Assignment: Feature Engineering

Q.1 What is a parameter?

Ans: A **parameter** is a numerical value that describes a characteristic of an entire population or a model.

Examples of Parameters

1. **Population Mean (μ)** – the average of all values in a population
   Example: The average height of all students in a university.
2. **Population Variance (σ^2)** – how spread out the population data is.
3. **Population Standard Deviation (σ)** – the square root of variance of the population.

Real-Life Example

Suppose a school has 1000 students.

- ✓ The average marks of all 1000 students = **Parameter (μ)**
- ✓ If you take only 50 students and calculate the average, that value is a **Statistic**, not a parameter.

Q.2 What is correlation? What does negative correlation mean?

Ans: Correlation is a statistical measure that shows the relationship between two variables — how they move together.

It tells us:

- ✓ Whether two variables are related
- ✓ The direction of the relationship
- ✓ The strength of the relationship

What Does Negative Correlation Mean?

Negative correlation means that two variables move in opposite directions.

- ✓ If one variable increases, the other decreases
- ✓ If one decreases, the other increases

Example of Negative Correlation

- ✓ Speed of a car and time to reach a destination
  (Higher speed → Less time)
- ✓ Price and demand
  (Higher price → Lower demand)

Numerical Understanding

- ✓ **r = -1** → Perfect negative correlation (strongest opposite relationship)
- ✓ **r = -0.5** → Moderate negative correlation
- ✓ **r = 0** → No correlation

Q3. Define Machine Learning. What are the main components in Machine Learning?

Ans: Machine Learning (ML) is a branch of Artificial Intelligence (AI) that enables computers to learn from data and make predictions or decisions without being explicitly programmed for every task.

Main Components of Machine Learning

1.Data

Data is the foundation of Machine Learning.

- ✓ Training Data → Used to train the model
- ✓ Testing Data → Used to evaluate performance

Example:
If building a house price prediction model, data may include:

- ✓ Area
- ✓ Location
- ✓ Number of rooms
- ✓ Price

2. Features (Input Variables)

Features are the measurable properties or characteristics of the data.

Example:
For predicting student marks:

- ✓ Study hours
- ✓ Attendance
- ✓ Previous grades

Good features → Better model performance.

3.Model

A model is the mathematical representation that learns patterns from data.

Examples:

- ✓ Linear Regression
- ✓ Decision Tree
- ✓ Neural Network

The model finds relationships between inputs and outputs.

4.Algorithm

An algorithm is the procedure used to train the model.

Example:

- ✓ Gradient Descent
- ✓ K-Nearest Neighbors
- ✓ Support Vector Machine

The algorithm adjusts the model parameters to minimize error.

5.Loss Function (Error Function)

Measures how wrong the model's predictions are.

Example:

- ✓ Mean Squared Error (MSE)
- ✓ Cross-Entropy Loss

Lower loss → Better model.

6.Training

The process of feeding data into the model and adjusting parameters to improve performance.

7. Evaluation / Testing

After training, we check model performance using:

- ✓ Accuracy
- ✓ Precision
- ✓ Recall
- ✓ F1-score
- ✓ RMSE

Q4. How does loss value help in determining whether the model is good or not

Ans: The **loss value** measures how far the model's predictions are from the actual (true) values. It tells us how much error the model is making.

1.Basic Idea

- ✓ **Low loss** → Predictions are close to actual values → Good model
- ✓ **High loss** → Predictions are far from actual values → Poor model

So, loss is a direct indicator of model performance during training.

2. How Loss Works in Practice

When a model makes predictions, the loss function compares:

Actual Value−Predicted Value\text{Actual Value} - \text{Predicted Value}Actual Value−Predicted Value

and calculates the error.

Example:

Actual	Predicted	Loss (Error)
100	98	Small loss
100	60	Large loss

Smaller loss means the model is learning correctly.

3.Role of Loss During Training

During training:

- ✓ The model makes predictions
- ✓ Loss is calculated
- ✓ The algorithm adjusts parameters (weights)

- ✓ Loss gradually decreases

This process continues until the model reaches minimum loss.

If loss decreases over epochs → Model is improving.

4. Training Loss vs Validation Loss (Very Important)

To judge if a model is truly good, we compare:

- ✓ **Training Loss** → Error on training data
- ✓ **Validation Loss** → Error on unseen data

Cases:

- ✓ Low training loss + Low validation loss → Good model (well learned)
- ✓ Low training loss + High validation loss → Overfitting

Q5. What are continuous and categorical variables?

Ans: Variables are mainly classified into **continuous** and **categorical** variables based on the type of values they can take.

1. Continuous Variables: A continuous variable **is a variable that can take any numerical value within a given range (including decimals).**

Key Features

- ✓ Numeric values
- ✓ Infinite possible values within an interval
- ✓ Measured, not counted
- ✓ Can include fractions and decimals

Examples

- ✓ Height (170.2 cm, 165.5 cm)
- ✓ Weight (60.3 kg, 72.8 kg)
- ✓ Temperature (36.6°C, 98.4°F)
- ✓ Time (2.5 hours, 3.75 hours)
- ✓ Age (21.5 years)

Example:
If you measure a student’s height, it can be 160 cm, 160.5 cm, 160.55 cm, etc.
So it is continuous.

2. Categorical Variables: A categorical variable **represents data that can be divided into groups or categories.**

Key Features

- ✓ Non-numeric labels (or numeric codes representing groups)
- ✓ Finite number of categories
- ✓ Counted, not measured
- ✓ Cannot have meaningful decimal values

Examples

- ✓ Gender (Male, Female)
- ✓ Color (Red, Blue, Green)
- ✓ Blood Group (A, B, AB, O)
- ✓ Department (HR, IT, Finance)
- ✓ Pass/Fail (Yes/No)

Example:
A student's gender can only be Male or Female (or other categories).
You cannot have 1.5 Male → so it is categorical.

Q6. How do we handle categorical variables in Machine Learning? What are the common techniques?

Ans: In Machine Learning, most algorithms work with **numerical data**, so categorical variables (like Gender, City, Color) must be converted into numbers before training a model. This process is called **encoding categorical variables**.

Common Techniques to Handle Categorical Variables in Machine Learning

1. Label Encoding

Label Encoding converts each category into a unique integer.

Example

Color	Label Encoded
Red	0
Blue	1

Color	Label Encoded
Green	2

When to Use

- ✓ Ordinal data (with order)
- ✓ Tree-based models (Decision Tree, Random Forest)

Limitation

It may create a false order (e.g., Blue < Green < Red), which can mislead some models like Linear Regression.

2. One-Hot Encoding (Most Common)

One-Hot Encoding creates separate binary columns for each category.

Example

Color	Red	Blue	Green
Red	1	0	0
Blue	0	1	0
Green	0	0	1

When to Use

- ✓ Nominal data (no order)
- ✓ Linear models, Logistic Regression, Neural Networks

Python Example

Advantage

- ✓ No false ranking between categories
- ✓ Works well for most ML models

3. Ordinal Encoding

Used when categories have a meaningful order.

Example

Education	Encoded

High School	0
Graduate	1
Postgraduate	2

When to Use

- ✓ Ordered categorical data
- ✓ Ranking-based features

4. Binary Encoding

Converts categories into binary numbers and then splits them into binary columns.

Example

Category → Number → Binary Columns

When to Use

- ✓ High-cardinality features (many unique categories)
- ✓ To reduce dimensionality compared to one-hot encoding

5. Target (Mean) Encoding

Each category is replaced with the mean of the target variable.

Example

If target = salary:

City	Avg Salary	Encoded Value
Delhi	50k	50000
Mumbai	60k	60000

When to Use

- ✓ Large datasets
- ✓ High-cardinality categorical variables

Caution

Can cause data leakage if not applied carefully.

6. Frequency Encoding

Categories are replaced by their frequency (count or proportion).

Example

City	Frequency
Delhi	0.40
Mumbai	0.35
Pune	0.25

When to Use

- ✓ High-cardinality features
- ✓ Tree-based model

Q7. What do you mean by training and testing a dataset?

Ans: In Machine Learning, training and testing a dataset refer to how we teach a model and then evaluate how well it has learned.

a. Training Dataset

The training dataset is the portion of data used to teach the model.

What happens during training?

- ✓ The model sees input data (features).
- ✓ It compares predictions with actual values (labels).
- ✓ It calculates loss (error)**.**
- ✓ It updates its parameters (weights) to reduce that error.

Think of training like studying for an exam.

Example

Suppose we want to predict house prices:

- ✓ Inputs: size, number of rooms, location
- ✓ Output: price

The model learns the relationship between inputs and price using training data.

b. Testing Dataset

The testing dataset is used to evaluate the model after training.

- ✓ The model has never seen this data before**.**
- ✓ We check how well it predicts new, unseen data.
- ✓ This tells us if the model generalizes well.

Q8. What is sklearn.preprocessing?

Ans: sklearn.preprocessing is a module in the scikit-learn library that provides tools to prepare and transform data before feeding it into a Machine Learning model.

In simple words:

It helps convert raw data into a clean, numerical, and scaled format that ML models can understand.

Most Machine Learning algorithms:

- ✓ Cannot handle categorical text data directly
- ✓ Are sensitive to feature scales (e.g., age vs salary)
- ✓ Require normalized or standardized inputs

Q9. What is a Test set?

Ans: A **test set** is a portion of the dataset that is kept separate from the training data and used to evaluate the final performance of a machine learning model on unseen data**.**

Q10. How do we split data for model fitting (training and testing) in Python? How do you approach a Machine Learning problem?

Ans:

How to Split Data for Model Fitting (Training & Testing) in Python

In Python, we usually split data using the `train_test_split()` function from **scikit-learn**.

Syntax:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

Meaning of Parameters

- ✓ X → Features (input variables)
- ✓ y → Target (output variable)
- ✓ test_size=0.2 → 20% data for testing
- ✓ random_state=42 → Ensures reproducible results

How to Approach a Machine Learning Problem

A structured approach helps build accurate and reliable ML models. Steps are given below.

Step 1: Define the Problem

Step 2: Collect and Understand Data

Step 3: Data Preprocessing

Step 4: Exploratory Data Analysis (EDA)

Step 5: Split the Data (Train/Test)

Step 6: Choose and Train the Model

Step 7: Evaluate the Model

Step 8: Hyperparameter Tuning (Improve Model)

Step 9: Deployment (Real-World Use)

Q11. Why do we have to perform EDA before fitting a model to the data?

Ans:

EDA (Exploratory Data Analysis) is performed before fitting a model because it helps us **understand the data deeply** and avoid serious modeling mistakes.

If you skip EDA, you are basically training a model blindly.

Why EDA is Important Before Model Fitting

1. Understand the Structure of Data

EDA helps answer:

- ✓ How many rows and columns?
- ✓ What are the data types?
- ✓ Which column is the target variable?
- ✓ Are variables numerical or categorical?

Without this, you cannot even choose the correct model.

2. Detect Missing Values

Missing values can break many ML algorithms.

Example:

```
df.isnull().sum()
```

If you don't handle missing data:

- ✓ Model may crash
- ✓ Model may learn wrong patterns

3. Identify Outliers

Outliers can distort models like:

- ✓ Linear Regression
- ✓ KNN
- ✓ SVM

EDA tools:

- ✓ Boxplots
- ✓ Histograms
- ✓ Scatterplots

If outliers are not handled:

- ✓ Model performance decreases
- ✓ Coefficients become unstable

4.Understand Data Distribution

Is the data:

- ✓ Normally distributed?
- ✓ Skewed?
- ✓ Highly imbalanced?

Example:
If your target variable is 95% "No" and 5% "Yes", accuracy alone will be misleading.

EDA helps detect class imbalance.

5. Check Relationships Between Variables

EDA helps answer:

- ✓ Which features strongly affect the target?
- ✓ Are some features irrelevant?
- ✓ Are two features highly correlated (multicollinearity)?

Example:
Correlation heatmap helps identify redundant variables.

6 Feature Engineering Insights

EDA may reveal:

- ✓ Need for scaling
- ✓ Need for encoding
- ✓ Need for transformation (log, square root)
- ✓ Need to create new features

Without EDA, you might miss important transformations.

7 Helps Choose the Right Model

After EDA, you can decide:

- ✓ Regression or classification?
- ✓ Linear model or tree-based?
- ✓ Need regularization or not?

Q12. What is correlation?

Ans:

Correlation is a statistical measure that shows the strength and direction of the relationship between two variables**.**

In simple words:

Correlation tells us how one variable changes when another variable changes.

Example to Understand Correlation

Suppose:

- ✓ Hours studied ↑ → Marks ↑
  This means both variables move in the same direction → Positive correlation.

Another case:

- ✓ Price ↑ → Demand ↓
  This means variables move in opposite directions → Negative correlation.

Q13. What does negative correlation mean?

Ans:

Negative Correlation (−)

When one variable increases and the other decreases.

Example

- ✓ Price and Demand
- ✓ Speed and Travel Time (for fixed distance)

If X increases and Y decreases → Negative correlation.

Q14. How can you find correlation between variables in Python?

Ans:

You can find correlation between variables in Python mainly using **Pandas**, NumPy, or visualization tools like heatmaps. The most common and simple method is using `pandas.DataFrame.corr()`.

Python example.

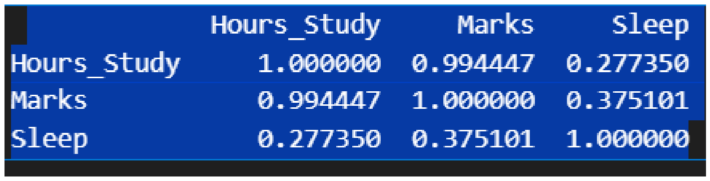
```
import pandas as pd

# Create a dataset
data = {
'Hours_Study': [1, 2, 3, 4, 5],
'Marks': [30, 40, 50, 65, 80],
'Sleep': [7, 6, 5, 6, 8]
}
```

```
df = pd.DataFrame(data)

# Find correlation matrix
corr_matrix = df.corr()
print(corr_matrix)
```

output:



```
             Hours_Study     Marks     Sleep
Hours_Study     1.000000  0.994447  0.277350
Marks           0.994447  1.000000  0.375101
Sleep           0.277350  0.375101  1.000000
```

Q15. What is causation? Explain difference between correlation and causation with an example.

Ans:

Causation means that **one variable directly causes a change in another variable**.

Difference Between Correlation and Causation

Feature	Correlation	Causation
Meaning	Two variables move together	One variable directly causes the other
Relationship	Association only	Cause-and-effect
Direction	May or may not imply cause	Clear cause → effect
Example	Height and weight	Exercise reduces body fat
Symbol	Measured by correlation coefficient (r)	Proven by experiments/research

Simple Example to Understand Clearly

Example 1: Correlation (No Causation)

- ✓ Number of umbrellas sold ↑
- ✓ Rainfall ↑

They are correlated because both increase together.
But umbrellas do NOT cause rain.

Example 2: True Causation

- ✓ More study time → Higher exam marks
- ✓ Here:
- ✓ Cause = Study time
- ✓ Effect = Marks
  So this is a causal relationship.

Q16. What is an Optimizer? What are different types of optimizers? Explain each with an example.

Ans:

An **optimizer** is an algorithm used in Machine Learning and Deep Learning to **adjust the model's parameters (weights and biases) in order to minimize the loss function**.

Different Types of Optimizers (With Explanation & Examples)

1. Gradient Descent (Batch Gradient Descent)

Updates weights using the **entire dataset** at once.

Features

- ✓ Stable updates
- ✓ Slow for large datasets
- ✓ High computational cost

Example (Concept)

If dataset = 10,000 samples
Gradient is calculated using all 10,000 samples before updating weights.

2. Stochastic Gradient Descent (SGD)

Updates weights using **one data point at a time**.

Features

- ✓ Faster updates
- ✓ Noisy but efficient

- ✓ Works well for large datasets

Example

Instead of using 10,000 samples, it updates weights after each single sample.

3. Mini-Batch Gradient Descent (Most Practical)

Updates weights using a small batch of data (e.g., 32 or 64 samples).

Features

- ✓ Faster than batch GD
- ✓ More stable than SGD
- ✓ Widely used in real-world ML

Example

Dataset = 10,000 samples
Batch size = 32
Updates happen every 32 samples.

4. Adam (Adaptive Moment Estimation) – Most Popular

Adam combines:

- ✓ Momentum
- ✓ RMSProp
  and adapts the learning rate automatically.

Features

- ✓ Fast convergence
- ✓ Works well for deep learning
- ✓ Handles noisy data efficiently

Example

Used in Neural Networks and NLP models.

5. RMSProp (Root Mean Square Propagation)

Adjusts the learning rate for each parameter using moving averages of squared gradients.

Features

- ✓ Good for non-stationary data
- ✓ Works well in RNNs and deep learning

Example

Used in time-series and deep neural networks.

```
optimizer = optim.RMSprop(model.parameters(), lr=0.001)
```

6. Adagrad (Adaptive Gradient)

Adapts learning rate based on frequency of parameter updates.

Features

- ✓ Good for sparse data
- ✓ Learning rate decreases over time

Example

Useful in NLP problems with sparse features.

7. Adadelta

Improved version of Adagrad that prevents learning rate from becoming too small.

Features

- ✓ No need to set learning rate manually (sometimes)
- ✓ More robust than Adagrad

Q.17 What is sklearn.linear_model ?

Ans:

`sklearn.linear_model` provides algorithms that model the relationship between input features and target using linear equations.

A linear model assumes a relationship like:

y=w1x1+w2x2+⋯+by = w_1x_1 + w_2x_2 + \dots + by=w1x1+w2x2+⋯+b

Where:

- ✓ xxx = input features
- ✓ www = weights (coefficients)
- ✓ bbb = bias (intercept)
- ✓ yyy = predicted output

Q18. **What does model.fit() do? What arguments must be given?**

Ans:

`model.fit()` teaches the model by learning the relationship between input features (X) and target values (y).

What Does model.fit() Actually Do Internally?

When you call:

```
model.fit(X, y)
```

the model:

1. Takes input data (features) `X`
2. Takes actual output (labels) `y`
3. Calculates predictions
4. Computes error (loss)
5. Updates parameters using an optimization algorithm
6. Repeats until the model learns the pattern

Required Arguments of `model.fit()`

1. X (Features) — Mandatory

- ✓ Input variables (independent variables)
- ✓ Usually a 2D array or DataFrame
- ✓ Shape: `(n_samples, n_features)`

Example:

```
X = [[1], [2], [3], [4]]
```

2. y (Target/Labels) — Mandatory (for Supervised Learning)

- ✓ Output variable the model should predict
- ✓ Shape: `(n_samples,)`

Example:

```
y = [2, 4, 6, 8]
```

Full Example

```
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X, y)
```

Q19. What does model.predict() do? What arguments must be given?

Ans:

model.predict() is a method used to make predictions using a trained machine learning model.

What does `model.predict()` do.

After you train a model using model.fit(), the model learns patterns from the training data. Then, model.predict() uses those learned patterns to:

- ✓ Predict labels (classification)
- ✓ Predict values (regression)
- ✓ Predict clusters (clustering models)

X (Input Features) — Mandatory

This is the only required argument.

Requirements:

- ✓ Must be a 2D array-like structure (list, NumPy array, or DataFrame)
- ✓ Shape: `(n_samples, n_features)`
- ✓ Must have the same number of features used during training

Example:

X_new = [[6], [7]]

Q20. What are continuous and categorical variables?

Ans:

1. Continuous Variables

A continuous variable is a numerical variable that can take any value within a given range**,** including decimals.

These variables are measured, not counted.

Key Characteristics

- ✓ Numeric values
- ✓ Can have decimal points
- ✓ Infinite possible values between two numbers
- ✓ Used in regression problems

Examples of Continuous Variables

- ✓ Height (e.g., 165.5 cm, 170.2 cm)
- ✓ Weight (e.g., 60.3 kg, 72.8 kg)
- ✓ Temperature (e.g., 36.6°C)
- ✓ Salary (e.g., ₹25,500.75)
- ✓ Time (e.g., 2.35 hours)

2. Categorical Variables

A categorical variable represents data that can be divided into groups or categories instead of numeric measurements.

These variables describe qualities or labels.

Key Characteristics

- ✓ Non-numeric labels (or encoded numbers)
- ✓ Finite number of categories
- ✓ Used in classification problems

Examples of Categorical Variables

- ✓ Gender (Male, Female)
- ✓ Color (Red, Blue, Green)
- ✓ City (Delhi, Mumbai, Bhubaneswar)
- ✓ Yes/No responses

- ✓ Education level (High School, Graduate, Postgraduate)

Q21. What is feature scaling? How does it help in Machine Learning?

Ans:

Feature scaling is the process of transforming numerical features so they are on a **similar scale or range**.

Feature scaling ensures that no single feature dominates others just because of its larger numeric value.

How Feature Scaling Helps in Machine Learning

1.Improves Model Performance

Algorithms like:

- ✓ K-Nearest Neighbors (KNN)
- ✓ Support Vector Machine (SVM)
- ✓ Logistic Regression
- ✓ Neural Networks

are distance-based or gradient-based.
If features are not scaled, results may be biased.

2. Faster Convergence

In gradient-based models (like Linear Regression, Neural Networks):

- ✓ Scaling helps the optimizer reach the minimum loss faster.
- ✓ Training becomes more stable.

3. Prevents Feature Dominance

Without scaling:

- ✓ A feature with large values can dominate the learning process.
- ✓ Small-range features may become irrelevant.

4.Required for Distance-Based Algorithms

Distance-based models calculate distance using formulas like:

Distance=(x1−y1)2+(x2−y2)2Distance = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}Distance=(x1−y1)2+(x2−y2)2

If one feature has large magnitude, distance calculation becomes biased.

Q22. How do we perform scaling in Python?

Ans:

Scaling in Python means transforming numerical data so that features are on a similar scale. This is very important in machine learning, statistics, and data analysis because many algorithms (like KNN, SVM, Gradient Descent) are sensitive to feature magnitudes.

Q23. **What is sklearn.preprocessing?**

Ans:

sklearn.preprocessing is a module in the scikit-learn library that is used to transform and prepare data before applying machine learning models.

It provides tools for:

- ✓ Scaling features
- ✓ Normalizing data
- ✓ Encoding categorical variables
- ✓ Handling missing values
- ✓ Generating polynomial features

In simple terms, it helps clean and convert raw data into a format that machine learning algorithms can understand.

Q24. How do we split data for model fitting (training and testing) in Python?

Ans:

To split data for training and testing in Python, we usually use the `train_test_split()` function from the scikit-learn library.

This is an essential step before model fitting because:

- ✓ **Training set** → Used to train the model
- ✓ **Testing set** → Used to evaluate model performance

Basic Syntax:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    random_state=42
)
```

Parameters:

- ✓ `X` → Features (independent variables)
- ✓ `y` → Target (dependent variable)
- ✓ `test_size=0.2` → 20% data for testing, 80% for training
- ✓ `random_state=42` → Ensures reproducibility

Q25. Explain data encoding?

Ans:

Data encoding is the process of converting categorical (text) data into numerical format so that machine learning models can understand and process it.

Most algorithms in machine learning (like regression, SVM, KNN) only work with numbers, not text. So categories like "Male", "Female", "Red", "Blue" must be converted into numeric values.

Why is Data Encoding Important?

Suppose you have this dataset:

Color	Price
Red	100
Blue	150
Green	120

A machine learning model cannot understand text values like “Red” or “Blue”, so we encode them into numbers before training.