MACHINE LEARNING ASSIGNMENT

Question 1: Explain the differences between AI, ML, Deep Learning (DL), and Data Science (DS).

Answer

- Artificial Intelligence (AI) is the broadest field. It's the concept of creating machines that can think and act like humans. This includes things like problem-solving, speech recognition, and decision-making. Think of it as the ultimate goal.
- Machine Learning (ML) is a subset of AI. It involves the use of algorithms that learn from data without being explicitly programmed. Instead of hard-coding rules, you give the model data and it learns the patterns itself. For example, a spam filter learns to identify spam by analyzing thousands of emails.
- Deep Learning (DL) is a more specialized subset of ML. It uses complex algorithms, specifically neural networks with multiple layers (hence "deep"), to find intricate patterns in data. DL is particularly effective for tasks like image and speech recognition, where the patterns are too complex for traditional ML.
- Data Science (DS) is a multidisciplinary field that uses scientific methods, processes, algorithms, and systems to extract knowledge and insights from structured and unstructured data. It combines elements of statistics, computer science, and domain expertise to solve real-world problems. While it uses ML and DL techniques, it also includes aspects like data cleaning, data visualization, and business intelligence.

Question 2: What are the types of machine learning? Describe each with one real-world example.

Answer

Machine Learning (ML) is a subset of Artificial Intelligence (AI) that enables systems to learn and improve from experience without being explicitly programmed. It uses algorithms to analyze data, identify patterns, and make predictions or decisions. ML is widely applied in areas like recommendation systems, fraud detection, autonomous vehicles, and medical diagnosis.

Types of Machine Learning

1.Supervised Learning

Supervised learning involves training a model on labeled data, where both input and output are known. The model learns to map inputs to the correct outputs and is commonly used for prediction and classification tasks.

Examples:

Predicting house prices (Regression).

Classifying emails as spam or not spam (Classification).

Algorithms:

Linear Regression, Logistic Regression, Decision Trees, Random Forest, and Support Vector Machines (SVM).

Applications:

Image classification, speech recognition, and fraud detection.

2.Unsupervised Learning

Unsupervised learning works with unlabeled data, aiming to uncover hidden patterns or groupings. It is often used for clustering and dimensionality reduction.

Examples:

Grouping customers based on purchasing behavior (Clustering).

Reducing data dimensions for visualization (Dimensionality Reduction).

Algorithms:

K-Means Clustering, DBSCAN, Principal Component Analysis (PCA).

Applications:

Customer segmentation, anomaly detection, and market basket analysis.

3.Reinforcement Learning

Reinforcement learning involves training an agent to make decisions by interacting with an environment. The agent learns through trial and error, receiving rewards or penalties based on its actions.

Examples:

Training AI to play chess or video games.

Optimizing robot movements.

Algorithms:

Q-Learning, SARSA, Deep Q-Networks (DQN).

Applications:

Autonomous vehicles, robotics, and game AI.

4.Semi-Supervised Learning

Semi-supervised learning combines labeled and unlabeled data. It is useful when labeled data is scarce or expensive to obtain.

Examples:

Language translation models using a mix of labeled and unlabeled sentence pairs.

Techniques:

Self-training, Co-training, and Graph-based methods.

Applications:

Medical imaging, speech recognition, and recommendation systems.

Question 3: Define overfitting, underfitting, and the bias-variance tradeoff in machine learning.

Answer

Overfitting:

This happens when a model learns the training data too well, including the noise and random fluctuations. While it performs great on the training data, it performs poorly on new, unseen data because it fails to generalize. It's like memorizing the answers to a test without understanding the concepts.

Underfitting:

This occurs when a model is too simple to capture the underlying patterns in the training data. It performs poorly on both the training and test data because it has not learned the fundamental relationship between the features and the target. It's like not studying enough for a test.

Bias-Variance Tradeoff:

This is a central concept in machine learning.

- Bias is the simplifying assumptions made by a model to make the target function easier to learn. High bias models are overly simple and tend to underfit.
- Variance is the sensitivity of the model to fluctuations in the training data. High variance models are complex and tend to overfit.
- The tradeoff is the balance between these two. A good model strikes a balance between bias and variance, avoiding both underfitting and overfitting to achieve optimal performance on new data.

Question 4: What are outliers in a dataset, and list three common techniques for handling them.

Answer

Outliers serve as captivating anomalies that frequently harbor profound insights within datasets. Despite appearing as erroneous data points, outliers possess the potential to offer valuable revelations about underlying processes or to reveal potential error in data collection.

Main Causes of Outliers

- Outliers can arise from various sources, making their detection vital:
- Data Entry Errors: Simple human errors in entering data can create extreme values.
- Measurement Error: Faulty device or experimental setup problems can cause abnormally high or low readings.
- Experimental Errors: Flaws in experimental design might produce facts factors that do not represent what they're presupposed to degree.
- Intentional Outliers: In some cases, data might be manipulated deliberately to produce outlier effects, often seen in fraud cases.
- Data Processing Errors: During the collection and processing stages, technical glitches can introduce erroneous data.
- Natural Variation: Inherent variability in the underlying data can also lead to outliers.

Three Common Techniques to Handle Outliers in ML

1.Statistical Methods (Detection + Removal)

- Use techniques like Z-score, IQR (Interquartile Range), or Mahalanobis distance to detect outliers.

- Remove or flag them before training.
- Best for models sensitive to noise, but risky if outliers carry important signal.

2. Robust Models

- Use algorithms that are less sensitive to outliers, such as:

  Decision Trees

  Random Forests

  Gradient Boosting
- These models split data based on thresholds rather than relying on distance or distribution.

3.Data Transformation

- Apply transformations like log, Box-Cox, or scaling to reduce the impact of extreme values.
- Helps normalize skewed distributions and stabilize variance.

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

Answer

Process of Handling Missing Values

1. Identify Missing Data

- Use tools like .isnull() or .info() in Python (e.g., with pandas) to detect missing entries.

2. Analyze the Pattern

- Determine if data is missing at random or follows a pattern.
- Check the percentage of missing values per column.

3. Decide on a Strategy

Options include:

- Deletion: Remove rows or columns with too many missing values.
- Imputation: Fill in missing values using statistical or model-based techniques.

- Flagging: Add a binary indicator column to mark missingness.

4. Apply the Chosen Technique

Implement the strategy using appropriate tools or libraries.

5.Validate

- Ensure the imputation or deletion hasn't distorted the data distribution or model performance.

Imputation Technique for Numerical Data

Mean Imputation Replace missing values with the mean of the non-missing values in that column. Example: If a column of ages has missing entries, fill them with the average age.

Imputation Technique for Categorical Data

Mode Imputation Replace missing values with the most frequent category (mode) in that column. Example: If a column for "City" has missing values, fill them with the city that appears most often.

Question 6: Write a Python program that:

● Creates a synthetic imbalanced dataset with make_classification() from sklearn.datasets.

● Prints the class distribution.

Answer

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

# Create synthetic imbalanced dataset
X, y = make_classification(n_samples=10000,       # total samples
                           n_features=20,        # number of features
                           n_informative=2,      # informative features
                           n_redundant=10,       # redundant features
                           n_clusters_per_class=1,
                           weights=[0.9, 0.1],    # imbalance ratio
                           flip_y=0,             # no label noise
                           random_state=42)

# Print class distribution
```

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

```
Class distribution: Counter({np.int64(0): 9000, np.int64(1): 1000})
```

Question 7. Implement one-hot encoding using pandas for the following list of colors:

['Red', 'Green', 'Blue', 'Green', 'Red']. Print the resulting dataframe.

Answer

```
import pandas as pd

# Original list of colors
colors = ['Red', 'Green', 'Blue', 'Green', 'Red']

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

# Apply one-hot encoding
one_hot_df = pd.get_dummies(df, columns=['Color'])

# Print the resulting DataFrame
print(one_hot_df)
```

```
   Color_Blue  Color_Green  Color_Red
0       False        False       True
1       False         True      False
2        True        False      False
3       False         True      False
4       False        False       True
```

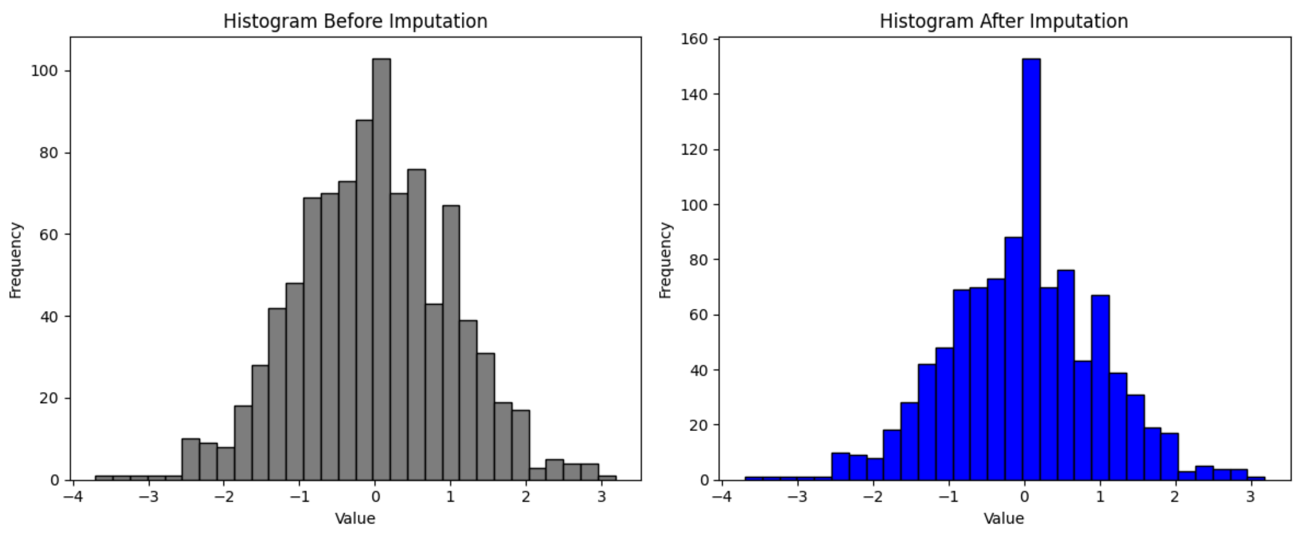
Question 8: Write a Python script to:

- 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.

Answer

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

data = np.random.normal(loc=0, scale=1, size=1000)
df = pd.DataFrame(data, columns=['value'])

missing_indices = np.random.choice(df.index, size=50, replace=False)
df.loc[missing_indices, 'value'] = np.nan

plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.hist(df['value'].dropna(), bins=30, color='grey', edgecolor='black')
plt.title('Histogram Before Imputation')
plt.xlabel('Value')
plt.ylabel('Frequency')

mean_value = df['value'].mean()
df_imputed = df.fillna(mean_value)

plt.subplot(1, 2, 2)
plt.hist(df_imputed['value'], bins=30, color='blue', edgecolor='black')
plt.title('Histogram After Imputation')
plt.xlabel('Value')
plt.ylabel('Frequency')

plt.tight_layout()
plt.show()

# Print a confirmation message
print("Histograms before and after imputation have been plotted.")
print(f"Mean value used for imputation: {mean_value:.4f}")
```



```
Histograms before and after imputation have been plotted.
Mean value used for imputation: -0.0214
```

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

Answer

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

# Original data
data = np.array([[2], [5], [10], [15], [20]])

# Initialize the scaler
scaler = MinMaxScaler()

# Fit and transform the data
scaled_data = scaler.fit_transform(data)
```

```
# Print the scaled array
print("Scaled array:\n", scaled_data)
```

```
Scaled array:
 [[0.        ]
 [0.16666667]
 [0.44444444]
 [0.72222222]
 [1.        ]]
```

Question 10: You are working as a data scientist for a retail company. You receive a customer transaction dataset that contains:

● Missing ages,

● Outliers in transaction amount,

● A highly imbalanced target (fraud vs. non-fraud),

● Categorical variables like payment method.

Explain the step-by-step data preparation plan you'd follow before training a machine learning model. Include how you'd address missing data, outliers, imbalance, and encoding.

Answer

Step-by-Step Data Preparation Plan

1. Understand the Data

- Perform exploratory data analysis (EDA) to understand distributions, missing values, outliers, and class imbalance.
- Use tools like df.info(), df.describe(), and visualizations (histograms, boxplots, bar charts).

2. Handle Missing Ages

Assess missingness: Is it random or systematic?

- Imputation strategy:

  If age is numerical, use:

  Mean/median imputation if distribution is normal/skewed.

  Model-based imputation (e.g., regression or KNN) if age correlates with other features.

- Optionally, add a missingness indicator column to preserve signal.

3. Treat Outliers in Transaction Amount

- Detect outliers using:

  IQR method: Values outside Q1 − 1.5×IQR or Q3 + 1.5×IQR.

  Z-score or visual inspection via boxplots.

- Handle outliers:

  Cap or clip extreme values (e.g., winsorization).

  Log transformation to reduce skew.

  Remove if clearly erroneous and not informative.

4. Address Class Imbalance (Fraud vs. Non-Fraud)

Check imbalance ratio (e.g., 95:5).

- Apply techniques like:
- Resampling:

  SMOTE (Synthetic Minority Over-sampling Technique)

  Random oversampling/undersampling

- Algorithmic solutions:

  Use models that support class weights (e.g., class_weight='balanced' in logistic regression or tree-based models).

- Evaluation metrics:

  Use precision, recall, F1-score, ROC-AUC instead of accuracy.

5. Encode Categorical Variables (e.g., Payment Method)

- Choose encoding based on cardinality:

Low cardinality (few unique values): Use One-Hot Encoding.

High cardinality: Use Target Encoding or Frequency Encoding.

- Avoid dummy variable trap by dropping one column if needed.

6. Feature Scaling

- Apply Min-Max Scaling or Standardization to numerical features, especially if using distance-based models (e.g., KNN, SVM).

7. Train-Test Split

- Split data into training and testing sets (e.g., 80/20) using train_test_split().
- Ensure stratification on the target variable to preserve class distribution.

8. Pipeline and Validation

- Build a data preprocessing pipeline using sklearn.pipeline.
- Use cross-validation to evaluate model performance robustly.