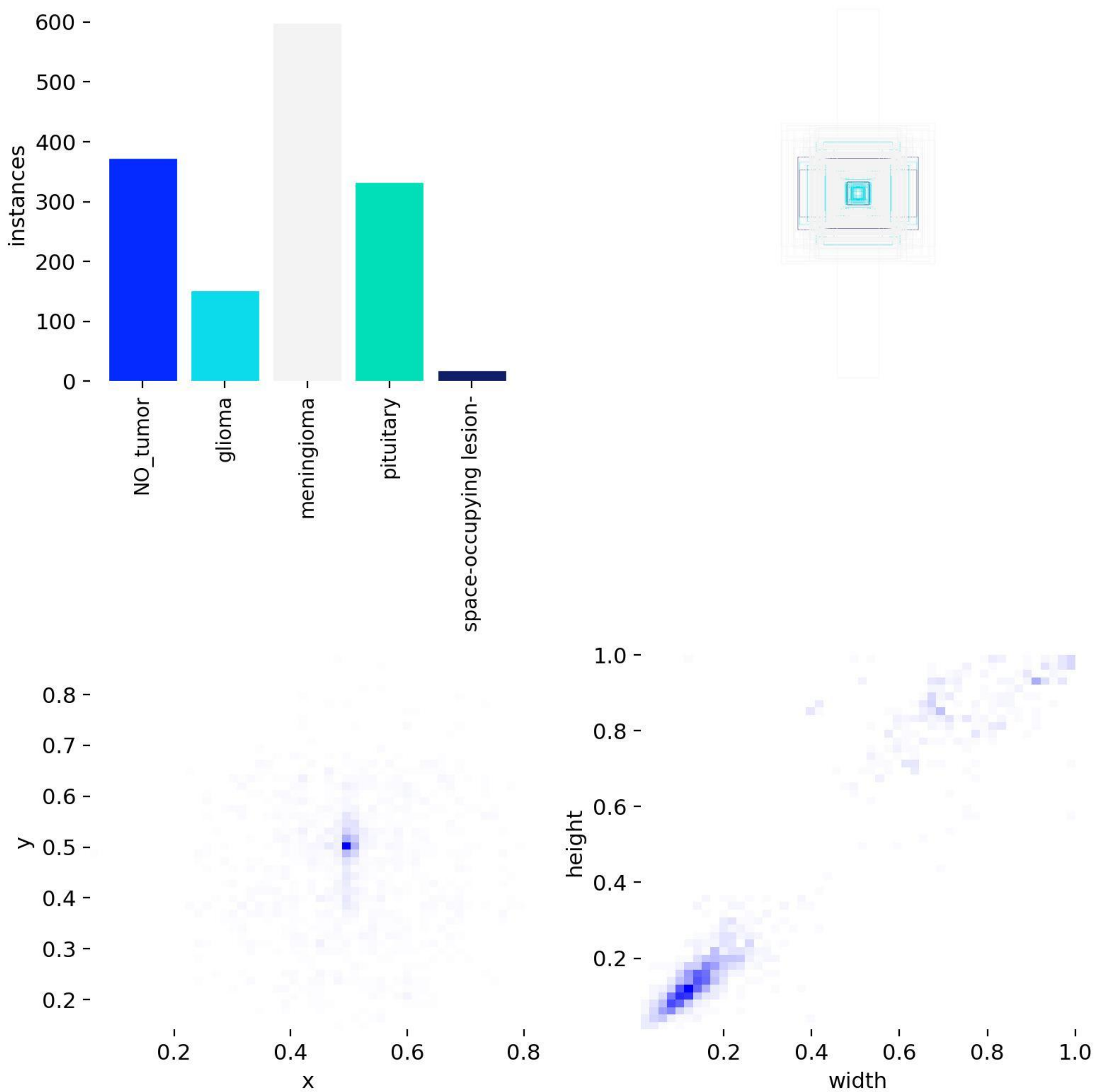


Dataset images:

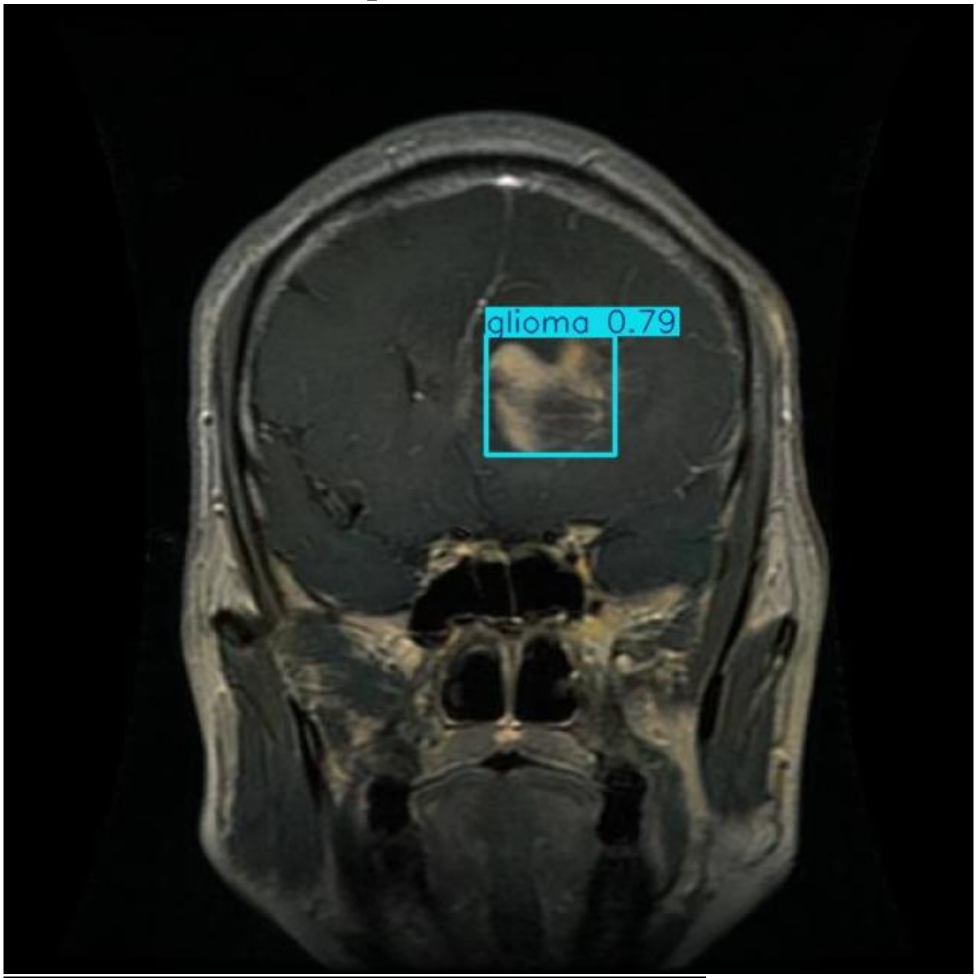
Yolov11 Model Training Graph:

```
 Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
 15/15         0G     0.7947     0.6052     0.9228          2        128: 100%|██████████| 172/172 [01:25<00:00,  2.01it/s]
                Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 25/25 [00:13<00:00,  1.91it/s]
                  all        395        415      0.819      0.573      0.596      0.417

ochs completed in 1.455 hours.
izer stripped from runs\detect\train11\weights\last.pt, 6.2MB
izer stripped from runs\detect\train11\weights\best.pt, 6.2MB

ating runs\detect\train11\weights\best.pt...
lytics 8.3.144  Python-3.13.3 torch-2.7.0+cpu CPU (Intel Core(TM) i5-8350U 1.70GHz)
 summary (fused): 72 layers, 3,006,623 parameters, 0 gradients, 8.1 GFLOPs
                Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 25/25 [00:13<00:00,  1.81it/s]
                  all        395        415      0.819      0.574      0.595      0.417
             NO_tumor        115        116      0.926      0.978       0.98      0.788
               glioma         30         36      0.608      0.306      0.364      0.173
           meningioma        144        148      0.889      0.864      0.927      0.691
            pituitary        106        111      0.673      0.721      0.706      0.433
-occupying lesion-          1          4          1          0    0.00106   0.000745
: 0.3ms preprocess, 17.1ms inference, 0.0ms loss, 1.3ms postprocess per image
ts saved to runs\detect\train11
\Users\Misbah>
```

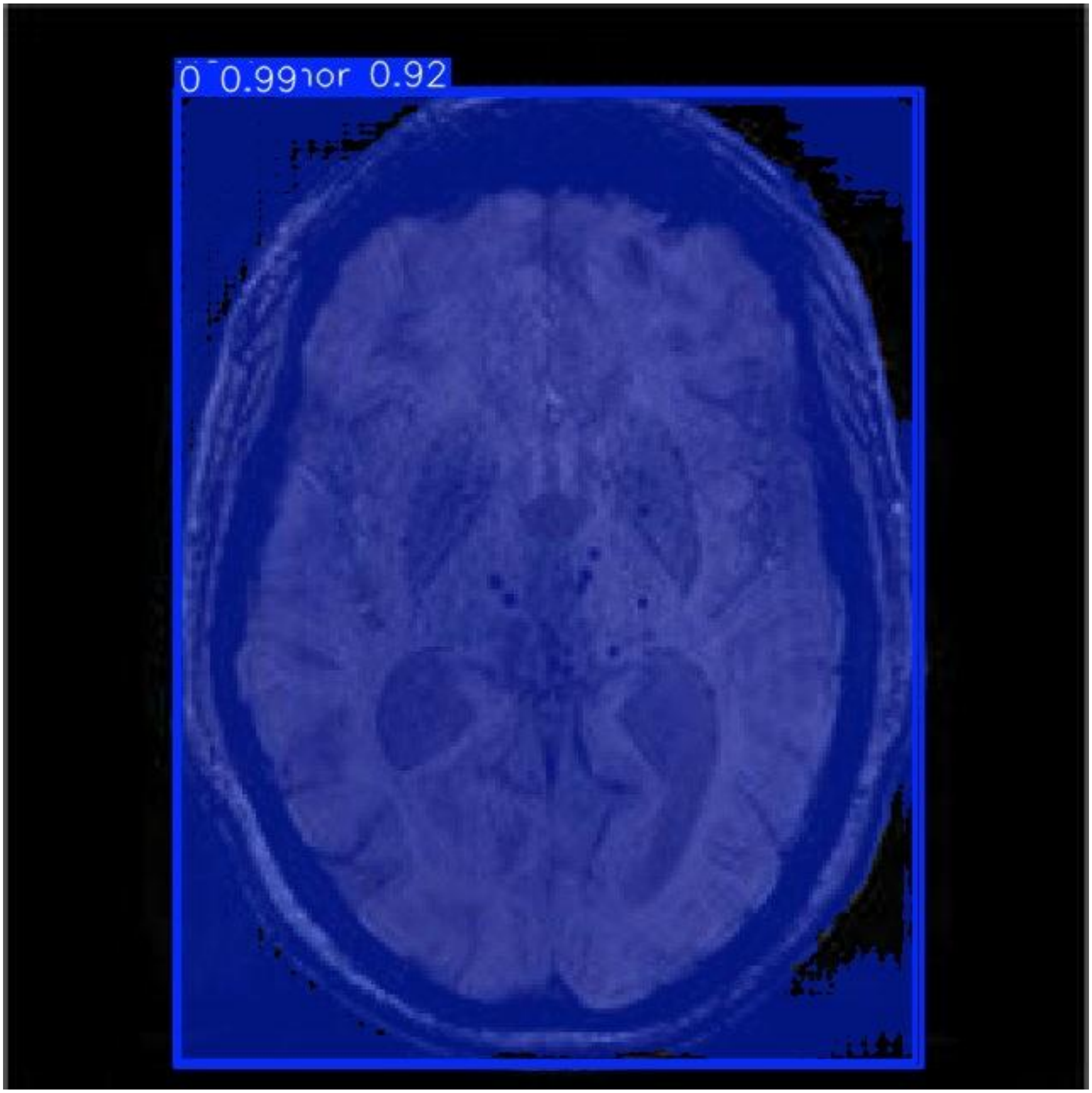



600
500
400
300
200
100
0
instances
NO_tumor
glioma
meningioma
pituitary
space-occupying lesion-
0.8
0.7
0.6
0.5
0.4
0.3
0.2
y
0.2
0.4
0.6
0.8
x
1.0
0.8
0.6
0.4
0.2
height
0.2
0.4
0.6
0.8
1.0
width

Yolov11 Detection Output:



SAM2 Segmentation Output:



Code with Explanation:

python ● Untitled-1.md ● W

C: > Users > Misbah > python

```python
from ultralytics import YOLO
from ultralytics import SAM
```

These lines import the YOLOv11 and SAM2 libraries from Ultralytics, which we will use for detection and segmentation.

C: > Users > Misbah > python > ...

```python
from ultralytics import YOLO

# Load pre-trained YOLOv11 model (or "YOLO('yolov8n.pt')" if using YOLOv8n)
model = YOLO("yolov8n.pt")  # You can use 'yolov8s.pt', 'yolov8m.pt' for larger models

# Train the model
model.train(
    data="brain_tumor.yaml",  # YAML file path
    epochs=50,
    imgsz=640,
    batch=8,
    device="cpu"  # or "0" for GPU
)
```

❖ This **trains the model** using your dataset.

- data points to the YAML file (it tells where images and labels are).
- epochs=50 means training will run for 50 rounds.
- imgsz=640 is the image size.
- batch=8 is how many images are processed at once.
- device="cpu" runs on CPU. Use "0" for GPU.

C: > Users > Misbah > python > ...

```python
from ultralytics import YOLO

# Load the trained YOLO model
model = YOLO("runs/detect/train/weights/best.pt")  # Change path if needed

# Run inference on test images
results = model.predict(source="test_images/", save=True, device="cpu")

```

YOLO(...): Loads the trained model from the path to best.pt (this is the best model saved during training).

source="test_images/": Path to the folder where your test images are saved.

save=True: Saves the output (with bounding boxes) in a new folder (usually runs/detect/predict).

device="cpu": Ensures the model runs on your CPU. (Use "0" if you have a GPU available.)

```python
# Step 3: Load the SAM2 model (make sure sam_b.pt is downloaded)
sam_model = SAM("sam_b.pt")

# Step 4: Loop through all YOLO results and segment using SAM2
for result in results:
    class_ids = result.boxes.cls.int().tolist()  # class ids of detections
    if len(class_ids):  # if detection exists
        boxes = result.boxes.xyxy  # get bounding boxes
        sam_model(
            image=result.orig_img,     # original input image
            bboxes=boxes,              # detected bounding boxes from YOLO
            verbose=False,
            save=True,
            device='cpu'               # use CPU
        )
```

The code loads the trained YOLO model to detect tumors in multiple images. For each detected tumor, it extracts the bounding box and passes it along with the original image to the SAM2 model. SAM2 then performs precise segmentation inside those boxes, and the segmented results are saved. The code runs on the CPU to ensure compatibility.

Task 2:

Chapter 1: Definitions and Types of Machine Learning

1. Machine Learning (ML)

Machine Learning is a field of computer science that enables systems to learn patterns from data and make decisions or predictions without being explicitly programmed. It focuses on developing models that improve over time with experience.

---

2. Supervised Learning

Supervised Learning involves training a model on labeled data, where the input comes with the correct output. The goal is to learn a function that maps inputs to desired outputs.

Example: Predicting house prices based on historical data.

---

3. Unsupervised Learning

Unsupervised Learning works with data that has no labels. The goal is to find hidden patterns or groupings in the data.

Example: Clustering customers based on purchasing behavior.

---

4. Semi-Supervised Learning

This method combines a small amount of labeled data with a large amount of unlabeled data. It helps improve learning accuracy when labeling is costly or time-consuming.

Example: Training a model with a few labeled medical images and many unlabeled ones.

5. Reinforcement Learning

An agent learns by interacting with an environment, receiving rewards or penalties based on its actions. It's commonly used in robotics, gaming, and autonomous driving.

Example: Training an AI to play chess or control a robot.

📊 Comparison Tables

📌 Supervised vs Unsupervised Learning

Feature	Supervised Learning	Unsupervised Learning
Data Type	Labeled data	Unlabeled data
Goal	Predict outcome/labels	Find structure/patterns
Examples	Regression, Classification	Clustering, Dimensionality Reduction
Output	Predicts labels	Groups or features
Difficulty	Easier to evaluate	Harder to evaluate objectively

📌 Batch vs Online Learning

Feature	Batch Learning	Online Learning
Data Processing	All at once (offline)	Incrementally (streaming)
Speed	Slower, resource-intensive	Fast and scalable
Use Case	Stable datasets	Dynamic data (e.g., stock market)
Retraining	Needs complete retraining	Learns from new data on the fly

💻 Chapter 2: Code Implementation with Output

Below are the **steps and code blocks** from Chapter 2 (End-to-End ML Project). Each block includes an explanation.

Step 1: Load Dataset

```
from sklearn.datasets import fetch_california_housing
housing = fetch_california_housing(as_frame=True)
df = housing.frame
df.head()
```

Step 2: Train-Test Split

```
from sklearn.model_selection import train_test_split
train_set, test_set = train_test_split(df, test_size=0.2, random_state=42)
```

Step 3: Data Exploration

```
import matplotlib.pyplot as plt
df.hist(bins=50, figsize=(20,15))
plt.show()
```

Step 4: Correlation Matrix

```
corr_matrix = df.corr()
corr_matrix["MedHouseVal"].sort_values(ascending=False)
```

Step 5: Feature Engineering

```
df['rooms_per_household'] = df['AveRooms'] / df['Households'
df['bedrooms_per_room'] = df['AveBedrms'] / df['AveRooms']
```

Step 6: Handling Missing Values

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy="median")
df_num = df.drop("MedHouseVal", axis=1)
imputer.fit(df_num)
X = imputer.transform(df_num)
```

Step 7: Feature Scaling with Pipeline

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler

pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy="median"))
    ('std_scaler', StandardScaler()),
])
df_prepared = pipeline.fit_transform(df_num)
```

Step 8: Training a Linear Model

```
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(df_prepared, df["MedHouseVal"])
```