1. What is a parameter?

Parameters are the values learned by a model during training. They determine the model's structure and performance

2. What is correlation?

What does negative correlation mean?

The correlation is the relationship between two variables the negative correlation means one variable relation opposite two the other like one variable increase and other variable decrease

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

The subfield of computer science that gives the computer the ability learn by itself without explicitly programmed .Machine learn from pattern and replicate the pattern for the future

Data:	Provides the input for training the model.
Features:	Describe the data and guide model training.
Model:	Maps input features to outputs.
Algorithm:	Defines the learning process.
Loss Function:	Measures error to improve learning.
Optimization:	Adjusts parameters to minimize loss.
Evaluation Metrics:	Validates model performance.
Deployment:	Uses the model in real-world scenarios.

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

The loss value measures the error or difference between the model's predictions and the actual (true) values. It is a numerical representation of how well or poorly a machine learning model is performing during training and testing.

5. **What are continuous and categorical variables**?

Continuous mean any value in a range eg: height , weight

Categorical variable are qualitative data that can grouped into different category

Eg: blood group , ranks , sex

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

Categorical variables represent data that can be grouped into categories rather than numerical values. In machine learning, models work primarily with numerical data, so categorical variables must be encoded into a numerical format. Proper handling ensures models can effectively use these variables for predictions.

- Nominal/OneHotEncoder
- Label Encoder
- Ordinal Encoder
- Target Guided encoder

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

When building machine learning models, the data is divided into two primary subsets: training data and testing data. This division allows the model to learn from one part of the data and be evaluated on another to ensure it generalizes well to unseen data.

Mostly the split take places

70-30 or 80-20

8. **What is sklearn.preprocessing?**

sklearn.preprocessing is a module in scikit-learn that provides tools for preprocessing and transforming data before using it in machine learning models. Preprocessing is a critical step to ensure that the data is clean, well-scaled, and in a format that models can work with effectively.

Preprocessing Module used for:

StandardScaler

MinMaxScaler

Normalize

DataEncoding:

Nominal/OneHotEncoder

LabelEncoder

OrdinalEncoder

9. What is a Test set?

A **test set** is a portion of data used to evaluate how well a machine learning model performs on unseen data. It is kept separate from the training and validation data to provide an unbiased assessment of the model's ability to generalize. Metrics like accuracy or error rate are calculated on the test set to measure the model's effectiveness.

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

Consider there is 1000 data it will split in to 700 for training and 150 for validation 150 for test

It can be done by using from sklearn.model_selection import train_test_split function perform this operation

Based on the fit and transform performed in training and validating according to it perform transform on the test data(unseen data)

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

Performing Exploratory Data Analysis (EDA) before fitting a model is essential because it helps you understand the data, identify issues, and make informed decisions for model building. EDA is a critical step to ensure the dataset is clean, well-understood, and ready for modeling. Skipping EDA can lead to poor model performance, misinterpretations, or wasted effort on irrelevant features.

12 – 13 same question of 2

14 How can you find correlation between variables in Python?

can find the correlation between variables in Python using the .corr() method provided by pandas.

Eg : df.corr(method='pearson')

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

Causation refers to a relationship between two variables where one variable directly influences or causes a change in another. In other words, causation implies that a change in variable A is responsible for a change in variable B

Eg: smoking causes lung desease

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

An **optimizer** is an algorithm or method used to adjust the parameters (weights and biases) of a machine learning model during training to minimize a given loss function. The goal is to improve the model's performance by finding the optimal set of parameters.

Optimizers play a crucial role in training neural networks by controlling:

1. **Step Size**: How much to update the parameters in each iteration.
2. **Convergence**: Ensuring the model reaches a minimum loss effectively and efficiently.

Types of Optimizers

Optimizers are broadly categorized into **Gradient-Based** and **Other Optimization Methods**. Below are commonly used optimizers in machine learning:

1. Gradient Descent (GD)

- **Description**: Updates model parameters by moving in the direction of the steepest descent of the loss function.
- **Variants**:
  - **Batch Gradient Descent**: Uses the entire dataset for each update.
  - **Example**:
  - `# Example with SGD in PyTorch`
  - `optimizer = torch.optim.SGD(model.parameters(), lr=0.01)`

2. Stochastic Gradient Descent (SGD)

- **Description**: Updates parameters using one data point (or a small batch) at a time, introducing randomness.
- **Advantages**:
  - Faster updates for large datasets.
- **Disadvantages**:
  - High variance in updates, leading to potential instability.
- **Example**:
- `optimizer = torch.optim.SGD(model.parameters(), lr=0.01)`

3. Momentum

- **Description**: Adds a "momentum" term to SGD to accelerate updates in the relevant direction and dampen oscillations.
- **Example**:
- `optimizer = torch.optim.SGD(model.parameters(), lr=0.01, momentum=0.9)`

4. AdaGrad (Adaptive Gradient Algorithm)

- **Description**: Adapts the learning rate for each parameter based on the historical gradient. Works well for sparse data.
- **Disadvantage**: Accumulated squared gradients can make the learning rate shrink too much.
- **Example**:
- `optimizer = torch.optim.Adagrad(model.parameters(), lr=0.01)`

5. RMSProp (Root Mean Square Propagation)

- **Description**: Like AdaGrad, but limits the growth of the accumulated gradients using an exponential moving average.
- **Example**:
- `optimizer = torch.optim.RMSprop(model.parameters(), lr=0.01)`

6. Adam (Adaptive Moment Estimation)

- **Description**: Combines the benefits of Momentum and RMSProp. Maintains separate learning rates for each parameter and uses moving averages of gradients and squared gradients.
- **Advantages**:
  - Adaptive learning rates.
  - Works well in practice for most applications.
- **Example**:
- `optimizer = torch.optim.Adam(model.parameters(), lr=0.001)`

7. AdamW (Adam with Weight Decay)

- **Description**: A variant of Adam that includes weight decay for better regularization.
- **Example**:
- `optimizer = torch.optim.AdamW(model.parameters(), lr=0.001)`

17.What is sklearn.linear_model ?

sklearn.linear_model is a module in the scikit-learn library in Python, which provides a collection of machine learning models and algorithms for linear regression, classification, and related tasks. These models are based on the assumption that the relationship between the input features and the target variable can be represented (or approximated) linearly.

Key Features of sklearn.linear_model:

Linear Regression:

Models the relationship between independent variables (features) and a continuous dependent variable.

Example: Predicting house prices based on size, location, etc.

Linear Classification:

Used for binary or multi-class classification tasks.

Example: Logistic regression for predicting whether an email is spam or not.

Regularization:

Offers variants like Ridge, Lasso, and ElasticNet that include regularization to prevent overfitting and handle multicollinearity.

Robustness:

Provides robust models like HuberRegressor and RANSACRegressor to handle outliers and noise.

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

model.fit() is the cornerstone of model training in scikit-learn, allowing the model to learn from the provided data. Ensure the arguments (X and y) are correctly formatted and aligned with the model's requirements.

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

model.predict() is the core method for inference in scikit-learn, used to predict target outcomes from new input data. It requires only the input features (X) and works seamlessly after the model has been trained.

20 same question of 5th

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

Feature scaling is a preprocessing technique used to normalize or standardize the range of independent variables (features) in a dataset. It ensures that all features contribute equally to the model by bringing them to a comparable scale, without distorting differences in their ranges.

For example:

A dataset with features like age (0-100) and income (0-100,000) has variables on vastly different scales. Feature scaling adjusts these scales to make them uniform.

1.Improve model performance

2.Enable fair feature comparison

3.Essential for regulation

4.Prevent Numerical instability

22. How do we perform scaling in Python?

Normalization(MinMaxScaler)

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)
```

Standardization(StandardScaler)

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

UnitVector(Normalize)

```
From sklearn.preprocessing import Normalize
Normalize(X)
```

23-24 Repeating the same questions

25 Explain data encoding?

Data encoding is the process of converting categorical or textual data into a numerical format that machine learning models can understand. Since most machine learning algorithms work with numeric data, encoding transforms non-numeric features into a format that can be processed effectively.

1.Nominal/OneHotEncoder

Used when cannot compare the categorical data

Performance are based as binary table

1 for corresponding category data and others kept as 0

```
from sklearn.preprocessing import OneHotEncoder
encoder  = OneHotEncoder
encoder.fit_transform(x_train).toarray()
```

2.LabelEncoder

LabelEncoder gives separate value for each category data

```
From sklearn.preprocessing import LabelEncoder
Encode  = LabelEncoder()
Encoder.fit_transform(x_train)
```

3.OrdinalEncoder

OrdinalEncoder gives values to category data according to the order

```
From sklearn.preprocessing import OrdinalEncoder
Encoder = OrdinalEncoder(categories = [["Cate_1","cate_2"...]])
Encoder.fit_transform(x_train)
```

4.Target_variable Encoder

Target encoder based on each categories corresponding target columns mean values

```
Target_encoder = Df.groupby("categorical_column")["target_column"].mean().to_dict()
Df["category"].map(Target_encoder)
```