# **Lecture Notes**

## [**Lecture 01**](https://docs.google.com/presentation/d/1YRtAG5rLwOEp4qCS_jk7d5w5Z8Ecc1jy/edit?usp=drive_link&ouid=101794246168461768671&rtpof=true&sd=true)

**Roadmap of AI**

**Agenda**

→What is AI

→AI Value creation

→Machine Learning vs Deep Learning

→Generative AI

extras(eagle movie, Q\* Blog Read , ANI,AGI,Cipher and DeCipher)

## [**Lecture 02**](https://docs.google.com/presentation/d/1OegiCFiyhvZRm7SLCvwB6O54_gIGf4KV/edit?usp=drive_link&ouid=101794246168461768671&rtpof=true&sd=true)

**Machine Learning Essentials**

**Agenda**

→Machine Learning

→Types of Machine Learning

→Deep learning

→Generative AI

advance(structured and unstructured data, inference(predict))

Task (Learning Read, Applications of Types of Machine Learning)

## [**Lecture 03**](https://docs.google.com/presentation/d/1qqqufBBB6BNLqtRnN9oI24iCYOkhTjGv/edit?usp=sharing&ouid=101794246168461768671&rtpof=true&sd=true)

**Discriminative vs Generative AI**

→Deep Learning

→Biological and Artificial Neuron

→(Dendrites & input)

→(cell nucleus & Node)

→(Axon & Output)

→(Synapse & Interconnected)

→(Discriminative(Farak krna (either image is dog or cat)) AI)

→is generative AI is Discriminative AI (Yes)

→NLP(Domain of AI (textual processing))

→CV(Domain of AI (visualize processing))

→Speech Processing (Processing Human Voice or Audio Processing)

## [**Lecture 04**](https://docs.google.com/presentation/d/1h6Ig5uFmzCjjDCNdymLd8RWwEMC7G7O0/edit?usp=sharing&ouid=101794246168461768671&rtpof=true&sd=true)

**A Deep Dive into Generative AI**

→Generative Ai have Generative Model

→(Text Generation) Chat GPT and Bard

→(Image Generation) Dall E and Stable Diffusion

→(Music Generation) Music LM

→(Video Generation) Runway ML Gen-2

→Prompt:(Personal(you are a programmer),Task,steps to perform this task,output,output style)

## [**Lecture 05**](https://docs.google.com/presentation/d/120CH38JWpWNuEQ9jo3bqvITIGhkqscpJ/edit?usp=sharing&ouid=104968968817780564391&rtpof=true&sd=true)

**Prompt Engineering**

→Chat Gpt Playground

## [**Lecture 06**](https://docs.google.com/presentation/d/1TxkZ6HX0iZUyIhWsGO1_c0XBeC6it1hz/edit?usp=sharing&ouid=104968968817780564391&rtpof=true&sd=true)

**Diffusion Models**

→latency

→Diffusion

→noice

## [**Lecture 07**](https://drive.google.com/drive/folders/1ofFrLbflhv967dBv9Z9bIDYe6BfO5ov8?usp=drive_link)

**Programming Fundamentals**

→Do Code Yourself not Copy 😂

→Servers

→DevOps

## [**Lecture 08**](https://drive.google.com/file/d/1kDCjzmgDTIumV8f8zMjH28HjZ41GFz5A/view?usp=drive_link)

**String Manipulation Basic Structure in Python**

Free AI Advance Course || Week 4 | Lecture 8 || String Manipulation || Python Basic Structure || Live Session

Yesterday was the 8th Lecture, Dive into Python's Playground: Strings and Structures!

Where we explored and implement the functions in operating strings and python basic structures

Lecture Highlights:

• Date: January 3, 2023

• Time: 7:30 PM - 8:30 PM

• Duration: 1 hour

Expert Instructors :

• Sir Mohammad Irfan.

• Dr. Sheraz Naser (Ph.D. in Artificial Intelligence).

• Sir Muhammad Mubashar

• Sir Farhan Ahmed

Hosted by : Xeven Solutions

Key Learnings:

I - String Manipulation.

1. Case Conversion:

• upper(): Converts all characters in a string to uppercase. Example: "hello".upper() ➞ "HELLO"

• lower(): Converts all characters in a string to lowercase. Example: "HELLO WORLD".lower() ➞ "hello world"

2. Whitespace Removal:

• strip(): Removes leading and trailing whitespace (spaces, tabs, newlines) from a string. Example: " hello ".strip() ➞ "hello"

3. String Replacement:

• replace(old, new, count=-1): Replaces occurrences of a substring with a new substring. Example: "hello world".replace("o", "a") ➞ "hella warld"

4. String Splitting:

• split(sep=None, maxsplit=-1): Splits a string into a list of substrings based on a specified separator. Example: "hello,world,how,are,you".split(",") ➞ ["hello", "world", "how", "are", "you"]

5. Other Useful String Methods:

• startswith(prefix, start=0, end=len(string)): Checks if a string starts with a specified prefix.

• endswith(suffix, start=0, end=len(string)): Checks if a string ends with a specified suffix.

• find(sub, start=0, end=len(string)): Returns the index of the first occurrence of a substring.

• rfind(sub, start=0, end=len(string)): Returns the index of the last occurrence of a substring.

• join(iterable): Joins elements of an iterable (like a list) into a string, using the string as a separator.

Remember:

• These methods generally return a new string rather than modifying the original string.

• You can chain multiple methods together to perform complex operations.

• Python offers a rich set of string methods for various manipulations, providing flexibility in text processing tasks.

II - Python Basic Structure.

1. Lists:

• Ordered collections of items, form of square brackets [ ].

• Mutable: You can change, add, or remove elements after creation.

• Access elements by index (starting from 0).

• Used for storing sequences of data, like shopping lists or arrays of numbers.

• Example:

my\_list = [1, "hello", True]

2. Tuples:

• Ordered collections of items, enclosed in parentheses ().

• Immutable: You can't change elements after creation.

• Access elements by index, similar to lists.

• Used for representing fixed collections of data, like coordinates or configurations.

• Example:

my\_tuple = (10, "apple", True)

3. Sets:

• Unordered collections of unique items, enclosed in curly braces {}.

• Mutable: You can add or remove elements.

• No duplicates allowed.

• Used for membership testing and removing duplicates.

• Example:

my\_set = {1, 2, 3, 3}

4. Dictionaries:

• Unordered collections of key-value pairs, enclosed in curly braces {}.

• Mutable: You can add, remove, or modify key-value pairs.

• Access values using keys (like a real-world dictionary).

• Used for storing and retrieving data by key, like a phonebook or a database.

• Example:

my\_dict = {"name": "Alice", "age": 30}

## [**Lecture 09**](https://drive.google.com/file/d/1TfAMWoFqc5WTW_Q7OHM2KLnXg1-lkW1C/view?usp=drive_link)

Free AI Advance Course || Week 5 | Lecture 9 || Operators in Python || Live Session

Today was the 9th Lecture, Dive into the Operators: Mathematical and Logical Operators!

where we explored and implement the functions involves in the Mathematical and Logical Operations.

Lecture Highlights:

• Date: 9 January, 2023

• Time: 7:30 PM - 8:30 PM, UTC+5:00

• Duration: 1 hour

Expert Instructors :

• Sir Mohammad Irfan.

• Dr. Sheraz Naser (Ph.D. in Artificial Intelligence).

• Sir Mohammad Harris Tariq

Hosted by: Xeven Solutions

Key Learnings:

**I - Operators**

• Operators are special symbols that carry out operations on values and variables. Python supports a variety of operators, each with a specific purpose.

• Here are the main types of operators in Python:

1. Arithmetic Operators:

• Perform basic mathematical calculations.

• Examples: + (addition), - (subtraction), \* (multiplication), / (division), // (floor division), % (modulo), \*\* (exponentiation).

2. Assignment Operators:

• Assigns the value on the right to the variable on the left.

• Examples: x = 10 assigns the value 10 to the variable x.

• x += 5: Adds 5 to x and assigns the result back to x.

• y -= 2: Subtracts 2 from y and assigns the result back to y.

• z \*= 3: Multiplies z by 3 and assigns the result back to z.

• a /= 4: Divides a by 4 and assigns the result back to a.

• b //= 2: Performs floor division of b by 2 and assigns the result back to b.

• c %= 7: Calculates the remainder of c divided by 7 and assigns the result back to c.

• d \*\*= 2: Raises d to the power of 2 and assigns the result back to d.

3.Comparison Operators:

== (equal to): Checks if two values are equal.

!= (not equal to): Checks if two values are not equal.

> (greater than): Checks if the left value is greater than the right value.

< (less than): Checks if the left value is less than the right value.

>= (greater than or equal to): Checks if the left value is greater than or equal to the right value.

<= (less than or equal to): Checks if the left value is less than or equal to the right value.

4.Logical Operators:

Combine multiple Boolean expressions (conditions) to create more complex logical statements.

Produce a Boolean result (True or False) based on the evaluation of the expressions.

and:

Returns True if both expressions are True.

Returns False otherwise.

Example: (x > 5) and (y < 10)

Membership Operators:

To check if two objects are the same object in memory, rather than just having the same value.

Operators:

is: Returns True if both objects point to the same memory location, False otherwise.

is not: Returns True if the objects point to different memory locations, False otherwise.

for writing Python code that performs calculations, makes decisions, and manipulates data.

## [**Lecture 10**](https://drive.google.com/file/d/1p4NWeGF-gdg5e69oboaGhsEnWvpEJ4uE/view?usp=drive_link)

Free AI Advance Course || Week 5 | Lecture 10 || Conditions and Loops in Python || Live Session

Today was the 10th Lecture, Dive into the Conditions and Loops!

where we explored and implement the Control functions involves in the Conditions and iteration.

Lecture Highlights:

• Date: 10January, 2023

• Time: 7:30 PM - 8:30 PM

• Duration: 1 hour

Expert Instructors :

• Sir Mohammad Irfan.

• Dr. Sheraz Naser (Ph.D. in Artificial Intelligence).

• Sir Mohammad Harris Tariq

Hosted by: Xeven Solutions

Key Learnings:-

\*\*I - Conditions:

• Used to make decisions and control the flow of your program.

• Based on Boolean expressions (True or False).

1. if statements: The main decision-maker. If the condition is True, the indented code block following the if is executed.

2. else statements: Used to define what happens if the if condition is False.

3. elif statements: Combine multiple checks in one if statement. These work like "if not the first condition, then check if the second one is true, and so on..."

• Here's an example:

age = 18

if age >= 18:

print("You are old enough to vote!")

else:

print("You are still too young to vote.")

• This checks if the user's age (stored in age) is greater than or equal to 18. If so, it prints a message about voting, otherwise it prints a different message.

\*\*II - Loops:

• Used to repeat a block of code multiple times.

• Two main types: for and while loops.

1. For loops:

• Ideal for iterating over a sequence (think lists, strings, even dictionaries!).

• Each element in the sequence is assigned to a variable in each loop, allowing you to access and process them individually.

• Example:

fruits = ["apple", "banana", "cherry"]

for fruit in fruits:

print(fruit)

• This loop prints each fruit in the fruits list.

2. While loops:

• Keep going as long as a condition remains True.

• You control the condition, deciding when to stop the loop.

• Useful for situations where the number of iterations isn't predetermined.

• Example:

guess = ""

while guess != "secret":

guess = input("Guess the secret word: ")

print("You guessed it!")

• This loop keeps asking for a guess until the user types "secret", then celebrates their success.

## [**Assignment 01**](https://drive.google.com/file/d/1crva6YbFHqWdyNLy_MsPvYJ6Sm54oD7j/view?usp=drive_link)

Collections, Operators, If Else, Loops

[**Notebook Solution**](https://colab.research.google.com/drive/1T-hv5FivbGtQRc_OdiRGr6WJWbZgo7J4?authuser=3#scrollTo=B8VJ8qwxhEXO)

## [**Lecture 11**](#_81qv9c77wszw)

Loops & Functions in python

[Notebook](https://drive.google.com/file/d/11GxwdNKzQxqq2JXMv6WRytbGmA-f6k40/view?usp=drive_link)

Free AI Advance Course || Week 6 | Lecture 11 || Loops and Functions in python || Live Session

Today was the 11th Lecture, Dive into the While loop and user define function!

where we explored and implement the Control functions involves in the While loops and user define functions.

Key Learnings:

\*\*I - While loops:

1.Purpose:

• Repeat a block of code as long as a certain condition remains True.

2.Syntax:

• while condition: # code to be repeated

3.Steps:

• The condition is evaluated.

• If the condition is True, the code within the loop body is executed.

• The condition is evaluated again.

• Steps 2 and 3 repeat until the condition becomes False.

4.Example:

• i = 0 while i < 5: print(i) i += 1 # Ensures the condition will eventually become False.

• Output:

0

1

2

3

4

\*\*II - User-defined functions:

1.Purpose:

• Create reusable blocks of code that perform specific tasks.

2.Syntax:

• def function\_name(parameters): # code to be executed

3.Key elements:

• def: Keyword to define a function.

• function\_name: Name you choose for the function.

• Parameters: Optional inputs the function can receive.

• code to be executed: The block of code that defines the function's actions.

• return: Optional statement to send a value back from the function.

4.Example:

• def square(x): return x \* x result = square(3) # Call the function with an argument print(result) # Output: 9

< Key points:

• Use while loops for indefinite repetition based on a condition.

• Use user-defined functions to organize code, improve readability, and avoid repetition.

• Functions can take parameters for flexibility and return values for further use.

## [**Lecture 12**](#_ym6zb4732778)

Introduction to APIS

[Notebook](https://colab.research.google.com/drive/1jtY3rkoVgGK1_6esnCUS_FLjfC-PAnXT?authuser=3)

Free AI Advance Course || Week 6 | Lecture 12 || Intoduction to APIs || OpenAI API || Live Session

Key Learnings:

\*\*I - APIs(Application Programming Interface):-

• An API, or Application Programming Interface, is a set of functions and protocols that allows different software programs to communicate with each other. Imagine it as a waiter taking your order from a menu and delivering it to the kitchen in the back.

• The waiter (API) understands what you want (your request) and translates it into a language the kitchen (software program) understands. The kitchen then prepares your food (processes the request) and sends it back to you (the response).

• APIs can be used for a variety of purposes, such as:

1.Sharing data: APIs can be used to share data between different applications, such as weather data, news articles, or social media posts.

2.Controlling devices: APIs can be used to control devices, such as smart TVs, thermostats, or even cars.

4.Building applications: APIs can be used to build new applications by combining the functionality of different software programs.

• There are two main types of APIs:

1.Public APIs: These are APIs that are available to anyone to use.

2.Private APIs: These are APIs that are only available to a limited number of users.

\*\*II - OpenAI API

• OpenAI API is a public API that allows developers to access OpenAI's large language models, such as GPT-3 and Jurassic-1 Jumbo. These models can be used for a variety of tasks, such as generating text, translating languages, writing different kinds of creative content, and answering your questions in an informative way.

• Here are some of the things you can do with the OpenAI API:

1.Generate text: You can use the OpenAI API to generate text, such as poems, code, scripts, musical pieces, email, letters, etc.

2.Translate languages: You can use the OpenAI API to translate text from one language to another.

3.Write different kinds of creative content: You can use the OpenAI API to write different kinds of creative content, such as poems, code, scripts, musical pieces, email, letters, etc.

4.Answer your questions in an informative way: You can use the OpenAI API to ask questions and get informative answers.

--> The OpenAI API is a powerful tool that can be used to create a variety of different applications. If you're interested in learning more about how to use the OpenAI API, I recommend checking out the OpenAI API documentation.

## [**Lecture 13**](https://docs.google.com/presentation/d/1bVkqyIueEKrrlCYQlVubs0Nb7r1B3tYF/edit?usp=drive_link&ouid=101794246168461768671&rtpof=true&sd=true)

Inroduction to Hugging Face

[Notebook](https://drive.google.com/file/d/1AnGqpaqi6S3lm7PauA82p7gz5pjnK_yq/view?usp=drive_link)

Free AI Advance Course || Week 7 | Lecture 13 || Introduction to Hugging Face || Live Session

Key Learnings:

\*I- Hugging Face:-

• Hugging Face is an open-source platform for building and sharing state-of-the-art machine learning (ML) models, particularly focused on natural language processing (NLP).

• Think of it as a giant library of pre-trained models in various tasks like text classification, translation, question answering, etc.

• It also offers tools and APIs for developers to easily work with these models, making ML more accessible and collaborative.

• It's constantly evolving: New models and tools are added regularly, reflecting the latest Advancements in AI research.

\*II- Image generation with the Dall-E

• Image generation with the Dall-E model is a Fascinating and powerful technology that allows you to create original, realistic images and art based on your text descriptions. Dall-E can translate such descriptive prompts into visually believable images.

Developed by OpenAI, Dall-E comes in two versions, each with its own capabilities. Here's a breakdown:

1.Dall-E 2:

• Creates original images from text Prompts: You can tell Dall-E 2 what you want to see, like "a photorealistic astronaut riding a horse" or "a watercolor painting of a bustling city square," and it will generate an image based on your description.

2.Dall-E 3:

• Builds upon the capabilities of Dall-E 2: It generates even more Realistic and detailed images, with sharper focus on intricate elements like text, hands, and faces.

\*III- Hugging Face Pipeline:

• In the world of machine learning, particularly Natural Language Processing(NLP) and other AI tasks, Hugging Face Pipelines are a game-changer. They offer an easy-to-use way to run powerful pre-trained models for various tasks without getting bogged down in the underlying complexities.

• ->Hugging Face Pipelines supports a wide range of tasks, including:-

1.Text analysis: Sentiment analysis, topic modeling, named entity recognition, factual language understanding, etc.

2.Content Creation: Text summarization, question answering, creative text generation, dialogue generation, etc.

3.Data classification and Predictions: Image classification, object detection, speech recognition, audio classification, etc.

4.Speech processing: automatic speech recognition, text-to-speech, etc.

5.Computer Vision: image classification, object detection, etc.

6.Multimodal tasks: combining different modalities like text and image for richer analysis.

## [**Lecture 14**](#_iika5suyy80m)

[Notebook](https://drive.google.com/file/d/161eVbL5eIOJmv82xxzxYcRzlSBnK9lU2/view?usp=drive_link)

**Free AI Advance Course || Week 7 | Lecture 14 || Hugging Face Pipelines || Live Session**

Key Learnings:

\*\* Hugging Face Pipelines:-

• Hugging Face Pipelines are essentially shortcuts for using powerful AI models from their platform for various tasks without getting bogged down in complex code. They simplify the process of inference, which means applying the trained model to new data to make predictions or get insights. Think of them as easy-to-use interfaces for tasks like:- Named Entity Recognition (NER), Masked Language Modeling (MLM), Sentiment Analysis, Question Answering, Audio Classification and Image Classification.

\* ° Code Snippet:

from transformers import pipeline

sentiment\_pipeline = pipeline("sentiment-analysis")

review = "This product is absolutely terrible! I wouldn't recommend it to anyone."

sentiment = sentiment\_pipeline(review)

print(f"Sentiment: {sentiment}")

•-->There are two main categories of pipelines:

• Task-specific Pipelines: These are pre-built Pipelines designed for specific tasks like: QuestionAnsweringPipeline, TextClassificationPipeline

• Generic Pipeline: This powerful and flexible pipeline() function allows you to build custom pipelines for any task by specifying the model and processing steps.

I- Text Summarization:

• Hugging Face Pipelines offer a simple and powerful way to use pre-trained NLP models for text summarization. They abstract away the complex code, letting you focus on providing the text to summarize and receiving a concise version. The Hugging Face pipeline makes text summarization accessible and efficient, empowering you to extract key information from large amounts of text with ease.

II- Image Classification

• In Hugging Face, the image classification pipeline is a pre-built tool that makes predicting image classes with powerful pre-trained models a breeze. It takes various forms of images (local path, URL, PIL object) as input and analyzes them using a chosen model to output predictions about their most likely classes.

III- Translation on Pipelines:

• Hugging Face provides a vast repository of pre-trained translation models covering multiple languages. Within the Pipelines framework, you can access these models by specifying the source and target languages.

• Translates text from one language to another using various pre-trained models like T5-base.Provides a user-friendly interface to access these models without diving into complex code.

## [**Lecture 15**](#_ccrbovo3zn2t)

[Lecture PPT](https://docs.google.com/presentation/d/1pPZM_Xn5qP7V9F-n7gAKk2DNdsmlf_kd/edit?usp=drive_link&ouid=101794246168461768671&rtpof=true&sd=true)

**Hope to Skill | Batch 02 || AI Advance Course || Week 8 | Lecture 15 || Data Science Processes || Live Session**

Key Learnings:-

**Deep Dive into Data Science Processes:**

* What is Data Science?

• Data science is a multidisciplinary field that involves extracting knowledge and insights from data. It combines statistical methods, programming, and domain expertise to solve problems, make predictions, and inform Decision-Making. The data science process is iterative and consists of several key stages:

1. Acquiring Data:

1.Identity: This involves understanding the type of data needed, identifying relevant data sources, and obtaining access to them.

2.Retrieve: Different techniques are used to retrieve data depending on the source, such as directly downloading files, scraping websites, or interacting with APIs.

3.Query: Structured data might require writing queries using languages like SQL to filter and extract specific information.

2. Data Preparation:

• Data preparation involves cleaning, manipulating, and transforming data to make it suitable for analysis and modeling. It consists of two main stages:

I- Exploratory Data Analysis(EDA):

a) Preliminary Analysis:

• Correlation: Identify relationships between variables.

• Outliers: Detect and potentially handle anomalous data points.

• General Trends: Discover patterns and trends in the data.

b) Understand the Nature of Data:

1.Types of Data: Categorical, numerical, ordinal, etc.

2.Types of Attributes: Independent, dependent, target, etc.

3--> Statistical Distribution of Attributes:

1.Central Tendency: Measures like mean, median, and mode indicate where the data is concentrated.

2.Dispersion: Measures like variance and standard deviation quantify how spread out the data is.

3.Proximity: Metrics like skewness and kurtosis capture the shape of the distribution.

II- Data Preprocessing:

• This involves specific techniques to improve data quality and suitability for modeling:

1.Data Cleaning: Identifying and correcting missing values, inconsistencies, and errors.

2.Data Integration: Combining data from multiple sources while resolving conflicts and ensuring consistency.

3.Data Reduction: Decreasing the data size while preserving information, using techniques like sampling or dimensionality reduction.

4.Data Transformation: Scaling, normalization, and encoding categorical variables to prepare data for specific algorithms.

## [**Lecture 16**](#_jjgneplv4j61)

[Lecture PPT](https://docs.google.com/presentation/d/1fI096Zk8yww3QExvc26a__VOil4ExuuO/edit?rtpof=true)

**Hope to Skill | Batch 02 || AI Advance Course || Week 8 | Lecture 16 || Data Science Processes || Live Session**

**Key Learnings:-**

**Deep Dive into Data Science Processes:**

* **Data Transformation**

Data transformation refers to the process of modifying raw data into a format suitable for analysis and modeling. It ensures the data is accurate, consistent, and relevant to the desired analysis. Some key aspects:

1.Smoothing: Aims to remove noise and reduce volatility in data by averaging neighboring data points. This helps identify underlying trends and patterns more clearly.

2. Feature Engineering: Involves creating new features or modifying existing ones to improve the performance of data analysis and modeling.

3. Normalization: Scales data into a specific range, often [0, 1] or [-1, 1], to ensure all features have equal influence on the analysis. Common techniques include min-max scaling and z-score normalization.

* **Data Analyzation**

Analyzation involves extracting insights and knowledge from transformed data using various statistical and machine learning techniques. Some key aspects:

1. Technique Selection: Choosing the appropriate data analysis technique depends on the type of data and the desired outcome.

2. Model Building: This involves developing algorithms or models based on the chosen technique to analyze the data and derive insights. Some types:

Classification: Categorizes data points into predefined classes (e.g., predicting if an email is spam or not).

Regression: Predicts continuous values based on independent variables (e.g., predicting house prices based on size and location).

Clustering: Groups similar data points together without predefined classes (e.g., identifying customer segments based on purchase history).

Associative Analysis: Discovers relationships and associations between variables (e.g., finding items frequently bought together).

Generative Analysis: Creates new data points that follow the underlying distribution of the existing data (e.g., generating realistic images or text).

3. Model Validation: Evaluating the performance of the developed model to ensure its accuracy and reliability. This involves testing the model on unseen data and analyzing metrics like accuracy, precision, recall, and F1 score.

* **Reporting**

Effectively communicating the findings of data analysis is crucial. Reporting involves summarizing results and presenting them in a clear and concise manner.

1. Visualization: Plays a key role in reporting, as it allows viewers to easily understand complex data patterns and relationships. Some key aspects:

Bar Charts: Compare different categories or groups of data.

Line Charts: Show trends and changes over time.

Pie Charts: Represent proportions of a whole.

Heat Maps: Visually represent the distribution of data across two dimensions.

Scatter Plots: Show relationships between two variables.

## [**Lecture 17**](#_9ataef9bltzv)

**Classification Using Scikit Learn**

**1. Visualization of Data**

- \*\*Overview\*\*: Before diving into classification using Scikit Learn, it's essential to understand the data you're working with. Visualization techniques help in gaining insights into the structure and characteristics of the dataset.

- \*\*Purpose\*\*: Visualization aids in identifying patterns, relationships, outliers, and distribution of data features.

- \*\*Techniques\*\*: Common visualization techniques include histograms, scatter plots, box plots, pair plots (for multivariate analysis), heatmaps, etc.

- \*\*Scikit Learn Integration\*\*: Scikit Learn provides compatibility with popular visualization libraries such as Matplotlib and Seaborn, making it easy to generate visualizations directly from your data.

**2. Preprocessing of Data**

- \*\*Overview\*\*: Data preprocessing involves preparing the dataset for model training by handling missing values, scaling features, encoding categorical variables, and splitting the data into training and testing sets.

- \*\*Purpose\*\*: Preprocessing ensures that the data is in a suitable format and quality for the machine learning model.

- \*\*Techniques\*\*: Techniques include handling missing data (imputation), feature scaling (e.g., normalization or standardization), encoding categorical variables (e.g., one-hot encoding), and splitting data into training and testing sets.

- \*\*Scikit Learn Integration\*\*: Scikit Learn provides various preprocessing modules such as `SimpleImputer`, `StandardScaler`, `OneHotEncoder`, and `train\_test\_split` for easy data preprocessing.

**3. Model Fitting Using Scikit Learn**

- \*\*Overview\*\*: Scikit Learn offers a wide range of classification algorithms that can be easily implemented for model fitting. Common algorithms include Logistic Regression, Decision Trees, Random Forests, Support Vector Machines (SVM), k-Nearest Neighbors (k-NN), etc.

- \*\*Purpose\*\*: Model fitting involves training the selected classification algorithm on the preprocessed training data to learn patterns and relationships between features and labels.

- \*\*Implementation\*\*: Choose the appropriate classification algorithm based on the problem and dataset characteristics. Then, instantiate the model object, fit it to the training data using the `fit()` method, and adjust hyperparameters if necessary.

- \*\*Scikit Learn Integration\*\*: Scikit Learn provides a consistent interface for fitting models, making it easy to switch between different algorithms without changing the code structure significantly.

**4. Model Evaluation**

- \*\*Overview\*\*: Once the model is trained, it needs to be evaluated to assess its performance and generalization capabilities on unseen data.

- \*\*Purpose\*\*: Model evaluation helps in understanding how well the classification algorithm is performing and whether it's suitable for the given task.

- \*\*Metrics\*\*: Common evaluation metrics for classification tasks include accuracy, precision, recall, F1-score, ROC curve, and confusion matrix.

- \*\*Scikit Learn Integration\*\*: Scikit Learn provides functions and modules to calculate these evaluation metrics conveniently, such as `accuracy\_score`, `precision\_score`, `recall\_score`, `f1\_score`, `roc\_curve`, and `confusion\_matrix`.

**5. Result Analysis**

- \*\*Overview\*\*: After evaluating the model, it's essential to analyze the results to gain insights into its strengths, weaknesses, and areas for improvement.

- \*\*Purpose\*\*: Result analysis helps in refining the model, optimizing hyperparameters, and making informed decisions about further steps in the machine learning pipeline.

- \*\*Techniques\*\*: Techniques for result analysis include examining misclassified instances, understanding feature importance, visualizing decision boundaries, and comparing different models' performances.

- \*\*Scikit Learn Integration\*\*: Scikit Learn provides tools for result analysis, such as feature importance scores for tree-based models, plotting decision boundaries, and comparing models using cross-validation techniques.

By covering these topics, the lecture aims to provide a comprehensive understanding of classification using Scikit Learn, from data visualization and preprocessing to model fitting, evaluation, and result analysis. These concepts form the foundation for building effective classification models in real-world scenarios.

## [**Assignment 02**](https://drive.google.com/drive/folders/1Hq4Slh8QA1Fs9ITxJT-ZxsoIPGatIens?usp=drive_link)

## [**Lecture 18**](#_nbc6eukwl5sn)

[Lecture Slide](https://docs.google.com/presentation/d/1bWGqgudFBUBcJYOIumm_UsrOrL90iK9y/edit?usp=drive_link&ouid=101794246168461768671&rtpof=true&sd=true)

[DATA](https://drive.google.com/drive/folders/1aXGB6ltKKKE4H8HErmxYl8L1iu0Zm3kE?usp=drive_link)

Hope to Skill | Batch 02 || AI Advance Course || Week 10 | Lecture 18 || Classification Algorithms

Today was the 18th Lecture, Dive into the Machine Learning Classifiers!

where we explored and Implement the different Classification Algorithms and techniques with using scikit-learn.

Key Learnings:

I- Classification Techniques in Machine Learning:

In machine learning, classification involves predicting the Category an unseen data point belongs to, based on Labeled examples. Here's a breakdown of the techniques:

1.Logistic Regression:

A Linear Model that predicts the probability of an instance belonging to a specific class.

Often used for binary classification (yes/no, spam/not spam), but can handle multi-class problems with adjustments.

2.Decision Tree Classifier:

Creates a tree-like structure where each node represents a feature and branches represent decision rules based on the Feature value.

Easy to interpret, but prone to Overfitting if not carefully tuned.

3.Random Forest Classifier:

Ensembles multiple decision trees trained on different subsets of data and aggregates their Predictions.

More robust to overfitting than single decision trees and handles complex relationships between features.

4.Gradient Boosting Classifier:

Sequentially builds an ensemble of weak learners (often decision trees) where each learner focuses on correcting the mistakes of the previous one.

Effective for both Binaryic and multi-class problems, particularly with complex data.

II- Training Models with scikit-learn:

Scikit-learn is a popular Python library for Machine Learning. Here's a brief Overview of training each model:

1.Logistic Regression:

from sklearn.linear\_model import LogisticRegression

Define Model parameters (solver, regularization, etc.)

Fit the model to training data: model.fit(X\_train, y\_train)

Use the model to PREDICT on new data: model.predict(X\_test)

2.Decision Tree Classifier:

from sklearn.tree import DecisionTreeClassifier

Define tree parameters (max depth, splitting criteria, etc.)

Fit the model: model.fit(X\_train, y\_train)

PREDICT: model.predict(X\_test)

3.Random Forest Classifier:

from sklearn.ensemble import RandomForestClassifier

Define forest parameters (number of trees, max depth, etc.)

Fit the Model: model.fit(X\_train, y\_train)

Predict: model.predict(X\_test)

4.Gradient Boosting Classifier:

from sklearn.ensemble import GradientBoostingClassifier

Define parameters (number of learning stages, learning rate, etc.)

Fit the model: model.fit(X\_train, y\_train)

Predict: model.predict(X\_test)

## [**Lecture 19**](#_tnf716gamxqn)

Hope to Skill | Batch 02 || ADVANCE.AI Course || Week 10 | Lecture 19 || Regression Algorithms

Key Learnings:

\*I- Regressionly Techniques in Machine Learning•

1.Linear Regression

• A Linear relationship between an independent variable(X) and a dependent variable(y) to predict y based on x. This Represents the relationship as a straight line.

Equation: fw,b(x) = w.x + b

<-Cost Function:-

• A mathematical Measure that quantifies the error or "cost" between the model's predicted values(ŷ) and the actual values(y) in the Training data

Equation: J(w,b) = 1/(2m)Σ(ŷ - y)^2

• The cost function tells us how well the line fits those points. The lower the cost, the better the fit. Mean Squared Error is a frequently used cost function.

2.Decision Tree Regressor:

Tree-like structure where each node represents a split on a feature, leading to leaf nodes with average target values.

Handles non-linear relationships, robust to outliers, easy to interpret.

Prone to Overfitting, less accurate than other methods for large datasets.

3.RANDOM Forest Regressor:

Ensemble of decision trees, averaging their predictions for robustness.

Less prone to overfitting than single decision trees, handles non-Linearity and dimensionality, robust to outliers.

Less interpretable than individual trees, may be computationally expensive for large datasets.

4.Gradient Boosting Regressor:

Builds decision trees Sequentially, each correcting the errors of the previous one.

Very powerful, handles non-linearity and complex relationships, robust to outliers.

Can be computationally expensive, harder to interpret than other methods.

\*II- Training Models•

scikit-learn provides efficient implementations of these techniques in Python. Here's a general workflow:

1.Import Necessary libraries:

from sklearn.linear\_model import LinearRegression

from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor

Create an instance of the Model: model = LinearRegression()

# Or other regressor object

2.Split data into Training and testing sets:

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)

# Split 20% for testing

3.Fit the model to the Training data:

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

mse = mean\_squared\_error(y\_test, y\_pred)

## [**Lecture**](#_l1ikfrltnb5h) **20**

Model Evaluation