What is ML, types of machine Learning (In Data Science).

Machine learning - A subset of artificial intelligence (AI) that focuses on the development of algorithms and statistical models that enable computers to learn and make predictions or decisions without being explicitly programmed. In the context of data science, machine learning plays a crucial role in extracting valuable insights and patterns from large datasets.

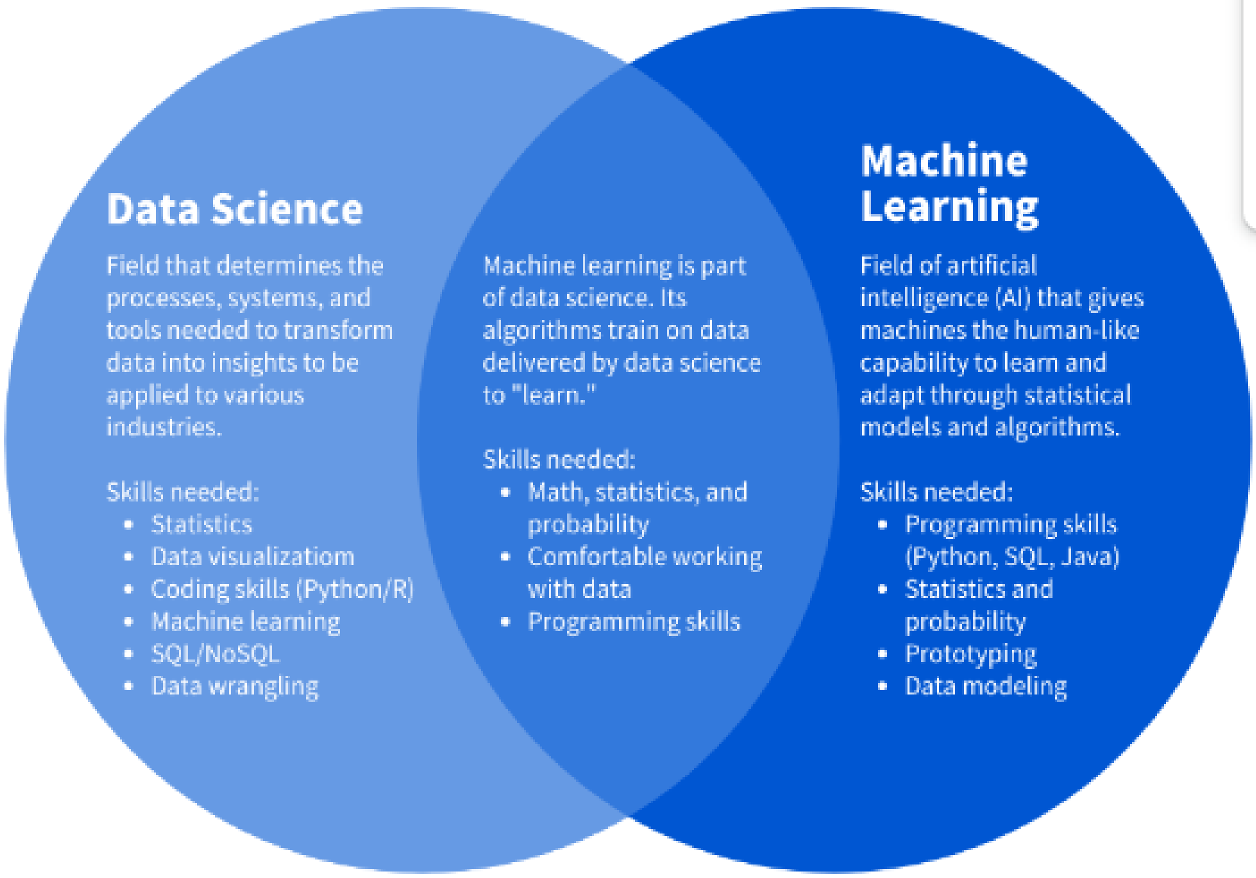
- **Machine learning** focuses on building ML models,
  - while **data science** is the field that works on extracting meaning from data.

Objectives

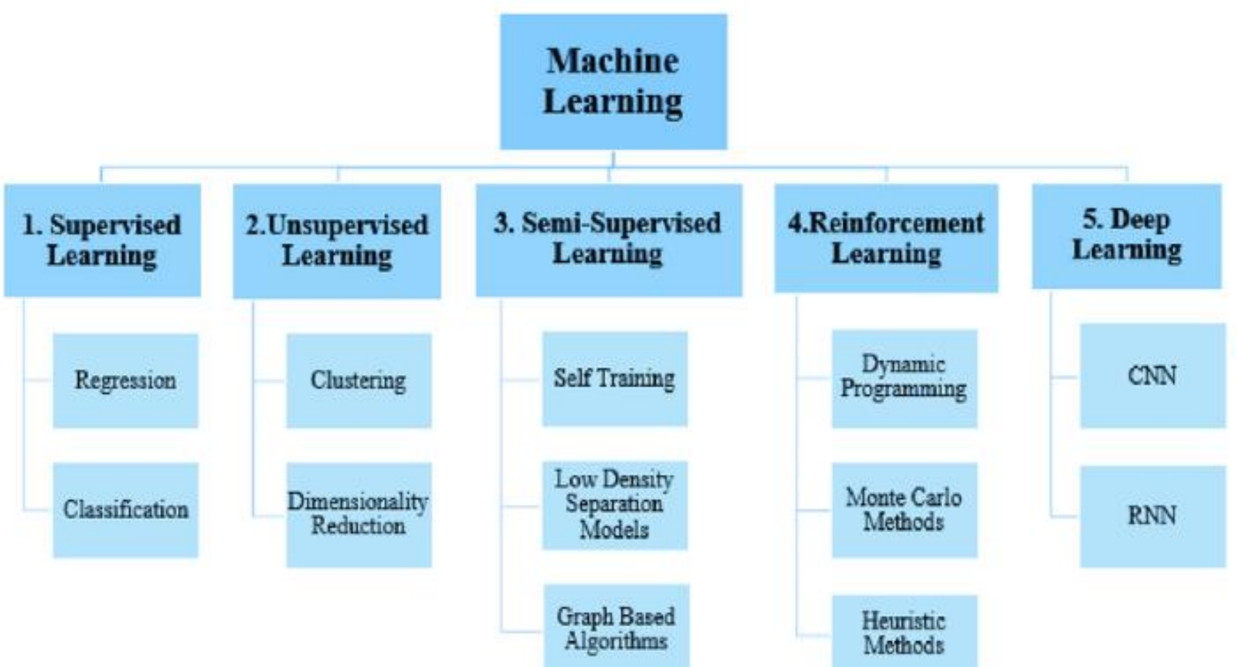
- **To apply** - *The basic principles, models, and algorithms of AI to recognize, model, and solve problems in the analysis and design of information systems.*
- **To discover** - Patterns *in the user data and then make predictions based on these and intricate patterns for answering business questions and solving business problems.*
- **To utilize data** - *For self-learning, eliminating the need to program machines in an explicit manner*

What is data science?

- Data science - is a field that studies data and how to extract meaning from it, using a series of methods, algorithms, systems, and tools to extract insights from structured and unstructured data. That knowledge then gets applied to business, government, and other bodies to help drive profits, innovate products and services, build better infrastructure and public systems, and more.
- Data Science as the Umbrella: Data Science is a broader field that encompasses data collection, cleaning, analysis, and interpretation. It sets the stage by providing the data and insights necessary for Machine Learning.
- Machine Learning as a Tool: Machine Learning, focused on creating predictive models from data. Data Scientists often use Machine Learning techniques to extract valuable patterns and predictions.
- Interdependence: Data Science and Machine Learning are highly interdependent. Data Science relies on Machine Learning to automate and improve decision-making, while Machine Learning relies on Data Science for high-quality data and domain knowledge.



Types of Machine Learning



Machine Learning Type.	Categories	Examples
Supervised learning is a type of machine learning in data science where an algorithm learns from a labeled dataset, which means that the input data is paired with the corresponding correct output or target. The primary goal of supervised learning is to learn a mapping from inputs to outputs, allowing the algorithm to make predictions or decisions based on the input data. In this process, the algorithm generalizes patterns and relationships in the labeled training data, enabling it to make accurate predictions or classifications on new, unseen data.	**-Regression:** where the goal is to predict a continuous numerical value or quantity based on input data. In other words, it's a statistical method used for modeling and analyzing the relationships between a dependent variable (the target or output) and one or more independent variables (the features or inputs) making it particularly useful for prediction and forecasting. Regression algorithms includes Linear Regression, Regression Trees, Non-Linear Regression, Bayesian Linear Regression, Polynomial Regression.	-Predicting the price of a house based on features like size, location, and number of bedrooms. -Estimating the temperature based on historical weather data and other variables.
	-Classification: the model is trained on labeled data, where each data point is associated with input features and a discrete class label. The primary objective is to categorize data points into predefined classes or labels. Classification algorithms are used when the output variable is categorical, which means there are two classes such as Yes-No, Male-Female, True-false, etc.	-Spam email detection, where emails are categorized as spam or not. -Medical diagnosis, such as classifying X-ray images as either normal or showing signs of a specific disease. -Image classification, such as determining whether an image contains a cat, dog, or another object.
Unsupervised learning is a type of machine learning in data science where an algorithm is used to analyze and make sense of unlabeled data, meaning that the data does not have predefined	**Clustering** aims to group similar data points together based on their inherent patterns or similarities, without any predefined	-Customer segmentation: Grouping customers based on purchasing behavior.

categories or target values. Instead of making predictions or classifications, the primary goal of unsupervised learning is to uncover hidden patterns, structures, and relationships within the data.	categories or labels. Clustering can uncover natural clusters in data. **Some of the popular clustering algorithms are** K-Means Clustering algorithm, Mean-shift algorithm, DBSCAN Algorithm, Principal Component Analysis, Independent Component Analysis.	-Document clustering: Categorizing text documents into topics. -Anomaly detection: Identifying unusual patterns in data (outlier detection).
	Dimensionality reduction techniques are used to reduce the number of features (dimensions) in a dataset while preserving important information. This simplifies complex data, aids visualization, and mitigates the "curse of dimensionality."	-Principal Component Analysis (PCA): Reducing the dimensionality of data for visualization or noise reduction. -t-Distributed Stochastic Neighbor Embedding (t-SNE): Visualizing high-dimensional data.
	Association rule learning is an unsupervised learning technique, which finds interesting relations among variables within a large dataset. The main aim of this learning algorithm is to find the dependency of one data item on another data item and map those variables accordingly so that it can generate maximum profit. It helps discover relationships between variables. Some popular algorithms of Association rule learning are Apriori Algorithm, Eclat, FP-growth algorithm.	-Market basket analysis: Identifying product associations in retail, e.g., "customers who bought A also bought B." -Recommender systems: Recommending products, movies, or content based on user behavior and preferences. -Web usage mining: Analyzing patterns in user navigation and website usage.
Semi-supervised learning is a type of machine learning in data science that	**In self-training**, the process begins with a small amount	-Text classification: Starting with a small set of manually

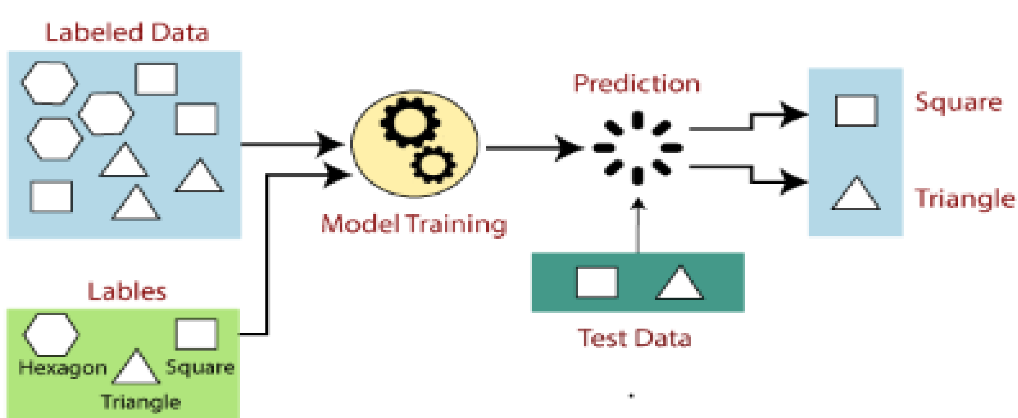
falls between the categories of supervised learning and unsupervised learning. In semi-supervised learning, the algorithm is trained on a dataset that consists of a combination of labeled and unlabeled data. This approach is often used when obtaining a large quantity of labeled data is challenging or expensive, while there is a relatively abundant supply of unlabeled data.	of labeled data, and a model is trained on this data. The model is then used to make predictions on unlabeled data, and the most confident predictions are added to the labeled dataset. This iterative process continues to refine the model.	labeled text data and iteratively adding confidently predicted text samples to the labeled dataset to train a better text classifier. -Speech recognition: Using a small set of transcribed speech data and iteratively refining the acoustic model.
	Low-density separation models are employed when there are sparse clusters or classes in the data, and labeled examples are scarce. These models aim to identify and separate these sparsely populated regions.	-Anomaly detection in network security: Identifying unusual network behavior or attacks when legitimate behavior forms the majority, and attacks are rare. -Rare disease diagnosis: Detecting and diagnosing rare medical conditions with limited labeled cases.
	Graph-based semi-supervised learning methods use relationships or connections between data points (nodes) to propagate labels from labeled data points to unlabeled data points. The data points and their relationships are often represented as a graph or network.	-Social network analysis: Predicting user preferences, friend recommendations, or community detection based on network connections. -Citation networks: Predicting the importance or relevance of research papers based on citation patterns and relationships. -Image segmentation: Labeling pixels in an image based on spatial relationships between neighboring pixels.
Reinforcement learning is a type of machine learning in data science that focuses on training intelligent agents to	**Dynamic Programming** methods involve solving reinforcement learning	-Game playing: In chess, a game state can be considered a subproblem,

make sequential decisions in an environment to achieve a specific goal. In reinforcement learning, an agent interacts with its environment, learns from these interactions, and takes actions to maximize a cumulative reward over time. This learning process is similar to how humans and animals learn through trial and error.	problems by breaking them down into smaller subproblems and finding optimal solutions for each subproblem. These methods are effective but are typically applied to problems with known models and relatively small state and action spaces.	and dynamic programming can be used to compute optimal moves. -Robotics: Planning and controlling the motion of a robot in a controlled environment.
	Monte Carlo methods are used to estimate the value of states or state-action pairs by averaging over a large number of sample episodes. These methods do not require a known model of the environment and are often used in problems with unknown dynamics.	-Game playing: In a board game like Backgammon, Monte Carlo methods can be used to estimate the value of moves by simulating many game sequences. -Autonomous driving: Estimating the value of different driving actions based on past experiences.
	Heuristic methods involve using domain-specific rules or strategies to make decisions in a reinforcement learning setting. These methods often incorporate expert knowledge or heuristics to guide the learning process.	-Game playing: In computer game AI, heuristics can be used to make strategic decisions in real-time strategy games. -Healthcare: Developing heuristics for treatment plans in medical applications based on expert medical knowledge. -Finance: Designing trading algorithms that incorporate heuristics based on market trends and financial indicators.
Deep learning is a subset of machine learning and a specialized field of artificial intelligence (AI) that focuses on training artificial neural networks to perform tasks. These neural networks	**Convolutional Neural Networks (CNN)** - CNNs are primarily designed for processing grid-structured data, such as images and	-Image classification: Identifying objects within images, e.g., recognizing cats or dogs in photographs.

consist of multiple layers of interconnected nodes (artificial neurons) and are inspired by the structure and function of the human brain. Deep learning has gained significant attention and popularity due to its ability to automatically learn features and patterns from data and make complex predictions or decisions.	videos. They use convolutional layers to automatically learn hierarchical features and patterns from data.	-Object detection: Locating and classifying multiple objects in images, such as in self-driving car applications. -Image segmentation: Labeling each pixel in an image, used in medical image analysis or satellite imagery.
	Recurrent Neural Networks (RNN) - RNNs are suited for processing sequences of data, making them ideal for tasks involving natural language, time series, and sequential data. RNNs have feedback loops that allow them to maintain internal state and handle sequential dependencies.	-Natural Language Processing (NLP): Language translation, sentiment analysis, and chatbot interactions. -Time series analysis: Predicting stock prices, weather forecasting, and demand forecasting. -Speech recognition: Transcribing spoken language into text. - Music generation: Creating music compositions that follow a temporal structure.

1. *Supervised learning.*

- Is a process of providing **input data** as well as **correct output** data to the machine learning model? **The aim of a supervised learning algorithm is to find a mapping function to map the input variable(x) with the output variable(y).**



Advantages of Supervised learning:

- With the help of supervised learning, the model can predict the output on the basis of prior experiences.
- In supervised learning, we can have an exact idea about the classes of objects.
- Supervised learning model helps us to solve various real-world problems such as fraud detection, spam filtering, etc.

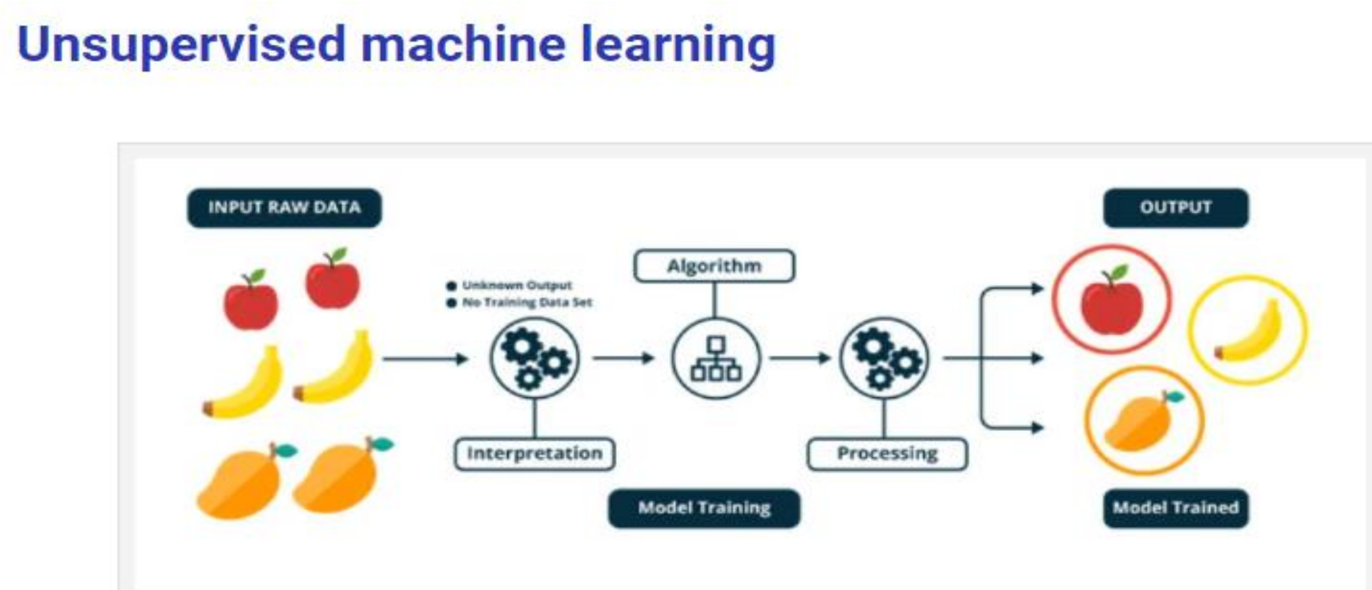
Disadvantages of supervised learning:

- Supervised learning models are not suitable for handling the complex tasks.
- Supervised learning cannot predict the correct output if the test data is different from the training dataset.
- Training required lots of computation times.
- In supervised learning, we need enough knowledge about the classes of object.

2. *Unsupervised Machine Learning.*

In unsupervised learning, the models are trained with the data that is neither classified nor labelled, and the model acts on that data without any supervision.

The main aim of the unsupervised learning algorithm is to group or categories the unsorted dataset according to the similarities, patterns, and differences. Machines are instructed to find the hidden patterns from the input dataset.



Advantages and Disadvantages of Unsupervised Learning Algorithm

Advantages:

- These algorithms can be used for complicated tasks compared to the supervised ones because these algorithms work on the unlabeled dataset.
- Unsupervised algorithms are preferable for various tasks as getting the unlabeled dataset is easier as compared to the labelled dataset.

Disadvantages:

- The output of an unsupervised algorithm can be less accurate as the dataset is not labelled, and algorithms are not trained with the exact output in prior.
- Working with Unsupervised learning is more difficult as it works with the unlabeled dataset that does not map with the output.

Applications of Unsupervised Learning

- ✓ **Network Analysis**: *Unsupervised learning is used for identifying plagiarism and copyright in document network analysis of text data for scholarly articles.*

- ✓ **Recommendation Systems:** *Recommendation systems widely use unsupervised learning techniques for building recommendation applications for different web applications and e-commerce websites.*
- ✓ **Anomaly Detection:** *Anomaly detection is a popular application of unsupervised learning, which can identify unusual data points within the dataset. It is used to discover fraudulent transactions*.

- ✓ **Singular Value Decomposition:** *Singular Value Decomposition or SVD is used to extract particular information from the database. For example, extracting information of each user located at a particular location.*

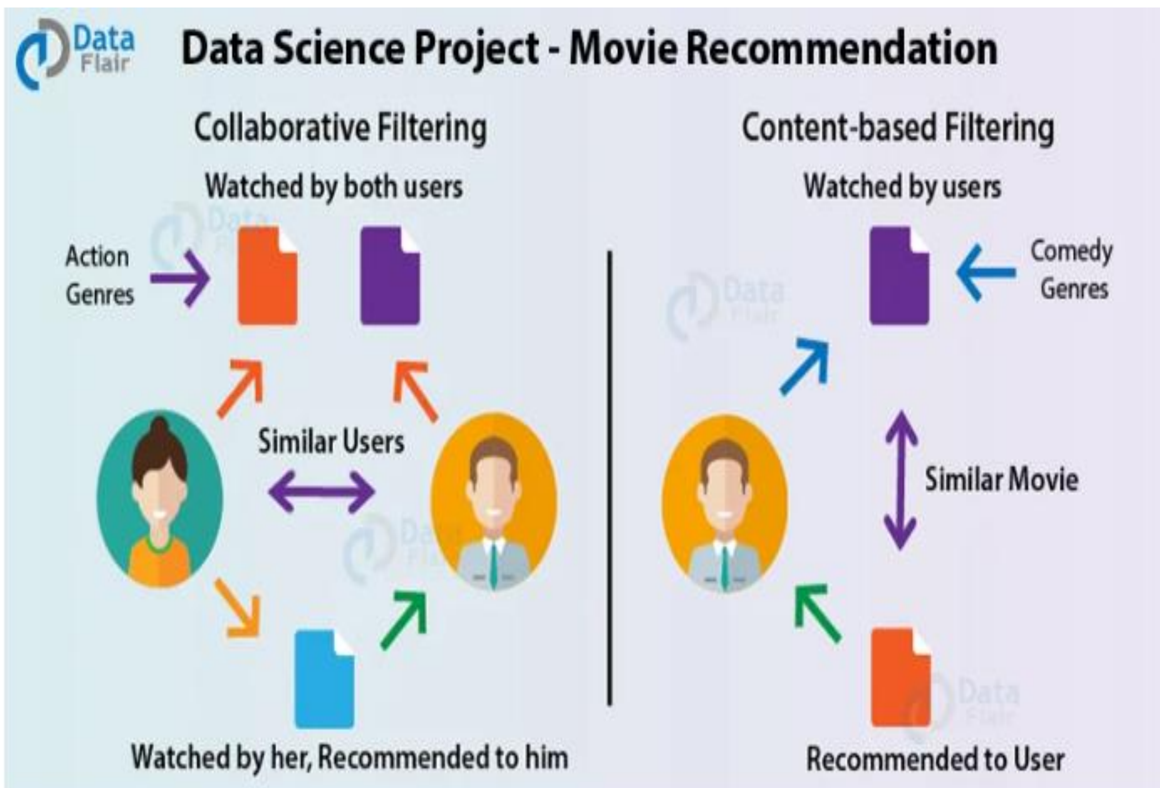
3. *Semi supervised Machine Learning.*

A recommendation *system:* In a recommendation system, the initial training of the system can involve supervised learning, where the system is trained on historical data that includes information about user preferences and item attributes (such as user ratings or purchase history). This data is used to predict user preferences for items or make recommendations. This part of the system can be seen as a classification or regression task in supervised learning.

However, recommendation systems also incorporate unsupervised learning techniques, such as collaborative filtering and matrix factorization, to discover patterns and similarities in user-item interactions without relying on explicit labels or ratings. These unsupervised techniques help in making recommendations based on user behavior and preferences without the need for labeled data.

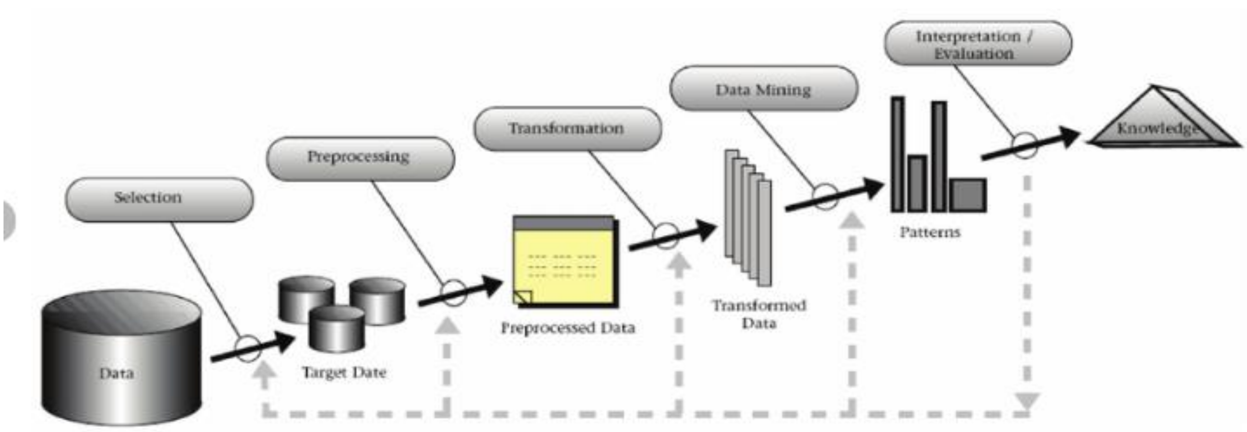
- **A recommendation** *system provides suggestions to the users through a filtering process that is based on user preferences and browsing history*.

- Recommendation engines are a subclass of machine learning which generally deal with ranking or rating products / users. Loosely defined, a recommender system is **a system which predicts ratings a user might give to a specific item**. These predictions will then be ranked and returned back to the user.



The Knowledge Discovery in Databases (KDD) process:

The Knowledge Discovery in Databases (KDD) process refers to a systematic and iterative approach to extracting useful knowledge, information, and insights from large volumes of data. It is a multidisciplinary field that combines techniques from data mining, machine learning, statistics, and database management to transform raw data into actionable knowledge.



KDD Process Model, adapted from Fayyad et al. (1996)

1) *Selection:* In this initial step, the relevant data is selected from various sources. This involves defining the scope and objectives of the data mining project, including what data is needed to achieve these objectives. Selection also includes data collection, where you gather and acquire the data required for analysis.
2) *Preprocessing:* Once the data is collected, it typically needs to be preprocessed. This step involves cleaning the data by addressing issues such as missing values, outliers, and noise. Data cleaning ensures that the data is accurate and complete. It may also involve data integration, where data from multiple sources is combined into a unified dataset.
3) *Transformation:* Data transformation is about preparing the data for analysis. This may include normalizing or scaling the data to make it more suitable for certain algorithms. Additionally, transformation can involve encoding categorical variables and reducing dimensionality to improve computational efficiency.
4) *Data Mining:* Data mining is the heart of the KDD process. It involves applying various algorithms and techniques to discover patterns, relationships, or insights within the data. Common data mining tasks include classification, clustering, association rule mining, and regression.
5) *Interpretation/Evaluation:* After data mining, the results need to be interpreted and evaluated. This step includes assessing the quality and significance of the patterns or models discovered. Evaluation often involves the use of metrics and visualizations to understand the value of the findings.
6) *Knowledge:* The ultimate goal of KDD is to generate knowledge that can be used for decision-making or problem-solving. The knowledge derived from the data mining process is often what provides insights, informs strategies, and supports informed decisions. This knowledge can be used to address the objectives defined in the selection phase.

Linear Regression Implementation:

Imagine you have data with two columns: X and Y. You want to know how Y is influenced by changes in X. Linear regression helps you find a straight line (a "linear" relationship) that best fits the data points. This line is called the regression line.

The regression line has two key components:

Slope (m): It represents how much Y changes for a one-unit change in X. If the slope is positive, an increase in X leads to an increase in Y. If the slope is negative, an increase in X leads to a decrease in Y.

Intercept (b): This is where the line crosses the Y-axis. It represents the starting value of Y when X is zero.

The regression equation looks like this:

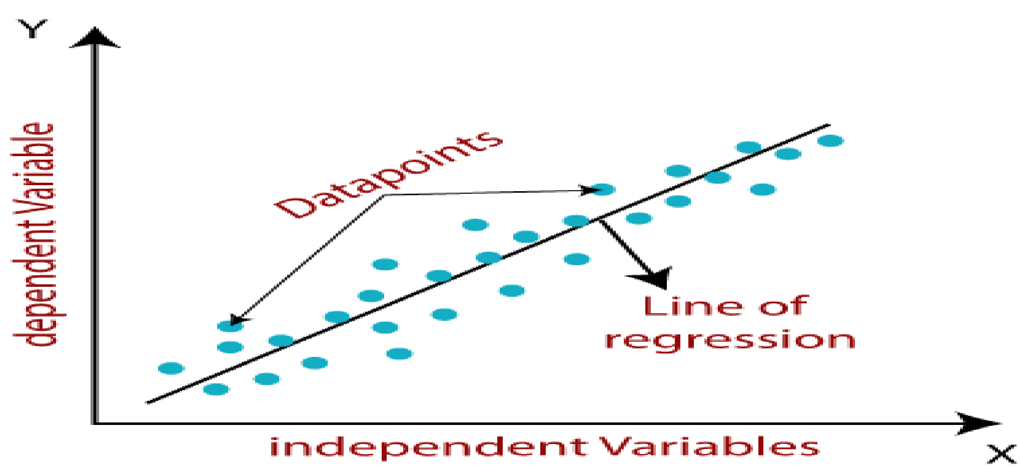
$Y = m * X + b$

With this equation, you can make predictions. Given a value of X, you can use the equation to estimate the corresponding Y.

In summary, linear regression helps you:

- ✓ Understand the relationship between variables.
- ✓ Make predictions based on that relationship.
- ✓ Identify the strength and direction of the relationship (positive or negative).

The linear regression model provides a sloped straight line representing the relationship between the variables. Consider the below image:



Linear Regression Formula

Linear regression shows the linear relationship between two variables. The equation of linear regression is similar to the slope formula what we have learned before in earlier classes such as linear equations in two variables. It is given by;

Y= a + bX

Now, here we need to find the value of the slope of the line, b, plotted in scatter plot and the intercept, a.

$$a = \frac{[(\sum y)(\sum x^2) - (\sum x)(\sum xy)]}{[n(\sum x^2) - (\sum x)^2]}$$

$$b = \frac{[n(\sum xy) - (\sum x)(\sum y)]}{[n(\sum x^2) - (\sum x^2)]}$$

$$a\ (intercept) = \frac{\sum y \sum x^2 - \sum x \sum xy}{(\sum x^2) - (\sum x)^2}$$

$$b\ (slope) = \frac{n \sum xy - (\sum x)(\sum y)}{n \sum x^2 - (\sum x)^2}$$

Where,

x and y are two variables on the regression line.

b = Slope of the line.

a = y-intercept of the line.

x = Values of the first data set.

y = Values of the second data set.

The slope indicates the steepness of a line and the intercept indicates the location where it intersects an axis

Solved Examples

Question: Find linear regression equation for the following two sets of data:

x	2	4	6	8
y	3	7	5	10

Solution:

Construct the following table:

x	y	x^2	xy
2	3	4	6
4	7	16	28
6	5	36	30
8	10	64	80
$\sum x$ = 20	$\sum y$ = 25	$\sum x^2$ = 120	$\sum xy$ = 144

$b = \frac{n\sum xy - (\sum x)(\sum y)}{n\sum x^2 - (\sum x)^2}$

$b = \frac{4\times 144 - 20\times 25}{4\times 120 - 400}$

b = 0.95

$$a = \frac{\sum y \sum x^2 - \sum x \sum xy}{n(\sum x^2) - (\sum x)^2}$$

$$a = \frac{25\times 120 - 20\times 144}{4(120) - 400}$$

a = 1.5

Linear regression is given by:

y = a + bx

y = 1.5 + 0.95 x

How to... Perform Simple Linear Regression by Hand Video Link…
https://www.youtube.com/watch?v=GhrxgbQnEEU

Hands-on: Linear Regression Using Python Scikit learn Hands-on-: Boston Housing Prices Dataset.

- **Environment:** Google Colab (Python)
- **Library:** Pandas
- **Module:** Scikit-learn

Step 1: Load the Boston dataset and Have a glance at the shape.

Boston dataset
https://www.kaggle.com/datasets/simpleparadox/bostonhousingdataset

This data frame contains following columns:

- **Crim:** Per capita crime rate by town
- **Zn:** Proportion of residential land zoned for lots over 25,000 sq. ft.
- **Indus:** Proportion of non-retail business acres per town
- **Chas:** Charles River dummy variable (= 1 if tract bounds river; 0, otherwise)
- **Nox:** Nitrogen oxides concentration (parts per 10 million)
- **Rm:** Average number of rooms per dwelling
- **Age:** Proportion of owner-occupied units built before 1940
- **Dis:** Weighted mean of distances to five Boston employment centers
- **Rad:** Index of accessibility to radial highways
- **Tax:** Full-value property tax rate per $10,000
- **Ptratio:** Pupil–Teacher ratio by town
- **Black:** 1000(Bk – 0.63) ^2, where Bk is the proportion of Blacks by town
- **Lstat:** Lower status of the population (percent)
- **Medv:** Median value of owner-occupied homes in $1000s

- *Now that we are familiar with the dataset, let us build the Python linear regression models.*
- *Consider '**Lstat**' as independent and '**Medv**' as dependent variables*

+ Code + Text

```
import pandas as pd
wainaina = pd.read_csv('/content/HousingData.csv')
print("Shape of the dataset:", wainaina.shape)
wainaina.head()
```

Shape of the dataset: (506, 15)

	Unnamed: 0	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.0
1	1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
2	2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7
3	3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.4
4	4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.2

Line	Explanation
import pandas as pd	This line imports the pandas library and gives it the alias **pd**. Pandas is a popular data manipulation library in Python, and **pd** is a common alias for it.
wainaina = pd.read_csv('/content/HousingData.csv')	This line reads a CSV file named "HousingData.csv" located at the path '/content/' and loads its data into a pandas DataFrame, which is assigned to the variable wainaina. This assumes that you have a CSV file named "HousingData.csv" in the '/content/' directory.
print("Shape of the dataset:", wainaina.shape)	This line prints the shape of the DataFrame wainaina. The shape of a DataFrame is a tuple that represents its dimensions, where the first element is the number of rows (samples) and the second element is the

	number of columns (features or variables). This line displays the message "Shape of the dataset:" followed by the shape (e.g., (506, 14)).
wainaina.head()	This line displays the first few rows of the DataFrame wainaina using the **.head()** method. By default, it shows the first 5 rows of the DataFrame, allowing you to quickly inspect the data's structure and content.

Step 2: Have a glance at the dependent and independent variables.

+ Code + Text

```
import pandas as pd
wainaina = pd.read_csv('/content/HousingData.csv')
data=wainaina.loc[:,['LSTAT','MEDV']]
data.head(5)
```

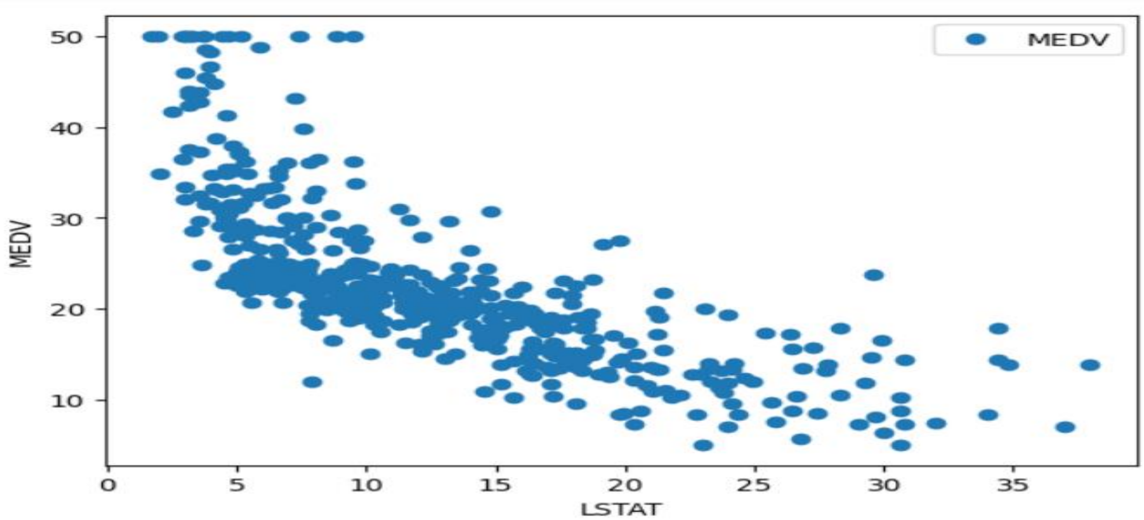
	LSTAT	MEDV
0	4.98	24.0
1	9.14	21.6
2	4.03	34.7
3	2.94	33.4
4	5.33	36.2

Line	Explanation
import pandas as pd	This line imports the pandas library and gives it the alias **pd**. Pandas is a data manipulation library in Python.
wainaina = pd.read_csv('/content/HousingData.csv')	This line reads a CSV file named "HousingData.csv" located at the path '/content/' and loads its data into a pandas DataFrame, which is assigned to the variable **wainaina**. The DataFrame will contain all columns from the CSV file.
data = wainaina.loc[:, ['LSTAT', 'MEDV']]	This line extracts specific columns from the DataFrame **wainaina**. It selects two columns, 'LSTAT' and 'MEDV', using the **.loc** method and assigns the resulting DataFrame to the variable **data**. The **:** before the comma means that we want all rows.
data.head(5)	This line displays the first five rows of the DataFrame **data** using the **.head(5)** method. It allows you to quickly examine the content of the selected columns in the DataFrame. It's a useful way to get an initial sense of the data.

Step 3: Visualize the change in the variables.

```
+ Code   + Text
import pandas as pd
import matplotlib.pyplot as plt
wainaina = pd.read_csv('/content/HousingData.csv')
wainaina.plot(x='LSTAT',y='MEDV',style='o')
plt.xlabel('LSTAT')
plt.ylabel('MEDV')
plt.show()
```

Line	Explanation
import pandas as pd	This line imports the pandas library, giving it the alias **pd**.
import matplotlib.pyplot as plt	This line imports the Matplotlib library for data visualization and gives it the alias **plt**.
wainaina = pd.read_csv('/content/HousingData.csv')	This line reads a CSV file named "HousingData.csv" located at the path '/content/' and loads its data into a pandas DataFrame, which is assigned to the variable **wainaina**.
wainaina.plot(x='LSTAT', y='MEDV', style='o')	This line creates a scatter plot using the **.plot()** method of the DataFrame **wainaina**. It specifies 'LSTAT' as the x-axis variable and 'MEDV' as the y-axis variable. The **style='o'** argument specifies that the data points should be plotted as circles ('o').
plt.xlabel('LSTAT')	This line sets the label for the x-axis to 'LSTAT' in the plot.
plt.ylabel('MEDV')	This line sets the label for the y-axis to 'MEDV' in the plot.
plt.show()	This line displays the plot on the screen. It's necessary to call **plt.show()** to visualize the plot in most Python environments.



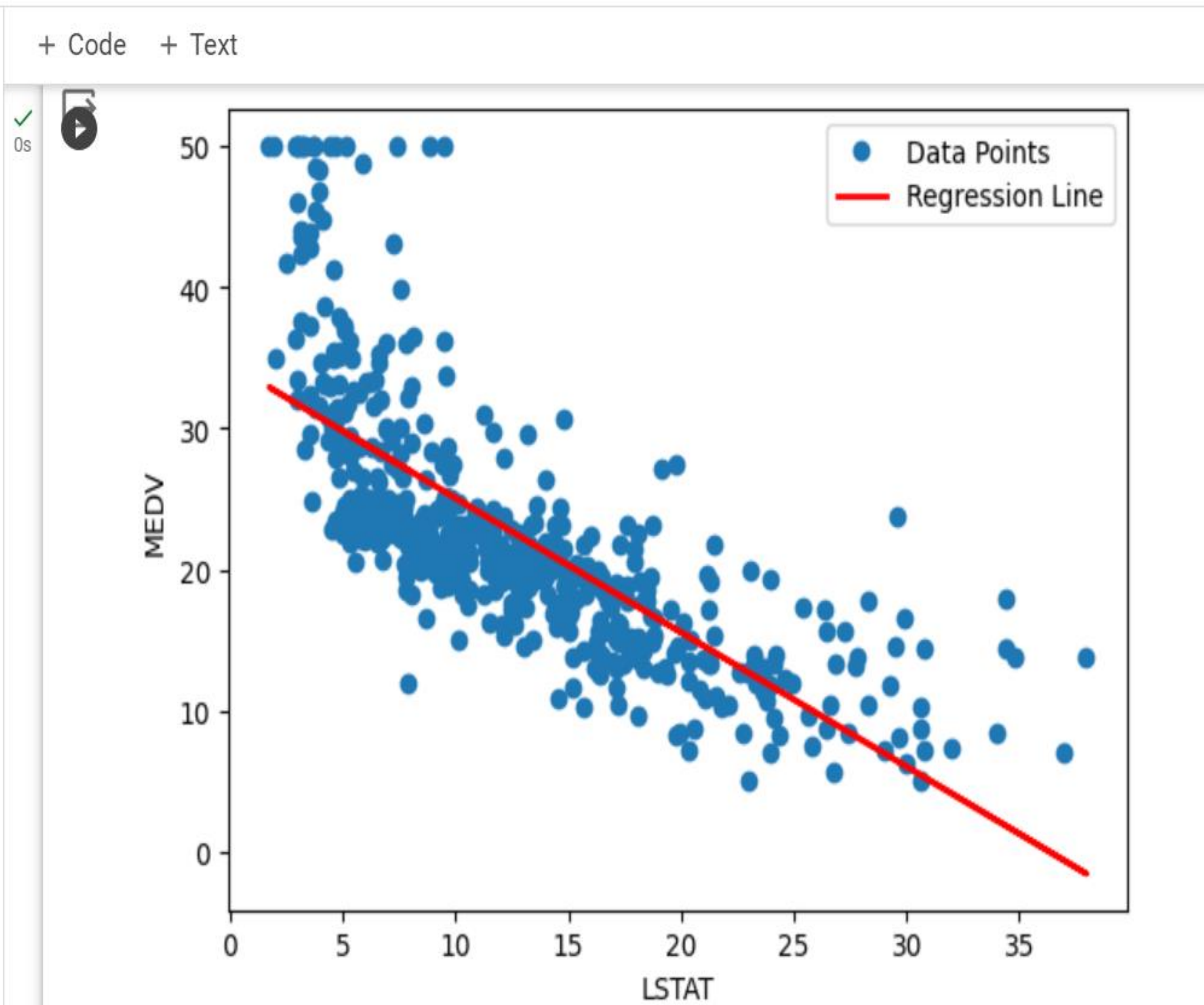
Adding a Regression Line:

+ Code + Text

```python
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
# Load the data
wainaina = pd.read_csv('/content/HousingData.csv')
# Extract the independent variable (X) and dependent variable (y)
X = wainaina[['LSTAT']]
y = wainaina['MEDV']
# Create and fit a linear regression model
regressor = LinearRegression()
regressor.fit(X, y)
# Plot the data points
wainaina.plot(x='LSTAT', y='MEDV', style='o', label='Data Points')
# Overlay the regression line
plt.plot(X, regressor.predict(X), color='red', linewidth=2, label='Regression Line')
plt.xlabel('LSTAT')
plt.ylabel('MEDV')
plt.legend()
plt.show()
```

Line	Explanation
import pandas as pd	Imports the Pandas library and assigns it the alias **pd**. Pandas is used for data manipulation and analysis.
import matplotlib.pyplot as plt	Imports the Matplotlib library and assigns it the alias **plt**. Matplotlib is a popular library for creating data visualizations, including plots and charts.
from sklearn.linear_model import LinearRegression	Imports the LinearRegression class from scikit-learn, a library for machine learning in Python.
wainaina = pd.read_csv('/content/HousingData.csv')	Reads a CSV file named 'HousingData.csv' located at the specified path ('/content/HousingData.csv') into a Pandas DataFrame named **wainaina**. It loads the dataset for further analysis.
X = wainaina[['LSTAT']]	Extracts the independent variable 'LSTAT' from the DataFrame and assigns it to the variable **X**.
y = wainaina['MEDV']	Extracts the dependent variable 'MEDV' from the DataFrame and assigns it to the variable **y**.
regressor = LinearRegression()	Creates an instance of the LinearRegression class from scikit-learn and assigns it to the variable **regressor**. This object will be used to fit a linear regression model.
regressor.fit(X, y)	Fits the linear regression model using the independent variable **X** and dependent variable **y**. The model is trained to predict 'MEDV' based on 'LSTAT'.
wainaina.plot(x='LSTAT', y='MEDV', style='o', label='Data Points')	Creates a scatter plot using the Pandas DataFrame **wainaina**. The 'LSTAT' column is used as the x-

	axis (independent variable), 'MEDV' as the y-axis (dependent variable), and 'o' as the style to plot data points as circles. A label 'Data Points' is added to the plot.
plt.plot(X, regressor.predict(X), color='red', linewidth=2, label='Regression Line')	Plots the regression line on the same plot. It uses the model's predictions for 'MEDV' based on 'LSTAT' (using **regressor.predict(X)**) as the y-values. The regression line is drawn in red with a linewidth of 2, and it is labeled 'Regression Line'.
plt.xlabel('LSTAT')	Sets the label for the x-axis of the plot to 'LSTAT'.
plt.ylabel('MEDV')	Sets the label for the y-axis of the plot to 'MEDV'.
plt.legend()	Adds a legend to the plot to distinguish between 'Data Points' and 'Regression Line'.
plt.show()	Displays the plot. This line is necessary to visualize the scatter plot and the regression line in the current environment.



Step 4: Divide the data into independent and dependent variables.

+ Code + Text

```
import pandas as pd
wainaina = pd.read_csv('/content/HousingData.csv')
x=pd.DataFrame(data['LSTAT'])
y=pd.DataFrame(data['MEDV'])
```

Line	Explanation
import pandas as pd	Imports the pandas library and assigns it the alias **pd**. Pandas is used for data manipulation and analysis.
wainaina = pd.read_csv('/content/HousingData.csv')	Reads a CSV file named "HousingData.csv" located at the path '/content/' and loads its data into a pandas DataFrame, which is assigned to the variable **wainaina**.
x = pd.DataFrame(data['LSTAT'])	Creates a new DataFrame **x** by extracting the 'LSTAT' column from a DataFrame named **data**. Note that **data** is not defined in the provided code, so this line will result in an error. You might want to replace **data** with **wainaina** to correctly extract the 'LSTAT' column from the **wainaina** DataFrame.
y = pd.DataFrame(data['MEDV'])	Similarly, creates a new DataFrame **y** by extracting the 'MEDV' column from a DataFrame named **data**. Again, you should replace **data** with **wainaina** to correctly extract the 'MEDV' column from the **wainaina** DataFrame.

Step 5: Split the data into train and test sets.

+ Code + Text

```
import pandas as pd
from sklearn.model_selection import train_test_split
wainaina = pd.read_csv('/content/HousingData.csv')
x=pd.DataFrame(data['LSTAT'])
y=pd.DataFrame(data['MEDV'])
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=1)
```

Line	Explanation
import pandas as pd	Imports the pandas library for data manipulation.
from sklearn.model_selection import train_test_split	Imports the **train_test_split** function from scikit-learn, which is used to split data into training and testing sets.
wainaina = pd.read_csv('/content/HousingData.csv')	Reads a CSV file named "HousingData.csv" located at the path '/content/' and loads its data into a pandas DataFrame, which is assigned to the variable **wainaina**.

x = pd.DataFrame(wainaina['LSTAT'])	Creates a new DataFrame **x** by extracting the 'LSTAT' column from the **wainaina** DataFrame.
y = pd.DataFrame(wainaina['MEDV'])	Creates a new DataFrame **y** by extracting the 'MEDV' column from the **wainaina** DataFrame.
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=1)	Uses the **train_test_split** function to split the data. **x** is used as the independent variable, **y** is used as the dependent variable. The data is split into training and testing sets. The **test_size** parameter specifies that 20% of the data will be used for testing, and the **random_state** parameter is set to 1 to ensure reproducibility. The resulting sets are stored in **X_train**, **X_test**, **y_train**, and **y_test**.

Step 6: Shape of the train and test sets.

+ Code + Text

```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(404, 1)
(102, 1)
(404, 1)
(102, 1)
```

Line	Explanation
print(X_train.shape)	Prints the shape (number of rows and columns) of the training set for the independent variable (**X_train**). This shows how many samples (rows) and features (columns) are in the training set for the independent variable.
print(X_test.shape)	Prints the shape of the testing set for the independent variable (**X_test**). It indicates the number of samples (rows) and features (columns) in the testing set for the independent variable.
print(y_train.shape)	Prints the shape of the training set for the dependent variable (**y_train**). This provides the number of samples (rows) in the training set for the dependent variable.
print(y_test.shape)	Prints the shape of the testing set for the dependent variable (**y_test**). It shows the number of samples (rows) in the testing set for the dependent variable.

Step 7: ***Train the algorithm.***

+ Code + Text

```python
from sklearn.linear_model import LinearRegression
regressor=LinearRegression()
regressor.fit(X_train,y_train)
```

```
▾ LinearRegression
LinearRegression()
```

Line	Explanation
from sklearn.linear_model import LinearRegression	This line imports the **LinearRegression** class from scikit-learn, which is used to create a linear regression model.
regressor = LinearRegression()	This line creates an instance of the **LinearRegression** model and assigns it to the variable **regressor**.
regressor.fit(X_train, y_train)	This line trains the linear regression model (**regressor**) with the training data. The **fit** method takes the independent variable (**X_train**) and the corresponding dependent variable (**y_train**) as arguments to fit the model. The model will learn the relationship between the independent and dependent variables during this training phase.

Step 8: ***Retrieve the intercept.***

+ Code + Text

```python
print(regressor.intercept_)
```

```
[34.33497839]
```

Line	Explanation
print(regressor.intercept_)	This line prints the intercept of a linear regression model, which is stored in the **intercept_** attribute of the **regressor** object. The intercept represents the value of the dependent variable when all independent variables are set to zero. In a linear regression equation ($y = mx + b$), this is the "b" or bias term. The **print** function is used to display the intercept value.

Step 9: ***Retrieve the slope.***

+ Code + Text

```python
print(regressor.coef_)
```

```
[[-0.92441715]]
```

Line	Explanation
print(regressor.coef_)	This line prints the coefficients (slopes) of a linear regression model. The coefficients represent the effect of each independent variable on the dependent variable. In a multiple linear regression model, like y = b0 + b1*x1* + *b2*x2 + ... + bn*xn, the **regressor.coef_** attribute contains the values of b1, b2, ..., bn, which correspond to the coefficients for the respective independent variables (x1, x2, ..., xn). The **print** function is used to display these coefficient values.

Step 10: Predicted Values.

```
+ Code    + Text

y_pred = regressor.predict(X_test)
y_pred

array([[27.37411725],
       [27.69766325],
       [16.95593597],
       [26.84719947],
       [24.91516763],
       [24.05545968],
       [29.99021779],
       [22.28057875],
       [17.76942306],
       [26.1908633 ],
       [27.17998965],
       [30.07341533],
       [21.75366098],
       [24.86894677],
       [23.50080939],
       [23.12179836],
```

Line	Explanation
y_pred = regressor.predict(X_test)	This line is used to make predictions using a trained linear regression model (regressor). Specifically, it's using the .predict() method of the linear regression model to generate predicted values for the dependent variable (y_pred) based on the independent variable values in the testing set (X_test). The X_test dataset contains the input values for which you want to predict the corresponding output values. In essence, it's applying the learned relationship between the independent and dependent variables from the training data to make predictions on new, unseen data.
y_pred	This line, when executed, displays the array of predicted values for the dependent variable. These values represent the model's estimations based on the independent variables in the testing dataset. It allows you to examine and work with the model's predictions.

Step 11: Actual Values.

```
#y_test
#print(y_test)
y_test.head(10)
```

	MEDV
307	28.2
343	23.9
47	16.6
67	22.0
362	20.8
132	23.0
292	27.9
31	14.5
218	21.5
90	22.6

Line	Explanation
y_test.head(10)	This line retrieves and displays the first 10 rows of the variable **y_test**. In this context, **y_test** typically contains the actual or observed values of the dependent variable for the testing dataset. Using the **.head(10)** method, you are displaying the first 10 values to examine the actual values that the model's predictions will be compared against. This helps in evaluating the model's performance by comparing predicted values (**y_pred**) to the actual values (**y_test**).

Step 12: Evaluate the algorithm.

```
from sklearn import metrics
import numpy as np
# Assuming y_test and y_pred are NumPy arrays or Pandas Series
mae = metrics.mean_absolute_error(y_test, y_pred)
mse = metrics.mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)  # RMSE is the square root of MSE
print("Mean Absolute Error (MAE):", mae)
print("Mean Squared Error (MSE):", mse)
print("Root Mean Squared Error (RMSE):", rmse)
```

```
Mean Absolute Error (MAE): 5.078127727696937
Mean Squared Error (MSE): 46.994820919547124
Root Mean Squared Error (RMSE): 6.855276866731724
```

1. **Mean Absolute Error (MAE)**:
   - **MAE** measures the average absolute difference between the actual (observed) values and the predicted values.
   - Formula: MAE = (1 / n) * Σ |actual - predicted|
   - Interpretation: On average, the model's predictions are off by approximately 5.078 units. It provides a sense of the magnitude of errors in the model's predictions without considering their direction.

2. **Mean Squared Error (MSE)**:
   - **MSE** measures the average of the squared differences between the actual values and the predicted values. It penalizes larger errors more than MAE.
   - Formula: MSE = (1 / n) * Σ (actual - predicted)^2
   - Interpretation: On average, the model's predictions result in an MSE of approximately 46.995. Squaring the differences emphasizes the impact of larger errors, which is why MSE is often higher than MAE.

3. **Root Mean Squared Error (RMSE)**:
   - **RMSE** is the square root of MSE and provides a more interpretable measure of error in the same units as the dependent variable. It's a widely used metric to evaluate the accuracy of regression models.
   - Formula: RMSE = √MSE
   - Interpretation: The RMSE of approximately 6.855 indicates that, on average, the model's predictions are off by around 6.855 units, which is more interpretable than the squared values of MSE. RMSE has the same units as the dependent variable, making it easier to understand the size of the errors.

Line	Explanation
from sklearn import metrics	This line imports the **metrics** module from scikit-learn (a popular machine learning library). The **metrics** module provides various evaluation metrics for assessing the performance of machine learning models.
import numpy as np	This line imports the **numpy** library and assigns it the alias **np**. NumPy is a widely used library for numerical and array operations in Python.
mae = metrics.mean_absolute_error(y_test, y_pred)	This line calculates the Mean Absolute Error (MAE) between the actual values in **y_test** and the predicted values in **y_pred**. MAE measures the average absolute

	difference between actual and predicted values, providing an assessment of the model's accuracy. The result is stored in the variable **mae**.
mse = metrics.mean_squared_error(y_test, y_pred)	This line calculates the Mean Squared Error (MSE) between the actual values in **y_test** and the predicted values in **y_pred**. MSE measures the average of the squared differences, emphasizing the impact of larger errors. The result is stored in the variable **mse**.
rmse = np.sqrt(mse)	This line calculates the Root Mean Squared Error (RMSE) by taking the square root of the MSE. RMSE provides a more interpretable measure of error in the same units as the dependent variable. The result is stored in the variable **rmse**.
print("Mean Absolute Error (MAE):", mae)	This line prints the calculated MAE, which quantifies the model's accuracy by measuring the average absolute difference between actual and predicted values.
print("Mean Squared Error (MSE):", mse)	This line prints the calculated MSE, which quantifies the model's performance by measuring the average of the squared differences between actual and predicted values.
print("Root Mean Squared Error (RMSE):", rmse)	This line prints the calculated RMSE, which provides an interpretable measure of prediction error in the same units as the dependent variable. It is the square root of MSE.

How to Save the Linear Regression Model you have Created:

```
import joblib
import pickle  # Import the pickle module
from sklearn.linear_model import LinearRegression
# Train your linear regression model
model = LinearRegression()
model.fit(x, y)  # Replace X and y with your training data
# Save the trained model to a file using joblib
joblib.dump(model, 'linear_regression_model.pkl')
# Alternatively, you can save the model using pickle
with open('linear_regression_model.pkl', 'wb') as file:
    pickle.dump(model, file)
```

Line	Code	Explanation
1	**import joblib**	Import the joblib library for model saving/loading.
2	**import pickle**	Import the pickle module for alternative serialization.
3	**from sklearn.linear_model import LinearRegression**	Import the linear regression model from scikit-learn.

4	**model = LinearRegression()**	Create an instance of the Linear Regression model.
5	**model.fit(x, y)**	Train the model with your training data x and target values y. Replace x and y with your actual training data.
6	**joblib.dump(model, 'linear_regression_model.pkl')**	Save the trained model to a file using joblib. The model will be stored in a file named 'linear_regression_model.pkl'.
7	**with open('linear_regression_model.pkl', 'wb') as file:**	Open a file in binary write mode for saving the model using pickle.
8	**pickle.dump(model, file)**	Save the trained model to the file using pickle.

Using the Trained Model to test more data set:

```
import joblib
import numpy as np
import pandas as pd
# Sample X values for prediction
new_X = np.array([6, 7, 8, 9, 10]).reshape(-1, 1)  # Reshape to a 2D array
# Load the trained model
model = joblib.load('linear_regression_model.pkl')  # Load your trained model here
# Make predictions on the new X values
predictions = model.predict(new_X)
# Create a DataFrame with one-dimensional arrays
new_data = pd.DataFrame({'X': new_X.flatten(), 'Predicted_Y': predictions.flatten()})
# Display the new_data DataFrame with the X values and the predicted Y values
print(new_data)
```

```
    X  Predicted_Y
0   6    28.853545
1   7    27.903495
2   8    26.953446
3   9    26.003397
4  10    25.053347
```

Line	Code	Explanation
1	**import joblib**	Import the joblib library for model saving/loading.
2	**import numpy as np**	Import the numpy library, aliasing it as np.
3	**import pandas as pd**	Import the pandas library, aliasing it as pd.
4	**new_X = np.array([6, 7, 8, 9, 10]).reshape(-1, 1)**	Create a NumPy array new_X with sample values [6, 7, 8, 9, 10], and reshape it into a 2D array with a single column.
5	**model = joblib.load('linear_regression_model.pkl')**	Load the trained linear regression model from the file 'linear_regression_model.pkl' using joblib.
6	**predictions = model.predict(new_X)**	Use the loaded model to make predictions on the new values in new_X.
7	**new_data = pd.DataFrame({'X': new_X.flatten(), 'Predicted_Y': predictions.flatten()})**	Create a DataFrame new_data containing two columns: 'X' (reshaped new_X) and 'Predicted_Y' (reshaped predictions).
8	**print(new_data)**	Display the new_data DataFrame with the X values and the predicted Y values.