Contents

[MACHINE LEARNING ALGORITHMS 2](#_Toc204797135)

[TYPES OF ML ALGORITHIM 3](#_Toc204797136)

[TYPES OF MACHINE LEARNING 3](#_Toc204797137)

[SUPERVISED LEARNING 3](#_Toc204797138)

[CLASSIFICATION (BINARY CLASSIFICATION) 5](#_Toc204797139)

[REGRESSION ALGORITHM 7](#_Toc204797140)

[UNSUPERVISED LEARNING 9](#_Toc204797141)

[SUPERVISED V/S UN-SUPERVISED LEARNING 11](#_Toc204797142)

[FEATURE SCALING 11](#_Toc204797143)

[NORMALIZATION 12](#_Toc204797144)

[STANDARDIZATION 12](#_Toc204797145)

[EXAMPLE – NORMALIZATION 13](#_Toc204797146)

[TRAINING ML MODELS 14](#_Toc204797147)

[LIFECYCLE 14](#_Toc204797148)

[FEATURE ENGINEERING 14](#_Toc204797149)

[MACHINE LEARNING ALGORITHMS CLASSIFICATION 15](#_Toc204797150)

[ANOMALY DETECTION ALGORITHM 15](#_Toc204797151)

[CLUSTERING ALGORITHM 15](#_Toc204797152)

[DATA PRE-PROCESSING USING PYTHON 15](#_Toc204797153)

[DATA PROCESSING USING PYTHON 20](#_Toc204797154)

[REGRESSION 20](#_Toc204797155)

[MULTIPLE LINEAR REGRESSION 21](#_Toc204797156)

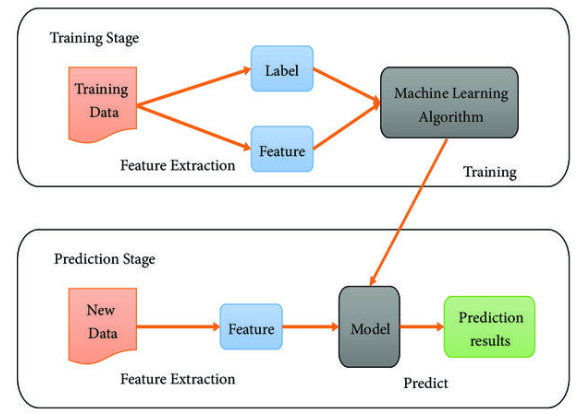
[POLYNOMIAL REGRESSION 21](#_Toc204797157)

[SUPPORT VECTOR REGRESSION 21](#_Toc204797158)

[DECISION TREE REGRESSION 21](#_Toc204797159)

[RANDOM FOREST REGRESSION 21](#_Toc204797160)

# MACHINE LEARNING ALGORITHMS



* Machines learn by observing examples and identifying patterns, similar to how humans (or babies) learn through experience.
* Machine learning relies heavily on data — both its **quality** and **quantity** — to train models effectively.

Steps

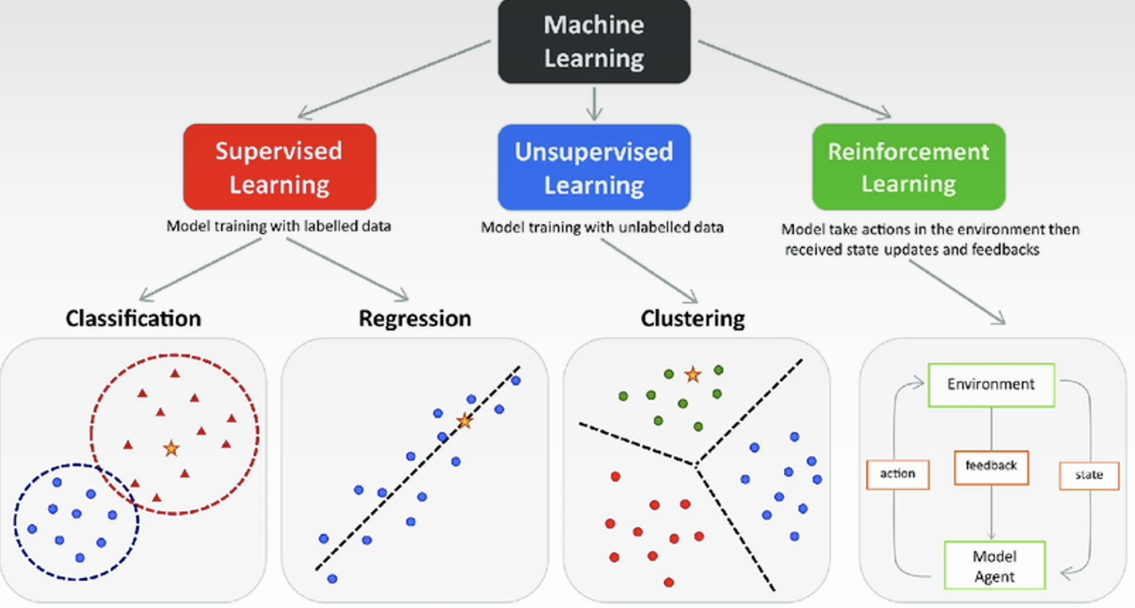
Step 1: The data is used to train algorithms.

Step 2: Algorithms analyze data to find patterns, once it recognize the pattern from the data, we can it a model

Step 3: Then models capable can able to do prediction for the new & unseen data.

|  |
| --- |
| **Algorithms**: These are sets of rules, often based on statistical and mathematical techniques, that guide the learning process. |

## TYPES OF ML ALGORITHIM



## TYPES OF MACHINE LEARNING

## SUPERVISED LEARNING

* The name supervised learning originates from the idea that training a machine while using this type of approach is like how humans are learning under the supervision of a teacher.
* Example - In regular school class, we have a group of students and a teacher. During a lecture about some specific topic, the teacher will provide several examples while teaching something. The students will use those examples to analyze and memorize them, something that will help them to extract the patterns from those examples. At a later stage, based on the information provided, the students will be able to solve similar problems. Overall, the teacher decided what kind of examples to present and how many, he or she basically supervised the learning process.
* **In supervised learning, we train machines by providing them with a set of examples, each provided example is a pair consisting of an input object and the desired output value for that object, it’s called label data set.**

|  |  |
| --- | --- |
| A diagram of a function  AI-generated content may be incorrect. | * **"X" is the input into the machine which can be a group of values called "features".** * **"Y" is the output of that machine, the target value.** * **Functions with the input x are basically some mathematical transformation function or mapping function discovered by the algorithm doing the training process** |

**NOTE: In Supervised Machine learning – the model tries to find the mapping function to map the input (feature) and output(label)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **RELATION BETWEEN MAPPING FUNCTION AND ML ALGORITHM**  **WHAT IS A MAPPING FUNCTION?**   * Think of a mapping function as a recipe or rule that tells us how to turn input into output. * In machine learning, this function is what the algorithm is trying to learn from data.   **REAL-WORLD EXAMPLE: PREDICTING HOUSE PRICES:** Let’s say we want to predict the price of a house based on its size.   * Input (X): Size of the house (e.g., 1000 sq ft) * Output (Y): Price of the house (e.g., ₹50 lakhs)   The ML algorithm’s job is to learn a function like:  **Price = f(Size)**  This function could be:   * A straight line (if price increases steadily with size) * A curve (if price increases faster for bigger houses)   **HOW ML ALGORITHM USES THE MAPPING FUNCTION?**   1. **Step 1**: We give it data: Sizes and prices of many houses. 2. **Step 2**: It finds a pattern: Learns the best function (mapping) that connects size to price. 3. **Step 3**: We use it to predict: For a new house size, it uses the function to predict the price.   Example in Simple Terms (Training Data)   |  |  | | --- | --- | | Size (sq ft) | Price (₹ lakhs) | | 1000 | 50 | | 1500 | 75 | | 2000 | 100 |  * The ML algorithm might learn: **Price = 0.05 X Size** * So, for a 1200 sq ft house: **Price = 0.05 X 1200 = ₹60** * A mapping function is the rule that connects input to output. * An ML algorithm learns this rule from data. * Once learned, it can make predictions on new data.   **Here's a simple chart that shows how a machine learning algorithm learns a mapping function from house size to price:**  **A graph with a red line and blue dots  AI-generated content may be incorrect.**   * *Blue dots: Real data points (house size vs. price).* * *Red line: The learned function (Price = 0.05 × Size), which the ML algorithm uses to make predictions***.**   **This is a basic example of how ML finds a pattern (a mapping function) in data and uses it to predict outcomes.** |

Two typical tasks performed by Supervised Learning

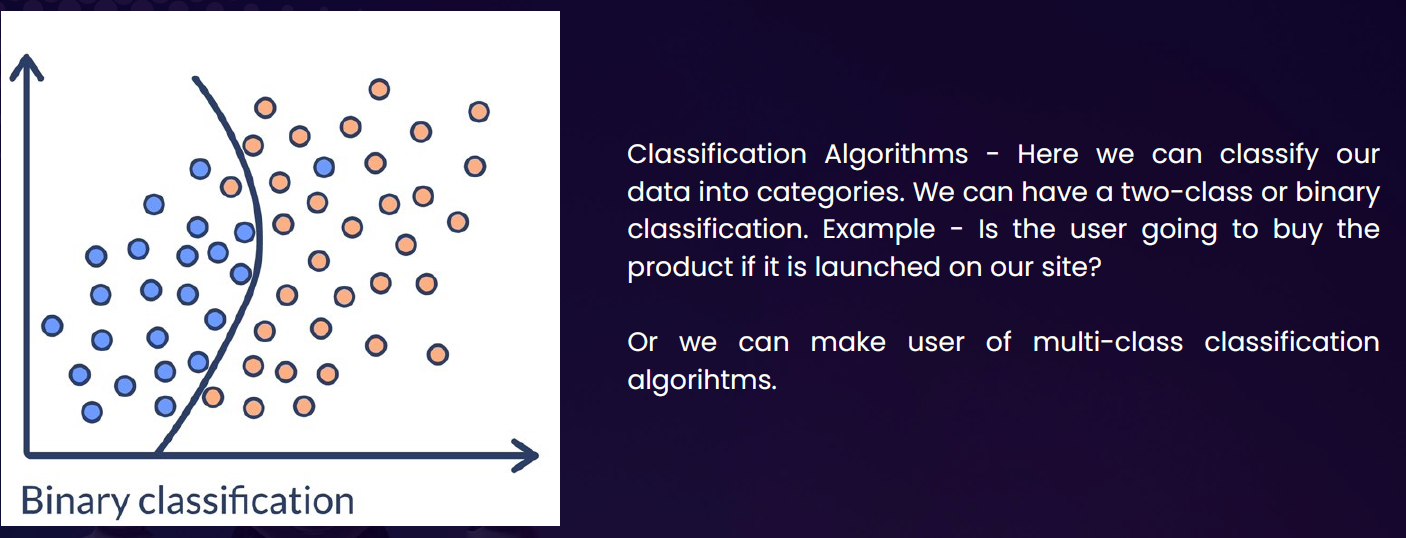
1. **CLASSIFICATION**
2. **REGRESSION**

### CLASSIFICATION (BINARY CLASSIFICATION)

A black and green rectangular sign with white text

AI-generated content may be incorrect.

**Classification** is a type of supervised learning where the goal is to **predict a category or class**. Instead of predicting a number (like in regression), classification predicts a **label**.



**Real-Life Example: Email Spam Detection**

Imagine if we have a bunch of emails, and each one is labeled as either:

* **Spam**
* **Not Spam**

We train a machine learning model with this data. The model learns patterns like:

1. Emails with “win money” → likely spam
2. Emails from your contacts → likely not spam

Once trained, the model can classify **new emails** as spam or not spam.

**Other Examples of Classification**

|  |  |
| --- | --- |
| Problem | Classes (Labels) |
| Disease diagnosis | Sick / Healthy |
| Image recognition | Cat / Dog / Bird |
| Customer feedback sentiment | Positive / Negative / Neutral |
| Loan approval | Approved / Rejected |

#### MULTICLASS CLASSIFICATION



Multiclass classification is when a machine learning model predicts **one label out of three or more possible categories**.

|  |  |
| --- | --- |
| **Problem** | **Classes (Labels)** |
| Handwritten digit recognition | 0 to 9 |
| Animal image classification | Cat / Dog / Bird / Horse |
| Exam grading | A / B / C / D / F |

* SVM(Support Vector Machines) are one such classification ML Model

### REGRESSION ALGORITHM

A graph with arrows and dots

AI-generated content may be incorrect.

* **Regression** is a type of **supervised learning** where the goal is to **predict a continuous value** (a number) based on input data.

**Real-World Example: Predicting House Prices**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Imagine a real estate agent and want to **predict the price of a house** based on certain features like:   * Size of the house (in square feet) * Number of bedrooms * Location * Age of the house | You collect data from past house sales. Each row in your dataset looks like this:   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Size (sqft)** | **Bedrooms** | **Age (years)** | **Location Score** | **Price (₹)** | | 1200 | 3 | 5 | 8 | 45,00,000 | | 1500 | 4 | 2 | 9 | 60,00,000 | | 1000 | 2 | 10 | 6 | 35,00,000 | |

* Now, we can train a **regression model** using this data. The model learns the relationship between the features and the price. Later, when we get a new house with known features but unknown price, the model can **predict the price**.

Great question! Let's break it down simply.

**What Does "Predict a Continuous Value" Mean?**

A **continuous value** is a number that can be **any value within a range**, not just fixed categories.

So, when we say **"predict a continuous value"**, we mean the model is trying to **guess a number** — not a label like "yes/no" or "cat/dog", but something like:

**Real-Life Examples**

|  |  |  |
| --- | --- | --- |
| **Problem** | **Input Features** | **Output (Continuous Value)** |
| Predict house price | Size, location, age | ₹50,00,000 |
| Predict temperature | Date, time, humidity | 32.5°C |
| Predict height of a child | Age, gender, parents' height | 145.2 cm |
| Predict fuel efficiency | Engine size, weight, speed | 18.7 km/l |

In each case, the **output is a number** that can vary smoothly — not just a fixed set of options.

**GRAPHICAL REPRESENTATION**

A graph of a house size

AI-generated content may be incorrect.

* Each dot represents a house.
* The x-axis is the size of the house (in square feet).
* The y-axis is the price of the house (in INR).
* The pattern shows that as house size increases, the price also tends to increase — this is the kind of relationship a regression model learns.
* Red line: The regression line — this is the model's prediction of house price based on size.
* The most common form of regression analysis is linear regression

#### LINEAR REGRESSION

* **Linear regression** is a fundamental concept in statistics and machine learning used to model the relationship between a **dependent variable** (target) and one or more **independent variables** (predictors or features).
* **Key Idea:** Linear regression assumes that there is a **linear relationship** between the input (X) and output (Y), meaning:

|  |  |
| --- | --- |
| **Y = aX + b + error** | * **Y** is the dependent variable (what we want to predict), * **X** is the independent variable (input), * **a** is the slope (how much Y changes with X), * **b** is the intercept (value of Y when X = 0), * **error** is the difference between the predicted and actual value. |

##### TYPES OF LINEAR REGRESSION

1. **Simple Linear Regression**: 1 independent variable: **Y = aX + b**
2. **Multiple Linear Regression**: more than 1 independent variable **Y = a1X1 + a2X2 + ... + anXn + b**

|  |
| --- |
| **Goal: Find the best values of a and b so that the predicted Y values are as close as possible to the actual Y values — usually by minimizing the mean squared error (MSE) between them.** |

**Other Applications:**

* Predicting sales based on ad spend
* Estimating house prices based on size, location, etc.
* Forecasting trends over time

##### EXAMPLE OF LINEAR REGRESSION

|  |  |  |
| --- | --- | --- |
| Problem: You have data showing how much a student studies (in hours), and their exam score. You want to predict the score based on study time. | **Study Hours (X)** | **Exam Score (Y)** |
| 1 | 50 |
| 2 | 55 |
| 3 | 65 |
| 4 | 70 |
| 5 | 75 |

**Goal:** Find a line Y = aX + b that best fits this data.

**Step 1: Plot the data :** If you plot X and Y on a graph, you'll notice a roughly straight-line trend.

**Step 2: Apply linear regression :** Using Python (with scikit-learn), you can fit a model:

|  |  |
| --- | --- |
| **from sklearn.linear\_model import LinearRegression**  **import numpy as np**  **# Data**  **X = np.array([[1], [2], [3], [4], [5]]) # 2D array for sklearn**  **Y = np.array([50, 55, 65, 70, 75])**  **# Model**  **model = LinearRegression()**  **model.fit(X, Y)**  **# Coefficients**  **slope = model.coef\_[0]**  **intercept = model.intercept\_**  **print(f"Model: Y = {slope:.2f}X + {intercept:.2f}")** | **Output:**  Model: Y = 6.5X + 43.0  This means:   * For every extra hour studied, score increases by ~6.5 * If a student studies **0 hours**, expected score = 43   **Prediction Example:**  Predict the score if a student studies for 6 hours:  **model.predict([[6]]) # Output: ~82.0**  So, **predicted score = 82**. |

## UNSUPERVISED LEARNING

* **Unsupervised learning** is a type of machine learning where the model learns from data **without any labels**.
* It tries to find **patterns, structures, or groupings** in the data on its own.

##### Examples

Imagine we have a bunch of photos of animals, but we don’t tell the model which ones are cats, dogs, or birds. The model looks at the photos and **groups similar ones together** — maybe all the cats in one group, dogs in another, and so on.

Common Use Cases

|  |  |
| --- | --- |
| Use Case | Description |
| Customer Segmentation | Grouping customers based on behavior or preferences. |
| Market Basket Analysis | Finding which products are often bought together. |
| Anomaly Detection | Spotting unusual patterns (e.g., fraud detection). |
| Document Clustering | Grouping similar news articles or research papers. |
| Image Compression | Reducing image size by identifying patterns in pixels. |

Popular Algorithms

* **K-Means Clustering**
* **Hierarchical Clustering**
* **DBSCAN**
* **Principal Component Analysis (PCA)**
* **Autoencoders**

## SUPERVISED V/S UN-SUPERVISED LEARNING

A comparison of words with black text

AI-generated content may be incorrect.

## FEATURE SCALING

Feature scaling **transforms the values of features to be on a similar scale**, typically to improve model performance and training stability.

We can understand feature scaling using the example below. Note: ***Feature scaling is always applied at column level***

A table with numbers and lines

AI-generated content may be incorrect.

There are 2 main techniques of feature scaling

1. **NORMALIZATION**
2. **STANDARDIZATION**

### NORMALIZATION

|  |  |
| --- | --- |
| A math equation with black text  AI-generated content may be incorrect. | * The normalization value lies between the closed interval of [0;1] |

|  |  |  |
| --- | --- | --- |
| **X1 (Price)** | **X-XMIN** | **Normalized Value(X1)** |
| $179.43 | $0.00 | 0.00 |
| $641.87 | $179.43 | 0.39 |
| $556.30 | $376.87 | 0.814959779 |
| $578.47 | $116.03 | 0.250908226 |
| $591.12 | $411.69 | 0.890256033 |
|  |  |  |
| X1-MAX |  | $641.87 |
| X1-MIN |  | $179.43 |
| X1MAX - X1MIN |  | **$462.44** |

### STANDARDIZATION

|  |  |  |
| --- | --- | --- |
| A math equation with numbers and symbols  AI-generated content may be incorrect. | µ | Average |
|  | Standard Deviation |
| * The value is lies in closes interval of [-3,3] * If data has some outliers – it will exist outside this range | |

### EXAMPLE – NORMALIZATION

A screenshot of a computer

AI-generated content may be incorrect.

1. Let's imagine we have a data set where we have two columns, annual income of a person and their age of

a blue, purple and red person.

1. **We must identify whether the purple person is like a “red” person or “blue” person . This is the task of clustering data. For that we need to do normalization of data as the units of the data is not uniform**

A close-up of a number

AI-generated content may be incorrect.

1. After normalizing, our values will look like above. Hence with the normalized data – From salary column perspective. The purple person is almost right in the middle between the red and the blue people(0.44), whereas in the age column, the purple person is closest to the blue person.

|  |
| --- |
| Scikit-learn (also written as scikit-learn or sklearn) is a powerful and widely used open-source machine learning library for the Python. It provides simple and efficient tools for:  🔍 Key Features   * Classification: Identifying which category an object belongs to (e.g., spam detection). * Regression: Predicting a continuous-valued attribute (e.g., house prices). * Clustering: Grouping similar data points (e.g., customer segmentation). * Dimensionality Reduction: Reducing the number of features (e.g., PCA). * Model Selection: Comparing, validating, and choosing parameters and models. * Preprocessing: Feature extraction, normalization, and transformation.   🧰 Built On  Scikit-learn is built on top of:   * NumPy: For numerical operations. * SciPy: For scientific computing. * Matplotlib: For plotting (indirectly used). * joblib: For model persistence and parallel processing. |

# TRAINING ML MODELS

A diagram of a training process

AI-generated content may be incorrect.

1. As part of the prerequisite of most machine learning projects, we need a training data set. A training data set is a large group of labeled examples. A machine-learning system is going to learn patterns inside the training data set and store that knowledge in something that is called a **model**.
2. This model is supposed to define as close as possible the relationship between features and the target label. In a common type of machine learning method called "supervised learning" the way to create this kind of model is based on analyzing a large group of labeled examples.
3. Once we have trained our model with those labeled examples, we can use that trained model to predict the label on unlabeled examples.

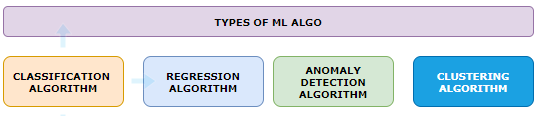
## LIFECYCLE

The lifecycle of a model in a machine learning system, we have two main phases

1. **TRAINING PHASE (LEARNING PHASE):** The main idea is to utilize or use some learning algorithms that will build the model using the training data set.
2. **INFERENCE PHASE (LEARNING PHASE):** In machine learning inference means applying the trained model in an actual machine learning system working in a production environment for making ongoing predictions.

## FEATURE ENGINEERING

# MACHINE LEARNING ALGORITHMS CLASSIFICATION



## ANOMALY DETECTION ALGORITHM

* This is used to detect if data deviates from the norm
* Example: used to detect fraudulent credit card purchase

## CLUSTERING ALGORITHM

A computer screen with text and a diagram

AI-generated content may be incorrect.

# DATA PRE-PROCESSING USING PYTHON

|  |  |
| --- | --- |
|  | * In this example we will perform data preprocessing step on this following data * It’s a user profile data – of an ecommerce website with a flag which says whether user has made purchase or not! * As a data processing step – we create two entities –  1. The first is **the matrix of features**, which contains separately these three columns (country, age, salary.) 2. And second is the dependent variable vector, which is last column(“Purchased”), because that's the column we want to predict.   *Note : This is exactly what we must do in this first data pre-processing phase.* |

|  |  |
| --- | --- |
| Step 1: Importing the libraries | import numpy as np import matplotlib.pyplot as plt import pandas as pd |
| Step 2: Importing the dataset | dataset = pd.read\_csv('Data.csv') X = dataset.iloc[:, :-1].values y = dataset.iloc[:, -1].values  OUTPUT  [['France' 44.0 72000.0]  ['Spain' 27.0 48000.0]  ['Germany' 30.0 54000.0]  ['Spain' 38.0 61000.0]  ['Germany' 40.0 nan]  ['France' 35.0 58000.0]  ['Spain' nan 52000.0]  ['France' 48.0 79000.0]  ['Germany' 50.0 83000.0]  ['France' 37.0 67000.0]]  ['No' 'Yes' 'No' 'No' 'Yes' 'Yes' 'No' 'Yes' 'No' 'Yes'] |
| Step 3: Missing Data   * For missing data, we can make use of Python library Scikit-learn. It is an open-source machine learning library built on top of **NumPy**, **SciPy**, and **matplotlib**. * It provides simple and efficient tools for data mining and data analysis. * For example - For missing salary - We will replace the missing salary with average salary in the column   **from sklearn.impute import SimpleImputer**  **imputer= SimpleImputer(missing\_values=np.nan, strategy='mean') imputer.fit(X[:, 1:3]) # Assuming columns 1 and 2 have missing values X[:, 1:3] = imputer.transform(X[:, 1:3]) # Transform the data to fill missing values**   * SimpleImputer from scikit-learn to fill missing values (NaN) in specific columns of the feature matrix X with the mean of each column. * It fits the imputer on columns 1 and 2 (indexing starts at 0), replaces missing values with the computed mean | |

Step 4: Encoding the categorial data

* Encoding categorical data is **crucial in machine learning** because most ML algorithms require **numerical input** to perform mathematical computations.

Common Encoding Techniques

|  |  |  |
| --- | --- | --- |
| Technique | Description | Best For |
| Label Encoding | Assigns A Unique Number To Each Category | Ordinal Data (E.G., "Low", "High") |
| One-Hot Encoding | Creates Binary Columns For Each Category | Nominal Data (E.G., "Red", "Blue") |
| Ordinal Encoding | Encodes Categories With Meaningful Order | Ordered Categories |
| Target Encoding | Replaces Categories With The Mean Of The Target Variable For Each Category | High-Cardinality Categorical Data |

Example

Suppose we have a column Fuel Type with values: ["Petrol", "Diesel", "Electric"]

* Label Encoding: Petrol → 0, Diesel → 1, Electric → 2 (May imply an order that doesn’t exist)
* One-Hot Encoding:

| Petrol | Diesel | Electric |
| --- | --- | --- |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |

Why Encode Categorical Data?

* ML Models Work with Numbers
  1. Algorithms like linear regression, decision trees, and neural networks **cannot interpret text** or labels directly. They need **numerical representations** to process the data.
* Preserves Information
  1. Encoding transforms categories into numbers **without losing the meaning** of the data. For example, converting "Red", "Blue", "Green" into numerical form allows the model to still distinguish between them.
* Improves Model Performance
  1. Proper encoding helps the model **understand relationships** between variables, which can lead to **better predictions** and **faster training**.
* Avoids Bias from Arbitrary Numbers
  1. Some encoding methods (like **One-Hot Encoding**) prevent the model from assuming an **ordinal relationship** where none exists.
  2. For example, assigning "Low", "Medium", "High" as 1, 2, 3 implies a ranking, which may or may not be appropriate.

CODE

Taking the example further will apply “Hot Encoding” of “County” column and “label encoding” to the “Purchased” column.

One Hot Encoding OF Country Column

|  |
| --- |
| from sklearn.preprocessing import OneHotEncoder  ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),[0])],remainder='passthrough') X = np.array(ct.fit\_transform(X)) # Apply one-hot encoding to the first column |

LABELLED Encoding OF Country Column

|  |
| --- |
| from sklearn.preprocessing import LabelEncoder  le = LabelEncoder() y = le.fit\_transform(y) # Apply label encoding to the dependent variable |

EXAMPLE – CODING EXERCISE

Dataset

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| PassengerId | Survived | Pclass | Name | Sex | Age | SibSp | Parch | Ticket | Fare | Cabin | Embarked |
| 2 | 1 | 1 | Cumings, Mrs. John Bradley (Florence Briggs Thayer) | female | 38 | 1 | 0 | PC 17599 | 71.2833 | C85 | C |
| 4 | 1 | 1 | Futrelle, Mrs. Jacques Heath (Lily May Peel) | female | 35 | 1 | 0 | 113803 | 53.1 | C123 | S |
| 7 | 0 | 1 | McCarthy, Mr. Timothy J | male | 54 | 0 | 0 | 17463 | 51.8625 | E46 | S |
| 11 | 1 | 3 | Sandstrom, Miss. Marguerite Rut | female | 4 | 1 | 1 | PP 9549 | 16.7 | G6 | S |
| 12 | 1 | 1 | Bonnell, Miss. Elizabeth | female | 58 | 0 | 0 | 113783 | 26.55 | C103 | S |
| 22 | 1 | 2 | Beesley, Mr. Lawrence | male | 34 | 0 | 0 | 248698 | 13 | D56 | S |

Coding Exercise 3: Encoding Categorical Data for Machine Learning

**1**: Import required libraries - Pandas, Numpy, and required classes for this task - ColumnTransformer, OneHotEncoder, LabelEncoder.

**2**: Start by loading the Titanic dataset into a pandas data frame. This can be done using the pd.read\_csv function. The dataset's name is 'titanic.csv'.

**3**: Identify the categorical features in your dataset that need to be encoded. You can store these feature names in a list for easy access later.

**4**: To apply OneHotEncoding to these categorical features, create an instance of the ColumnTransformer class. Make sure to pass the OneHotEncoder() as an argument along with the list of categorical features.

**5**: Use the fit\_transform method on the instance of ColumnTransformer to apply the OneHotEncoding.

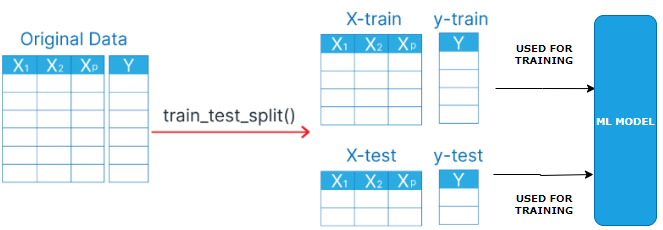
**6**: The output of the fit\_transform method should be converted into a NumPy array for further use.

**7**: The 'Survived' column in your dataset is the dependent variable. This is a binary categorical variable that should be encoded using LabelEncoder.

**8.**  Print the updated matrix of features and the dependent variable vector

|  |
| --- |
| # Importing the necessary libraries import pandas as pd import numpy as np from sklearn.compose import ColumnTransformer from sklearn.preprocessing import OneHotEncoder, LabelEncoder  # Load the dataset dataset = pd.read\_csv("titanic.csv")  # Identify the categorical data categorical\_features = ['Sex', 'Embarked', 'Pclass']  # Implement an instance of the ColumnTransformer class  ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),categorical\_features)],remainder='passthrough')  # Apply the fit\_transform method on the instance of ColumnTransformer ct\_fit = ct.fit\_transform(dataset) # Apply one-hot encoding to the first column  # Convert the output into a NumPy array X = np.array(ct.fit\_transform(dataset))  # Use LabelEncoder to encode binary categorical data le = LabelEncoder()  Y = le.fit\_transform(dataset["Survived"]) # Apply label encoding to the dependent variable  # Print the updated matrix of features and the dependent variable vector print(X) print(Y) |

Step 5: Training Versus Test Data



|  |
| --- |
| 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=1) |

The train\_test\_split method arrays or matrices into random train and test subsets.

Parameters:

* X: Features (input data).
* y: Labels (target data).
* test\_size: Fraction or number of samples to use as the test set (e.g., 0.2 means 20% for testing).
* train\_size: Fraction or number of samples for training (optional, usually inferred).
* random\_state: Seed for random number generator (ensures reproducibility).
* shuffle: Whether to shuffle data before splitting (default is True).
* stratify: Ensures the split preserves the proportion of labels (optional, usually for classification).
* Returns: Splits the data into x\_train, x\_test, y\_train, and y\_test.

EXAMPLE

1. Import necessary Python libraries: pandas, train\_test\_split from sklearn.model\_selection, and StandardScaler from sklearn.preprocessing.
2. Load the Iris dataset using Pandas read.csv. Dataset name is iris.csv.
3. Use train\_test\_split to split the dataset into an 80-20 training-test set.
4. Apply random\_state with 42 value in train\_test\_split function for reproducible results.
5. Print X\_train, X\_test, Y\_train, and Y\_test to understand the dataset split.
6. Use StandardScaler to apply feature scaling on the training and test sets.
7. Print scaled training and test sets to verify feature scaling.

Step 6: Feature Scaling

# DATA PROCESSING USING PYTHON

# REGRESSION

* A **regression model** is a type of statistical or machine learning model used to understand the relationship between a **dependent variable** (what you're trying to predict) and one or more **independent variables** (the inputs or predictors).
* **In simple terms:** A regression model helps answer questions like:
  + "How does the price of a house depend on its size, location, and number of bedrooms?"
  + "How does advertising spending affect sales?"

Types Of Regression Models

1. Linear Regression

* Assumes a straight-line relationship between variables.
* Example: y = a + bx
* Multiple Linear Regression:
* Like linear regression, but with multiple input variables.
* Example: y = a + b*1x*1 + b*2x*2 + … + b*nx*n
* Polynomial Regression:
* Models curved relationships by including powers of the input variables.
* Example: y = a + bx + cx2
* Logistic Regression:
* Used when the output is categorical (e.g., yes/no, 0/1).
* Despite the name, it's used for classification, not regression.
* Ridge, Lasso, And Elastic Net Regression:
* Variants of linear regression that include regularization to prevent overfitting.

What It’s Used For

* Predicting future values (e.g., stock prices, sales).
* Understanding relationships between variables.
* Making data-driven decisions in business, science, and engineering.

SIMPLE LINEAR REGRESSION

A diagram of equations

AI-generated content may be incorrect.

## MULTIPLE LINEAR REGRESSION

## POLYNOMIAL REGRESSION

## SUPPORT VECTOR REGRESSION

## DECISION TREE REGRESSION

## RANDOM FOREST REGRESSION