

Feature Engineering Answers

1. What is a parameter?

- A parameter is a value that you pass into a function, method, or procedure to customize its behavior or provide input it needs to operate. parameters act like placeholders for actual values, called arguments, that are supplied when the function is called.

2. What is correlation?

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

What does negative correlation mean?

- Negative correlation means that as one variable increases, the other

decreases they move in opposite directions.

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

- Machine Learning (ML) is a field of artificial intelligence (AI) that enables computers to learn from data and make decisions or predictions without being explicitly programmed. Instead of following hardcoded rules, ML systems improve their performance as they are exposed to more data over time.

Main Components of Machine Learning:

1.Data:i)The foundation of any ML system.

- ii) Includes inputs used to train and test the model (e.g., images, text, numbers, sensor data).
- iii) Good quality and relevant data are essential.

2.Features:i) Specific measurable properties or characteristics of the data.

- ii) For example, in house price prediction, features might include square footage, number of bedrooms, and location.

3.Model:i) A mathematical structure that makes predictions or decisions based on the data.

- ii) Learns patterns from the training data.
- iii) Examples include decision trees, neural networks, or support vector machines.

4.Algorithm:i)The method used to train the model.

ii)It adjusts the model's internal parameters to reduce prediction error.

iii)Examples: Linear regression, gradient descent, k-nearest neighbors.

5.Training:i)The process where the model learns patterns from a labeled dataset.

ii)The model adjusts itself to reduce the difference between

its

predictions and the actual results.

6.Evaluation:i)Testing the trained model on new, unseen data to measure its performance.

ii) Uses metrics like accuracy, precision, recall, or mean squared error.

7. Inference: i) After training and evaluation, the model is used to make predictions on new data.

8. Improvement: i) In some systems (like reinforcement learning), the model continues to improve based on feedback from its environment.

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

- The loss value plays a critical role in evaluating how well a machine learning model is performing. It

quantifies the difference between the model's predicted output and the actual output (true labels).

What is loss?

It is a numerical value that represents the model's error on a single prediction or a batch of predictions. The goal during training

is to minimize this loss the smaller the loss, the better the model's predictions.

How Loss Helps Determine Model Quality:

1. Training Guidance:-

i) During training, the model uses the loss value to update its parameters (weights and biases).

ii) Algorithms like gradient descent use the loss value to decide the direction and size of parameter updates.

2.Performance Indicator:-

i) A low loss usually indicates that the model's predictions are close to the actual values.

ii) A high loss suggests the model is making poor predictions and needs further training or adjustment.

3.Comparison tool:i) When evaluating multiple models or configurations (e.g., different architectures or hyperparameters), comparing their loss values helps identify which one performs best.

4. Monitoring Overfitting:

- i) If training loss is low but validation loss is high → Overfitting.
- ii) If both losses are high → Underfitting.
- iii) Monitoring the loss during training and validation helps detect these issues early.

5. What are continuous and categorical variables?

- These are two main types of variables used in data analysis and machine learning.

Continuous variables:

A continuous variable can take an infinite number of values within a given range.

Characteristics:

- i) Measurable

- ii) Can have decimal values

iii) Often used in regression problems.

Ex:- Height (e.g., 172.5 cm)

Categorical variables:

A categorical variable represents data

that can be divided into distinct groups or categories.

Characteristics: i) Qualitative (descriptive)

ii) Usually represented as labels or categories

iii) Can be either: Nominal (no natural order, e.g., colors: red, blue, green)

Ordinal (with a meaningful order, e.g., low, medium, high)

Ex:- Gender (male, female)

6. How do we handle categorical variables in Machine Learning?

What

are the common techniques?

- Handling categorical variables properly is essential in machine learning, since most ML algorithms require numerical input. Here are the most common techniques to convert categorical variables into a form that models can understand:

1.Label Encoding:

What it does: Assigns a unique integer to each category.

Example:Color: [Red, Green, Blue] → [0, 1, 2]

Use When: The variable is ordinal (i.e., categories have a meaningful order).

Warning: Not ideal for **nominal** (unordered) categories, as models may assume a ranking.

2.One-Hot Encoding:

What it does: Creates binary columns for each category (1 if the category is present, 0 otherwise).

Example:Color: [Red, Green, Blue] →

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

Use When: The variable is nominal (unordered).

Libraries like pandas

(`pd.get_dummies`) and

Scikit-learn (`OneHotEncoder`) can do this.

3.Ordinal Encoding: Similar to label encoding, but categories are encoded based on their meaningful rank/order.

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

- In machine learning, we split our data into training and testing sets to build and evaluate models properly.

1.Training Dataset:i)Purpose:

Used to teach the model.

ii)The model **learns patterns**

and

relationships from this data.

iii)Contains input features **and** their correct outputs (labels/targets).

Ex:-In a house price prediction model, the training set might include:

Size	Bedrooms	Location	Price
------	----------	----------	-------

1000	2	Urban	\$300K
------	---	-------	--------

1500	3	Suburban	\$400K
------	---	----------	--------

2. Testing Dataset: i) Purpose:

Used

to evaluate how well the trained model performs on new, unseen data.

ii) Helps check for overfitting or underfitting.

iii) The model does **not see** this data during training.

Ex:-After training, we test the model on:

Size	Bedrooms	Location	Price
------	----------	----------	-------

1200 | 2 | Suburban | \$350K

Typical Data Split:

Training Set: ~70–80% of the data

Testing Set: ~20–30%

Sometimes a Validation Set is also used (e.g., Train 70% / Validation 15% / Test 15%)

8. What is sklearn.preprocessing?

- sklearn.preprocessing is a module in Scikit-learn (sklearn) that provides a set of tools for preprocessing and transforming data before training a machine learning model.

9. What is a Test set?

- A test set is a portion of your dataset that is kept separate from

the training process and used only to evaluate the final performance of a trained machine learning model.

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

- The most common way to split data into training and testing sets is using `train_test_split` from `scikit-learn`.

Ex:-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:

- i)test_size=0.2: 20% of the data is used for testing, 80% for training.
- ii)random_state=42: Ensures the split is reproducible (any number works).

Here's a structured step-by-step ML workflow:

Step 1: Define the problem:

- i)What is the goal? (classification, regression, clustering, etc.)
- ii)What is the target variable?

Step 2: Gather and understand the data

- i)Load the data (pandas, numpy, etc.)
- ii)Explore data using:
 - .head(), .info(), .describe()
 - Visualizations (matplotlib, seaborn)

Step 3:Data Preprocessing

- i)Handle missing values
- ii)Handle categorical variables (e.g., one-hot encoding)
- iii)Scale/normalize features
- iv)Feature engineering (if necessary)

Step 4:Split the data

- i)Use `train_test_split` to divide into training and testing sets

Step 5:Choose a model

Select a suitable algorithm:

- i)Classification: Logistic Regression, Decision Tree, Random Forest, etc.
- ii)Regression: Linear Regression, Ridge, XGBoost, etc.

Step 6:Train the model

`model.fit(X_train, y_train)`

Step 7: Evaluate the model

`y_pred = model.predict(X_test)`

Use metrics like:

Accuracy, Precision, Recall, F1 Score (for classification)

RMSE, MAE, R^2 (for regression)

Step 8: Tune hyperparameters

Use Grid Search or Random Search to improve performance from `sklearn.model_selection`

`import GridSearchCV`

Step 9: Test and Deploy

i) Validate performance on test

set

ii) Save and deploy the model using tools like pickle, joblib or ML platforms

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

- EDA (Exploratory Data Analysis) is a critical first step in any machine learning or data science project. It helps you understand the structure, patterns, and issues in your dataset before building a model.

Key Reasons to Perform EDA:

1.Understand the data:

- i)What features are available?
- ii)What does the target variable look like?
- iii)What types of variables (categorical, numerical) are present?

2.Detect missing or incorrect values:

- i)Models can't handle missing or corrupted data well.
- ii)EDA helps identify:

Missing values

Outliers

Duplicate records

3.Spot Outliers:

i)Outliers can skew model training and predictions.

ii)You may decide to remove or transform them.

4.Feature relationship:

i)Helps you see how variables

relate to each other or to the target.

ii)You may uncover strong correlations or nonlinear relationships.

5.Decide on feature

Engineering:

EDA may show that some features need to be:

- i)Transformed (log, scaling)
- ii)Combined (e.g., "year" and "month" into "date")
- iii)Encoded (e.g., one-hot encoding for categorical variables)

6.Avoid common pitfalls:

- i)Prevent issues like:
- ii)Data leakage (using info that wouldn't be available at prediction time)

7.Build better models:

- i)A model built on well-understood, clean data is more accurate and reliable.
- ii)It also trains faster and avoids unpredictable behavior.

12. What is correlation?

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

13. What does negative correlation mean?

- Negative correlation means that as one variable increases, the other decreases — they move in opposite directions.

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

- **Step 1: Import libraries**

```
import pandas as pd
```

```
import seaborn as sns
```

```
import matplotlib.pyplot as plt
```

Step 2:Load your data

```
df =
```

```
pd.read_csv('your_dataset.csv')
```

Step 3:Compute the correlation

matrix

```
correlation_matrix = df.corr()
```

```
print(correlation_matrix)
```

Step 4:Visualize with a heatmap

```
plt.figure(figsize=(10, 8))
```

```
sns.heatmap(correlation_matrix,
```

```
annot=True, cmap='coolwarm',
```

```
linewidths=0.5)
```

```
plt.title('Correlation Matrix')
```

```
plt.show()
```

Optional:Correlation between two specific columns

```
correlation =
```

```
df['feature1'].corr(df['feature2'])
```



```
print("Correlation between  
feature1  
and feature2:", correlation)
```

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

- Causation means that one variable directly affects another a cause-and-effect relationship. If A causes B, then changing A will change B.

Correlation vs. Causation – Key Difference:

Aspect	Correlation	Causation
What it	Two variables	One variable produces a

mean	move	change in
s	together	another
Direct	No direction	Has a clear
ion	implied	cause → effect
		path
Exam	Might be	Direct,
ple	coincidence	scientifically
Link	or indirect	explainable
Prove	 No	 Yes
s		
impac		
t?		

Example:- Correlation

Ice cream sales and drowning incidents both increase in summer.

Example:- Causation

Smoking causes lung cancer.

16. What is an Optimizer? What are different types of optimizers?

Explain each with an example.

- An optimizer is an algorithm or method used in machine learning and deep learning to minimize (or maximize) the loss function by adjusting the model's weights and biases during training. The goal of an optimizer is to find the best parameters (weights) so that the model performs well (i.e., has the least error or loss).

Types of Optimizers:

Optimizers are typically categorized into two types:

1.First-order optimizers (use gradients)

2.Second-order optimizers

Below are **popular first-order optimizers**, used frequently in practice:

1. **Gradient Descent:**

Concept: It calculates the gradient of the entire dataset to update weights.

Formula: $\theta = \theta - \eta \cdot \nabla_{\theta} J(\theta)$

θ : parameters

η : learning rate

$\nabla_{\theta} J(\theta)$: gradient of the loss with respect to the parameters

Ex:- Training a linear regression model on a small dataset.

Pros: Simple and intuitive.

Cons: Slow with large datasets

Can get stuck in local minima.

2. **Stochastic Gradient**

Descent: Concept: Instead of

using the entire dataset, it updates weights for each training sample.

Update Rule:

$$\theta = \theta - \eta \cdot \nabla_{\theta} J(\theta; x(i), y(i))$$

Ex:-Training on streaming or large datasets.

Pros:Faster updates.

Can escape local minima due to randomness.

Cons:High variance — may not converge smoothly.

3. Mini-Batch Gradient

Descent:Concept: A

compromise between GD and SGD. Uses mini-batches of data.

Ex:-Batch size of 32 or 64 in neural network training.

Pros:Faster convergence.

More stable than SGD.

4. Momentum:Concept: Helps accelerate SGD by adding a fraction of the previous update to the current one.

Formula: $v_t = \gamma v_{t-1} + \eta \nabla_{\theta} J(\theta)$

$$\theta = \theta - v_t$$

γ : momentum term

Ex:-Image classification with deep networks.

Pros:Faster convergence.

Avoids oscillations.

5.Adagrad:Concept: Adapts learning rate to each parameter, performing larger updates for infrequent parameters.

Formula: $\theta = \theta - \eta / \sqrt{G + \epsilon} \cdot \nabla_{\theta} J(\theta)$

Ex:-NLP tasks with sparse data (like word embeddings).

Pros:Good for sparse data.

Cons: Learning rate decreases too much over time.

6.RMS Prop:Concept:

Improves Adagrad by using exponential decay on the squared gradients.

Formula: $E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma)g^2_t$

$$\theta = \theta - \eta / \sqrt{E[g^2]_t + \epsilon} \cdot g_t$$

Ex:-Recurrent Neural Networks (RNNs)

Pros: Works well for non stationary objectives.

7.Adam:Concept: Combines Momentum and RMSprop.

Maintains both first moment (mean) and second moment (variance) of gradients.

Formula: $m_t = \beta_1 m_{t-1} + (1-\beta_1)g_t$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)g_t^2$$

$$\theta = \theta - \eta / \sqrt{v_t + \epsilon} \cdot m_t$$

Ex:-Deep learning models
(CNNs, RNNs)

Pros:Adaptive learning rates.

Fast and robust.

Default choice in many
frameworks.

8.AdamW:Concept: A variant
of Adam that decouples weight
decay from the gradient
update.

Ex:-Transformer models like
BERT, GPT

Pros:Better generalization than
Adam.

17. What is sklearn.linear_model ?

- sklearn.linear_model is a
module in the Scikit-learn
(sklearn) library in Python that

provides linear models for regression and classification tasks.

18. What does `model.fit()` do? What arguments must be given?

- The `.fit()` method in Scikit-learn is used to train (fit) a machine learning model using training data.

What Does `model.fit()` Do?

When you call `model.fit(X, y)`, it:

- Takes input features (X) and target labels (y).
- Learns the relationships (i.e., computes model parameters like weights and bias).
- Stores these learned

parameters in the model object so you can use it for prediction later with `.predict()`.

General syntax: `model.fit(X, y)`

Required arguments:

Argument	Description
X	Feature matrix (shape: <code>[n_samples, n_features]</code>)
y	Target values (shape: <code>[n_samples]</code> or <code>[n_samples, n_targets]</code>)

Example for regression:

```
from sklearn.linear_model import  
LinearRegression
```

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

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

```
model = LinearRegression()
```

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

19. What does `model.predict()` do?

What arguments must be given?

- The `.predict()` method in Scikit-learn is used to make predictions on new (unseen) data using a trained model.

What Does `model.predict()`

Do?

After training a model with `.fit(X, y)`, you use `.predict(X_new)` to:

- i) Apply the learned model parameters to new input features.
- ii) Return the **predicted target values** (labels for classification, values for regression).

General syntax:

```
y_pred = model.predict(X_new)
```

Required argument:

Argument	Description
----------	-------------

X_new Feature matrix for
prediction (shape:
[n_samples, n_features])

This must have the same
number of features as the
data used in .fit().

Linear regression:
from sklearn.linear_model
import LinearRegression

```
X_train = [[1], [2], [3]]
```

```
y_train = [2, 4, 6]
```

```
model =  
LinearRegression()
```

```
model.fit(X_train, y_train)
```

```
X_new = [[4], [5]]
```

```
predictions =  
model.predict(X_new)
```

```
print(predictions)
```

Ex:-Classification

```
from sklearn.linear_model  
import LogisticRegression
```

```
from sklearn.datasets  
import load_iris
```

```
X, y =  
load_iris(return_X_y=True  
)
```

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

```
predictions =  
model.predict(X[:5])
```

```
print(predictions)  
Internally: For regression,  
it returns continuous  
numerical values.
```

For **classification**, it
returns class labels (like 0,
1, 2, etc.).

If you want **probabilities** instead of class labels in classification, use:
`model.predict_proba(X)`

20. What are continuous and categorical variables?

- A continuous variable is a numerical variable that can take any value within a range. It is measurable and often represents quantities.

A categorical variable is one that has a fixed number of categories or labels. It is qualitative and represents groups or classes.

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

- Feature scaling is a preprocessing technique in machine learning that standardizes or normalizes the range of independent variables (features) so they are on a similar scale.

Why Feature Scaling?

Many machine learning algorithms perform better or converge faster when input features are on the same scale.

Without scaling:

- i) Features with larger magnitudes can dominate the learning process.
- ii) Some algorithms assume or are sensitive to the distribution and range of features.

When is Feature Scaling Important?

Algorithm	Need	Reason
-----------	------	--------

Scaling?

Linear
Regression



Yes

Distance and
gradient-based

Logistic
Regression



Yes

Assumes
equal
contribution

K-Nearest
Neighbors
(KNN)



Yes

Based on
Euclidean
distance


Support
Vector
Machines
(SVM)



Yes


Distance-based
kernel
functions

Principal Component Analysis	 Yes	Sensitive to variance and scale
------------------------------	---------------------------------------------------------------------------------------	---------------------------------

Tree-based models (e.g. RF)	 No	Not affected by scale
-----------------------------	--------------------------------------------------------------------------------------	-----------------------

How Feature Scaling Helps in ML

Benefit

 Faster convergence

Explanation

Especially for gradient descent-based algorithms



More
accurate results

No feature
dominates due to
scale



Distance-based
models work
better

KNN, SVM,
K-means perform
more meaningfully



Fair feature
contribution

Ensures fair
treatment of
features during
training

22. How do we perform scaling in Python?

- In Python, feature scaling is commonly performed using the Scikit-learn (`sklearn`) library.

Here's how to perform different types of scaling step by step:
Step-by-Step: How to Perform Scaling in Python

1.Important required modules:

```
from sklearn.preprocessing  
import MinMaxScaler,  
StandardScaler, RobustScaler  
import numpy as np
```

2.Create sample data:

```
X = np.array([[25, 50000],  
              [45, 120000],  
              [35, 80000]])
```

1.Min-Max scaling:

Scales features to a range of [0, 1]

```
scaler = MinMaxScaler()  
X_minmax = scaler.fit_  
transform(X)
```

```
print("Min-Max Scaled Data:\n",  
X_minmax)
```

2.Standardization:

Scales features to have mean = 0
and standard deviation = 1

```
scaler = StandardScaler()
```

```
X_standard = scaler.fit_  
transform(X)
```

```
print("Standardized Data:\n",  
X_standard)
```

3.Robust scaling:

Uses median and IQR, ideal for
handling outliers

```
scaler = RobustScaler()
```

```
X_robust = scaler.fit_transform(X)
```

```
print("Robust Scaled Data:\n",  
X_robust)
```

23. What is sklearn.preprocessing?

- sklearn.preprocessing is a module in Scikit-learn that provides preprocessing utilities and transformers to prepare your data before feeding it into a machine learning model.

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

- In Python, you can split your dataset into **training and testing sets** using the function:

train_test_split() from
sklearn.model_selection

This is the standard way to split

data when building machine learning models using Scikit-learn.

Why Split the Data?

i) Training set: Used to train (fit) the model.

ii) Testing set: Used to evaluate the model's performance on unseen data.

iii) This helps detect overfitting or underfitting.

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:

Parameter	Description
-----------	-------------

X	Feature matrix
---	----------------

y	Target vector (labels)
---	------------------------

test_size	Proportion of data to be used as test set (e.g., 0.2 = 20%)
-----------	-------------------------------------------------------------

train_size	(Optional) Proportion of data to be used for training
------------	-------------------------------------------------------

random_state	Ensures reproducibility by setting the seed
--------------	---------------------------------------------

shuffle Shuffle data before
splitting (default = True)

stratify Maintain label
distribution (especially
for classification)

Example splitting data:

```
from sklearn.datasets
import load_iris

from
sklearn.model_selection
import train_test_split

X, y =
load_iris(return_X_y=True)
```

```
X_train, X_test, y_train,  
y_test =  
train_test_split(X, y,  
test_size=0.2,  
random_state=1)
```

```
print("Training size:",  
X_train.shape)
```

```
print("Testing size:",  
X_test.shape)
```

Example with stratified
sampling:

```
X_train, X_test, y_train,  
y_test =  
train_test_split(
```

```
X, y, test_size=0.3,  
random_state=42,  
stratify=y)
```

25. Explain data encoding?

- Data encoding is the process of converting categorical (non-numeric) data into numerical

format, so it can be used in machine learning algorithms.

Most ML models (like logistic regression, SVM, or KNN) cannot handle text or category labels directly—they require numbers.

Encoding transforms these categories into numbers in a meaningful or model-friendly way.

Why Is Encoding Important?

i) ML models need numeric input

ii) Categorical data like "Male", "Female" or "Red", "Blue" must be converted

iii) Ensures fairness and accuracy in training

Types of Encoding:

1. Label encoding: Assigns a unique number to each category.

Gender

Encoded

Male	0
Female	1

```
from sklearn.preprocessing import  
LabelEncoder
```

```
encoder = LabelEncoder()
```

```
gender_encoded =  
encoder.fit_transform(['Male', 'Female',  
'Male'])
```

```
print(gender_encoded)
```

2.One hot encoding:Creates binary columns for each category (no order assumed).

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

```
from sklearn.preprocessing import  
OneHotEncoder
```

```
data = [['Red'], ['Blue'], ['Red']]  
  
encoder =  
OneHotEncoder(sparse=False)  
  
encoded = encoder.fit_transform(data)  
  
print(encoded)
```

3.Ordinal Encoding:Encodes categories with an explicit order.

Size	Encoded
Small	0
Medium	1
Large	2

```
from sklearn.preprocessing import  
OrdinalEncoder
```

```
data = [['Small'], ['Medium'], ['Large']]  
  
encoder =  
OrdinalEncoder(categories=[['Small',  
'Medium', 'Large']])  
  
encoded = encoder.fit_transform(data)  
print(encoded)
```

- 4. Binary Encoding:**
- i) Converts categories to binary, then splits bits into columns.
 - ii) Useful when you have high-cardinality categorical data (lots of unique values).
 - iii) Libraries like `category_encoders` offer this.

Summary table:

Encoding Type	Best For	Avoid When
---------------	----------	------------

Label Encoding	Ordinal data	Nominal data
----------------	--------------	--------------

One-Hot Encoding	Nominal data (no order)	Many unique categories
------------------	-------------------------	------------------------

Ordinal Encoding	Ordered categories	Categories without meaningful order
------------------	--------------------	-------------------------------------

Binary/ High-cardi When
Other nality interpretability
columns is key

Example: Combined use in a pipeline
from sklearn.compose import
ColumnTransformer

from sklearn.preprocessing import
OneHotEncoder

from sklearn.ensemble import
RandomForestClassifier

transformer =
ColumnTransformer(transformers=[
 ('cat', OneHotEncoder(), ['color'])
], remainder='passthrough')

```
pipeline = make_pipeline(transformer,  
RandomForestClassifier())
```