**Python**

**Question: 1**

1. Write an efficient algorithm that searches for a value target in an m x n integer matrix. This matrix has the following properties:
   1. Integers in each row are sorted from right to left.
   2. The first integer of each row is greater than the last integer of the previous row.

**Solution:** We can use a modified binary search algorithm to search for a value target in the given matrix efficiently. Here is the algorithm:

Step 1: Initialize variables rows and cols to store the matrix's number of rows and columns.

Step 2: Set left to 0 and right to rows\*cols-1 as the indices of the leftmost and rightmost elements in the matrix.

Step 3: While left<= right, do the following:

1. Calculate the middle index as mid= (left + right)/2.
2. Convert the middle index to the corresponding row and column indices using integer division and modulo operations:

row= mid/cols

col= mid%cols

1. Return True if matrix[row][col] equals the target.
2. If matrix[row][col] is less than the target, update left= mid+1 to search in the right half of the matrix.
3. If matrix[row][col] is greater than the target, update right= mid-1 to search in the left half of the matrix.

Step 4: Return False if the target is not found after the binary search.

Example:

# Set the left and right pointers for the binary search

left = 0

right = rows \* cols - 1

# Binary search loop

while left <= right:

# Calculate the midpoint

mid = (left + right) // 2

# Calculate the row and column of the midpoint

row = mid // cols

col = mid % cols

# Check if the midpoint value is equal to the target

if matrix[row][col] == target:

return True

# If the midpoint value is less than the target, update the left pointer

elif matrix[row][col] < target:

left = mid + 1

# If the midpoint value is greater than the target, update the right pointer

else:

right = mid - 1

# Return False if the target is not found

return False

**Question: 2**

**2.**  Write a program that takes a string as input and counts the frequency of each word in the string. There might be repeated characters in the string. Your task is to find the highest frequency and return the length of the highest-frequency word.

**Note -** You have to write at least 2 additional test cases in which your program will run successfully and provide an explanation for the same.

Example input - string = “write write write all the number from from from 1 to 100”

Example output - 5

Explanation - From the given string, we can note that the most frequent words are “write” and “from,” and the maximum value of both the values is “write,” and its corresponding length is 5

Solution: [Python Question 2ipynb - Colaboratory (google.com)](https://colab.research.google.com/drive/1NUtrJfj9tfXoQVbzNoRrGHFPH9sJGm6t#scrollTo=xWYfMf_bVFI9)

**Machine Learning**

**Question: 1**

1. Imagine you have a dataset where you have different Instagram features like u**sername , Caption, Hashtag, Followers, Time\_Since\_posted, and likes,** now your task is to predict the number of likes and Time Since posted, and the rest of the features are your input features. Now you have to build a model that can predict the number of likes and Time Since posted.

[Dataset](https://www.kaggle.com/datasets/rxsraghavagrawal/instagram-reach) This is the Dataset You can use this dataset for this question.

Solution: [Machine Learning Question No-1.ipynb](https://colab.research.google.com/drive/1hO2wz048LlEO8_yu4CIoPVlP3GzIlrYf#scrollTo=aIWFyp31eM7A)

**Question: 2**

2.

1. Explain how you can implement ML in a real-world application.
2. Train an SVM regressor on [the Bengaluru housing dataset](https://www.kaggle.com/datasets/amitabhajoy/bengaluru-house-price-data)

Must include in details:

- EDA

- Feature engineering

Solution:

Step 1: Exploratory Data Analysis (EDA):

* + EDA helps us understand the dataset, identify patterns, and uncover insights. Here’s what you can do:
    - [Load the dataset (you can find it on Kaggle1](https://www.kaggle.com/datasets/amitabhajoy/bengaluru-house-price-data)).
    - Explore the features (columns) and their data types.
    - Check for missing values and decide how to handle them (impute or drop).
    - Visualize distributions, correlations, and outliers.
    - Understand the target variable (house prices).

Step 2: Feature Engineering:

* + Feature engineering involves creating new features or transforming existing ones to improve model performance. Here are some steps:
    - Handle categorical features (like location, size, etc.) by encoding them (one-hot encoding or label encoding).
    - Extract relevant information from features (e.g., total square footage from size).
    - Create interaction features (e.g., multiplying bedrooms and bathrooms).
    - Normalize numerical features (scaling them to a similar range).
    - Consider adding features related to amenities, distance from landmarks, etc.

Step 3: Model Selection and Training:

* + We’ll use an SVM regressor (Support Vector Machine) for predicting house prices.
  + Split the dataset into training and testing sets.
  + Train the SVM model using features like location, size, bedrooms, bathrooms, etc., and the target variable (price).
  + Tune hyperparameters (kernel type, regularization, etc.) using cross-validation.

Step 4: Model Evaluation:

* + Evaluate the model’s performance using metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), or R-squared.
  + Compare the SVM model with other regression models (e.g., linear regression, random forest, etc.).

Step 5: Predictions:

* + Once the model is trained and evaluated, use it to predict house prices for new data.
  + Input the relevant features (location, size, etc.) and obtain predicted prices.

Code: [Machine Learning Question 2.ipynb](https://colab.research.google.com/drive/1kFBLNsXKvKSxtGMWx3dtkY6vAjEjdGc0#scrollTo=N5_y68kpkCGp)

**Deep Learning**

**Question: 1**

1. Train a Pure CNN with less than 10000 trainable parameters using the MNIST Dataset having minimum validation accuracy of 99.40%

Solution: [Deep Learning Question1.ipynb](https://colab.research.google.com/drive/16xXvr47a-XEe33qwemNdktM-EK1tHqEx#scrollTo=T7RmElntlqMR)

**Question: 2**

2.

Solution: Implementing deep learning (DL) in real-world applications involves creating models to learn from data and make predictions or decisions. Let’s focus on building an industry safety detection model to detect whether employees are wearing helmets. Here are the steps to achieve this:

1. Problem Definition:
   * Clearly define the problem: detecting helmet-wearing by employees in an industrial setting.
   * Specify the context (e.g., construction sites, manufacturing plants, etc.).
2. Data Collection:
   * Gather labeled data containing images of employees with and without helmets.
   * You can create your own dataset or use publicly available datasets.
3. Data Preprocessing:
   * Resize images to a consistent size (e.g., 224x224 pixels).
   * Normalize pixel values (usually in the range [0, 1]).
   * Split the dataset into training and validation sets.
4. Model Selection:
   * Choose an appropriate DL model architecture. For this task, Convolutional Neural Networks (CNNs) work well.
   * Popular architectures include YOLO (You Only Look Once), Faster R-CNN, or custom CNNs.
5. Model Training:
   * Train the selected model using the labeled dataset.
   * Use transfer learning if you have limited data. Fine-tune a pre-trained model (e.g., YOLOv7) on your helmet dataset.
   * Set up an appropriate loss function (e.g., binary cross-entropy) and optimizer (e.g., Adam).
6. Model Evaluation:
   * Evaluate the model on a separate validation set.
   * Metrics to consider: accuracy, precision, recall, F1-score, etc.
7. Deployment:
   * Deploy the trained model to an edge device (e.g., cameras, sensors) in the industrial environment.
   * Real-time inference: Capture live video frames and run the model to detect helmets.
8. Integration with Safety Systems:
   * Integrate the helmet detection system with existing safety protocols.
   * Set up alerts or notifications when an employee is detected without a helmet.
9. Monitoring and Maintenance:
   * Continuously monitor the model’s performance in real-world scenarios.
   * Retrain the model periodically with new data to adapt to changing conditions.