1. Explain the differences between AI, ML, Deep Learning (DL), and Data Science (DS)?
    - Artificial Intelligence (AI)= AI is the broadest concept—it refers to creating machines or systems that can perform tasks that normally require human intelligence.
    - Machine Learning (ML)= ML is a subset of AI that allows machines to learn from data and improve their performance without being explicitly programmed.

    - Deep Learning (DL)= : DL is a subset of ML that uses artificial neural networks with many layers (hence "deep") to model complex patterns.
    - Data Science (DS)= DS is a broader field that involves collecting, cleaning, analyzing, and interpreting data to extract insights and support decision-making.

2. What are the types of machine learning? Describe each with one real-world example?
    - Supervised Learning= The model is trained on a labeled dataset (i.e., input data + correct output).
    - Unsupervised Learning= The model is given unlabeled data (only inputs, no outputs) and must find hidden patterns or groupings

3. Define overfitting, underfitting, and the bias-variance tradeoff in machine leaning ?
    - Overfitting= When a model learns too much from the training data, including noise and random fluctuations, so it performs very well on training data but poorly on unseen/test data.
    - Underfitting= When a model is too simple and cannot capture the underlying patterns in the data. It performs poorly on both training and test data.

4. What are outliers in a dataset, and list three common techniques for handling them.
    - Outliers in a Dataset= Outliers are data points that differ significantly from most other observations in a dataset.

Three Common Techniques for Handling Outliers

1. Removal (Deleting Outliers)= If outliers are due to errors or irrelevant data, we can remove them.

2. Transformation (Reducing Impact)= Apply mathematical transformations (like log, square root, or Box-Cox) to reduce the effect of extreme values.

3. Imputation / Capping (Replacing Outliers)= Replace outliers with median, mean, or nearest acceptable values.

5. Explain the process of handling missing values and mention one imputation technique for numerical and one for categorical data?

Handling Missing Values in a Dataset

1. Identify Missing Data= Use functions like `isnull()` / `isna()` in Pandas to detect missing values.

2. Analyze Missingness= Check how much data is missing (small % or large %).

Imputation Techniques

Numerical Data – Mean/Median Imputation= Replace missing numerical values with the mean (average) or median (middle value) of the column.

6. Write a Python program that: ● Creates a synthetic imbalanced dataset with make_classification() from sklearn.datasets. ● Prints the class distributionfrom

```
sklearn.datasets import make_classification
 from collections import Counter


X, y = make_classification(n_samples=1000,      # total samples
                           n_features=10,       # number of features
                           n_informative=2,     # informative features
                           n_redundant=0,       # redundant features
                           n_clusters_per_class=1,
                           weights=[0.9, 0.1],  # imbalance ratio (90% vs 10%)
                           random_state=42)


print("Class distribution:", Counter(y))
```

7. Implement one-hot encoding using pandas for the following list of colors: ['Red', 'Green', 'Blue', 'Green', 'Red'].

```
import pandas as pd

colors = ['Red', 'Green', 'Blue', 'Green', 'Red']


df = pd.DataFrame({'Color': colors})


one_hot = pd.get_dummies(df['Color'])

print("Original DataFrame:")
print(df)
print("\nOne-Hot Encoded DataFrame:")
print(one_hot)

output=    Color
0    Red
1  Green
2   Blue
3  Green
4    Red
```

8. Write a Python script to: ● Generate 1000 samples from a normal distribution. ● Introduce 50 random missing values. ● Fill missing values with the column mean. ● Plot a histogram before and after imputation.

-
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Step 1: Generate 1000 samples from a normal distribution
np.random.seed(42)  # for reproducibility
data = np.random.normal(loc=50, scale=10, size=1000)  # mean=50, std=10

# Convert to DataFrame
df = pd.DataFrame(data, columns=['Value'])

# Step 2: Introduce 50 random missing values
missing_indices = np.random.choice(df.index, size=50, replace=False)
df.loc[missing_indices, 'Value'] = np.nan

# Step 3: Fill missing values with the column mean
df_filled = df.copy()
mean_value = df['Value'].mean()
df_filled['Value'].fillna(mean_value, inplace=True)

# Step 4: Plot histograms before and after imputation
plt.figure(figsize=(12,5))

# Histogram before imputation
plt.subplot(1,2,1)
plt.hist(df['Value'].dropna(), bins=30, edgecolor='black')
plt.title("Before Imputation")
plt.xlabel("Value")
plt.ylabel("Frequency")

# Histogram after imputation
plt.subplot(1,2,2)
plt.hist(df_filled['Value'], bins=30, edgecolor='black')
plt.title("After Imputation (Mean Filled)")
plt.xlabel("Value")
plt.ylabel("Frequency")

plt.tight_layout()
plt.show()
```

9. Implement Min-Max scaling on the following list of numbers [2, 5, 10, 15, 20] using sklearn.preprocessing.MinMaxScaler. Print the scaled array.

-
```
import numpy as np
from sklearn.preprocessing import MinMaxScaler

data = np.array([2, 5, 10, 15, 20]).reshape(-1, 1)

scaler = MinMaxScaler()

scaled_data = scaler.fit_transform(data)

# Print results
print("Original Data:\n", data.flatten())
print("Scaled Data:\n", scaled_data.flatten())

Output = Original Data:
 [ 2  5 10 15 20]

Scaled Data:
 [0.   0.158 0.421 0.684 1.   ]
```