

**Name : Hira Shehzad**

**Intern ID : ARCH-2505-0452**

**Phone No : +92 3245686812**

**TASK 01 & 02**

**Task 01 – Brain Tumor Segmentation using YOLOv11 and SAM2**

**Project Overview**

This project applies advanced computer vision models — YOLOv11 for object detection and SAM2 (Segment Anything Model 2) for image segmentation — to identify and segment brain tumors in medical images. The goal is to accurately detect tumor regions and generate segmentation masks that can assist medical diagnosis.

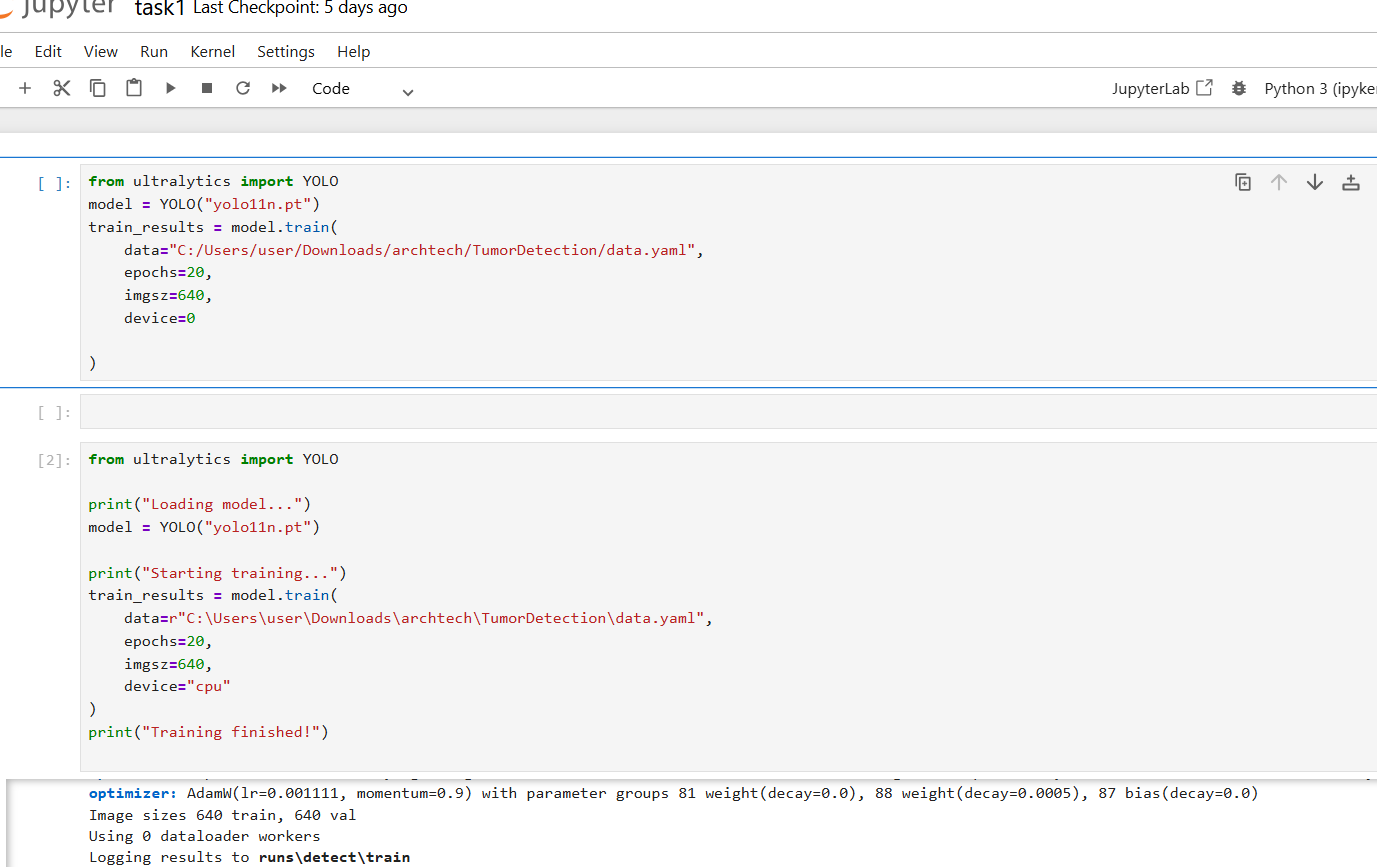
**Objective**

* Understand YOLOv11 & SAM2
* Implement tumor detection and segmentation
* Evaluate model performance
* Document the results and challenges

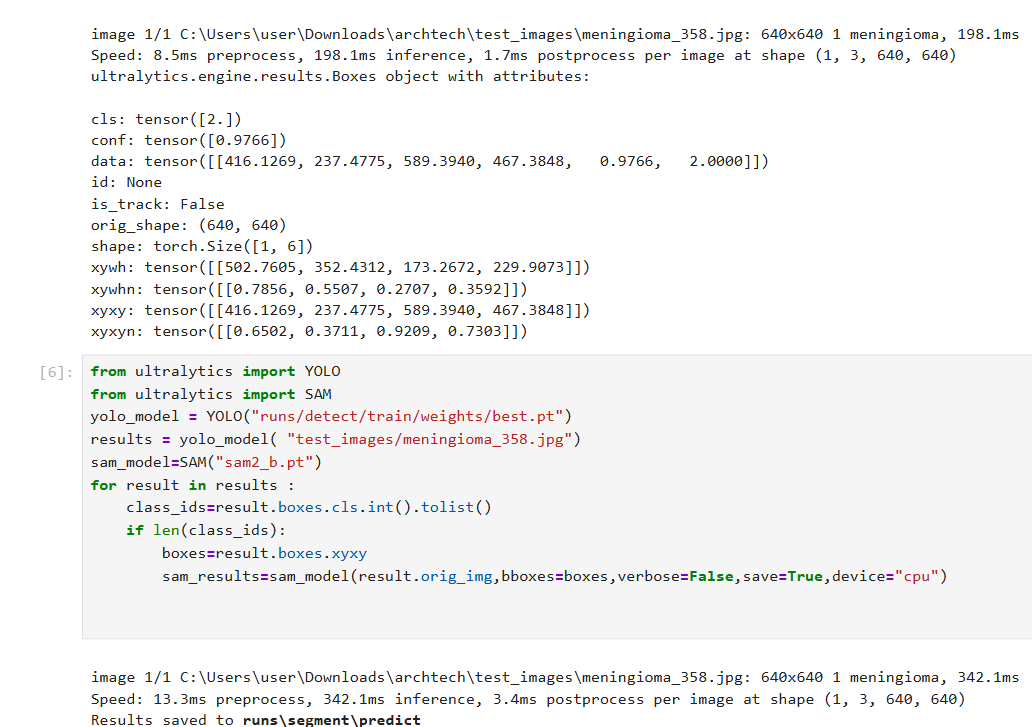
**Tools and Environment**

* **Python**
* **Google Colab / VS Code**
* ultralytics, segment-anything, OpenCV, matplotlib, torch, etc.
* YOLOv11 and SAM2 models
* Brain tumor image dataset (e.g., Kaggle dataset)

**Implementation of Code with Explanation :**







**Code :**

from ultralytics import YOLO

model = YOLO("yolo11n.pt")

train\_results = model.train(

data="C:/Users/user/Downloads/archtech/TumorDetection/data.yaml",

epochs=20,

imgsz=640,

device=0

)

**Explanation:**

* The code begins by importing the YOLO class from the Ultralytics library.
* It initializes a YOLOv8n (nano) model using a pre-trained weights file (yolo11n.pt).
* The model is trained using a custom dataset, the structure of which is defined in the data.yaml file.
* Training runs for 20 epochs with an image size of 640x640 pixels.
* device=0 indicates that the training will utilize the GPU (if available).

**Code :**

from ultralytics import YOLO

print("Loading model...")

model = YOLO("yolo11n.pt")

print("Starting training...")

train\_results = model.train(

data=r"C:\\Users\\user\\Downloads\\archtech\\TumorDetection\\data.yaml",

epochs=20,

imgsz=640,

device="cpu")

print("Training finished!")

**Explanation:**

* This block does the same as the first, but it runs training on the **CPU** instead of the GPU.
* The print statements are added for clarity during execution.

**Code:**

from ultralytics import YOLO

model = YOLO("C:/Users/user/Downloads/archtech/runs/detect/train/weights/best.pt")

results = model("C:/Users/user/Downloads/archtech/test\_images/meningioma\_358.jpg", save=True)

results[0].show()

**Explanation:**

* After training, the best-performing model is loaded from the path runs/detect/train/weights/best.pt.
* It is then used to make a prediction on a single test image (meningioma\_358.jpg).
* The result is saved and displayed using the .show() method.

**Code :**

from ultralytics import YOLO

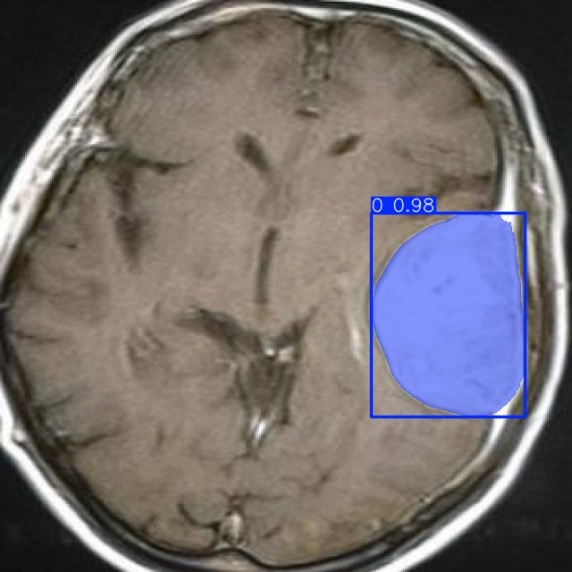
model = YOLO("runs/detect/train/weights/best.pt")

results = model("test\_images", save=True)

**Explanation:**

* The trained model is applied to an entire folder named test\_images.
* This allows batch processing of multiple images.
* The save=True option saves the predicted images with bounding boxes.





**Conclusion**

This project provided hands-on experience with rel-world computer vision tasks. YOLOv11 and SAM2 proved to be powerful tools for accurate tumor localization and segmentation. The project helped me understand the practical workflow of combining detection + segmentation in medical AI applications.

**TASK 02**

**Project Overview :**

This project is based on **Chapters 1 and 2** of *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow* by Aurélien Géron. The main objective of this task was to understand the foundational concepts of Machine Learning (ML), explore different types of learning algorithms, and apply those concepts in a practical end-to-end ML project using real-world data.

The project is divided into two key parts:

1. **Chapter 1: The Machine Learning Landscape**  
   This section introduced the fundamental concepts of Machine Learning, including the different types of learning (supervised, unsupervised, reinforcement), various learning styles (batch vs. online, instance-based vs. model-based), and common challenges like overfitting and underfitting. I studied key terminologies, problem types, and the structure of ML systems to develop a strong theoretical foundation.
2. **Chapter 2: End-to-End Machine Learning Project**  
   In this section, I implemented a complete ML pipeline using the **California Housing Dataset**. The project involved:
   * Loading and exploring the dataset
   * Creating income categories and applying **stratified sampling**
   * Preparing the data through **pipelines** (imputation + scaling)
   * Training and evaluating different models including **Linear Regression**, **Decision Tree**, and **Random Forest**
   * Using **GridSearchCV** for hyperparameter tuning
   * Evaluating the final model on a test set

The goal of this hands-on project was not just to train a model but to understand the **entire lifecycle** of a real ML project — from problem framing to data cleaning, model training, and evaluation.

By completing this task, I strengthened both my theoretical understanding and practical skills, laying the foundation for more advanced machine learning work in future chapters.

**Notes :**

**Chapter 1: The Machine Learning Landscape**

**Key Concepts & Terminology**

* **Machine Learning (ML):** Systems that learn from data and improve performance over time without explicit programming.
* **Dataset:** A collection of data, typically in rows (instances) and columns (features).
* **Labels:** Target values the model is trying to predict.
* **Feature:** A measurable property or characteristic of an instance.

**Types of ML**

* **Supervised Learning:** The algorithm is trained on labeled data. E.g., regression (predicting values) and classification (predicting classes).
* **Unsupervised Learning:** The model tries to learn patterns without labeled output. E.g., clustering, anomaly detection.
* **Semi-supervised Learning:** Mix of labeled and unlabeled data.
* **Reinforcement Learning:** Agents learn to make decisions through rewards and penalties.

**Learning Styles**

* **Batch Learning:** Model is trained on the full dataset.
* **Online Learning:** Model learns incrementally from incoming data.

**Challenges in ML**

* **Bad Data:** Insufficient, noisy, irrelevant, or inconsistent data.
* **Overfitting:** Model performs well on training data but poorly on unseen data.
* **Underfitting:** Model is too simple to capture the data's complexity.

**Hyperparameter Tuning**

* **Hyperparameters:** Parameters not learned directly from data (e.g., learning rate).
* **Model Selection:** Choosing the best model using tools like cross-validation.

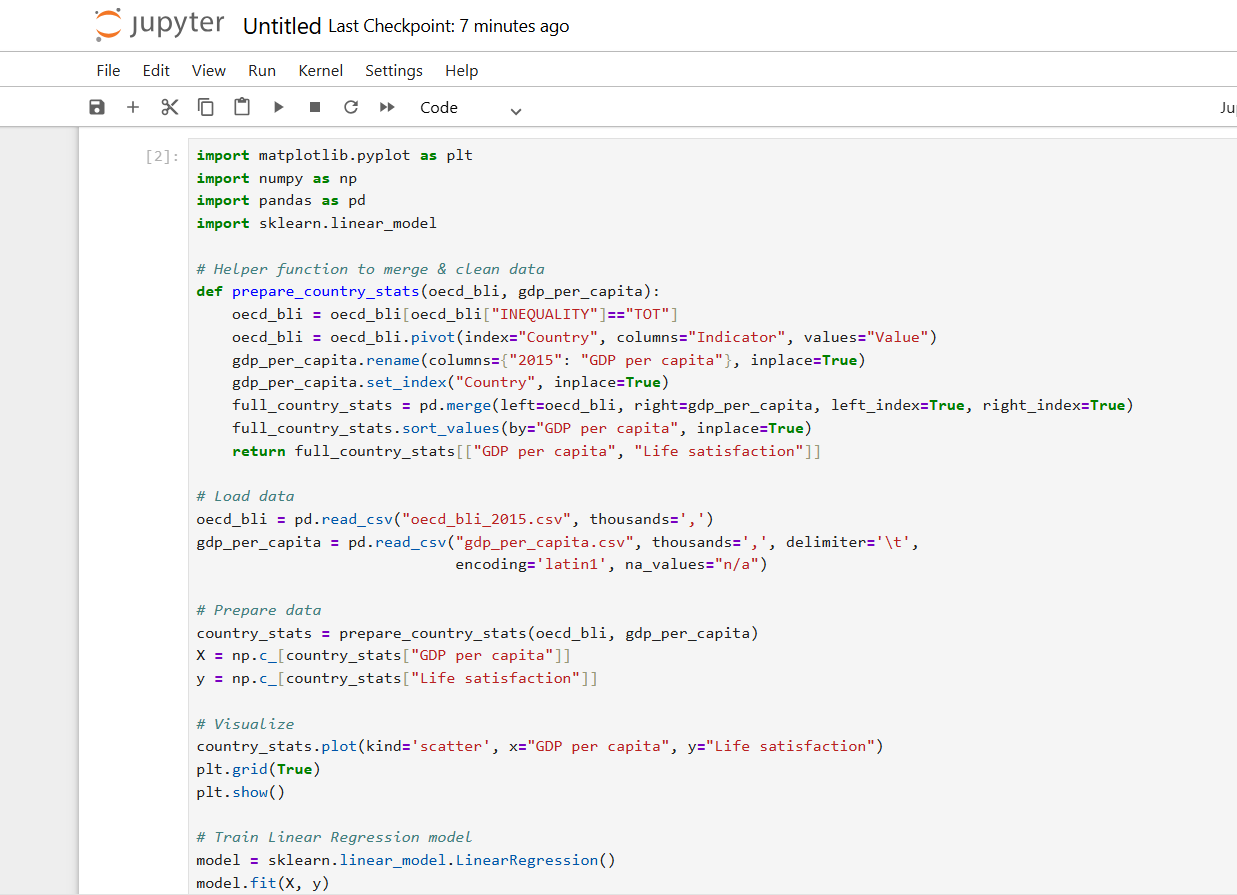
**Chapter 2: End-to-End Machine Learning Project**

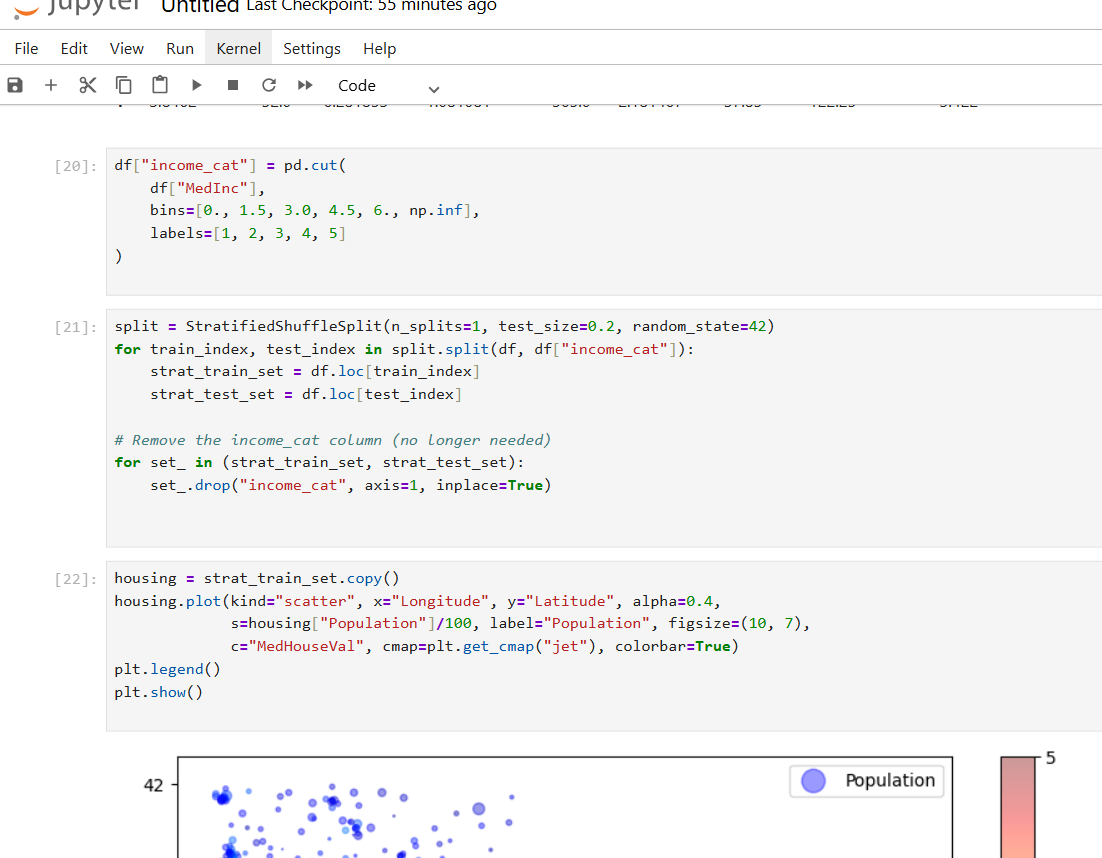
This chapter walks through a complete ML project using the **California housing dataset**.

**Steps in a Typical ML Project:**

1. **Look at the Big Picture:**
   * Define the problem (e.g., regression).
   * Select performance metrics (e.g., RMSE).
   * Identify constraints and goals.
2. **Get the Data:**
   * Automate fetching and loading.
   * Ensure reproducibility.
3. **Explore and Visualize the Data:**
   * Use scatterplots, histograms.
   * Look for correlations, outliers, missing data.
4. **Prepare the Data:**
   * Handle missing values (e.g., imputation).
   * Feature scaling (MinMaxScaler, StandardScaler).
   * Encoding categorical attributes (OrdinalEncoder, OneHotEncoder).
   * Build custom transformers and pipelines for preprocessing.
5. **Select and Train a Model:**
   * Try various algorithms like LinearRegression, DecisionTree, RandomForest.
   * Measure performance using cross-validation.
6. **Fine-Tune the Model:**
   * Use **GridSearchCV**: Exhaustive search over hyperparameters.
   * Use **RandomizedSearchCV**: Random sampling of hyperparameters for faster results.
   * Treat data preprocessing choices as hyperparameters.
7. **Present the Solution:**
   * Communicate insights to stakeholders.
   * Save and document the model (e.g., using joblib).
8. **Launch, Monitor, and Maintain:**
   * Deploy the model.
   * Monitor data drift and retrain when needed.

**Implementation of code with outputs ( Chap 2) :**

****

****

**Code:**

housing.plot(kind="scatter", x="Longitude", y="Latitude", alpha=0.4,

s=housing["Population"]/100, label="Population", figsize=(10, 7),

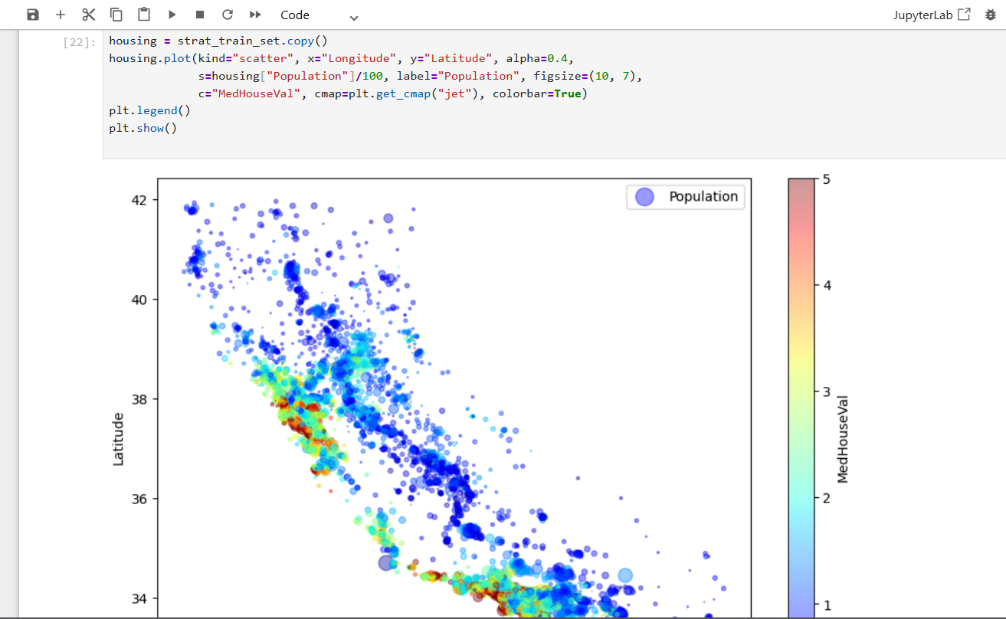
c="MedHouseVal", cmap=plt.get\_cmap("jet"), colorbar=True)

plt.legend()

plt.show()

**Explanation and output :**

* Creates a geographical scatter plot of the California housing data.
* x="Longitude", y="Latitude": Plots houses based on their geographic coordinates.
* alpha=0.4: Adds transparency to make dense regions visible.
* s=housing["Population"]/100: Sets marker size based on population (larger = more people).
* c="MedHouseVal": Colors each point based on median house value.
* cmap="jet": Applies a color gradient (blue → red).
* colorbar=True: Adds a legend showing the color scale for house values.
* Helps visualize housing prices across different locations, especially high-value areas along the coast.

****

**Code:**

from sklearn.datasets import fetch\_california\_housing

housing = fetch\_california\_housing(as\_frame=True)

df = housing.frame

df.head()

df.info()

df.describe()

df.hist(bins=50, figsize=(12, 8))

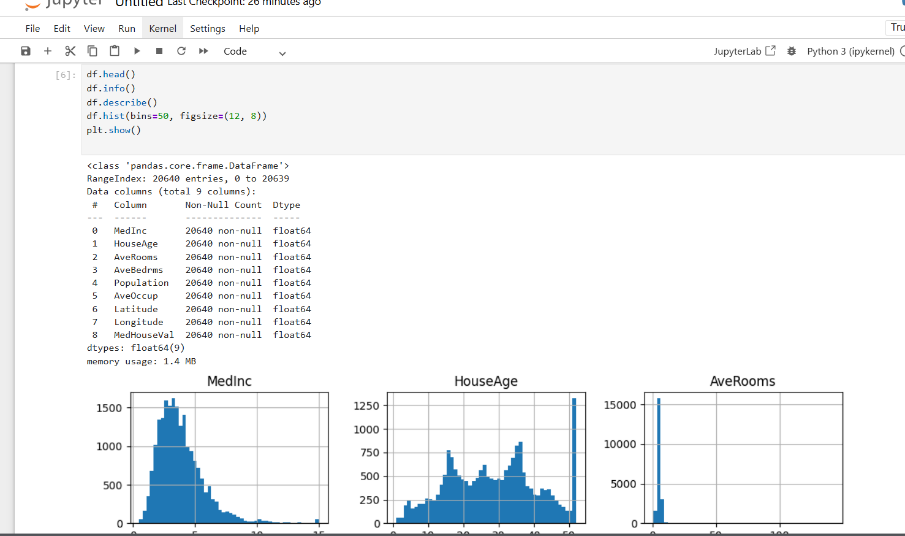
plt.show()

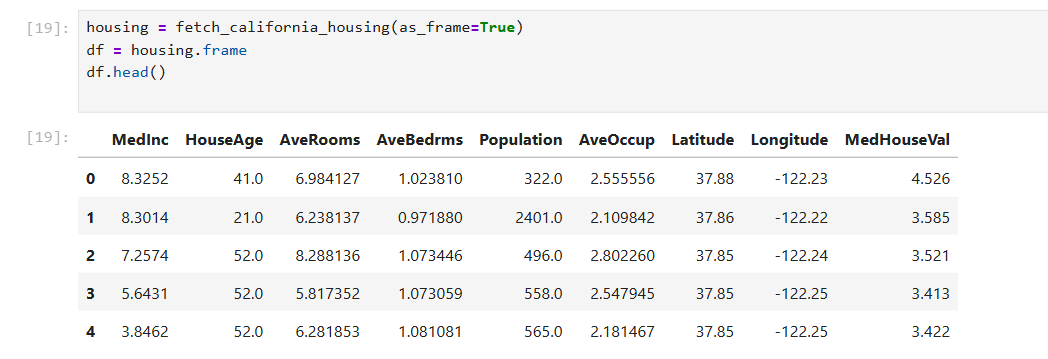
**Explanantion and output :**

 Loads the California housing dataset using Scikit-learn.

 as\_frame=True returns the data as a Pandas DataFrame.

 df contains all input features and the target column (MedHouseVal).

****



**Chapter 1 Excersice:**

**Q1) How would you define Machine Learning?**

Machine Learning is a branch of AI that allows systems to learn patterns from data and make decisions or predictions without being explicitly programmed for every specific task.

**2. Can you name four types of problems where it shines?**

* **Spam detection** – filters emails automatically
* **Image recognition** – identifies faces, objects, etc.
* **Recommendation systems** – suggests movies or products
* **Medical diagnosis** – predicts diseases from symptoms or images

**3. What is a labeled training set?**

It's a dataset that includes both **input features** (like income, age) and **correct output labels** (like whether the person bought a product). It’s used to train supervised models.

**4. What are the two most common supervised tasks?**

* **Classification** – predicting categories (e.g., spam or not spam)
* **Regression** – predicting continuous values (e.g., house price)

**5. Can you name four common unsupervised tasks?**

* **Clustering** – grouping similar data (e.g., customer segmentation)
* **Dimensionality reduction** – simplifying data (e.g., PCA)
* **Anomaly detection** – spotting unusual patterns (e.g., fraud)
* **Association rule learning** – finding rules (e.g., "people who buy X also buy Y")

**6. What type of algorithm would you use for a robot walking in unknown terrains?**

**Reinforcement Learning**, where the robot learns through trial and error using rewards (like staying balanced) and penalties (like falling).

**7. What type of algorithm would you use to segment customers into groups?**

**Clustering** (an unsupervised learning technique), like K-Means, to group similar customers based on features like age, purchase history, etc.

**8. Is spam detection supervised or unsupervised?**

**Supervised learning**, because the model learns from a labeled dataset where each email is marked as spam or not spam.

**9. What is an online learning system?**

It’s a system that **learns incrementally**, updating its model as new data comes in. Useful for live systems like stock predictions or news feeds.

**10. What is out-of-core learning?**

It’s used when the dataset is **too large to fit in memory**. The model processes small chunks of data at a time (e.g., using partial\_fit() in Scikit-learn).

**11. Which algorithm uses similarity measures to make predictions?**

**Instance-based learning**, like **K-Nearest Neighbors**, which compares new data to the most similar examples from the training set.

**12. Difference between model parameter and hyperparameter?**

* **Model parameter**: Learned from data (e.g., weights in linear regression).
* **Hyperparameter**: Set by the user (e.g., number of neighbors in KNN, learning rate).

**13. What do model-based algorithms search for, and how do they make predictions?**

They search for a mathematical model that **best fits the training data** (like minimizing error).  
Prediction is made by **applying this learned function** to new data.

**14. What are four main challenges in ML?**

* **Insufficient data** – not enough examples to learn from
* **Poor quality data** – noisy or incomplete
* **Overfitting** – too much focus on training data
* **Underfitting** – model too simple to capture patterns

**15. If a model performs well on training data but poorly on new data?**

That’s **overfitting**.  
**Solutions include**:

* Use simpler model
* Add more training data
* Use **regularization** techniques

**16. What is a test set and why use it?**

A test set is **completely unseen data** used to evaluate how well the final model generalizes to new data. It ensures an honest estimate of real-world performance.

**17. What is the purpose of a validation set?**

Used during model development to **tune hyperparameters** and compare models, without touching the test set.

**18. What can go wrong if you tune hyperparameters using the test set?**

You might overfit to the test set, making your evaluation **unrealistic**. The model would perform well only on that test set, not on new data.

**19. What is repeated cross-validation and why prefer it over a single validation set?**

It runs **multiple train-validation splits** and averages results. It gives **more reliable estimates** than a single validation set, especially when data is limited.

**Conclusion :**

This task helped me build a solid understanding of core Machine Learning concepts and workflows. I learned about different learning types, key challenges, and the ML pipeline. Through hands-on practice with the California Housing dataset, I applied preprocessing, model training, and hyperparameter tuning techniques. Overall, the task strengthened both my theoretical knowledge and practical skills in building end-to-end ML projects.

**THE END**